ENHANCING USER VIEW FEATURE EXTENSIONS FOR OPENJUMP

USING GIS MARKERS

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Abhinandan Sheshadri
Summer 2011
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Abhinandan Sheshadri:

Enhancing User View Feature Extensions for OpenJUMP Using GIS Markers

__________________________
William Root, Chair
Department of Computer Science

__________________________
Carl Eckberg
Department of Computer Science

__________________________
David Lesley
Department of Mathematics and Statistics

4/22/11
Approval Date
DEDICATION

I would like to dedicate my thesis to my wife, parents and friends who are always with me in my bad and good times. They are the true source of inspiration for me and because of their blessings; I could complete this thesis today.
ABSTRACT OF THE THESIS

Enhancing User View Feature Extensions for OpenJUMP Using GIS Markers
by
Abhinandan Sheshadri
Master of Science in Computer Science
San Diego State University, 2011

This thesis provides an implementation of OpenJUMP plug-ins for identification and maintenance of new locatable geographic locations. Implementation consists of user view extensions for faculties and GIS Markers using Hotlink Editor and Tool.

The user view allows students / faculties to easily add and view a geographic location and attach images, videos or textual descriptions to identify their originality and presence, also enables users to import images and videos from Google and YouTube, respectively.

The faculty view allows teachers to easily add a geographic location and attach images, videos or textual descriptions to identify their originality and presence, also enables users to import images and videos from Google and YouTube, respectively.

The main purpose of thesis is to provide a common platform to aid easy learning platform for students at the same time provide an easy tool for teachers to add or remove new topics or areas of interest as and when required.

The implementation also consists of two additional tools added to the open source product. A help tool providing more information on each and every tool present in the application, which also has a complete index and searching option.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>xi</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>xii</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>OPEN SOURCE GIS COMPONENT JAVA BASED ARCHITECTURES</td>
<td>4</td>
</tr>
<tr>
<td>2.1 What is OpenJUMP</td>
<td>4</td>
</tr>
<tr>
<td>2.2 What is uDig</td>
<td>4</td>
</tr>
<tr>
<td>2.3 MapObjects – Java Edition Introduction</td>
<td>5</td>
</tr>
<tr>
<td>SOFTWARE REQUIREMENTS SPECIFICATIONS</td>
<td>6</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>6</td>
</tr>
<tr>
<td>3.1.1 Purpose</td>
<td>6</td>
</tr>
<tr>
<td>3.1.1.1 Document Conventions</td>
<td>6</td>
</tr>
<tr>
<td>3.1.1.2 Intended Audience and Reading Suggestions</td>
<td>6</td>
</tr>
<tr>
<td>3.1.2 Project Scope</td>
<td>6</td>
</tr>
<tr>
<td>3.1.3 Description of .CSV and .CHM Files</td>
<td>7</td>
</tr>
<tr>
<td>3.1.3.1 CSV File</td>
<td>7</td>
</tr>
<tr>
<td>3.1.3.2 CHM File</td>
<td>8</td>
</tr>
<tr>
<td>3.1.4 Definitions, Acronyms, and Abbreviations</td>
<td>8</td>
</tr>
<tr>
<td>3.1.5 Overview</td>
<td>8</td>
</tr>
<tr>
<td>3.2 General Description</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1 Product Perspective</td>
<td>9</td>
</tr>
<tr>
<td>3.2.2 Product Functions</td>
<td>9</td>
</tr>
<tr>
<td>3.2.3 User Characteristics</td>
<td>10</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 3.1. Intended Audience and Reading Suggestions...........................................................7
Table 3.2. Acronyms and Expansions.......................................................................................9
Table 3.3. FR1 Create New Theme from CSV File.................................................................10
Table 3.4. FR2 Create New Theme from Geo-data Entered Manually ..................................11
Table 3.5. FR3 Search for Geo-data from Web .......................................................................11
Table 3.6. FR4 Validate Geo-data for its Correctness .............................................................11
Table 3.7. FR5 Create New Schema to Store Geo-information ..............................................11
Table 3.8. FR6 Create a Collection to Associate with New Theme ........................................11
Table 3.9. FR7 Add Geometries/Map Objects to Theme (Feature) Collection .......................12
Table 3.10. FR8 Create New Geometries/Map Objects from Geo-data ..................................12
Table 3.11. FR9 Display Tool’s Usage Information.................................................................12
Table 3.12. FR10 Search for Topic Information .....................................................................12
Table 3.13. Software Interfaces .............................................................................................14
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>OpenJUMP workbench/UI</td>
<td>18</td>
</tr>
<tr>
<td>5.2</td>
<td>Components of JUMP workbench</td>
<td>20</td>
</tr>
<tr>
<td>5.3</td>
<td>JUMP project/task window</td>
<td>21</td>
</tr>
<tr>
<td>5.4</td>
<td>JUMP attribute window</td>
<td>21</td>
</tr>
<tr>
<td>5.5</td>
<td>JUMP schema window</td>
<td>22</td>
</tr>
<tr>
<td>5.6</td>
<td>JUMP feature table view</td>
<td>22</td>
</tr>
<tr>
<td>5.7</td>
<td>JUMP feature HTML view</td>
<td>22</td>
</tr>
<tr>
<td>5.8</td>
<td>JUMP output window</td>
<td>23</td>
</tr>
<tr>
<td>5.9</td>
<td>JUMP editing toolbox</td>
<td>23</td>
</tr>
<tr>
<td>6.1</td>
<td>OpenJUMP architecture</td>
<td>29</td>
</tr>
<tr>
<td>6.2</td>
<td>Extensions</td>
<td>30</td>
</tr>
<tr>
<td>6.3</td>
<td>Workbench data-structures</td>
<td>33</td>
</tr>
<tr>
<td>6.4</td>
<td>OpenJUMP feature model</td>
<td>34</td>
</tr>
<tr>
<td>6.5</td>
<td>OpenJUMP layer-task model</td>
<td>35</td>
</tr>
<tr>
<td>6.6</td>
<td>OpenJUMP task frame model</td>
<td>35</td>
</tr>
<tr>
<td>6.7</td>
<td>OpenJUMP cursor tool hierarchy</td>
<td>37</td>
</tr>
<tr>
<td>7.1</td>
<td>Plugin architecture for GIS marker</td>
<td>43</td>
</tr>
<tr>
<td>7.2</td>
<td>Hot link architecture</td>
<td>44</td>
</tr>
<tr>
<td>7.3</td>
<td>Hot link tool flow diagram</td>
<td>45</td>
</tr>
<tr>
<td>7.4</td>
<td>Hot link editor flow diagram</td>
<td>46</td>
</tr>
<tr>
<td>7.5</td>
<td>Hot link editor and tool icon</td>
<td>46</td>
</tr>
<tr>
<td>7.6</td>
<td>New theme flow diagram</td>
<td>47</td>
</tr>
<tr>
<td>7.7</td>
<td>New theme dialog XY coordinates tab</td>
<td>47</td>
</tr>
<tr>
<td>7.8</td>
<td>New theme dialog from CSV file tab</td>
<td>48</td>
</tr>
<tr>
<td>7.9</td>
<td>New theme dialog get coordinates tab</td>
<td>48</td>
</tr>
<tr>
<td>7.10</td>
<td>Help topic plug-in class diagram</td>
<td>50</td>
</tr>
</tbody>
</table>
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assumptions and Dependencies</td>
<td>AD</td>
</tr>
<tr>
<td>Comma Separated Value</td>
<td>CSV</td>
</tr>
<tr>
<td>Feature ID</td>
<td>FID</td>
</tr>
<tr>
<td>Functional Requirement</td>
<td>FR</td>
</tr>
<tr>
<td>General Constraints</td>
<td>GC</td>
</tr>
<tr>
<td>Geographic Information Science</td>
<td>GIS</td>
</tr>
<tr>
<td>Geography Markup Language</td>
<td>GML</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>GUI</td>
</tr>
<tr>
<td>Integrated Development Environment</td>
<td>IDE</td>
</tr>
<tr>
<td>Java Development Tools</td>
<td>JDT</td>
</tr>
<tr>
<td>JTS Topological Suite</td>
<td>JTS</td>
</tr>
<tr>
<td>JUMP Pilot Project</td>
<td>JPP</td>
</tr>
<tr>
<td>Multiple Document Interface</td>
<td>MDI</td>
</tr>
<tr>
<td>Scalable Vector Graphics</td>
<td>SVG</td>
</tr>
<tr>
<td>Single Document Interface</td>
<td>SDI</td>
</tr>
<tr>
<td>User Characteristics</td>
<td>UC</td>
</tr>
</tbody>
</table>
I would like to thank Almighty God and my family members for giving me the strength in all my good deeds. I would like to thank Dr. Carl Eckberg who guided me throughout the whole thesis. With his knowledge and guidance I think I have improved my knowledge of GIS based application and component based GIS architectures. I would also like to thank Prof. William Root who allowed and encourage me to go into this direction and inspiring me by telling that nobody has tried this before so let us do it.
CHAPTER 1

INTRODUCTION

Geographic Information Science (GIS) involves in displaying and analyzing of location based data where maps are the most common example and census data are the most common example of a large database with spatial components [1]. It provides set of tools that captures, stores, analyzes, manages, and presents data that are linked to location(s). In the simplest terms, GIS is the merging of cartography, statistical analysis, and database technology. In a general sense, any information system which integrates, stores, manipulates, analyzes, shares, and displays geographic related information for informing decision making. GIS applications are tools that allow users to create interactive queries, analyze spatial information, edit data, maps, and present the results of all these operations [2], [3].

OpenJUMP, ESRI, uDig, etc. sells products to the GIS community. These are very large applications with thousands of features. The builder of a GIS application, using these tools, must select the features he needs, and in some cases explicitly discard others. In most cases, programming skills are not needed.

Component architectures like OpenJUMP, MOJO, uDig, etc. takes a different approach. The idea of a component is that it is an easily modifiable piece of software that a developer can incorporate into an application. It provides a mechanism or exposes various ways to the developer through which they can merge their innovative and effective solution so called new features to the existing set of features or update the existing feature as per their requirement. Thus it allows developer to make the application more robust, interactive and innovative. The user can, by easy to use interfaces, modify the state of the object, e.g., its color, shape or label. In addition, the user can modify the behavior of the component. There may be dozens of functions the component can execute, and the developer can write code to control what those functions do. Initial code is often provided. For example a component that is visible when the application is running can be programmed to behave in a certain way when the user clicks a mouse on that component. Component architecture has some real advantages.
1. It provides more precise and specific control over behavior of the application.
2. The user makes the application from the bottom up, adding only those features that are necessary and sufficient for their use and discard the remaining.
3. Components from various different companies are easily combined.
4. A fully fledged programming language as well as development platform is available to the developer.

The drawback is that such powerful features in a big package are not available without huge work and programming skills are really important and necessary.

Following the second approach, it makes the platform open to the developer community which is termed as Open Source. Several benefits have been achieved due to such an open source characteristics of these platforms. They are as follows:

1. Source code is made available for use or modification as users or other developers see fit.
2. Software being distributed must be redistributed to anyone else without any restriction.
3. Source code must be made available (so that the receiving party will be able to improve or modify it).
4. License can require improved versions of the software to carry a different name or version from the original software.
5. The idea is very similar to that behind free software and the Free Software Foundation.

Several different open source architectures are available to developer community that can be used to develop a robust, interactive, effective and innovative GIS solution. Among those I have used OpenJUMP to understand the architecture and platform to develop a different way to create your own new plug-ins or extensions or add-ins. This plug-ins will help the user to add new geographic location from the available geo-data (longitudes, latitudes and its respective geometry) and attach information like images, videos or textual descriptions to identify their originality and presence, also enables users to import images and videos web. The main purpose of thesis is to provide a common platform to aid easy learning for students at the same time provide an easy tool for teachers to add or remove new topics or areas of interest as and when required. The implementation also consists of two additional tools added to the open source product. A help tool providing more information on each and every tool present in the application, which also has a complete index and searching option. To display this information associated with the newly created locations or existing
ones, we have used GIS markers. GIS marker is also known as “Hotlink” or “Pin-Point” or “Point of Interest.” It requires a hotlink tool, i.e., tool with the lightning bolt as a cursor icon, to pop up the information associated with this geo-locations or geometries which can also be one of the useful learning tools.
CHAPTER 2
OPEN SOURCE GIS COMPONENT JAVA BASED ARCHITECTURES

This chapter discusses about commonly used open source component Java based GIS architectures; used to developed robust GIS based applications. They are as follows.

2.1 WHAT IS OPENJUMP

OpenJUMP previously known as JUMP, is a GIS, graphical user interface (GUI)-based application for viewing and processing spatial data. It has been developed originally by the two Canadian companies Vivid Solutions and Refractions Research under the name JUMP. JUMP is an abbreviation for Unified Mapping Platform. It includes many common spatial and GIS functions for mapping and data representation. The “J” points to the used programming language “Java.” “Open” is for “Open Source,” which means that the quell code is accessible for everybody. It underlies the GNU General Public License and is maintained and improved by programmers around the globe. It provides an interactive workbench; API’s giving full programmatic access, highly modular and extensible, supports Geography Markup Language (GML) and the OpenGIS Consortium spatial object model. The source code of OpenJUMP is made available for use or modification as users or other developers see fit. It has no software redistribution restrictions; license can require improved versions of the software to carry a different name or version from the original software. Please refer Chapters 5 and 6 to get detail description about OpenJUMP as an application and GIS based application development platform.

2.2 WHAT IS uDIG

uDig is a open source desktop GIS software development program developed by a community led by Canadian-based consulting company Refractions Research. It is based on the Eclipse platform and features full layered Open Source GIS. It is written in Java and released under GNU Lesser General Public License. uDig has a walkthrough in Flash and also quick start directions for those who wish to complete a full version build to write
plug-ins or contribute to the main build. uDig can use GRASS for complex vector operations and also embeds JGRASS and specialized hydrology tools from the Horton Machine. It provides read/write supports for ESRI shape-files, PostGIS, WMS, WFS, DB2, Oracle Spatial, ArcSDE and many other data sources natively [4]. Also to handle even largest data sets to provides high performance rendering pipeline. It also provides complete integrated access to free public online data like NASA Landsat archives and daily satellite monitoring and coordinate reference system support for all data sources, also on-the-fly coordinate system transformation [5], [6].

2.3 MapObjects – Java Edition Introduction

MapObjects - Java Edition is the version of ESRI’s pure Java components that can be used for developing and creating GIS based applications. Two core groups of components are: client-side components that can be used for developing GIS and mapping user interfaces and applications and server-side constructs to enable integration with Internet and enterprise applications. Developers are benefited from having access to server-side components, allowing them to build map services, Java Server Pages, and Java Servlets. Developers can also construct custom Enterprise JavaBeans and take advantage of industry-standard J2EE application server providers. It allows user to display maps with dynamic, real-time geographic data, to navigate and zoom through map layers, perform queries on spatial information and geometric operations, labeling and customizing map interfaces. Additional features included in 2nd edition are:

1. Support for additional file formats such as MrSID, CAD (DGN, DXG, and DWG), SDC, military formats, and much more.
2. Projections and datum transformations.
3. Layout view for comprehensive map composition.
4. Support for Mac OS X and IBM AIX operating systems in addition to continued support for HP-UX, Linux, Solaris, and Windows.
5. Project save and load.
6. Additional samples with source.
CHAPTER 3
SOFTWARE REQUIREMENTS SPECIFICATIONS

3.1 INTRODUCTION

This section talks about the purpose, documents conventions, intended auditions and reading suggestions, project scope, any description related supporting files and data, some definitions, abbreviations and acronyms and finally about brief overview about the system.

3.1.1 Purpose

This document describes the specific requirements for Enhancing admin view feature extensions for OpenJUMP using GIS markers. The general requirements give an overview of the users’ characteristics, product perspective, and overview of functional and data requirements. The specific requirements give a more refined version of the general requirements.

3.1.1.1 DOCUMENT CONVENTIONS

All the requirements in this software requirement specification are mentioned in descending order of their priorities. Every requirement is provided with the unique requirement ID, so that developers can identify the requirements very easily.

3.1.1.2 INTENDED AUDIENCE AND READING SUGGESTIONS

The reading suggestions and intended audience is mentioned below in Table 3.1.

3.1.2 Project Scope

The name of this thesis is enhancing admin view feature extensions for OpenJUMP using GIS markers. The main purpose of thesis is to provide a common platform to aid easy learning platform for students at the same time provide an easy tool for users to add or remove new topics or areas of interest as and when required. This thesis provides
Table 3.1. Intended Audience and Reading Suggestions

<table>
<thead>
<tr>
<th>Intended Audience</th>
<th>Reading Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Manager</td>
<td>PM has to read the whole document including purpose and scope of the project and all the requirements.</td>
</tr>
<tr>
<td>Developers</td>
<td>Developers have to focus mainly on functional &amp; non–functional requirement and have to keep watch on every other requirement and also maintain the scope of the product.</td>
</tr>
<tr>
<td>Testers</td>
<td>Testers have to focus on functional &amp; non functional requirements</td>
</tr>
<tr>
<td>Users</td>
<td>Users have to focus on the General requirements</td>
</tr>
<tr>
<td>Document Writers</td>
<td>Document Writers have to know the whole document and also need to keep record on changes in the system, if any.</td>
</tr>
</tbody>
</table>

implementation of OpenJUMP plug-ins for identification and maintenance of new locatable geographic locations. The faculty view allows user to easily add a geographic location and attach images, videos or textual descriptions to identify their originality and presence, also enables users to import images and videos from Google and YouTube, respectively. The implementation also consists of two additional tools added to the open source product. A help tool providing more information on each and every tool present in the application, which also has a complete index and searching option.

3.1.3 Description of .CSV and .CHM Files

This section provides detailed description related .csv and .chm files.

3.1.3.1. CSV FILE

Comma Separated Value (CSV) file abbreviates comma separated values files, i.e., a text file which contains information separated by ‘,’. These files either accessed locally or are store inside project directory under “/lib/ext/resource/CSV Files.” These files contain geographical information that will be used to create new geometry or maps within the given theme. Each line in this file contains information related to single geometry. Each line is divided into two parts as: geotypes and geoCoordinates

Where,

1. geoType: should be one of these following (point, multipoint, linestring, multilinestring, line-ring, polygon, multipolygon)
2. geoCoordinates: For each above mentioned geometry please follow below specified format to enter its coordinates:
   a. point/multipoint: long1,lat1[,long2,lat2,....,longN,latN]
   b. linestring/multi-line-string:
      long1,lat1,....,longN,latN[|long1,lat1,....,longN,latN|....|....]
   c. linering: long1,lat1,....,longN,latN
   d. polygon/multipolygon:
      (long1,lat1,....,longN,latN)[(long1,lat1,....,longN,latN)...]
      (long1,lat1,....,longN,latN)[(long1,lat1,....,longN,latN)...]...]
   e. where in each case both the 1st and nth pair of (long, lat) are same to form close geometry

   In all of the above specified formats long stands for longitudes while lat stands for latitudes.

3.1.3.2 .CHM FILE

.CHM file is compiled file generated by Microsoft HTML Help Tool. It is used to display the help related to usage a specific tool in the application. This file is displayed whenever user clicks any tool after clicking “Help Topic Tool.” The html files that are part of this .chm file can be obtained at location “/lib/ext/resource/HelpDocs/Help_Html_Files” inside project directory. The name of the .chm file used to display help is SampleHelp.chm. Install htmlhelp.exe, modify and recompile SampleHelp.chm file to reflect the changes that are made to existing html help files or newly added help information to the .chm file. Since the tool used to show help information is Microsoft Html Help Tool, these files are not recognized on other platforms.

Note: All these help files as well as htmlhelp.exe executable files are there in the respective directories if you check-out the version of OpenJUMP, that I have used for my thesis work, from Google SVN project hosting site.

3.1.4 Definitions, Acronyms, and Abbreviations

Please refer to Table 3.2 for definitions, acronyms and abbreviations.

3.1.5 Overview

The rest of SRS contains two parts General Description and Specific Requirements. General Description consist information about an outline of the Product, hardware and
software requirements. It also provides details about the users involved with the system. The specific requirement presents a more detailed version of the available functional and other requirements of the Project.

### 3.2 General Description

The general description in SRS talks about the product perspective, product functions, user characteristics and constraints and assumptions and dependencies taken into consideration while developing the software (plug-ins).

#### 3.2.1 Product Perspective

This thesis is a self-contained desktop GIS based application. The product is designed to run in its supported environment mentioned in the software requirements. Its GUI as well as functionality is implemented using Java.

#### 3.2.2 Product Functions

Some of the major functions that this product will perform are:

- Allow user to create new geometry based on geo-coordinates by manually entering it.
- Allow user to create new geometry or maps based on geo-coordinates by specifying location of “.csv” file.
- Allow user to find geo-coordinates of any location based on its address, zip code, city, etc.
- Provide support to create all type geometrical map objects like point, multipoint, line-strings, multi-line-strings, line-ring, polygon and multi-polygon.
- Provide user with help tool to understand usage of any tool in the toolbar.
- Search help for a specific topic within the system.
3.2.3 User Characteristics

The user (student/faculty) should follow the geo-formats mentioned in the dialog while entering geo-data manually. User is considering being aware about the basic information like name, address, zip code, city, etc. related to the new theme in case if he/she is unaware of the geo-data of the geometries that are part of these new theme. User should also know the location of .csv file or else can create or store in the default location (refer to Section 1.3). User should create .csv file following description mentioned in Section 1.3.

3.2.4 General Constraints

The work station should be properly configured with Eclipse Java integrated development environment (IDE) with appropriate sub-versioning plug-in subclipse.

3.2.5 Assumptions and Dependencies

Some of the assumptions for this project include:

1. We assume that User of the system will follow the correct geo-format according to the possible values specified in the Section 1.3.
2. The .chm file should contain help information for all the tools that are available in the application.

3.3 Specific Requirements

The specific requirements go into details about the particular requirements targeting areas like functional, design constraints, attributes and external interfaces required like hardware and software support, etc.

3.3.1 Functional Requirements

Please refer to Tables 3.3 to 3.12 for functional requirements (FR).

<table>
<thead>
<tr>
<th>Table 3.3. FR1 Create New Theme from CSV File</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td><strong>Processing</strong></td>
</tr>
<tr>
<td><strong>Output</strong></td>
</tr>
</tbody>
</table>
### Table 3.4. FR2 Create New Theme from Geo-data Entered Manually

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing user to create new theme from geo-data entered manually</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Geo-data should match its respective geo-format</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Validation of the geo-data to check for correctness. And invoke appropriate new theme API to carry on further process</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Create new theme resulting in listing the theme name in layer-name panel and displaying geometry in the layer-view panel.</td>
</tr>
</tbody>
</table>

### Table 3.5. FR3 Search for Geo-data from Web

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing user to get geo-coordinates from web</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Entering location’s name/address/zip code/city, etc.</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Search for geo-coordinates of desired location.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Geo-coordinates of desired location copied into clip-board of the system.</td>
</tr>
</tbody>
</table>

### Table 3.6. FR4 Validate Geo-data for its Correctness

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>Allows system to check the correctness of the geo-data.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Geo-data of the geometry or map object.</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Validating entered geo-data whether it matches its specified format.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>True/False based on the match.</td>
</tr>
</tbody>
</table>

### Table 3.7. FR5 Create New Schema to Store Geo-information

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing system to create new schema to store geo-information of geometries or map object belonging to the current new theme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Attributes names and their types</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Create new schema</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>New schema for theme is created</td>
</tr>
</tbody>
</table>

### Table 3.8. FR6 Create a Collection to Associate with New Theme

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing system to create new collection to hold geometries or map objects belonging to new theme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Validating the edited Case Sheet detail and Updating record in database</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>New theme (feature) collection is instantiated.</td>
</tr>
</tbody>
</table>
### Table 3.9. FR7 Add Geometries/Map Objects to Theme (Feature) Collection

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing system to add geometries/map object to theme (feature) collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Feature collection and feature schema</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Add geometries/map object to collection. Associate feature schema to each geometry/map object with basic information.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Feature collection with all geometries belonging to current new theme</td>
</tr>
</tbody>
</table>

### Table 3.10. FR8 Create New Geometries/Map Objects from Geo-data

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing system to create new geometries or map objects from geo-data.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Validated geo-data.</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Create new geometries/map objects.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Creates new geometry or map object based on geo-data</td>
</tr>
</tbody>
</table>

### Table 3.11. FR9 Display Tool’s Usage Information

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing user to display tool’s usage information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Name of the tool used to index in the help .chm file.</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Check whether help topic tool is activated. If yes then display the tool usage information by using system API.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>.chm file is popped up and shows tool’s usage information.</td>
</tr>
</tbody>
</table>

### Table 3.12. FR10 Search for Topic Information

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>For allowing user to search for any help related to any topic in the application.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Invoke system API to display help information</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>.chm file is popped up and shows help information.</td>
</tr>
</tbody>
</table>

### 3.3.2 Design Constraints

Design constraints talk about the hardware and software limitations related to the software as well as the platform required to run this software.

#### 3.3.2.1 HARDWARE LIMITATIONS

Following are the hardware limitations:

- The minimum RAM required for this project is 1 GB.
• Should have a Hard-Disk Free space of at least 600 MB.
• Requires an Internet Connection Speed of at least 56 KB/s.
• Processor Clock speed for the server system should have a minimum of 1 Ghz.

3.3.2.2 SOFTWARE LIMITATIONS

Following are the software limitations:
• Any operating system that supports TCP/IP protocol.
• All platforms that provide support for Eclipse 3.5.0 or newer, Subclipse 1.6.x, Java JDK 1.5 or newer.
• For Microsoft Html Help Tool is compatible with Microsoft Windows XP or newer.
• Browser’s: Internet Explorer 5.5 or newer, Netscape 6.2.x or newer and Mozilla 1.4+.

3.3.3 Attributes

Following are the attributes related to the developed plug-in.

3.3.3.1 AVAILABILITY

The following factors guarantee defined availability level for the entire system:
• Internet connection should be active.
• Eclipse should be configured with proper subclipse version to check out and check in code changes.

3.3.3.2 MAINTAINABILITY

For maintaining the project, the operations involved in the system are grouped under specific classes, so problems in the system can be checked by the module that it has occurred and can be fixed quickly.

3.3.3.3 TRANSFERABILITY

Since this a desktop-based project, porting the project to another OS is an issue. Hence files that are accessed locally need to be present on other system to make the project create in OpenJUMP workbench to work smoothly. Also refer to Section 3.2 to get information about system’s design constrains.
3.3.4 External Interface Requirements

External Interface requirements exposes the required external hardware, software and communication interfaces to run this software plug-in as well as the platform supporting this software.

3.3.4.1 Hardware Interfaces

The project requires a Computer with a mouse and keyboard for accessing the project source code. A router or modem for connecting to Internet.

3.3.4.2 Software Interfaces

Table 3.13 lists the software interfaces required to run the software as well as platform efficiently.

Table 3.13. Software Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Source</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>3.5.0</td>
<td><a href="http://www.eclipse.org/">http://www.eclipse.org/</a></td>
<td>Used as IDE to develop GIS based application for OpenJUMP platform.</td>
</tr>
<tr>
<td>Subclipse</td>
<td>1.6.x</td>
<td><a href="http://subclipse.tigris.org/">http://subclipse.tigris.org/</a></td>
<td>To check out OpenJUMP SDK from Google SVN.</td>
</tr>
<tr>
<td>Java</td>
<td>1.5 and newer</td>
<td><a href="http://java.sun.com">http://java.sun.com</a></td>
<td>As programming development environment.</td>
</tr>
</tbody>
</table>

3.3.4.3 Communication Interfaces

TCP/IP protocol is required for accessing the OpenJUMP SDK from Google SVN.
CHAPTER 4

DEVELOPMENT ENVIRONMENTS AND
TECHNOLOGIES

This chapter briefs you about the developments environments, tools, plug-ins and technologies used to accomplish the thesis work. They are as follows.

4.1 ECLIPSE JDT

Eclipse Java Development Tools (JDT) is an open extensible software development framework. Many developers have start using and developing it, and researchers try to build their own research ideas on top of the Eclipse platform because of its openness and extensibility with plug-in architecture. It is a multi-language software development environment comprising an IDE and an extensible plug-in system. It is written primarily in Java programming language and can be used to develop applications in Java and, by means of various plug-ins, other languages including C, C++, COBOL, Python, Perl, PHP, Scala, Scheme and Ruby (including Ruby on Rails framework). The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java and Eclipse PDT for PHP. The initial codebase originated from Visual-Age. In its default form it is meant for Java developers, consisting of the JDT.

Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. It is free and open source software, released under the terms of the Eclipse Public License.

4.2 SUBCLIPSE

Subclipse is a plug-in provided by Eclipse Team Provider as a support for Subversion within the Eclipse IDE. The software is released under the Eclipse Public License 1.0 Open Source license. It is subversion plug-in for Eclipse and aims to provide all subversion functionality to the eclipse development environment. It’s an open source project as well, and available at the project.
Its versions are tied to specific versions of the subversion client API. So you must have a matching version of the subversion client API (JavaHL) for your version of subclipse. Any 1.x version of a Subversion client can talk to any 1.x version of a Subversion server, so generally the version does not matter too much. However, if you use multiple client tools on the same subversion working copy, then it is important that the version of those clients is all the same. In addition, if you are on Linux, your distribution might only support a specific version of Subversion and JavaHL. So you might want to stick with a specific version of subclipse for that client.

### 4.3 Microsoft HTML Help Tool

Microsoft HTML Help is the standard help system for the Windows platform. Authors can use HTML Help to create online help for a software application or to create content for a multimedia title or Web site. Developers can use the HTML Help API to program a host application or hook up context-sensitive help to an application. As an information delivery system, HTML Help is suited for a wide range of applications, including training guides, interactive books, and electronic newsletters, as well as help for software applications.

HTML Help offers some distinct advantages over standard HTML, such as the ability to implement a combined table of contents and index and the use of keywords for advanced hyper linking capability. The HTML Help compiler (part of the HTML Help Workshop) makes it possible to compress HTML, graphic, and other files into a relatively small compiled help (.chm) file, which can then be distributed with a software application, or downloaded from the Web.

HTML Help consists of an online Help Viewer, related help components, and help authoring tools from Microsoft Corporation. The Help Viewer uses the underlying components of Microsoft Internet Explorer to display help content. It supports HTML, ActiveX, Java, scripting languages (JScript, and Microsoft Visual Basic Scripting Edition), and HTML image formats (.jpeg, .gif, and .png files). The help authoring tool, HTML Help Workshop, provides an easy-to-use system for creating and managing help projects and their related files.
4.4 Google SVN

Google are offering free Subversion hosting for open source projects. Log in using your Gmail account and you can create a project. There is a choice of open source licenses. I have chosen Google’s Open Source hosting because they have a very clean UI with integrated issue tracking and also offer native support for Subversion. One of the many aspects that make Subversion a decent choice is the ability to import existing SVN repositories to Google Code. That is, the whole history is uploaded and your soon-to-be open source project doesn’t lose valuable undo history. We currently use Subversion 1.6; made available via WebDAV Project Hosting on Google Code provides a free collaborative development environment for open source projects. Each project comes with its own member controls, Subversion/Mercurial repository, issue tracker, wiki pages, and downloads section. Our project hosting service is simple, fast, reliable, and scalable, so that you can focus on your own open source development.

To get started with how to use the Google SVN as project hosting and other related question please refer to URL listed in references section [7], [8]. It contains all details related to how check-in code for the first time, how to check-out, how to manage multiple versions, what plug-ins required to be used with your IDE to make effective use of SVN and many more. For our thesis we have used subclipse, which is an Eclipse JDT plug-in, to carry out task related to Google SVN. You can also refer to Chapter 5 for getting details about how to check-out OpenJUMP project for the Google SVN repository that we have created for this thesis and other information related to it.

4.5 OpenJUMP SDK

As discussed before, OpenJUMP SDK stands for open source Java unified mapping platform software development kit for developing GUI based GIS that can be used to view and process geo-spatial information. Due to its openness and extensibility feature, it exposes to the developer community the core framework as well as set of plug-ins contributed by the developer community to make OpenJUMP one of the effective GUI based GIS application. Please refer to Chapters 5 and 6 to get details related to OpenJUMP SDK, how to download, how to upgrade, how make effective use of OpenJUMP core framework and details about it, etc.
CHAPTER 5

OPENJUMP/JUMP

5.1 INTRODUCTION

OpenJUMP, previously known as JUMP, is a GIS, GUI-based application for viewing and processing spatial data. It includes many common spatial and GIS functions for mapping and data representation. It has been developed originally by the two Canadian companies Vivid Solutions and Refractions Research under the name JUMP. JUMP is an abbreviation for Unified Mapping Platform. The “J” points to the used programming language “Java.” “Open” is for “Open Source,” which means that the quell code is accessible for everybody. It underlies the GNU General Public License and is maintained and improved by programmers around the globe, see Figure 5.1.

Figure 5.1. OpenJUMP workbench/UI.
It provides an interactive workbench; API’s giving full programmatic access, highly modular and extensible, supports GML and the OpenGIS Consortium spatial object model. The source code of OpenJUMP is made available for use or modification as users or other developers see fit. It has no software redistribution restrictions; license can require improved versions of the software to carry a different name or version from the original software.

5.2 CAPABILITIES/FUNCTIONALITIES

OpenJUMP is a vector GIS that can also read raster image files. It is not just simple demo GIS viewer but also provide functionalities to edit, save, analyze, conflate and display GIS data. It is developed in Java programming language hence provides support for multiple platforms including Windows, MacOSX, Linux, etc. It has an excellent open source and extensible framework for rapid prototyping. Its advance editing, drawing and easy to use capabilities helps user to moved, scaled and duplicates the features. Due to extensibility and openness of JUMP framework it allows developer to provide new capabilities and plug-ins that made OpenJUMP to load vector, raster and database files. The capabilities, functionalities and support provided by OpenJUMP are as follows:

1. **Data Formats**: Provides read support for GML, SHP, DXF*, JML, MIF* & TIFF, JPG, MrSID, ECW as well as from PostGIS, ArcSDE*, Oracle* and MySQL*; write support for GML, SHP, DXF*, PostGIS*, JML & JPG. Also supports OGC WMS & WFS (-T), GML 2, SLD. Also provide support to export images in Scalable Vector Graphics format. Please refer to [9] in Bibliography for details information about listed different data formats.

2. **Editing and Conflating**: It provides support for drawing various geometric objects like lines, circles, polygons, points; add, move and delete the vertices; can rotate and scale auto-complete polygons; cut and merge polygons and line. It also provides tools for wrapping and quality assurance. It also provides advance set of selection tools, objects or image scaling tools, etc.

3. **Analysis (Vector) and Query**: It provide tools to carry out spatial and attribute query functions; calculating length, area, centroid, buffer, convex hull, etc.; topology tools for line noder, polygonizer, planar graph, etc.; attribute transfer tools (1:1 transfer, or n:1 transfer); raster analysis via sextante connector.

4. **Customized**: It provides an open source and extensible development framework that consist of rich collection of core API’s as well as extended plug-in contributed by developer communities. It provided support for scripting via bean shell and python. It also supports I18N (language internationalization) for various different locales.
5.3 JUMP GUI

JUMP GUI also known as workbench (window) is the one that gets displayed once you launch OpenJUMP. It displays the main workbench component called the Project or Task Window. Other than project or task window, JUMP also has many other displayed components called windows and views, which are activated during the editing/drawing and querying processes. Below is the detail description of different windows or views that are part of JUMP application, see Figure 5.2.

Figure 5.2. Components of JUMP workbench.

Figure 5.3 depicts the picture of Project/Task window. It is the main window in JUMP workbench. It is divided into two parts:

1. **Layer List**: Display list of all opened layers in the application.
2. **Layer View**: Graphically displays all the opened layers listed in the layer list. It also provide various tools and functionalities that can be perform in the portion of the window like adding new layer, editing existing opened once, zoom in/out, analyze shapes or geo-data related to layers, etc.
JUMP workbench also has different types of views or windows including main project or task window. They are as follows:

1. **Attribute View**: Displays the attributes of the features. Attributes can be added using Schema View window. It has a toolbar on the left side with tools used to select and edit items on the Layer View along with zooming tools. See Figure 5.4.

   ![Figure 5.4. JUMP attribute window.](image)

2. **Schema View**: It allows you to add, delete attributes to a layer or to change attribute data types. See Figure 5.5.

3. **Feature View**: It shows attributes and values for selected items. Two viewing options are available:
   a. **Table View**: Displays feature information in tabular form. See Figure 5.6.
   b. **HTML View**: Displays attributes and the coordinates of features both in GML and WKT syntax. See Figure 5.7.
4. **Output Window**: Shows processes or error messages. Record control on the bottom allows seeing previous messages. See Figure 5.8.

5. **Editing Toolbox**: It is a collection of tools for editing, generating and modifying geometries on the Layer View. With the help of these tools, user can make complex
drawings from basic elements, such as lines, circles or polygons. Features can be duplicated, modified or rotated. It can be opened by selecting the editing toolbox tool on the Toolbar or from the View menu-bar. The working layer must be editable in order to use these tools. See Figure 5.9.

This toolbox contains below listed tools:

- Selection and move tools
- Drawing objects tools
- Vertices and resize tools
- Snap and split tools
- Draw constrained object tools
- Other miscellaneous tools

Please refer to [10] in Bibliography to get detail information about the tools in editing toolbox. Apart from windows and view, JUMP workbench also has two important components named Menu-Bar and Toolbar which provides access to different functions or tools that are available in these components of JUMP workbench. For detail information about these components, please refer to [11] in Bibliography.
5.4 Releases

When I started to explore more about OpenJUMP and related other topic, the latest version available at that time was 1.3.1, released on April 2009. Currently the latest version available is 1.4.0.2, released by November 8, 2010, which is completely backward compatible with previous version and is easy to upgrade from 1.3.1 to 1.4.0.2 versions. For more details related to how to upgrade from 1.3.1 to 1.4.

5.5 File Extensions

JUMP framework supports and provides mechanism for various data formats mentions in Section 5.2. Along with that it has its own extension for saving projects and other GIS information. Project files are saved as .jmp extension while the layers or feature-sets are saved as .jml extension. It also provides ways to come up with your own data-sources, that read, write and save in a particular format. During this thesis I was not able to explore much on these areas but in next chapter a brief description is mentioned about how to make effective use of JUMP framework to write a user defined data source.
CHAPTER 6

JUMP FRAMEWORK AND ARCHITECTURE

This chapter discusses about the OpenJUMP framework and architecture and its different components, which will help you to get insight about how to use OpenJUMP framework and develop new plug-ins or add-ons. In beginning, it has explained reasons why to use OpenJUMP as framework or as tool-kit, followed by OpenJUMP Package structure i.e., How entire code is been divided into different packages and class hierarchies and then covers basic programming guidelines related to OpenJUMP. In later part it has explained immensely and precisely about various components of OpenJUMP Architecture and it ends with the existing JUMP Pilot Project (JPP) Development Group.

6.1 JUMP AS FRAMEWORK OR TOOL-KIT

OpenJUMP provides an excellent GUI based framework as well as tool kit platform to the developer and user communities. This section of the chapter will elaborate the facts depicting why to use JUMP as framework or tool-kit.

1. JUMP as Framework: Below mentioned are nine reasons why to use JUMP as framework.
   a. Splash Screen: JUMP framework display’s a splash screen that disappears as soon as your main window appears.
   b. Multiple Document Interface (MDI): JUMP framework provides both MDI as well as regular single document interface (SDI) application configuration. By default its MDI, which like a key benefit, but if you don’t require it you can still use rest of the framework with SDI.
   c. Reusable Actions: Functionalities or actions defined once can used/invoked from multiple places i.e., either as menu items in menu bar or sub menu items, popup menus, toolbars, main menus, etc. All you need is to define this operation as plug-ins or tools interfaces. You will come to know more about plug-ins and tools in next chapter.
   d. Advanced Menuing: The JUMP Workbench makes it easy for you to add logic about when menus should be enabled and disabled. Many frameworks do this, but this framework also allows you to specify the reason for disabling the menu – the user simply holds the mouse cursor over the disabled menu item and gets a tooltip reason, like “Fence needs to be drawn” or “At least 1
layer must be selected.” Simply write an anonymous class on the fly that implements the EnableCheck interface. Example: ChangeStylesPlugIn#createEnableCheck.

e. Cursor Tools: JUMP workbench makes it really easy in case if you want your application or functionality to handle mouse click or need to handle mouse events (e.g., for drawing), the JUMP Workbench makes it easy. Simply subclass DragCursorTool (if you want the user to drag out a box) or MultiClickTool (if you want the user to click several places, for example, as when drawing a polygon), or one of the other useful classes in the AbstractCursorTool hierarchy. Example: DrawRectangleFenceTool.java.

f. Unobtrusive Warnings: JUMP framework notify user regarding the interrupting warnings without interrupting user work flow in a very innovative and stylish way. With the JUMP framework, you can show warnings in the status bar. The warning is yellow and flashes for a second, to get the user’s attention, but doesn’t interrupt the user’s train of thought because there is no OK button to push. Simply call WorkbenchFrame#warnUser.

g. Error Dialog: Of course, a full-blown error dialog is available if you really need to grab the user’s attention. In fact, unhandled exceptions automatically percolate up to the error dialog. The error dialog has a Show/Hide Details button that reveals the Java exception stack trace, which is useful for debugging. (To do: build a framework for reporting errors to the user in a friendly manner. For example, exception must have a friendly message, as well as cause, and recommended actions.)

h. Window List: Another little bonus you get is a Window menu that lists the open. Again, not hard to write, but not fun to write either!

i. Options Dialog: The Edit/Options menu will open a tabbed dialog that you can store your program options under, so that the user can set his or her preferences, e.g., InstallGridPlugIn#initialize.

2. JUMP as Tool-Kit: Below are five reasons why one can still consider JUMP to be useful source for developing powerful components.

a. Wizard: There isn’t much on the Internet in the way of a free Wizard framework. JUMP has one, and it looks pretty good!

b. AddRemovePanel: This is a frequently used GUI idiom, and there isn’t one in Swing. Try JUMP’s - it’s easy to use, and you can plug-in whatever you want into the panels - even trees!

c. LayerViewPanel. This is JUMP’s core component — the panel that displays all the maps. You get lots of functionality for free: zoom logic, fast draws, and dozens of pre-built cursor tools that you can plug-into your own app (drawing, moving, panning, selecting, etc.). Add it to your web app!

d. Java2XML: Lots of people are interested in this tool. Its a little class that turns the objects you feed it into XML documents, and vice versa. If you do
not want to use Castor because you don’t want to bother with XML Schema, and Bewitched is giving you inexplicable errors, give Java2XML a try!

e. **MultiInputDialog:** Sometimes it’s fun to build a dialog; sometimes it’s a pain. MultiInputDialog will let you whip up a new dialog in minutes rather than hours. Simply give it the types of the fields (e.g., Integer, Boolean, String), the label strings, perhaps an optional image, and presto! Instant dialogs that looks pretty good! Examples: the **Boundary Match Data** dialog under the **Tools > Generate** menu, or the (nice looking!) **Validate Selected Layers** dialog under the **QA** menu.

### 6.2 JUMP LIBRARIES AND SUPPORTING LIBRARIES

This section of the chapter discusses about the OpenJUMP package structure and other supporting libraries that are used to built such a robust GIS application. Once you download and configure Eclipse JDT with OpenJUMP SDK, you will see that entire package structure of JUMP, which is been divided into two main package structures, one providing core JUMP functionalities belonging to JUMP application or workbench and another consisting of plug-ins and add-ons contributed by the developer communities. Below is the brief description about these structures:

1. com.vividsolutions.jump contains the original sources. Classes from this package can be fixed or improved, but functional improvements and new capabilities should be added to org.openjump package.

2. org.openjump.core contains all new contributions, i.e., the improvements added after the initial JUMP development stopped and since OpenJUMP was created.

Most of the new plug-ins should be placed in one of the org.openjump.core.ui.plug-in.xxx packages. Please refer to [12] for getting more insight into JUMP packages and API’s expose by it. Apart from these package structures there are many supporting libraries have been used that exhibit JUMP to provide robust capabilities as mention in Section 5.2. Out of those I have made immensely use of one of the library in my thesis work and below is the brief description of it

**JTS Topological Suite (JTS)** is an API of spatial predicates and functions for processing geometry. It exposes a robust and rich collection of API’s that are used to carry out spatial analysis of geo-data, also to draw various geometric objects, many algorithms, etc. It has the following design goals:

1. It conforms to the Simple Features Specification for SQL published by the Open GIS Consortium.
2. It provides a complete, consistent, robust implementation of fundamental algorithms for processing linear geometry on the 2-dimensional Cartesian plane.

3. It is fast enough for production use.

4. It is written in 100% pure Java.

It is an open source Java software library that provides an object model for Euclidean planar linear geometry together with a set of fundamental geometric functions. It is primarily intended to be used as a core component of vector-based geometrics software such as geographical information systems. It can also be used as a general-purpose library providing algorithms in computational geometry. It implements the geometry model and API defined in the OpenGIS Consortium Simple Features Specification for SQL. It provides a foundation for building further spatial applications, such as viewers, spatial query processors, and tools for performing data validation, cleaning and integration. The software is published under the GNU Lesser General Public License.

6.3 PROGRAMMING GUIDELINES

It is really important from any developer or programmer to follow certain standards defined by the community to have consistent and persistent representation of the contribution to the product. Similarly, in case of OpenJUMP, there are certain guidelines which are important to be kept in mind while writing your own plug-ins. They are as follows:

1. **General Java Programming Guidelines**: OpenJUMP is entirely developed in Java programming language and hence it’s important for each developer or programmer to follow general Java programming language practices. To get more insight in it please refer to [13] in Bibliography. Indentation is one of the key aspects of writing any piece of code in terms of making code easily readable, understandable and separable. Hence it is highly recommended to use whitespaces instead of tabs to make indentation consistent across all IDE as the tab settings may vary among different IDEs.

2. **OpenJUMP Packages**: Embedd your contribution to right place in the package structure exposed by JUMP framework. Please refer to Section 5.6 to get insight into OpenJUMP package structures.

3. **Plug-Ins and Extensions**: JUMP framework provides two ways through which programmer or developer can contribute to OpenJUMP, i.e., plug-ins and extensions. Extensions are the mechanism used in JUMP framework to initialize one or several related plug-ins. They are recognized by JUMP as jar file placed in /lib/ext directory, where as plug-ins are the mechanism that used by programmer to add new functionalities and capabilities to JUMP workbench or application. Detailed
description is explained in Chapter 6 while discussing about various components of JUMP framework.

4. **Plug-In names and initialization**: Please refer to Section 6.5 for detail description about plug-ins names and initialization.

### 6.4 JUMP ARCHITECTURE

The architecture of the JUMP helps user to understand JUMP Framework. It helps user to develop their own plug-ins, cursor tools, renderers, or data-sources. The major components of the JUMP architecture are shown in Figure 6.1.

![Figure 6.1. OpenJUMP architecture.](image)

OpenJUMP architecture is divided into several different components. These components play important in making JUMP as a robust GIS application. Developer need to follow some standards to defined their own sub-components that are part of one these components of JUMP architecture. Components of JUMP architecture are as follows.
6.4.1 Workbench

The Workbench loads extensions, which are JAR files adding functionality to the workbench. This additional functionality may take the form of:

1. Plug-Ins (menu items)
2. Cursor tools (toolbar buttons)
3. Renderers (ways to draw the data)
4. Data-sources (ways to load and save various data formats)

All this functionalities also consist of many miscellaneous tools and components, and the operate of common workbench data structures.

6.4.2 Extensions

An extension is a collection of classes and supporting resources that provides additional functionality to JUMP. Extensions are packaged as JAR files. From the user’s perspective, extending JUMP is as easy as copying an extension JAR files into the JUMP application’s workbench plug-in directory (see Section 6.5.4).

Typically, an Extension will add plug-ins (menu items) and cursor tools (toolbar buttons) to the Workbench. We will discuss more about Plug-Ins and cursor tools in later sections. The JUMP Workbench will search the JAR file for subclasses of Extension. It is important to note that the class consisting code of user defined extension must also be named as “…Extension.” It will then calls the #configure() method on each Extension class it finds.

Figure 6.2 explains diagrammatically, how extensions configure the workbench with new user defined plug-ins, whose respective JAR files are in workbench plug-in directory.

```
CustomExtension()
  |______________________________|
  | Configure()                 |
  |                             |
  | Plugin_1_initialize()       |
  |                             |
  | Plugin_2_initialize()       |
```

Figure 6.2. Extensions.
Let’s take an example code showing how to write a Java class named as “Extension” which configure JUMP workbench with your plug-ins. To do that let me create a MyFirstPlug-In which will write “Hello World!!!!” on the workbench’s output window.

1. First create a plug-in called “MyFirstPlug-In”.

```java
package example;
import com.vividsolutions.jump.workbench.plugin.AbstractPlugIn;
import com.vividsolutions.jump.workbench.plugin.PlugInContext;
public class MyFirstPlugIn extends AbstractPlugIn {
    public void initialize(PlugInContext context) throws Exception {
        context.getFeatureInstaller().addMainMenuItem(this, new String[]
            {"Sample", "Test"}, getName(), false, null, null);
    }
    public boolean execute(PlugInContext context) throws Exception {
        context.getWorkbenchFrame().getOutputFrame().createNewDocument();
        context.getWorkbenchFrame().getOutputFrame().addText("Hello, World!!");
        context.getWorkbenchFrame().getOutputFrame().surface();
        return true;
    }
}
```

2. Then create an extension named “MyFirstExtension” which will load the plug-in in Workbench.

```java
package org.openjump.core.ui.plugin.NewTheme;
import com.vividsolutions.jump.workbench.plugin.Extension;
import com.vividsolutions.jump.workbench.plugin.PlugInContext;
public class MyFirstExtension extends Extension {
    public void configure(PlugInContext context) throws Exception {
        new MyFirstPlugin().initialize(context);
    }
}
```

Now, create a JAR file containing these two classes and save it into the Workbench’s plug-in directory (see Section 6.5.4). You will see a new menu item: Sample > Test > My First, when you restart JUMP. Selecting it will open the Output Window, which will display the “Hello World!!” message.

One might wonder where the Workbench got the menu name “My First” – it is not anywhere in the MyFirstPlug-In code. Generating a friendly name from the class name is one of the useful functions provided by AbstractPlugIn (and is an incentive to create meaningful plug-in class names!).
It is really tedious every time to generate and install JAR file even if you are testing a minor change in your plug-in. It is better to specify the name of your plug-in class in the workbench properties file (refer to Section 6.5.4).

### 6.4.3 Plug-Ins

Plug-Ins is nothing but an object that performs an action, in response to a menu item selection or button press. There are many examples of plug-ins in JUMP source code. All functionalities in form of menus, menu items (including pop-up menus) as well as some of tools in toolbar fall into the category of plug-ins. To accomplish my thesis work I too have created 2 plug-ins named as NewThemePlug-In and HelpTopicPlug-In.

To add a plug-in to the main menu, use

```
PlugInContext.getFeatureInstaller().addMainMenuItem(...)
```

To add a plug-in to the toolbar, use

```
PlugInContext.getWorkbenchFrame().getToolBar().addPlugIn(...)
```

A plug-in has three methods: `#initialize`, `#execute`, and `#getName`. The `#initialize` method is called when the Workbench starts up. `#execute` is called when the plug-in is triggered, e.g., by the user selecting a menu item or clicking a toolbar button. Most plug-ins, in order to do something useful, must manipulate Workbench data structures.

You can customize the plug-in name in `#getName` method of the plug-in interface. AbstractPlug-In returns a default name, based on the class name, if not overloaded or implemented. To support internationalized name or locales, you must overload, e.g.:

```java
public String getName() {return I18N.get("org.openjump.core.ui.plug-in.MyFirstPlugIn");}
```

If the plug-in is called from the menu, you generally want to place it in an appropriate sub-menu and call it by its name. You define where the plug-in is installed in the user interface (e.g., as main menu item, or context menu item or mouse menu item) in plug-in’s initialize method. For details refer to [14] in Bibliography.

To activate/deactivate the plug-in based on certain conditions, you must use an EnableCheck created with the EnableCheckFactory class, by using MultiEnableCheck, or by directly implementing EnableCheck. EnableCheckFactory can be obtain from the PlugInContext. For details refer to [14] in Bibliography.
All plug-ins need some useful information about the JUMP core like the current selection or the current LayerManager, etc. to perform their functions. This information can be obtained from methods belonging to PlugInContext and the more general WorkbenchContext. It might have notes on the usefulness of each method in the contexts; these notes can also be used as Java docs.

6.4.4 Workbench Data-Structures

Workbench data-structure is the common data structure or data-set which is used by the component of workbench like plug-ins, cursor tools, menu items, etc. It mainly comprises of feature, feature collection, feature schema, feature data set, layers, layer manager, tasks, blackboards, task frames, etc. The block diagram in Figure 6.3 depicts the relationship between the data structures.

![Block diagram of Workbench data-structures](image)

**Figure 6.3. Workbench data-structures.**

Let’s take a look at different types of workbench data structures:
1. **OpenJUMP Feature Model:** JUMP feature model defines the relationship between features, feature collections, geometry, layers. Figure 6.4 defines the relationship between these data-structures.

![OpenJUMP Feature Model Diagram](image)

**Figure 6.4. OpenJUMP feature model.**

A feature is a representation of a ‘geographic’ object in the world, including its location, geometry, and other attributes (spatial and non-spatial). In the current Workbench model, each feature has one spatial attribute (its Geometry, imported from the JTS Java Topology Suite) and zero or more non-spatial attributes. Aggregating 1 or more such homogenous or heterogeneous feature into a collection forms a feature collection, which is an interface that supports special methods for querying the features that lie within a given envelope. Other than features, it also defined feature schema that represents a schematic view of feature information like geometry, feature specific information, feature id (FID), etc. Each schema by default has two attribute geometries and FIDs, but it can also have more than two attributes and it all depends on the user. Each attribute holds data similar to hash or map (key-value pairs) i.e., attribute name and attribute type. Attribute can be of type geometry, integer, double, date and object. Feature dataset is concrete class that
implements all the behavior of feature collection. You can also define your own feature collection by either inheriting your class from feature dataset or by defining methods of feature collection. At the broader scene, each feature collection is a part of Layer, whose internal data and representation with respect to workbench is managed by a LayerManager. Layer Manager is nothing but container for or registry of layers. Each project window has a layer manager that manages activities of 1 or more layers that are part of the project.

2. **OpenJUMP Layer-Task Model**: Figure 6.5 depicts the different component of this model.

![Figure 6.5. OpenJUMP layer-task model.](image)

JUMP workbench/UI supports both MDI or regular SDI. In either of the case, each document interface is also known as task frame or project window that represent given task in the interface. Each task is divided into multiple categories. By default if you start new project, you will see two categories named Working and System listed in layer list view of a task frame/project window. Each category further consists of list of layers in the given task. All of these layers in a given task is been managed by a single instance of layer manager. A LayerManager is a container for registry of Layers. A Task is a thin wrapper around LayerManager that associates a name, project file, and Category.

3. **OpenJUMP TaskFrame Model**: The project window in OpenJUMP is also known as Task frame, which consists of following components. See Figure 6.6.

![Figure 6.6. OpenJUMP task frame model.](image)

a. **Task**: A Task is a thin wrapper around LayerManager that associates a name, project file, and Category.
b. **LayerViewPanel**: Right side of project widow that displays maps or geometric objects.

c. **LayerTreePanel/LayerNamePanel**: List the layers that are part of current project.

d. **LayerManager**: A LayerManager is a container for/registry of Layers.

e. **WorkbenchContext**: The context that stores important JUMP core information.

### 6.4.5 Cursor Tools

A CursorTool is a button on the toolbar that sets the mode of mouse interaction (e.g., selection mode, or draw-polygon mode), like the buttons on the Drawing toolbar of Microsoft Word. In other words, when you need to perform any action or operation like mouse click, move, drag, etc. on the geometric objects or map displayed in layer view panel, then you need to define that operation as CursorTool.

To define your own CursorTool, need to create a subclass of AbstractCursorTool, which takes care of all the XOR logic involved in drawing. This class will also automatically generate label names from class name, similar to the way that menu item names are automatically generated by AbstractPlugIn.

Built-in cursor tools are loaded in JUMPConfiguration. There are many examples of CursorTool in JUMP workbench source code. Also I have developed two cursor tools name HelpTopicTool and SearchTopicTool for my thesis work.

To create a new CursorTool, you need to define a subclass of AbstractCursorTool and also have to define three abstract methods that plays really important role in tool life cycle:

1. **#activate**: It is called when the toolbar button is pressed. It will inform the workbench about current active tool and hence take necessary action relate to it.

2. **#deactivate**: It is called when another CursorTool’s button is pressed.

3. **#gestureFinushed**: It is called when the CursorTool’s gesture is complete.

If your application requires more than one or two new CursorTools, it is recommended that you make these available to the User in a ToolBox (see Section 6.5.9, Toolboxes). Cursor tools are another example of a commonly supplied resource that developers might supply as part of an Extension. Figure 6.7 shows the CursorTool’s class hierarchies.
6.4.6 Menu-Items

Menu-items are the fundamental element of JUMP user interface that forms the binding between user action and plug-in functionalities. Most probably you would like to add menu items either in menu-bar or in popup menus. Following are different ways through which you can add menu items:

1. Use `addMainMenuItem` to add an item which is always available.
2. Use `addLayerViewMenuItem` to add a menu item which is only enabled when a task window has the focus.
3. Use `addLayerNameViewMenuItem` to add a menu item which is only enabled when a task window has the focus, and that Task window has a Layer Name panel visible.
4. Use `addPopupMenuItem` to add right click popup menu-items.
Menu-items need not to be enabled all the time during the application and also it creates a confusion if the reason for it disability is not clearly known. To resolve the mysteries enable-checks, the JUMP workbench provides a mechanism that makes it easy for developer when a menu item should be disabled and why.

When you add a plug-in as a menu, one of the parameters in `#addMainMenuItem` is an Enable-Check. An enable-check has a single method that returns null if the menu item should be enabled or a String if the menu item is disabled. This String gives the reason for disabling the menu (e.g., “At least one feature must be selected”). When the user places the mouse cursor over the disabled menu item, the reason will be displayed as a tooltip. The enable-check is called whenever the parent menu is opened. For more information about this mechanism please refer to [14] in Bibliography or JUMP developer Guide (i.e., Section 4.2, Enable-Checks).

### 6.4.7 Renderers and Styles

A renderer is an object that draws on the Workbench using a `java.awt.graphics`. A style is used by a `LayerRenderer` to draw a feature. Styles are a bit easier to write than fullblown renderers because you can focus on rendering one feature. Most plug-ins do not need to write their own renderers or styles. They already have control over the colour, line width, etc. If they need to “decorate” features with arrowheads, etc., there are a few existing Styles, for example, `ArrowLineStringEndPointStyle`.

The `BasicStyle` specifies style elements such as colours, line widths, etc. `ColorScheme` is used to select from a range of professionally designed color schemes. You can pass one of the fill patterns from `FillPatternFactory` into `BasicStyle#setFillPattern`, or make your own. The factory contains all of the very cool textures from IBM. For example, you can instantly give a fake mountain texture to your mountain features.

### 6.4.8 DataSources

A `DataSource` is an object that mediates to move data between the Workbench and a file or other data source; it will replace the `Readers/Writers` found in earlier versions of JUMP.

Similarly, `DataSourceQueryChooser` replaces the now-deprecated Driver. These are the UI panels associated with a `DataSource`. Note: Reader and Writer are have not been
deprecated in JUMP 1.1, because there is no plan to re-write the existing Readers/Writers in the near future.

Developers may need to supply custom DataSources in their JUMP Extensions in order to provide access to data for their applications.

Note: Use workbench property file to avoid generating JAR file every time.

6.4.9 ToolBoxes

A toolbox is a small modeless dialog that can contain cursor-tool buttons and custom GUI components. Toolboxes are always visible because they float above the Workbench, like the Picture toolbar in Microsoft Word. If your application requires the addition of more than one or two new cursorTools, using a Toolbox is recommended: it avoids cluttering the main toolbar (which is already pretty packed!); and it allows the user to interact with data even while the dialog is up (modal dialogs can be annoying).

To setup a toolbox, following are the required steps:
1. Implement initialize Toolbar.
2. Add CursorTools to the toolbar. (Optional)
3. Add a panel to the toolbox.
4. Write event handlers for the components in the panel.

6.4.10 Miscellaneous Components

Following are the other miscellaneous components provided by JUMP framework:

1. **Java2XML**: Java2XML is a utility class that lets you turn objects into XML documents and vice versa. It’s dead simple to use, yet quite powerful — you can create complex-looking XML from it. Java2XML has advantages over existing Java XML bindings: it is simpler than Castor (for example, there is no need for a schema or DTD); unlike Digester, it can both read and write; unlike Bewitched, it gives clear and helpful error messages when a problem occurs. Since there are very few classes (3), it’s relatively easy to track down problems. Please refer to [14] to get detail information about this topic.

2. **Undo**: A plug-in can indicate via AbstractPlugIn#reportNothingToUndoYet that it does not modify the system, or that it is undoable, but has not modified the system yet.

3. **Feature Text Writer**: There are different ways to display the contents of a Feature. Examples as WKL, GML and Coordinate list button on left side of feature info window. We can also provide customized way of displaying feature information by simply subclass AbstractFeatureTextWriter (and implement its one abstract method); then in your plug-ins #initialize method, say:
context.getWorkbenchContext().getFeatureTextWriterRegistry().register(FTW)
This code will add a button to the Feature Info window.

There several other topic related to JUMP architecture and I will highly encourage you to refer [14] in Bibliography to get information about those area.

6.5 JUMP How To’s

This section discusses about general question related OpenJUMP and the supporting tools and IDE’s used for accomplishing the thesis work.

6.5.1 How to Get OpenJUMP?

There two locations from where you can download OpenJUMP SDK:

1. JPP resource section is the location from where you can download the latest OpenJUMP SDK version as well as executable of OpenJUMP GIS application. Please refer to [15] in Bibliography for more details. This version will include all the latest and current development as well as contribution made by developer communities to OpenJUMP, approved by JPP and Vivid Solutions as well as Refraction Research. JPP is a development group that currently working on maintaining as well as adding new features and capabilities to OpenJUMP. For more information about JPP please refer to Section 6.6.

2. Apart from this, one can use the OpenJUMP version that has been used for this thesis work. The IDE should be capable to get source code from Google SVN that has been created from this thesis work. For this thesis, Eclipse JDT has been used. Please refer to Section 6.5.2 to get information about how to integrate or use Google SVN with Eclipse JDT and other related information. Also the version I have used is 1.3.1, which is not the latest one. Hence refer to Section 6.5.3 for more information about how to upgrade it to latest.

6.5.2 How to Integrate Google SVN and Eclipse JDT?

Google SVN is a free subversion hosting of open source projects. It provides a project hosting services along with the standard control and versioning system features. To use Google SVN or any CVS repository from Eclipse JDT, you need a plug-in called subclipse. Subclipse provides you a cvs client that will allow you perform basic functionalities like check-in/out code, synchronize to latest, sync to version#, merge, differentiate between local and CVS repository, etc. To download subclipse please refer to [16] in Bibliography. Once you installed subclipse, you can simple follow step mentioned in Section 6.5.3 to check-in/check-out code from repository.
6.5.3 How to Check-In/Out Code from Google SVN?

Please refer to wiki pages related to Checking-in and checking-out, from SVN in Google SVN project hosting site, where it has clearly explained the process step-by-step. For site information refer to [17] in Bibliography.

6.5.4 How to Configure Your Plug-Ins into JUMP?

There are two major things you need to keep in mind whenever you want to configure your plug-in into OpenJUMP.

1. **Properties File:** The Workbench allows developers to specify the name of plug-in classes in the workbench properties file, an XML file whose location can be specified in a command-line argument for the JUMP Workbench. Thus, the Workbench looks for plug-ins in two places: in JAR files in the workbench plug-in directory, and in classes specified in the workbench properties file. These class names are passed to the JVM's classloader, which searches for them in the Java CLASSPATH. Follow this steps to create your own property file:
   a. Create a file somewhere called “workbench-properties.xml” with the format below. Put the name of your class between the <plug-in> tags.
      
      ```xml
      <workbench>
        <plug-in>example.HelloWorldPlugIn</plug-in>
      </workbench>
      ```
   b. In your IDE, where you specify JUMPWorkbench as the class to run, specify the location of the workbench properties files as a program argument. Under Windows, this would require a command-line parameter like the following:
      
      ```
      -properties C:\Sandbox\HelloWorld\workbench-properties.xml
      ```
      Under MacOS X, GNU/Linux, or other versions of UNIX, the command-line parameter would look like:
      
      ```
      -properties /home/jumpuser/Sandbox/HelloWorld/properties.xml
      ```
      Now when you run JUMPWorkbench, it will call your plug-in’s #initialize method during startup.

2. **Plug-In Directory:** JUMP’s Extension mechanism makes it easy for users to add functionality to the Workbench: they simply need to copy a JAR file into their application’s workbench plug-in directory. By default, this directory is lib/ext/ in the directory where JUMP is installed (lib/ext/ in the Unix/Linux/MacOS world). It is, however, possible to specify another directory by specifying the --plug-in-directory command line option when JUMP is started. You can provide this command-line as program arguments under arguments tab inside run configuration settings of your JUMP project.

6.5.5 How to Run JUMP from Eclipse JDT?

Please refer to [18] in Bibliography which will explain you steps that you need to follow to execute OpenJUMP from eclipse JDT.
6.6 JUMP Pilot Project (JPP)

It is an organization dedicated to provide software development solutions based on open source GIS program JUMP. It coordinates and encourages the cooperative development of technologies related to OpenJUMP. It provides developers from around the world an opportunity to work together and avoid duplicated efforts. Please refer to [19] for more information about it. Currently JPP mainly coordinates OpenJUMP software platform. This is a version of the original JUMP being developed and maintained by a group of volunteer developers from around the globe. This list is for discussion between developers working on JPP plug-ins, the JPP core, or other JPP enhancements. It is also an appropriate place for questions about operation and use of the version of JUMP administered by the JPP. To see the collection of prior postings to the list, visit the JUMP-pilot-devel Archives [20]. Use JUMP-Pilot-Devel to post a message to all the list members, send email to jump-pilot-devel@lists.sourceforge.net. You can subscribe to the list, or change your existing subscription.
CHAPTER 7
IMPLEMENTATION AND CONTRIBUTION

This chapter describes the implementation details for the extensions developed for OpenJUMP. Figure 7.1 shows the blocks diagram of the GIS Marker Plug-in for OpenJUMP.

![Diagram of GIS Marker Plugin Architecture](image)

**Figure 7.1. Plugin architecture for GIS marker.**

It consists of various different components that play an important role in user’s perspective. My thesis work primarily focused towards user view and provides feature extension that can be used to efficiently execute GIS Marker. The components of GIS Marker plug-in that play effective role in user perspective are Hot Link tool, Hot Link Editor and the Theme component. Along with these components there are two tools. All these plug-ins and tools are part of org.openjump.core.ui. plug-in package library. The tools developed in this thesis are external plug-ins that act as additional functionality to OpenJUMP workbench.
7.1 HOT LINK TOOL

Hot Link Interface has the following characteristics:

1. It acts as an interface between the Hotlink Editor, Hotlink Tool and New Theme plug-in to manipulate the feature schema associated with the feature collections or layer.
2. Each layer’s newly created or existing schema can modified to add new attributes related to Hotlink and New Theme plug-ins.
3. We can modify name, text, image files, web-links, audio files based on the feature-ids.
4. It exposes API’s like Get, Set and Has methods to query the selected schema.
5. For example, getFeatureText(), setFeatureText(), hasFeatureText() etc.

Figure 7.2 explains HotLink architecture in a better way and Figure 7.3 explains the flow diagram associated with HotLink tool. Figure 7.4 explains HotLink editor’s functional flow diagram and Figure 7.5 shows HotLink tool and HotLink icon in the openJUMP editor.

Figure 7.2 Hot link architecture.
Figure 7.3. Hot link tool flow diagram.
7.2 New Theme Plug-In

As an admin perspective, New Theme provides the user a plug-in or functionality that will allow creating new features by entering features geographic information. Figure 7.6 explains the flow diagram for new theme flow plug-in. When you click on tool in the toolbar it will display dialog as shown in Figure 7.7.
Figure 7.6. New theme flow diagram.

Figure 7.7. New theme dialog XY coordinates tab.
New Theme dialog provides user three ways to enter geographic information about their features. They can either select “XY Coordinates” tab to manually enter the geo-data or they can create a CSV file as shown in Figure 7.8, which contains same geo-data but in a specific format (Section 3.1.3.1 explains in detail about geo-data format of various geometry and how to create a CSV file for any new feature that you want to create using this plug-in). To enter geo-data manually, dialog will show format of the geometries that you select in the geometry type. There is a third option/tab called “Get Coordinates” that will allow us to get the geo-coordinates of the place we are looking for by entering its address, as shown in Figure 7.9.

Figure 7.8. New theme dialog from CSV file tab.

Figure 7.9. New theme dialog get coordinates tab.
7.3 Help Topic Plug-In

This plug-in provides utility to search for any topic related OpenJUMP application or to get the usage information of any tools present in the tool bar. Whenever you try to get familiar with new application you need to have help tool to make you understand with tools and functionalities provided by the application. This plug-in is divided into two tools:

1. **Help Topic Tool**: It allows user to get familiarize with the behavior of the tools and functionalities available on the toolbar component of the OpenJUMP workbench. To understand the usage of any tool including Help Topic Tool, you need to first click ![help icon] to activate the tool and after click on any tool on the toolbar to shows its usage in samplehelp.chm file. Please refer to Section 3.1.3.2 for getting brief information about help files and .chm file.

2. **Search Topic Tool**: This tool will allow you to search any topic related to OpenJUMP GIS application. It will also display same .chm html help file but will have searching and indexing capabilities embedded into it. You need to click on tool on toolbar to display the help file to search for the required information.

The .chm html help file is not fully developed for its intended purpose. But the main objective of this tool is to provide any interface/utility through which you can display the help or usage information of various tools and topic within the application. For better help information, you can modify the samplehelp.chm file or can construct your own file and change its respective file path in these tools to display your version of the help information.

Also its important to understand why I decided to implement this functionalities as tools instead of plug-ins because these tools require to perform some mouse click events on the components of OpenJUMP workbench like layer view panel, toolbar, etc. By declaring these functionalities as cursor-tools, it allows tools to provide its own implementation to handle these events when the user performs them and execute their behaviors. Figure 7.10 shows the class diagram of these plug-in and relationship among them.
Figure 7.10. Help topic plug-in class diagram.
CHAPTER 8

POSSIBLE ENHANCEMENTS

After the development of this tool, I still feel that there is a possibility of the following enhancements in the future:

- The tool can be upgraded to support new geometries that will be included into JTS Topological coming suite in future.
- We can come up in standardized way to define the look and feel of the html help files that is compiled into chm html help file.
- Also there various different areas in OpenJUMP where you can contribute to make more robust GIS tools. Please refer to OpenJUMP developer guide about the future to-dos and enhancements.
CHAPTER 9

CONCLUSION

The main objective behind development of this plug-in is providing admin perspective of GIS marker as an effective tool for learning purpose for the OpenJUMP user. Also while going through the development phase of this plug-in, I tried my best to explore OpenJUMP framework and architecture at my best which can help you to start using it as development platform to provide your valuable contribution in the field of GIS application. I successfully completed the development of this tool that will allow user (students or faculties) to create their own themes over various maps and embedded their require information with it that can be used to improve their learning purpose. This can successfully allows you to create all kinds of geometries that are currently been supported by JTS library and will only take geographic coordinates (i.e., longitudes and latitudes) of these geometrical objects.

There are lots of enhancements that can be done on the OpenJUMP GIS application using its robust open source and extensible nature. I tried my best to expose its framework and architecture through my thesis work but there are lot many areas and supporting library that it uses for various different purpose that might be useful to you to accomplish your contributions. Please refer to Bibliography and find extra informational source for more information on your required areas.
BIBLIOGRAPHY

WORKS CITED


WORKS CONSULTED


