VISIOSVN

System Document

Contents

1 System Design

This chapter will first describe the Functional view of the proposed system followed by the use case diagram, use case descriptions and the model diagram. After that the GUI design will be described in detail. Then it will further describe the Architectural design used for the system development.

1.1 **Use case diagram**

1.2 **Use case descriptions**

Use case name	Set URL and Authentication.
Summary	This use case is for users to log in to the system and set URL of a repository. This encapsulates the authentication process of the users.
Actors	User.
Preconditions	User run the application and displays the logging interface.
Basic course of events	Step 1) User enters a valid User Name. Step 2) User enters a valid password for the User Name. Step 3) User enters a valid URL of a repository. Step 4) The system Validate the User name, Password and URL. Step 5) The System finishes the authentication process and let the user work with further interfaces.
Exceptional course of events	1. User enters an invalid username. 2. User enters an invalid password. 3. User enters an invalid URL.
Post Conditions	User log in to the system and view the main interface.

Use case name	View repository image.
Summary	This use case is for viewing content of the repository.
Actors	User
Preconditions	User logs in to the system.
Post Conditions	User can use other functions such as drag, zoom in/out, view revision.

Use case name	Zoom in/out
Summary	This use case is for viewing inner contents of a project or folder (zoom in) and return to the previous stage (zoom out).
Actors	User.
Preconditions	User logs in to the system and views repository image.
Basic course of events	1) User clicks on a folder or project folder (nodes) in a repository to view inner contents. 2) User clicks on the main (middle) node to return to previous stage.
Exceptional course of events	1. User clicks on a file where inner contents cannot be existed.
Post Conditions	View inner contents/ view previous stage.

Use case name	Search files/folders
Summary	Most of repositories contain lots of files and folders. Therefore it is hard to identify a specific file or folder. This function searches folders and view inner contents of it and search files.
Actors	User.
Preconditions	User enters the search keyword in the search area. Then user selects the appropriate name from the suggestion list and clicks the search button.
Basic course of events	Step 1) User enters the keyword in search area. Step 3) Select the right name from the suggestion list. Step 2) Click search button.
Exceptional course of events	User enters invalid name of a file/folder.
Post Conditions	User views the inner content of the searched folder and identifies the searched file.

Use case name	Drag nodes.
Summary	This use case is for moving nodes on the canvas. If a node overlap with another user can drag in to another location to view the repository image more clearly.
Actors	User.
Preconditions	User logs in to the system and views repository image.
Basic course of events	Step 1) User clicks on a node and drags it on canvas o another location.
Post Conditions	Displays a better repository image.

Use case name	Read externals.
Summary	This use case is for reading and viewing svn externals in the repository.
Actors	User.
Preconditions	User logs in to the system and views content of a project folder.
Basic course of events	Step 1) User clicks on a folder which contains externals. Step 2) System displays the externals.
Post Conditions	Externals are displayed on canvas and user can see content of them.

Use case name	View revision (full or in a range)
Summary	This use case is for viewing the revision graph of a project/folder/file. This displays revision types such as added, modified, renamed, merged and deleted. User can filter the revision graph to view revisions in a given range.
Actors	User.
Preconditions	User logs in to the system and views repository image.
Basic course of events	Step 1) User right clicks on a file/folder on which the revision graph is needed. Step 2) System displays a popup menu. Step 3) User selects the "revision graph" menu item. Step 4) System displays compressed revision graph of the selected file/folder. Step 5) User can view full revision graph by clicking expand button. Step 6) User clicks on the filter button in the canvas to view revision in a given range. Step 7) User gives a range using revision numbers in the filter interface and clicks ok. Step 8) System displays revision in given range on the canvas.
Exceptional course of events	User enters invalid range of revisions.
Post Conditions	Revision graph is displayed on the canvas.

Use case name	View tags and revision data.
Summary	This use case is for viewing revision data such as author, date, revision number, log message, url.
Actors	User.
Preconditions	User logs in to the system and views revision graph.
Basic course of events	1) User move mouse pointer to a revision to view the tool tip which contains revision data.
Post Conditions	Revision data displayed in the revision graph.

Use case name	Authentication management
Summary	This use case is for checking correct user names, passwords and urls.
Actors	System.
Preconditions	User enters user name, password and url.
Basic course of events	Step 1) Check for correct url and make a repository. Step 2) Check for correct username and password.
Post Conditions	Allows system to proceed.

Use case name	Read data from URL
Summary	This use case is for getting data from a repository and displays them.
Actors	System.
Preconditions	Authentication pass.
Basic course of events	Step 1) Identify content of a repository. Step 2) Display them.
Post Conditions	A repository data list is updated.

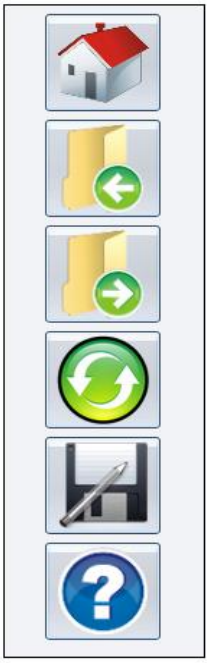
Use case name	Identify file/folder/root/revision types
Summary	This use case is for identifying different node kinds (file/folder/root/revision types) to use different shapes to display them.
Preconditions	System reads repository data.
Basic course of events	Step 1) Identify node kinds/ revision types. Step 2) Assign a shape to node kinds.
Post Conditions	A data list is updated.

Use case name	Display data
Summary	This use case is for displaying all the data on the canvas.
Actors	System.
Preconditions	Exists updated data lists.
Basic course of events	Step 1) System get data from updated data lists. Step 2) Draw nodes according to data. Step 2) Connect nodes according to data.
Post Conditions	Repository image/ Revision graph is displayed.

1.3 **GUI Design**

Since the users of this application are subversion users, the GUI should be easy to adapt and easy to navigate through the data. To get the maximum use out of it, GUI contains integration of both Windows Forms. The GUI contains a light theme for repository graph and it has many more user friendly facilities. For navigation through data and do the other processes with the application, it contains a JToolBar which is very familiar with most of the users in repository graph canvas and revision graph canvas. Below is the description of main features the GUI has build with.

JToolBar in repository graph canvas.



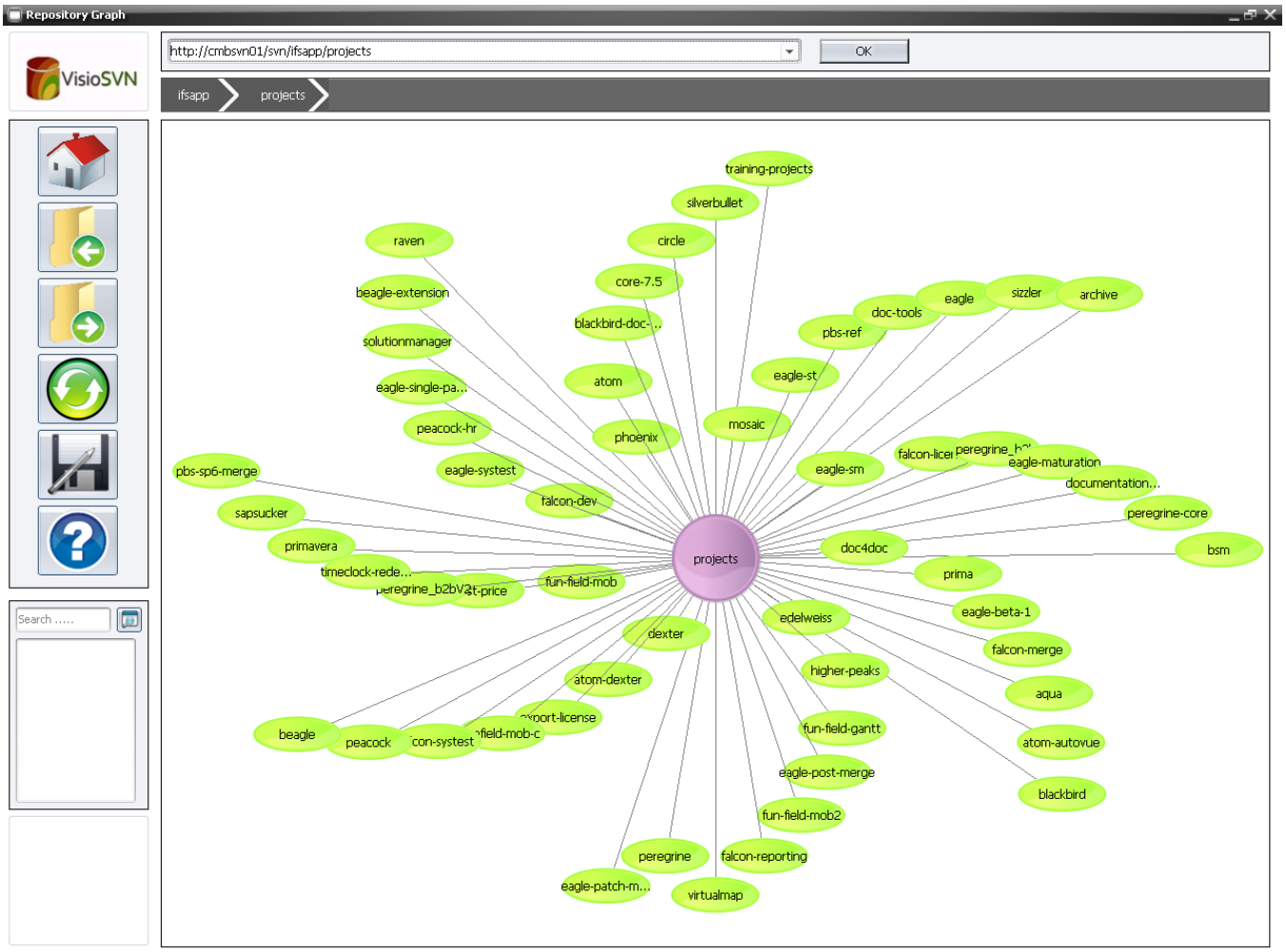
JToolBar in revision graph canvas

- This tool bar control consumes a little area out of the whole area.
- Most of the users are familiar with this kind of tool bars.

- Features in the tool bar control such as quick access is useful when navigating through data.

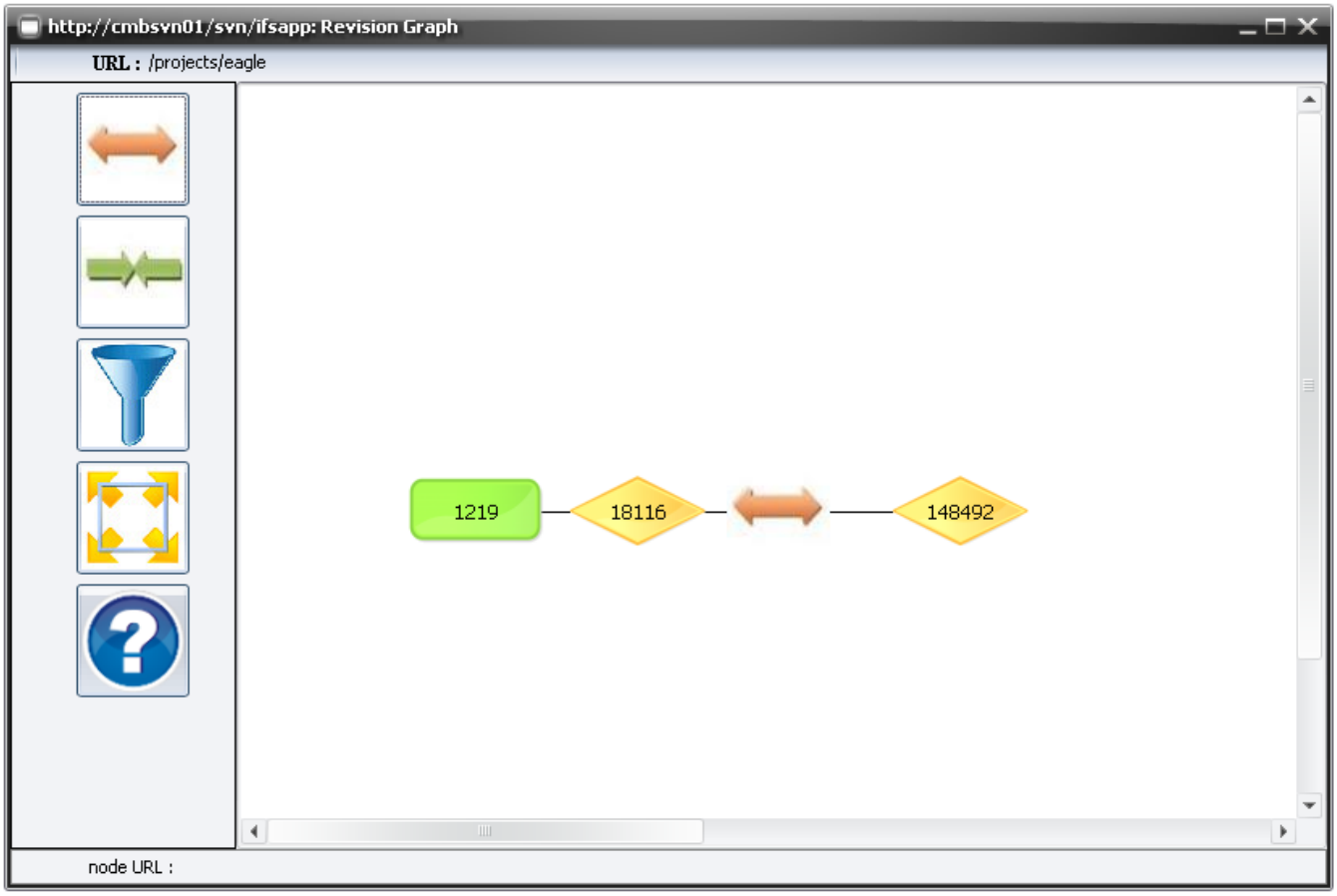
View Styles

Repository Graph



View Styles is a newly added and very useful feature in this application. This is some kind of MDI implementation in a different way. The main advantage of this feature is to view and work simultaneous interfaces in one time. That is, user can divide the working area as requirement and have separate number of working areas. Application has provided one minimum number of working area and four maximum numbers of working areas in different styles.

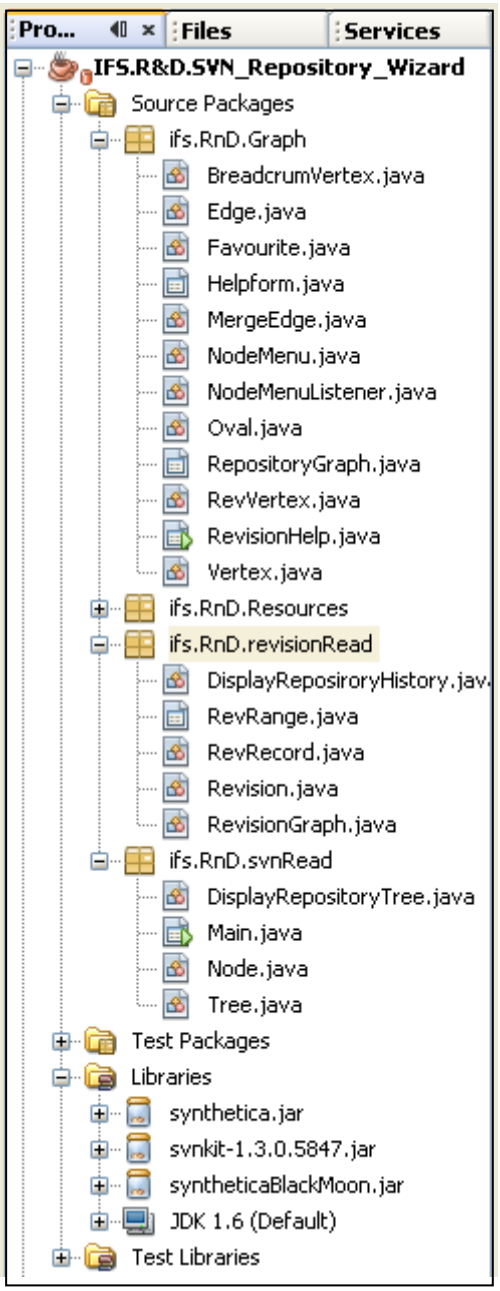
Revision graph



1.4 **Architectural design**

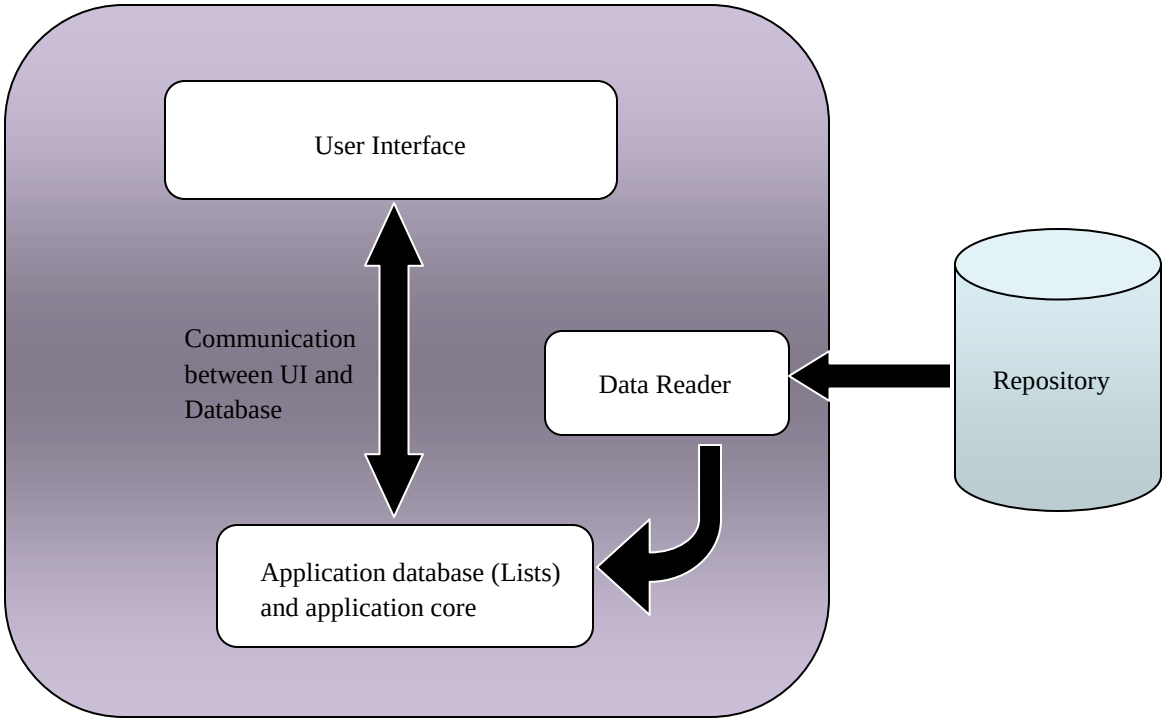
Logical view

This section describes the architecturally significant parts of the design model such as its decomposition in to sub systems and packages.



In this project the source is devided in to three packages. svnRead pakage contains java classes which are used to read a repository. revisionRead package contains revision data handling classes. Resources package contains image icons and Graph package contains GUI classes and other drawing classes. The main class is designed for first user interface and communicating between other classes. Above java libraries are used for implementing repository reading methods and for look and field of the software.

This application is developed for Windows and the diagram below represents the architecture of the Subversion Repository Visualization tool.



- **Data reader** communicates with the repository and reads data.
- **Application database** keeps repository details from Data reader and **application core** contains main algorithems to perform the functions of the application.
- The results in the core are displayed using the **user interface**. Also the information required to perform the functions of the core are also taken from the user interface.