Visualization of Concurrent Versions Systems

by

Bharath Suresh

Bachelor of Engineering, Computer Science
Anna University, India
2007

A project submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
June 2010
ABSTRACT

Visualization of Concurrent Version Systems

by

Bharath Suresh

Dr. Jan B. Pedersen, Examination Committee Chair
Assistant Professor, Department of Computer Science
University of Nevada, Las Vegas

Information Visualization is a computing technique used to analyze large data sets through visual representation. Information Visualization transforms abstract data sets into a form that facilitates human interaction for exploration and understanding.

In this project, tool called Visual CVS [Concurrent Versions Systems] has been developed to visualize CVS log files. The tool enables the end user to effectively explore the CVS variables like revision, date and time in the form of graphs. The tool makes it easier to observe the extent of changes to any file in the CVS database. A number of interaction techniques such as filtering the graph based on CVS variables are included in the tool to understand the data better. This tool has been successfully tested on a large data set.
TABLE OF CONTENTS

CHAPTER 1    INTRODUCTION ..................................................................................... 1

Organization of the project .................................................................................. 3

CHAPTER 2    CONCURRENT VERSIONS SYSTEMS................................................. 4

Concurrent Versions Systems Log Files.................................................................... 5

CHAPTER 3    USER INTERFACE .................................................................................. 8

Tool usage............................................................................................................. 10

CHAPTER 4    IMPLEMENTATION.............................................................................. 15

Database Design...................................................................................................... 16

Need to choose three different plots for the visualization ..................................... 22

SQL querying in the interface............................................................................... 22

Temporary Table.................................................................................................. 25

CHAPTER 5    RESULTS .............................................................................................. 27

CHAPTER 6    CONCLUSION ....................................................................................... 38

Future Work .......................................................................................................... 38
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>CVS- Client Server Architecture</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>A CVS log file sample</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Components used in the interface</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>Scatter plot</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Line graph</td>
<td>12</td>
</tr>
<tr>
<td>3.4</td>
<td>Bar graph</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>Format of a CVS log file</td>
<td>15</td>
</tr>
<tr>
<td>4.2</td>
<td>Revision number with branching</td>
<td>19</td>
</tr>
<tr>
<td>4.3</td>
<td>Revision numbers not sorted properly</td>
<td>20</td>
</tr>
<tr>
<td>4.4</td>
<td>Table sorted properly with the addition of Rev_whole field</td>
<td>21</td>
</tr>
<tr>
<td>4.5</td>
<td>The user interface</td>
<td>23</td>
</tr>
<tr>
<td>5.1</td>
<td>An example of a scatter plot using revision as dimension 1, date as dimension 2 and revisioncnt as dimension 3</td>
<td>28</td>
</tr>
<tr>
<td>5.2</td>
<td>An example of a line graph showing revision as dimension 1, date as dimension 2 and authorcnt as dimension 3</td>
<td>29</td>
</tr>
<tr>
<td>5.3</td>
<td>An example of a scatter plot graph showing revision as dimension 1, time as dimension 2 and authorcnt as dimension 3 with avila, biddi, childs selected as authors</td>
<td>31</td>
</tr>
<tr>
<td>5.4</td>
<td>An example of a bar graph showing time as dimension 1 and revisioncnt as dimension 2 with a list of authors selected</td>
<td>33</td>
</tr>
</tbody>
</table>
Figure 5.5  An example of a bar graph showing author as dimension 1 and revisioncnt as dimension 2.

Figure 5.6  Line graph showing date as dimension 1, time as dimension 2 and authorcnt as dimension 3 with list of authors selected.

Figure 5.7  Error message displayed while choosing a line graph.

Figure 5.8  A snapshot of fit to screen mode.

LIST OF TABLES

Table 4.1  The structure of Headtable.

Table 4.2  The structure of Maintable.

Table 4.3  The structure of Commentstable.

Table 4.4  The structure of tempgraph table.
CHAPTER 1

Introduction

Since the advent of computing, significant amount of research has been done on
improving the human computer interaction. As computers have grown prevalent
across the world and information on various disciplines is becoming widely available,
it is important to organize and present the available data in easily usable format.
Since over 50% of the brains neurons are connected to vision, a visual representation
can communicate some kinds of information much more rapidly and effectively than
any other method. Such representation of information in a visual system is called
Information Visualization [IV].

IV is defined as any tool or method for interpreting image data fed into a computer
and for generating images from complex multi-dimensional data sets. IV deals with
large abstract data sets that do not have a direct physical correspondence and maps
them to a two or three dimensional representation. Strong interaction techniques
enable the user to modify the visualization in real-time, thus affording perception of a
wide array of patterns and relations in the abstract data in question. IV is widely
recognized as a powerful tool in scientific research, digital libraries, data mining,
financial data analysis, market studies, manufacturing production control, and drug
discovery.

This project deals with using Information Visualization for Concurrent Version
Systems [CVS]. CVS is an example of a Version Control System. Version Control is
widely used in software development, where a team of people may change the same
set of files at the same time. Changes are usually identified by a number or letter code, termed the "revision number". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. The version control software maintains a log of all the changes made to any file. If 100 changes are made to a particular file by 100 users, log files track each of these changes. This leads to an explosion of log file sizes and makes it extremely difficult to analyze the extent of changes. Analysis of CVS log files is a typical application where the use of IV can ease the burden of users trying to analyze and understand changes to their code repositories.

In this project, a standalone application called **Visual CVS** to visualize the CVS log files has been developed. In Visual CVS, the log file is analyzed by plotting them as graphs. The graphs are plotted using four attributes. The first two attributes are the data obtained from the log files, third attribute is the size of the graphical marker (it can be size of an oval or height of a bar used in the graph) and the fourth attribute is the color of the graphical structure(color of an oval or a bar in the graph).

The graphs used in this project are scatter plot, line and bar graphs. These graphs can be analyzed in greater detail using many interaction techniques. For example, graphs can be filtered just to show all the changes made to a file by a particular author. Graphs can also be viewed in fit to screen mode.
Organization of the project

An overview of Concurrent Versions Systems is presented in Chapter 2. The components of a CVS log file, which form the data set for this project is explained in detail in the same chapter. The third chapter deals with the user interface designed for this tool. It also provides the details involved in using the interface. The fourth chapter details the database design and the implementation of the tool. Chapter 5 describes the results and snapshots taken from this tool. The last chapter concludes this report and includes a section on the possible enhancements that can be made to this project in the future.
CHAPTER 2

Concurrent Versions Systems

Computer software is often developed by multiple programmers across multiple
time zones. So, it is extremely important that all changes made to the design files are
captured correctly. The role of managing access to shared code belongs to Version
Control Systems. Version Control systems can be defined as software tools that
manage changes to documents, programs, and other information stored as computer
files.

Concurrent Versions Systems (CVS) is an example of a version control system.
CVS keeps track of all the changes in a set of files and allows several developers to
work together. CVS works on a client-server architecture as illustrated in the figure
below.

Figure 2.1: CVS- Client Server Architecture.

[Pic Courtesy: Zef Hemel]
The server is responsible for maintaining several details of the project including the current version and its history. A version control server is like the heart and soul of CVS. It contains the latest versions of all directories and files in the project. It also contains all the older versions as well.

The client checks out the copy of the current version of the project, makes the changes and checks in the copy of his project. A client can also add new files to the project as well as delete older files from the project. If the client wants these changes to be reflected in the version of the project stored in the server, these changes need to be committed and the database in the server will be updated with the latest changes to the files.

**Concurrent Version System Log Files**

The details regarding the changes to the files are maintained in the CVS log file. The log file contains information about the modules checked in, checked out, status of the work done on the module, users who have checked out the module and the lines of code added or modified by the user. An example of a CVS log file is shown in figure 2.2.

A CVS Log file typically contains a revision number, date of revision, time of revision, author of revision, and description of revision. These elements of a CVS log file are known as CVS variables.
Working file: ./vtk/common/vtkProcessObject.cxx
head: 1.22
symbolic names:
Kitware-PCD1: 1.22
VolView-12-Branch: 1.14.0.2
release-2-3-0: 1.4
release-3-3-branch-point: 1.22
release-3-2: 1.21.0.2
release-3-1-2: 1.11
release-3-3: 1.22.0.4
pv10-branch: 1.22.0.2
release-3-1: 1.11
Kitware-PCD0: 1.12
MPI-paper: 1.19.0.2
pv10: 1.22
Kitware-VV11: 1.5
VAV_8_11_98: 1.2
GE_BASELINE3_PATCH: 1.4.0.2
release-2-2-beta2: 1.4
BASELINE3: 1.4
release-3-2-branch-point: 1.21
release-2-2: 1.4
Kitware-IP20: 1.11

keyword substitution:
total revisions: 22;
description:
-------------------------------------------------------------
revision 1.1
date: 1998-03-26 22:50:19 +0000; author: Schroede;
state: Exp;
ENH: Consolidated "typed" attribute data; added fields and cell data.
-------------------------------------------------------------
revision 1.2
date: 1998-04-16 21:06:43 +0000; author: martink; state: Exp;
lines: +19 -3
BUG: fixed memory leaks
-------------------------------------------------------------
revision 1.3
date: 1998-09-03 17:51:30 +0000; author: millerjv;
state: Exp;
lines: +26 -8
Style
-------------------------------------------------------------

Figure 2.2: A CVS log file sample.
In Figure 2.2, working file shows the file on which the changes are done, total revisions shows the number of revisions made on the file, revision shows the revision number, date shows the date on which the change was done, time shows the time on which the change was done, author shows the name of the person associated with the change, lines shows the number of lines added and removed by the user.

The above figure just shows a snapshot of a CVS log file. As projects grow larger, the log files also grow larger making it in nearly impossible to understand the effect of the changes made to the working file. The user interface described in the next chapter will greatly help in reducing the complexity involved in analyzing large CVS log files.
CHAPTER 3

User Interface

CVS log files discussed in the previous chapter is the input to the user interface developed in this project. A Java code snippet processes the CVS log file and splits each CVS variable into individual tokens. These tokens are stored in a table structure in the database. Depending on the data required for the interface, variables are selected and retrieved from the database. The user interface is designed to visualize the data obtained from the database. Figure 3.1 shows the interface developed in this project. GUI components used in this interface are generated using SQL queries.

Figure 3.1 Components used in the Interface.

Figure 3.1 shows the user interface of this tool. The user interface contains a frame on which all the components are drawn. There is drawing panel in the middle of the screen where all the graphs are drawn. The drawing panel has been implemented as a
scrollable interface. Files that need to be visualized are selected from the file list box and are drawn in this drawing panel.

All files taken from the database are displayed in the file list box. Users can select single files or multiple files from the file list box for visualization.

The tool provides three different graphs for visualizing the data. The three graphs are bar, scatter and line graph. There are three radio buttons displayed on the screen which are used to select the type of graphical representation.

The variables in CVS log files are known as dimensions in this tool. These dimensions can be chosen from the combo boxes displayed on the screen. SQL queries are written to retrieve all the information related to the selected dimensions. The data can also be filtered.

The author list box in this tool is used to filter the data according to the author. This list box is capable of accepting single or multiple selections. Each author is assigned a specific color in the list box. The color of the graph depends on the color associated with the selected author.

“Fit to screen” is another interaction technique provided in this tool. This option provides the user to view the data in a normal mode and fit to screen mode. If the fit to screen option is selected, data is displayed in the “Fit to Screen Mode”.

The queries required for the graph visualization are triggered by clicking the “GO” button. This button is finally where the visualization comes up on screen. The next section illustrates the manner in which the tool needs to be used.
Tool Usage

The tool has to go through the following steps to visualize the data. First, the list of log files to be visualized has to be chosen from the file list box. The type of graph has to be chosen from radio buttons below the file list box. This selection can be a single or multiple files in case of scatter plot and bar graph. In the case of line graph, selection is limited to five files. This restriction on the selection of files for the line graph is to make sure that lines do not get congested and hence making the visualization unclear.

The dimensions will be displayed depending on the type of graph. If scatter plot or line graph is chosen, three dimensions will be displayed but in the case of bar graph only 2 dimensions will be displayed. The first 2 dimensions in the case of scatter plot or line graph, will be revision, date, or time. The 3rd dimension can be count of revision, date, time, or author. In the case of a bar graph, the first dimension will always be revision, date, time, or author, and the second dimension will always be count of revision, date, time, or author. Count variables give the count of those variables in selected files and according to the variables selected as dimension 1 and dimension 2 in the interface. As an example, Count of revision gives the total number of revision according to the variables selected as dimension 1 and dimension 2 in those selected files.

Once the dimensions are chosen, the two combo boxes below populated with data from the list of files selected. The range of data for each dimension can be chosen
from the “from” and “to” combo box. The range of data can be specified only for dimensions like

- Revision,
- Date,
- Time and
- Author.

The user is also given an option to filter the data according to the author who made the changes. The list of authors involved in the project is displayed in the author list box. Depending on the selection, the data will be filtered and displayed on the screen.

Figures 3.2, 3.3 and 3.4 show examples of the scatter plot, the line graph and the bar graph visualizations respectively.

![Figure 3.2 Scatter plot](image-url)
Figure 3.2 shows the example of a scatter plot graph with revision as dimension 1, date as dimension 2 and count of date as dimension 3. The numbers displayed on top of the red dots shows the file number with which it is associated.

Figure 3.3 shows an example of a line graph with revision as dimension 1, date as dimension 2 and count of author as dimension 3.

Figure 3.3 Line graph

Figure 3.4 shows an example of a bar graph with time as dimension 1 and count of revision as dimension 2. The colors displayed on the bar graph shows the authors associated it.
It is clear from the previous section that only certain combinations can be selected from the combo box in case of each plot. However, all combinations of dimensions are not legal. The following sets of rules govern the usage of this tool.

**Rule 1: Certain combinations are illegal and hence not allowed.**

In the case of a scatter plot graph with *revision* as dimension 1, *date* as dimension 2 and *author* as dimension 3, the plot gives the details of the every revision made on that particular date with author as dimension 3. But this plot did not give a good visualization as the plot did not reveal any clear purpose in the selection of the third dimension.
Rule 2: For three dimensional visualizations, only two dimensions can be selected from the variables of the log file.

Similarly having third dimension as variable from the log file did not provide good visualization results as the plot did not clearly reveal any useful information. So, it was decided to have the third dimension as count of the one of the variable used.

Rule 3: Author is solely represented in terms of colors in scatter plot and line graphs

Let us consider another example of a scatter plot graph with author as dimension 1, revision as dimension 2 and count of author as dimension 3. This plot will have the details of every author working on every revision in the selected file with count of author as 3rd dimension. This plot does not provide any useful data because there can be only one author per revision and count of author in the 3rd dimension is always going to be one. Hence having author as one of the variables in the first two dimensions does not give enough details about the plot. In order for the visualization to look interesting author variable has been shown in terms of colors. In order to make the visualization meaningful, author was not given as one of the dimension for scatter plots and line graphs. If the authors were selected in the author list box, they are represented in terms of colors in the scatter plot and line graphs. However, author is one of the dimensions for a bar graph representation.

All the above mentioned rules were taken into consideration and it was decided to have first two dimensions from the variables in the CVS log file, third dimension from the count of the variables and fourth dimension to be from the color of the authors involved in the project.
CHAPTER 4

IMPLEMENTATION

Visual CVS consists mainly of two packages namely splitlogfile and graphics_cvs. Input for the tool comes from the logfile1.txt which consists of the CVS log file entries. This file contains many log files entries. A sample of logfile1.txt is shown in figure 4.1. This file is split into tokens using splitfile.java which is a java code that basically splits the file into meaningful tokens. These split tokens are stored in the database. The next section shows how the database was designed.

Figure 4.1. Format of a CVS log file.
Database Design

One of the main challenges of this project was to devise a way to store the CVS variables. It was decided to store these variables in a database to help in easy information retrieval.

The database was designed using MS Access. The CVS variables were stored in a database with details of every file along with the date, time, author and other details. The following tables were created and the fields in the table are described in Table 4.1, Table 4.2, Table 4.3 and Table 4.4 respectively.

1. Headtable,
2. Maintable,
3. Commentstable and
4. Tempgraph.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working_file</td>
<td>Text</td>
</tr>
<tr>
<td>Head</td>
<td>Text</td>
</tr>
<tr>
<td>Total_Revision</td>
<td>Text</td>
</tr>
</tbody>
</table>

Table 4.1 The structure of Headtable.

Table 4.1 shows the structure of Headtable. This table contains the Working_file which shows the file on which the changes were made, Head shows the last added revision number of the file and Total_Revision shows the total number of revision made to that file.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working_file</td>
<td>Text</td>
</tr>
<tr>
<td>Revision</td>
<td>Text</td>
</tr>
<tr>
<td>Date</td>
<td>Date/Time</td>
</tr>
<tr>
<td>Author</td>
<td>Text</td>
</tr>
<tr>
<td>State</td>
<td>Text</td>
</tr>
<tr>
<td>Time</td>
<td>Date/Time</td>
</tr>
<tr>
<td>Rev_whole</td>
<td>Text</td>
</tr>
</tbody>
</table>

Table 4.2 The structure of Maintable.

Table 4.2 shows the structure of Maintable. This table contains the Working File. Revision shows each and every revision number of the revision done in that file, Date shows the date on which that revision was made, Time shows the time at which that revision was made, Author shows the author who made changes to that revision, State shows the state assigned to the revision and Rev_whole shows the field which is got from the revision field by converting its revision number into a 16 digit number. Rev_whole is stored as text data type. The reason for storing it as a text data type is discussed later in this chapter.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working_file</td>
<td>Text</td>
</tr>
<tr>
<td>Revision</td>
<td>Text</td>
</tr>
<tr>
<td>Comments</td>
<td>Memo</td>
</tr>
</tbody>
</table>

Table 4.3 The structure of Commentstable.

Table 4.3 shows the structure of Commentstable. In this table, Working_file, Revision and Comments gives the comments made on that revision in the file.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimen1*</td>
<td>Text</td>
</tr>
<tr>
<td>Dimen2*</td>
<td>Date/Time</td>
</tr>
<tr>
<td>Dimen3</td>
<td>Number</td>
</tr>
<tr>
<td>Rev_wh</td>
<td>Text</td>
</tr>
<tr>
<td>Author</td>
<td>Text</td>
</tr>
<tr>
<td>File</td>
<td>Text</td>
</tr>
</tbody>
</table>

Table 4.4 The structure of tempgraph table.

- The data type of these fields change according to what is chosen as Dimension 1 and Dimension 2 in the interface.

Table 4.4 gives the structure of tempgraph table. In this table Dimen1, Dimen2 and Dimen3 change according to what is chosen as dimension 1, dimension 2 and dimension 3 in the interface. Rev_wh field gives the 16 digit number of the revision,
Author gives the name of author involved in that revision and File gives the name of file.

The revision field is stored as a text data type since some of the revisions have branching included in their revision number. For example,

```
revision 1.4.2.1
date: 2000-09-14 15:24:45 +0000; author: martink;
state: Exp; lines: +12 -3
ENH: add debug leak call for non-factory object.
```

Figure 4.2 Revision number with branching.

In figure 4.2, the revision number is of the format 1.4.2.1. Since there is no number format that supports this kind of number, revision numbers were stored as text.

Storing of revision as text does not aid in sorting entries according to revision numbers. Figure 4.3 shows the sorting problem, when sorting revision numbers as text strings.
Since revision numbers were stored as text, the order in which they were sorted was incorrect. If we try to visualize the data from the figure 4.3, we end up getting revision 1.10 and 1.11 before revision 1.2 and 1.3. This will lead to incorrect visualization.
In order to overcome this issue, the revision field was stored in a string variable and was split with a string split operation. Then the variables were formatted with leading zeros and combined to get a 16 digit formatted text. This was stored in Rev_whole column in the table. This field stores a 16 digit formatted text of the revision number. For example, revision number 1.1 is converted into 0001000100000000. In this way, the revisions can be sorted properly and the visualization of the data can be done accurately.

![Table sorted properly](image)

**Figure 4.4** – Table sorted properly with respect to their revision numbers with the addition Rev_whole field.
Need to choose three different plots for the visualization

The graphics_cvs package consists of files that deal with the actual visualization. The files in this package allow the user to visualize the data using three different types of plots. As discussed earlier, not all the combinations in the dimension used in the graphical representation make sense.

The following table details all the legal combinations of dimensions for each type of graph.

<table>
<thead>
<tr>
<th>Plot Name</th>
<th>Dimension 1</th>
<th>Dimension 2</th>
<th>Dimension 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Revision</td>
<td>Date</td>
<td>Time</td>
</tr>
<tr>
<td>Scatter</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Line</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bar</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

SQL querying in the interface

SQL querying is done to get the input of the dimensions in the interface. According to the dimension chosen, corresponding SQL query is generated to display the results on screen. Figure 4.5, shows the interface in which revision is selected as dimension 1 and date is selected as dimension 2.
As discussed earlier, the revision field in the database was stored as text, so an additional field called rev\_whole was added. Here, while writing SQL queries we make use of the field. Whenever the revision field is chosen, we write a SQL query to order that by the rev\_whole field. For example, when revision was chosen as dimension 1, the query is written in this form

```
SELECT DISTINCT Revision,Rev\_whole FROM logTable2\_mylog
Where
Working\_File='./vtk/common/vtkAbstractTransform.cxx' OR
Working\_File ='/vtk/common/vtkActor2D.cxx' OR
```

Figure 4.5 The user interface.
Working_File = './vtk/common/vtkActor2DCollection.cxx' OR
Working_File = './vtk/common/vtkAssemblyNode.cxx' OR
Working_File = './vtk/common/vtkActor2DCollection.cxx' OR
Working_File = './vtk/common/vtkAssemblyPath.cxx' ORDER
BY Rev_whole;

The rest of the dimension field was generated using SQL query that contains its
dimension and order by its dimension. Here is an example,

SELECT DISTINCT Date FROM logTable2_mylog Where
Working_File='./vtk/common/vtkAbstractTransform.cxx' OR
Working_File='./vtk/common/vtkActor2D.cxx' OR
Working_File='./vtk/common/vtkActor2DCollection.cxx' OR
Working_File='./vtk/common/vtkAssemblyNode.cxx' ORDER
BY Date.

Dimension 3 in the scatter plot and the line graph are always ‘count’ variables of the
dimensions. Count variables are usually taken as the count of that variable used in the
table by SQL querying. Here is an example of the query,

SELECT COUNT(Revision) as A FROM logTable2_mylog Where
Working_File='./vtk/common/vtkAbstractTransform.cxx' OR
Working_File='./vtk/common/vtkActor2D.cxx' OR
Working_File='./vtk/common/vtkActor2DCollection.cxx' OR
Working_File='./vtk/common/vtkAssemblyNode.cxx' GROUP
BY Revision.
Temporary Table

When the GO button is pressed, the data chosen from the selected files and dimension is stored in the table tempgraph. This table structure is altered according to dimensions chosen. Every time a dimension is chosen, its data type is returned to this table and an insert statement is written to store the data in the table. Here is an example of alter and insert table query.

```
ALTER TABLE tempgraph ALTER COLUMN Dimen1 text(255)
ALTER TABLE tempgraph ALTER COLUMN Dimen2 Date
ALTER TABLE tempgraph ALTER COLUMN Dimen3 Number
INSERT INTO tempgraph(Dimen1,Dimen2,Dimen3,Rev_wh,Author,File)
SELECT Revision,Date,COUNT(Revision),Rev_whole,Author,Working_File FROM logTable2_mylog WHERE(Date Between #1994-02-22# And #1996-10-09#)AND(Rev_whole Between '0001000100000000' And '0001001300000000')AND(Working_File = './vtk/common/vtkAttributeData.cxx' OR Working_File = './vtk/common/vtkBitArray.cxx' OR Working_File = './vtk/common/vtkByteSwap.cxx' OR Working_File = './vtk/common/vtkPolyDataSource.cxx' OR Working_File = './vtk/common/vtkPolygon.cxx' OR Working_File = './vtk/common/vtkPolyLine.cxx') GROUP BY Rev_whole,Revision,Date,Author,Working_File
```
The tempgraph table shows only the filtered values of the dimensions chosen. The values from these tables are used for plotting the graphs. Before the data is altered in this table, the existing data is cleared.

The author list box is used to filter information according to the authors selected. Each author is assigned a color associated with which the data displayed. Depending on the user’s choice, single or multiple authors can be selected for visualization.

Fit to screen option is also provided at the bottom.
CHAPTER 5

Results

In this chapter, the results of visualization are discussed and several examples of the tool are shown. The tool was tested with logfile1.txt which consisted of 67 files. Due to space constraints, only a few snapshots of the tool are shown in this report. This chapter also contain illustrations of the key features in this tool.

Figure 5.1 shows the scatter plot of revision between 1.1 and 1.15, date between 1997-12-05 and 2000-04-25 and revisioncnt with none of the authors selected from author list. Following query gives the result for figure 5.1,

```
SELECT Revision,Date,COUNT(Revision),Rev_whole,Author,Working_File FROM logTable2_mylog WHERE(Date Between '#1997-12-05#' And '#2000-04-28#')AND(Rev_whole Between '0001000100000000' And '0001001500000000')AND(Working_File = './vtk/common/vtkProp.cxx' OR Working_File = './vtk/common/vtkPropAssembly.cxx' OR Working_File = './vtk/common/vtkPropCollection.cxx' OR Working_File = './vtk/common/vtkProperty2D.cxx') GROUP BY Rev_whole,Revision,Date,Author,Working_File;
```
Figure 5.1 An example of a scatter plot using revision as dimension 1, date as dimension 2 and revisioncnt as dimension 3.

The above figure shows scatter plot of all the changes made on that particular revision and on that particular date. Since none of authors are selected from author list, it shows all the authors in the plot with default red colored oval. The file numbers are shown on top of the oval indicating the files to which the changes belong.

Figure 5.2 shows a line graph of revision between 1.1 and 1.24, date between 1994-03-03 and 2000-10-05 and authorcnt with none of the authors selected from the author list. The following query gives the result for figure 5.2,
SELECT Revision, Date, COUNT(Time), Rev_whole, Author, Working_File
FROM logTable2_mylog WHERE (Date Between #1994-03-03# And #2001-08-10#) AND (Rev_whole Between '0001000100000000' And '0001002400000000') AND (Working_File = './vtk/common/vtkAssemblyPath.cxx' OR Working_File = './vtk/common/vtkBitArray.cxx' OR Working_File = './vtk/common/vtkPolyLine.cxx' OR Working_File = './vtk/common/vtkProp.cxx') GROUP BY Rev_whole, Revision, Date, Author, Working_File;

Figure 5.2 An example of a line graph showing revision as dimension 1, date as dimension 2 and authorcnt as dimension 3.
The above figure shows a line graph of particular revision made on a particular date with timecnt as third dimension. Each plot starts with a number written on it and this number corresponds to the working file associated with it. All changes that have the same file number are joined together by a line and a common color is given to it to show that it belongs to the same file. Since none of authors are selected from author list, it shows the all the authors in the plot with default red colored oval.

Figure 5.3 shows a scatter plot of revision between 1.1 and 1.49, time between 00:00 and 20:00 and authorcnt with avila, biddi and childs selected as authors from author list. Following query gives the result for figure 5.3,

```
SELECT Revision, Time, COUNT(Author), Rev_whole, Author, Working_File
FROM logTable2_mylog
```
Figure 5.3 An example of a scatter plot graph showing revision as dimension 1, time as dimension 2 and authorcnt as dimension 3 with avila, biddi, childs selected as authors.

Figure 5.3 shows plot of revisions made on a specific time of the day. In this plot even though some of the authors are selected in author list, still the plot shows only
the default red colored ovals. It is because the selected authors did not have any changes made on that time of the day. Here again the number displayed on top of the oval shows the file number associated with it. Key panel shows the color associated with the author. The color in the panel corresponds to a unique author.

Figure 5.4 shows bar graph of time between 00:00 and 13:00 as dimension 1, datecnt as dimension 2 and entire list of authors selected from author list. Following query gives the result for figure 5.4,

```sql
SELECT Time,COUNT(Revision),Rev_whole,Author FROM logTable2_mylog WHERE (Time Between #00:00#And#13:00#)AND(Working_File='./vtk/common/vtkAssemblyPaths.cxx' OR Working_File='./vtk/common/vtkAttributeData.cxx' OR Working_File='./vtk/common/vtkBitArray.cxx' OR Working_File='./vtk/common/vtkByteSwap.cxx' OR Working_File='./vtk/common/vtkPolyDataSource.cxx' )GROUP BY Time,Author,Rev_whole
```
Figure 5.4 An example of a bar graph showing time as dimension 1 and revisioncnt as dimension 2 with a list of authors selected.

Figure 5.4 shows an example of a bar graph showing number of revisions made on a particular time of the day. Since all the authors are selected, each bar is shown in a different color. Each color corresponds to a specific author. The height of the bar shows the number of revisions made by the author at that time. Key panel shows the color associated with the author. Each color in the panel corresponds to an unique author.

Figure 5.5 shows a bar graph with author as dimension 1 and count of revision made by these authors across various files as dimension 2. The following query gives the result for figure 5.5,
SELECT Author, COUNT(Revision), Rev_whole, Author FROM logTable2_mylog WHERE (Author Between 'avila' And 'will') AND (Working_File = './vtk/common/vtkAssemblyPaths.cxx' OR Working_File = './vtk/common/vtkPolyDataSource.cxx' OR Working_File = './vtk/common/vtkPriorityQueue.cxx') GROUP BY Author, Rev_whole;

Figure 5.5 An example of a bar graph showing author as dimension 1 and revisioncnt as dimension 2.

Figure 5.5 shows the number of revisions made by each author in the selected list of files. The default black color of the bars is shown in this figure as it was not needed to show the bars in different colors. The bars are shown in the default color because the plot in itself shows which author made the changes.
Figure 5.6 shows a line graph with **date** between 1999/01/10 and 2001/06/10 as dimension 1, **time** between 00:00 and 23:00 as dimension 2 and the **authorcnt** as dimension 3 with the entire list of authors selected from the author list. Following query gives the result for figure 5.6,

```
SELECT Date, Time, COUNT(Author), 0, Author, Working_File FROM logTable2_mylog WHERE (Date Between #1999-01-07# And #2001-07-12#) AND (Time Between #00:00# And #23:00#) AND (Working_File = './vtk/common/vtkPyramid.cxx' OR Working_File = './vtk/common/vtkQuadric.cxx' ) GROUP BY Date, Time, Author, Working_File;
```

Figure 5.6 Line graph showing **date** as dimension 1, **time** as dimension 2 and **authorcnt** as dimension 3 with list of authors selected.
The above figure depicts a line graph that shows the distribution of the changes that takes place during the different times of the day. This graph looks like a distribution graph. Numbers displayed at the beginning of the plot refer to the file number and the colored ovals specify the author associated with that change. Key panel shows the color associated with the author. Each color in the panel corresponds to a unique author.

Figure 5.7 shows an error message when the user tries to choose more than five files in case of line graph. It was decided to restrict the number of files in case of line graph to make sure that visualization does not look congested and plot can be seen clearly.
Figure 5.8 A snapshot of fit to screen mode.

Figure 5.8 shows an example of fit to screen option. In this example, a scatter plot graph is drawn using date as dimension 1, time as dimension 2, revisioncnt as dimension 3 and lymbdemo, martink, millerjv and schroede as selected authors. Key panel shows the key associated with the author.
CHAPTER 6

CONCLUSION

In this project, a tool called Visual CVS that will visualize CVS log files has been implemented. The tool tracked CVS log files and the changes were visualized in terms of scatter plot graphs, line graphs and bar graphs. The data was filtered using possible variables like revision, data, time and author. Various interaction techniques like fit to screen, coloring according to particular author were also implemented. The tool was tested with a log file having large number of files. All possible combinations were tested in each and every type of graph and the desired results were obtained.

FUTURE WORK

Here are some of the possible future enhancements to the tool,

• Currently, the project does not handle overlapping of coordinates. When two points share the same coordinates, it just displays one point on top of another. So, a better algorithm can be implemented to overcome this issue.

• In its present implementation, the project works only on one set of CVS log files. It may be enhanced in the future to handle more sets of log files.

• This whole application can be made dynamic. Log files can be linked dynamically to the database so that as soon as the data comes from the file it can be stored in the database and visualized immediately.

• Interaction techniques like zooming, panning can be implemented in the project to view the visual representation in an interactive way.