ClubUML

CSYE7945 Spring 2013

Use Cases

Chris Serrano

Sarvesh Vichare

Naveen Satya

Steven Huang

Yidu Liang

Huichao Tan

Minhee Song

Rui Hou

2013-03-10

Revision: 2.0

Table of Contents

[Revision History 3](#_Toc350688572)

[Introduction 4](#_Toc350688573)

[Use Case Diagram 5](#_Toc350688574)

[RegisterNewAccount Use Case 6](#_Toc350688575)

[Log in Use Case 8](#_Toc350688576)

[UploadDiagram Use Case 9](#_Toc350688577)

[CompareDiagrams Use Case 10](#_Toc350688578)

[MergeDiagram Use Case 12](#_Toc350688579)

[DownloadProject 15](#_Toc350688580)

[DownloadProject Use Case 15](#_Toc350688581)

[DownloadProject Sequence Diagram 16](#_Toc350688582)

[RemoveDiagram 17](#_Toc350688583)

[RemoveDiagram Use Case 17](#_Toc350688584)

[RemoveDiagram Sequence Diagram 18](#_Toc350688585)

[Acknowledgements 18](#_Toc350688586)

[Appendix A: Comment Use Case 19](#_Toc350688587)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author(s)** | **Notes** |
| 1.0 | 2/5/13 | Chris Serrano & others listed | Includes previous docs as described in Introduction |
| 2.0 | 3/10/13 | Chris Serrano & others listed | Added DownloadProject, RemoveDiagram, clarification to UploadDiagram, CompareDiagrams, MergeDiagram |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

This document is a compilation of all use cases for the ClubUML system. Here is a summary of the work breakdown:

* Document compiled by Chris Serrano
* Use case diagram is by Minhee Song
* RegisterNewAccount use case is by Huichao Tan and Chris Serrano
* Log in use case is by Yidu Liang and Chris Serrano
* UploadDiagram use case is by Chris Serrano
* MergeDiagram use case is by Steven Huang
* CompareDiagrams use case is by Sarvesh Vichare and Naveen Satya
* Comment use case is by Sarvesh Vichare and Naveen Satya
* DownloadProject use case and sequence diagram are by Minhee Song
* RemoveDiagram use case is by Rui Hou and sequence diagram is by Steven Huang

# Use Case Diagram

A diagram of all use cases in the ClubUML system is shown in Figure 1. There is a RegisterNewAccount use case for new users to the system who need to register a user name and password. UploadDiagram describes a user which merely wants to upload a diagram to the database, without doing any compare or merge functions. MergeDiagram describes the user manually choosing which elements to keep and which to discard when comparing the differences in two diagrams. CompareDiagrams describes the user comparing two diagrams and generating a report of the differences between diagrams (without making a merged diagram). The UploadDiagram, MergeDiagram, and CompareDiagrams use cases all include the basic Log in use case. Additionally, there is a Comment use case in Appendix A which is not included in the use case diagram since it is a maintenance task and we have not yet determined if it will be added or if the current comment implementation will be kept.

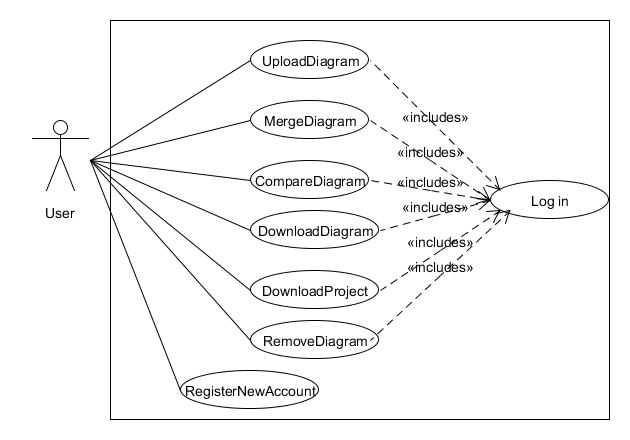


Figure : ClubUML System Use Case Diagram

# RegisterNewAccount Use Case

**Brief Description**

This use case describes the account registration process for ClubUML.

**Actors**

User: The user is the actor who needs to create a new account to access ClubUML.

**Preconditions**

1. The user does not have an account.
2. The ClubUML webapp is correctly deployed either on a server or on User’s local machine
3. The user meets the Java Runtime Environment and browser requirements to run ClubUML.

**Flow of events**

1. The user navigates to the ClubUML login page.
2. The user clicks the button to register a new account.
3. The user is redirected to the new account creation page.
4. The user enters the desired username, password, security question and answer, and his or her email address (for password recovery).
   * User name must be 1-10 alphanumeric characters
   * Password must be 1-10 alphanumeric characters
   * Two password fields must match
5. The user clicks the register button to request the account be created.
6. The user is redirected to the ClubUML login page.

**Alternate Flow**

**User leaves registration fields blank**

If the user does not complete all the fields, the registration page is refreshed but with an error message at the top of the page telling the user to complete all required fields.

**User enters invalid entries**

If the user enters too many characters into any field or unacceptable special characters, the registration page is refreshed but with an error message at the top of the page telling the user to enter the specified number of characters with no special characters.

**Password fields don’t match**

If the two password fields don’t match, the registration page is refreshed but with an error message at the top of the page telling the user to re-enter the same password in both fields.

**User requests user name which is already taken**

If the user enters a user name which already exists in the database, the registration page is refreshed but with an error message at the top of the page telling the user that the requested user name already exists.

**Post-conditions**

1. **Valid registration request**

User arrives at ClubUML login page.

1. **Invalid registration request**

User is returned to ClubUML registration page with an error message on the page.

# Log in Use Case

**Brief Description**

This use case describes the log in process for a user to access the ClubUML system.

**Actors**

User: The user is the actor who logs in to his or her account.

**Preconditions**

1. The user has an account (see RegisterNewAccount for account creation process).
2. The ClubUML webapp is correctly deployed either on a server or on User’s local machine
3. The user meets the Java Runtime Environment and browser requirements to run ClubUML.

**Flow of events**

1. The user navigates to the ClubUML login page.
2. The user enters his or her user name and password in the appropriate boxes.
3. The user clicks the log in button.
4. The user is redirected to the successful log in page.
5. The user clicks the button to continue to ClubUML.
6. The user is redirected to the main ClubUML webapp page.

**Alternate Flow**

**User enters incorrect user name and/or password**

If the user does not enter their correct account information, the ClubUML login page is refreshed with an invalid login error message now at the top of the page.

**Post-conditions**

1. **Correct login credentials entered**

User arrives at main ClubUML webapp page.

1. **Incorrect login credentials entered**

User is returned to ClubUML login page with an error message on the page.

# UploadDiagram Use Case

**Brief Description**

This use case describes the process for uploading a UML diagram to ClubUML.

**Actors**

User: The user is the actor who is uploading a UML diagram.

**Preconditions**

1. The user follows Login use case flow to access ClubUML.
2. The diagram to be uploaded is either an Ecore diagram or the set of three Papyrus file types.
3. The diagram is either a class diagram (Ecore or Papyrus) or a sequence diagram (Papyrus only).

**Flow of events**

1. The user selects Ecore or Papyrus file type.
2. The user clicks on the “Choose file” button.
3. The file dialog box appears.
4. The user selects one file to upload from the file dialog box.
5. The user repeats steps 2-4 if there are other files to be uploaded.
6. The user clicks the “Upload” button after files have been selected.
7. After upload is successful, the ClubUML page shows the new diagram in the list of diagrams uploaded to the project.
8. The uploaded diagram image is shown next to the list of files in the project.

**Alternate Flow**

**User selects an invalid file type**

If the user selects a file that isn’t the appropriate type, or doesn’t select the three types of Papyrus file (.uml, .notation, .di), the system informs the user that the upload is not valid.

**User selects a Papyrus XMI diagram which is not a Class or Sequence diagram**

If the user selects an invalid diagram type, the system alerts the user with an error that lists the compatible diagram types (class or sequence).

**There is an error parsing the file**

If there is an error in the parser, implying a corrupt file, the system alerts the user that there was an error parsing that file and the file may be corrupt.

**Post-conditions**

1. **Successful Completion**

The file is successfully added to the ClubUML file list.

1. **Failure Condition**

No file is uploaded and an error is displayed.

# CompareDiagrams Use Case

**Brief Description**

This use case portrays the comparison of two diagrams selected by the user, with support for comments to aid in collaboration.

**Actors**

User: The user is the actor who invokes the comparison and reviews the report.

**Preconditions**

1. The user follows Login use case flow to access ClubUML.
2. The user has already uploaded two valid diagrams of the same type (following UploadDiagram flow of events).

**Flow of events**

1. The user follows Login use case flow to access ClubUML.
2. The user selects two diagrams from the list of diagrams in the project, which are the same type and have been output by the same piece of software to ensure compatibility.
3. The user clicks “Go to Compare” button for the selected diagram of the same type.
4. ClubUML generates a detailed comparison report in a display panel within the web page.
5. The user has the option to click the “save” button to download a PDF report of the comparison (the PDF report does not pop up automatically as in the previous version).
6. The user decides which diagram is preferred and adds a comment under the preferred diagram.
7. The user then clicks the Promote button next to the comment box.
8. The user is returned to the main ClubUML page and the comment can be seen under the promoted diagram.

**Alternate Flow**

**User selects less than two diagrams**

If the user selects less than two diagrams the system alerts with a pop-up saying, “*Please select at least two diagrams”*

**User selects more than two diagrams**

If the user selects more than two diagrams the system alerts with a pop-up saying, *“Please select at most two diagrams”*

**No Response from Server**

If in process of preparing the comparison report there is no response from the server, then:

1. ClubUML application shall display the message "Network unavailable – try again".

2. The use case ends with a failure condition.

**Key Scenarios**

If the button “save” has not been clicked, no file would be saved.

**Post-conditions**

1. **Successful Completion**

A detailed comparison report will be generated.

1. **Failure Condition**

No comparison report is generated.

# MergeDiagram Use Case

**Brief Description**

This use case describes a manual merge functionality, in which the user selects which elements to keep from two related diagrams.

**Actors**

User: The user selects the diagrams to merge and makes decisions on merging.

**Preconditions**

1. The user follows Login use case flow to access ClubUML.
2. The user has already uploaded two valid diagrams of the same type (following UploadDiagram flow of events).
3. The uploaded diagrams to be merged are from Papyrus (no Ecore merge support).

**Basic Flow of Events**

1. The user selects two diagrams of the same type (class or sequence), both Papyrus files, to be merged.
2. The user clicks the “merge” button.
3. The webpage shows the two diagrams side by side. Under each diagram, there is a list of:
   1. Classes in diagram A only.
   2. Classes in diagram B only.
   3. Classes in common with diagram A and diagram B.
4. The user can select any pair of classes to merge if they are not already in common with both diagrams, e.g. class Bike in diagram A and class Bicycle in diagram B.
5. The user makes the selection of which elements to be kept in the merged class by checking a box (or some similar type of input) next to each desired element.
6. The user clicks another “merge” button to merge the class.
7. The user will be prompted to resolve any conflicts with associations in the newly merged class by picking which type of association is appropriate in each situation.
8. The user can repeat steps 4-7 to merge other classes which are not in common.
9. When the user is finished merging classes, the user clicks another Merge button to finalize the process and generate the overall merged diagram.
10. The UI displays a new merged diagram with the versions of each element combined.
11. The user can save the merged diagram.
12. The user can export Papyrus files of the merged diagram (see DownloadProject use case).
13. The user can comment on the merged diagram.
14. The user can return to the main ClubUML page after doing any combination of saving the drawing, exporting the merged diagram, or commenting.
15. The merged diagram will appear on the main ClubUML page along with the other diagrams that have been uploaded.

**Alternative Flows**

**Incorrect type of diagrams**

If user selects two different types of diagrams, then:

1. An error dialog will be prompted and ask the user to select two diagrams with the same type.
2. The use case returns to basic flow step 1.

**Number of diagrams error**

If user selects more or less than two diagrams, then:

1. An error dialog will be prompted and ask the user to select more or less diagrams to be merged.
2. The use case returns to basic flow step 1.

**Two exact diagrams are selected**

If user selects two diagrams that are exactly the same, then

1. A dialog will be prompted notifying user that the two diagrams are the same. Neither selections of merging nor saving of the diagram is allowed.
2. The use case returns to basic flow step 1.

**Subflows**

**No merging**

1. The user doesn’t make the request to merge diagrams
2. The user can return to the main ClubUML page, and the use case ends.

**No saving diagram**

1. The user chooses not to save an image of the merged diagram.
2. No saving will be done.

**No exporting XMI**

1. The user chooses not to export an XMI file of the merged diagram.
2. No exporting will be done.

**No commenting**

1. The user chooses not to leave any comment.
2. No comment will be added to the diagram.

**Key Scenarios**

No diagrams will be saved unless the user makes the request.

**Post-conditions**

1. The merged diagram is saved with comments, if any.
2. No diagrams are lost.

**Special Requirements**

The merged diagram has the same type as the two original diagrams.

The merged diagram can be merged again with other diagrams, if the user makes the request.

# DownloadProject

This use case is for improving the download functionality. The current download functionality gets only a png file of the diagram. User needs to download all uploaded diagrams and the XMI files to re-open the documents in Papyrus at the same time. In order to improve download functionality, it provides with ‘download project’ functionality that zip /uploads folder and generate .zip file.

## DownloadProject Use Case

**Brief Description**

This use case describes the diagrams downloading process for ClubUML.

**Actors**

User: The user is the actor who needs to download previously uploaded UML diagrams.

**Preconditions**

1. The user has an account and has already logged in successfully.
2. Server has existing or uploaded diagram.

**Flow of events**

1. The user follows Login use case flow to access ClubUML.
2. The user clicks “Download Project” button.
3. ClubUML prompts the user for download location.
4. User enters the location directory.
5. .zip file is saved to the specified location.
6. Project page is redisplayed.
7. The user case ends successfully.

**Post-conditions**

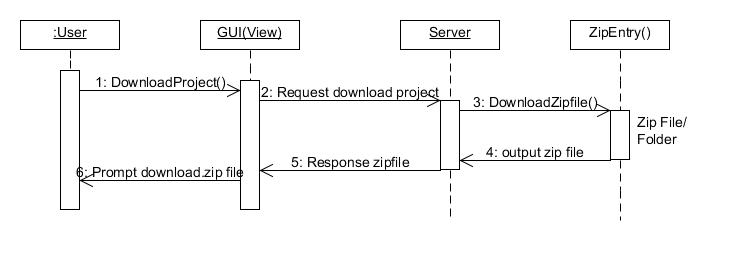
1. **Successful Completion**

Diagram downloaded and internal logs updated if any.

1. **Failure Condition**

Logs updated if any.

## DownloadProject Sequence Diagram



# RemoveDiagram

It is one inevitable situation: in most cases, users of ClubUML upload plenty of diagrams and their diagram list look messy, which will definitely cause unsatisfaction and make us lose our users. The solution is adding one “Remove” button to clear things up.

## RemoveDiagram Use Case

**Brief Description**

This use case describes the diagrams removing process for ClubUML.

**Actors**

User: The user is the actor who needs to remove diagrams from diagram list.

**Preconditions**

1. The user has an account and has already logged in successfully.
2. There is at least one diagram on user’s diagram list. If there is none diagram on the list, then the remove button will be disabled(in gray color) until user upload at least one diagram.
3. The user can remove only one diagram at a time.

**Flow of events**

1. The user follows Login use case flow to access ClubUML.
2. The user selects at least one diagrams from the list of diagrams in the project.
3. The user clicks “Remove” button.
4. The selected diagram is deleted from user’s diagram list.

**Alternate Flow**

**User selects no diagram and click “Remove”**

If the user selects no diagram and click on “Remove” button, the system will alert the user with an error that says “no file selected”

**Post-conditions**

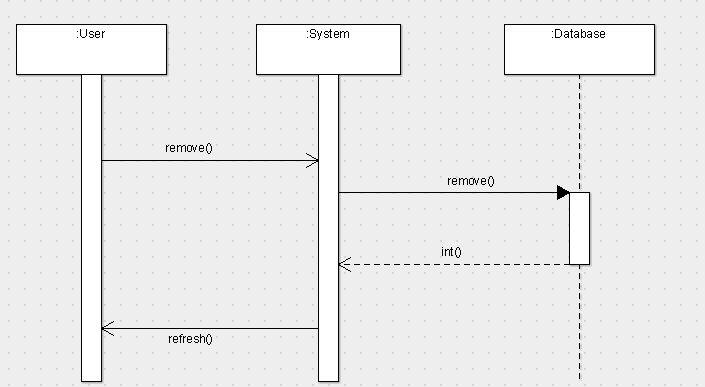
1. **Successful Completion**

The file is successfully remove from user’s diagram list.

1. **Failure Condition**

No diagram is deleted and an error is displayed.

## RemoveDiagram Sequence Diagram



## Acknowledgements

Pranshant suggestion for gray out the remove button if there is none diagram on list.

Zessie and Christian’s suggestions for user’s authorizations to remove diagrams.

# Appendix A: Comment Use Case

**This use case was written with improved comment capabilities in mind as a set of maintenance tasks. It has not yet been determined whether it will be implemented. If it is to be implemented, it will need to be included as an additional use case in the use case diagram.**

**Brief Description**

This use case portrays improvised comparison of two diagrams after rightful suggestion from the user with comment collaboration.

**Actors**

User: The user is the actor who invokes the comparison and reviews the report.

**Preconditions**

1. The user follows Login use case flow to access ClubUML.
2. Current logged in user can only edit/delete his previous comments. *[In other words, under current user settings he would be seeing only edit/ delete button for his comments]*
3. Current login user can reply to the comments from others user only.
4. Comments cannot exceed 250 characters.

**Flow of events**

1. The user clicks on “Review Compared Reports” button.
2. ClubUML displays all reports uploaded by multiple users in the file selection panel.
3. The user browses all the reports and selects the one for review purpose.
4. The user clicks “Review” button for the selected report.
5. ClubUML displays a detailed comparison report in a display panel with download and commenting option.
6. The user can click on “save” button and ClubUML will save the report to your local machine.
7. The user clicks on “Comment” button, which results in a dialogue box appearance in the commenting panel.
8. The user proceeds by posting the comment for the desired report with “post” button.
9. After successfully posting the comment the current user can perform “edit” or “delete” function if required.
10. While clicking the “edit” button a dialogue box appears with the scope to improvise on the comment and after improvising, “Submit” button is clicked.
11. While clicking the “reply” button a dialogue box appears displaying the comment on which the current user will be replying. “Submit” button takes the current user to the main panel.
12. On clicking the “delete” button the main panel gets refreshed.
13. The use case ends successfully.

**Alternate Flow**

**User exceeds 250 characters**

If the user exceeds 250 characters the system alerts with a pop-up saying, “*Comment exceeds 250 characters”*

**No Response from Server**

If in process of commenting there is no response from the server, then:

1. ClubUML application shall display the message "Network unavailable – try again".

2. The use case ends with a failure condition.

**Key Scenarios**

As long as the button “submit/post” has not been clicked, no comments will be posted.

**Post-conditions**

1. **Successful Completion**

Comment is posted and is available for future edit/delete functionality.

1. **Failure Condition**

Comment not posted.

Comment posted but doesn’t display “edit” and “delete” button.