Visualization of Statistical Contents

Raja Majid Mehmood
&
Gulraiz Iqbal
Abstract
Our project presents the research on visualization of statistical contents. Here we will introduce the concepts of visualization, software quality metrics and proposed visualization technique (line chart). Our aim to study the existing visualization techniques for visualization of software metrics and then proposed the visualization approach that is more time efficient and easy to perceive by viewer.

In this project, we focus on the practical aspects of visualization of multiple projects with respect to the versions and metrics. This project also gives an implementation of proposed visualization techniques of software metrics. In this research based work, we have to compare practically the proposed visualization approaches. We will discuss the software development life cycle of our proposed visualization system, and we will also describe the complete software implementation of implemented software.

Keywords: Software Visualization, Software Metrics, Charts (line chart), Statistical Contents, Quality Metrics.
Acknowledgement
First of all we are very thankful to our supervisor Prof. Welf Lowe for introducing us to software visualization and for all the help along the way. We have a sincere thanks to Mr. Rudiger Lincke for the many discussions, advice and proof-readings. Many thanks are also directed to our teacher Dr. Jonas Lundberg for great collaboration and many interesting discussions during our Master degree in Software Technology. Thanks as well to all of our colleagues and friends at Vaxjo University whose support through all the ups and downs has been so very much appreciated. Finally, we would like to thank our family for always supporting and encouraging us to make our studies possible.
Table of Contents

1. Introduction........................................................................................................... 1
   1.1. Problem Description ...................................................................................... 1
   1.2. Goals ............................................................................................................. 1
   1.3. Objectives....................................................................................................... 2
   1.4. Background .................................................................................................... 2
       1.4.1. Software Visualization ........................................................................... 2
       1.4.2. Software Metrics .................................................................................... 2
       1.4.3. Why Visualization Can Help ................................................................. 3
       1.4.4. Statistical Contents .................................................................................. 3
   1.5. Outline ............................................................................................................ 4
2. Visualization and Proposed Techniques ................................................................. 5
   2.1. Related Work .................................................................................................. 5
       2.1.1. Charts ..................................................................................................... 5
       2.1.2. Glyphs .................................................................................................... 6
       2.1.3. Java Open Source Charting Tools .......................................................... 6
   2.2. Visualization Approaches .............................................................................. 7
       2.2.1. Line Charts ............................................................................................... 7
       2.2.2. 2D Environment ....................................................................................... 8
       2.2.3. Proposed Line Chart Visualization ....................................................... 8
   2.3. Summary ......................................................................................................... 8
3. Requirement Analysis and Design ......................................................................... 9
   3.1. Methodology .................................................................................................. 9
       3.1.1. Iterative and Incremental Process Model ............................................. 9
   3.2. Requirements ................................................................................................. 11
       3.2.1. Functional Requirements ....................................................................... 11
       3.2.2. Non-Functional Requirements ............................................................. 13
       3.2.3. Use Case Diagram .................................................................................. 15
   3.3. Test Cases ...................................................................................................... 15
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.1.</td>
<td>Software Complexity</td>
<td>16</td>
</tr>
<tr>
<td>3.3.2.</td>
<td>Architecture and Structure</td>
<td>20</td>
</tr>
<tr>
<td>3.3.3.</td>
<td>Design Guidelines</td>
<td>23</td>
</tr>
<tr>
<td>3.4.</td>
<td>Database Schema</td>
<td>23</td>
</tr>
<tr>
<td>3.5.</td>
<td>Summary</td>
<td>26</td>
</tr>
<tr>
<td>4.</td>
<td>Implementation</td>
<td>27</td>
</tr>
<tr>
<td>4.1.</td>
<td>Eclipse Development Environment</td>
<td>27</td>
</tr>
<tr>
<td>4.2.</td>
<td>AWT</td>
<td>28</td>
</tr>
<tr>
<td>4.3.</td>
<td>Swing</td>
<td>28</td>
</tr>
<tr>
<td>4.4.</td>
<td>Applets</td>
<td>29</td>
</tr>
<tr>
<td>4.5.</td>
<td>Visualization System Components</td>
<td>29</td>
</tr>
<tr>
<td>4.6.</td>
<td>Software Development</td>
<td>30</td>
</tr>
<tr>
<td>4.6.1.</td>
<td>Database Class Diagram</td>
<td>31</td>
</tr>
<tr>
<td>4.6.2.</td>
<td>Visualization Class diagram</td>
<td>32</td>
</tr>
<tr>
<td>4.6.3.</td>
<td>User Interface Class Diagram</td>
<td>33</td>
</tr>
<tr>
<td>4.6.4.</td>
<td>Color Manager</td>
<td>34</td>
</tr>
<tr>
<td>4.6.5.</td>
<td>CV2DChartPanel</td>
<td>35</td>
</tr>
<tr>
<td>4.6.6.</td>
<td>CVApplet</td>
<td>37</td>
</tr>
<tr>
<td>4.6.7.</td>
<td>CVColorDialog</td>
<td>38</td>
</tr>
<tr>
<td>4.6.8.</td>
<td>CVD</td>
<td>39</td>
</tr>
<tr>
<td>4.6.9.</td>
<td>CVFormComponent</td>
<td>41</td>
</tr>
<tr>
<td>4.6.10.</td>
<td>CVMetricComponent</td>
<td>42</td>
</tr>
<tr>
<td>4.6.11.</td>
<td>CVProjectComponent</td>
<td>43</td>
</tr>
<tr>
<td>4.6.12.</td>
<td>CVTabs</td>
<td>44</td>
</tr>
<tr>
<td>4.6.13.</td>
<td>CVToolBar</td>
<td>45</td>
</tr>
<tr>
<td>4.6.14.</td>
<td>ReadTextFile</td>
<td>47</td>
</tr>
<tr>
<td>4.6.15.</td>
<td>ReadToolTextFile</td>
<td>48</td>
</tr>
<tr>
<td>4.6.16.</td>
<td>WriteTextFile</td>
<td>49</td>
</tr>
<tr>
<td>4.6.17.</td>
<td>CheckNode</td>
<td>50</td>
</tr>
<tr>
<td>4.6.18.</td>
<td>CheckNodeTree</td>
<td>51</td>
</tr>
</tbody>
</table>
4.6.19. CheckRenderer ................................................................. 52
4.6.20. Collaboration Diagram of Visualization System .................. 54
4.6.21. Collaboration Diagram of Tool Input File ....................... 55
4.6.22. Collaboration Diagram of Metrics Input File ..................... 56

5. User Interaction and System Dialogs ......................................... 57
  5.1. Main Controller Window ..................................................... 57
  5.2. Upload Tool Information .................................................... 57
  5.3. Upload Metric Data ............................................................ 59
  5.4. Make Selection ................................................................. 60
  5.5. Exporting Metric Data ....................................................... 62
  5.6. Data Points ................................................................. 63
  5.7. Zoom Out ................................................................. 63
  5.8. Export Drawing as Image .................................................. 64
  5.9. Print Drawing ............................................................... 64

6. Case Study ............................................................................. 66
  6.1. Case Study A ................................................................. 66
  6.2. Case Study B ................................................................. 66
  6.3. Case Study C ................................................................. 67
  6.4. Case Study D ................................................................. 67
  6.5. Case Study E ................................................................. 68
  6.6. Case Study F ................................................................. 69
  6.7. Summary ................................................................. 69

7. Conclusion .............................................................................. 70
  7.1. Results ........................................................................... 70
    7.1.1. Graphical Representation ........................................... 70
    7.1.2. Visualization Techniques ........................................... 70
    7.1.3. Database Interface .................................................. 71
    7.1.4. Web Application ....................................................... 72
    7.1.5. Utilities ............................................................... 72
  7.2. Future Work .................................................................. 72
List of Tables

Table 3.1. Functional requirements ............................................................... 13
Table 3.2. Non-Functional requirements ......................................................... 14
Table 3.3. Table tabanalysistools Fields ....................................................... 24
Table 3.4. Table tabanalysistoolversions Fields ............................................ 24
Table 3.5. Table tabanalysistype Fields ......................................................... 24
Table 3.6. Table tabmetrics Fields ............................................................... 24
Table 3.7. Table tabclasses Fields ............................................................... 25
Table 3.8. Table tabprojects Fields ............................................................. 26
Table 3.9. Table tabprojectversions Fields ................................................... 26
Table 4.1. List of Classes ........................................................................... 30
Table 4.2. Class Description: ColorManager .............................................. 35
Table 4.3. Class Description: CV2DChartPanel ........................................... 37
Table 4.4. Class Description: CVApplet ..................................................... 38
Table 4.5. Class Description: CVColorDialog ............................................ 39
Table 4.6. Class Description: CVDb ......................................................... 40
Table 4.7. Class Description: CVFormComponent ...................................... 42
Table 4.8. Class Description: CVMetricComponent ................................... 42
Table 4.9. Class Description: CVProjectComponent .................................. 44
Table 4.10. Class Description: CVTabs ...................................................... 44
Table 4.11. Class Description: CVToolBar ............................................... 46
Table 4.12. Class Description: ReadTextFile .............................................. 48
Table 4.13. Class Description: ReadTooltextFile ...................................... 49
Table 4.14. Class Description: WriteTextFile ............................................. 50
Table 4.15. Class Description: CheckNode ............................................... 51
Table 4.16. Class Description: CheckNodeTree ......................................... 52
Table 4.17. Class Description: CheckRenderer ......................................... 53
List of Figures

Figure 2.1. Proposed Visualization: Chart Visualization.................................. 8
Figure 3.1. System Development Life cycle [28]........................................... 9
Figure 3.2. Overview of Iterative and Incremental Model [28]......................... 11
Figure 3.3. Use Case Diagram..................................................................... 15
Figure 3.4. Test Case: Visualizing complexity of different projects by LOC.. 16
Figure 3.5. Test Case: Visualizing complexity of different projects by NOM 18
Figure 3.6. Test Case: Visualizing complexity of different projects by WMC 19
Figure 3.7. Test Case: Visualizing structural analysis of projects by DIT ...... 20
Figure 3.8. Test Case: Visualizing architectural analysis by CBO................ 21
Figure 3.9. Test Case: Visualizing architectural analysis by LCOM ............ 22
Figure 3.10. Test Case: Visualizing architectural analysis by LOD.......... 23
Figure 4.1. Eclipse Platform Structure[34] ............................................... 27
Figure 4.2. Components of Visualization System ....................................... 29
Figure 4.3. Class Diagram: Database Interaction System............................ 31
Figure 4.4. Class Diagram: Visualization System ........................................ 32
Figure 4.5. Class Diagram: Web Application ............................................ 33
Figure 4.6. ColorManager: Class Design Diagram ....................................... 34
Figure 4.7. CV2DChartPanel: Class Design Diagram ................................. 35
Figure 4.8. CVApplet: Class Design Diagram ........................................... 37
Figure 4.9. CVColorDialog: Class Design Diagram .................................... 38
Figure 4.10. CVDb (Database Handler): Class Design Diagram ................... 39
Figure 4.11. CVFormComponent: Class Design Diagram........................... 41
Figure 4.12. CVMetricComponent: Class Design Diagram.......................... 42
Figure 4.13. CVProjectComponent: Class Design Diagram........................ 43
Figure 4.14. CVTabs (Tabs Controller): Class Design Diagram .................. 44
Figure 4.15. CVToolBar (Tool Bars): Class Design Diagram......................... 45
Figure 4.16. ReadTextFile: Class Design Diagram .................................... 47
Figure 4.17. ReadToolTextFile: Class Design Diagram............................... 48
Figure 4.18. WriteTextFile: Class Design Diagram .................................... 49
Figure 4.19. CheckNode: Class Design Diagram....................................... 50
Figure 4.20. CheckNodeTree: Class Design Diagram .................................. 51
Figure 4.21. CheckRenderer: Class Design Diagram .................................... 52
Figure 4.22. Collaboration Diagram: Visualization System ......................... 54
Figure 4.23. Collaboration Diagram: ReadToolTextFile............................. 55
Figure 4.24. Collaboration Diagram: Read Metrics Dataset ......................... 56
Figure 5.1. Screen Shorts: Main Window ................................................... 57
Figure 5.2. Screen Shorts: Three steps to store tool data.......................... 58
Figure 5.3. Screen Shorts: Tool File Format ............................................. 58
Figure 5.4. Screen Shorts: Metric data format .......................................... 59
Figure 5.5. Screen Shorts: Three steps to store metrics data....................... 59
Figure 5.6. Screen Shorts: Main Form ...................................................... 60
Figure 5.7. Screen Shorts: List of Projects and Metrics ............................... 60
Figure 5.8. Screen Shorts: Make selection for visualization....................... 61
Figure 5.9. Screen Shorts: Visualization result .............................................. 61
Figure 5.10. Screen Short: Exporting Metric Data ........................................ 62
Figure 5.11. Screen Short: New directory .................................................... 62
Figure 5.12. Screen Short: Data Point ......................................................... 63
Figure 5.13. Screen Short: Zooming data point ........................................... 63
Figure 5.14. Screen Short: Zooming result of selected point ......................... 64
Figure 5.15. Screen Short: Export chart as a image format ............................ 64
Figure 5.16. Screen Short: Print chart ......................................................... 65
Figure 6.1. Case Study-A: Visualizing LOC, CYC Metrics ........................... 66
Figure 6.2. Case Study-B: Visualizing LOC, CYC Metrics ........................... 67
Figure 6.3. Case Study-C: Visualizing LOC, CYC Metrics ........................... 67
Figure 6.4. Case Study-D: Visualizing LOC, CYC Metrics ........................... 68
Figure 6.5. Case Study-E: Average data points of LOC and CYC ................. 68
Figure 6.6. Case Study-F: Visualizing text glyph LOC, CYC Metrics ........... 69
1. Introduction
Systems can contain a big amount of software entities that allows users to visually analyze, explore and compare a statistical data of different projects. Software visualization tools are used by software designers to raise the level of abstraction and reduce the amount of information to the one needed. Mostly these are different projects versions, data can be import and export which force the user to switch between different windows and contexts. The development of software visualization frameworks is a significant step to bring visualization tools in the forward engineering process [1], [2] and [3].

1.1. Problem Description
Bulk amount of data is produced due to the rapid advances in computer technology. Mainly, the data comes from statistical data gathering, automated measurements and simulations, and this will become the primary source for a large dataset, which contains a large number of items, variables and time steps. It is very difficult to display a huge number of attributes in a small graphical representation. Our target is to design visualization application that is useful to display as much as possible software metrics within small drawing canvas with high quality display and time efficiency.

In our work, we have to visualize the statistical contents using proposed technique, which contains either too many data objects or attributes for a visualization to be efficiently carried out using a propose techniques, is referred to as being large. Depending on the application area, efficient can have different meanings. In general, representations that enable efficient information visualization are those which convey information about data in a clear and interpretable way, and in a reasonable time. Therefore, we need to develop an effective (time efficient, low memory consumption, accuracy) approach to visualize software metrics for a large statistical content. We will try to prove the efficiency of our proposed application in test scenarios, experiments and conclusion, describing in later chapters.

1.2. Goals
This project presents a technique that allows viewers to visually analyze, explore, and compare a statistical data of different projects. We present an algorithm that visualizes data along 2D-Charts. We use simple graphical objects that vary in color and placement properties to represent the attribute values contained in a data element. When shown together, the glyphs form visual patterns that support exploration, facilitate discovery of data characteristics, and highlight trends and exceptions.
1.3. Objectives
In the following we identify some important objectives for our research:

- Design graphical representation of statistical contents using $2D$-Graphics.
- To implement a visualization technique that is more user friendly and time efficient for a large dataset.
- Develop a Database-Interface that provides the communication between online Visualization-Tool and Database.
- Construct the new dataset by computing the average values of all attributes of each project.
- Develop the Web-Based visualization tool on client-side.
- Develop the server-side tool to upload the metrics data of projects along their version information into database.

Our primary objective in this project is to design a graphical glyph that supports flexibility in its placement and in its ability to represent statistical data objects. We focus to accommodate a high number of attributes by a small graphical representation.

The main part of this project is to develop a software tool that can accept any type of multivariate raw data, compute statistically and represent it graphically. Our software is able to read the dataset from some external data source; it can be some file system.

1.4. Background
The background of our research includes a brief explanation of software visualization, software metrics and measuring the statistical contents and also why we use visualization in computer science and other fields.

1.4.1. Software Visualization
Software Visualization either static or dynamic (animated) 2D or 3D is used to visual representations of information about software systems based on their structure, size, or behavior. The information used for visualization is software metric data from measurement activities differently with different behaviors. The actual objectives of software visualizations are to support the understanding of software systems (i.e., its structure) and algorithms (e.g., by animating the behavior of sorting algorithms) [4] as well as the analysis of software systems and their anomalies (e.g., by showing classes with height coupling) [7], [5], [6].

Software Visualization uses graphical representation to show code, control flows, classes, data, objects and dependencies among them. Therefore the programmer and software designer would have software visualization instead of lines of code displayed in a textual editor [3].

1.4.2. Software Metrics
Software metric is standard of measurement. Software metric computes property of a piece of software or its specifications. Since quantitative methods have been proved so powerful in other sciences, computer science practitioners and
theoreticians have worked hard to bring similar approaches to software development. Tom DeMarco stated, “You can’t control what you can’t measure”. There are countless examples of software metrics: like lines of code, cyclomatic complexity, function point analysis, bugs per line of code, code coverage, number of lines of customer requirements, number of classes and interfaces, Robert Cecil Martin’s software package metrics cohesion or coupling [8]. Management methodologies, such as the Capability Maturity Model or ISO 9000, have therefore focused more on process metrics which assist in monitoring and controlling the processes that produce the software [8].

1.4.3. Why Visualization Can Help
Visualization is an area that presents data in a visual form to facilitate rapid, effective, and meaningful analysis and interpretation. Example application domains include scientific simulations, land and satellite weather information, geographic information systems, and molecular biology [10]. Visualization is also used in more abstract settings, for example, software engineering, data mining, and network security. A key challenge is designing visualizations that are effective for the user’s data and analysis tasks [9].

The Software Visualization approach constructs visual representations that harness the strengths of the low-level human visual system. These perceptual visualizations display the data in ways that allow items of interest to capture the user’s focus of attention [10].

It is a very critical situation to analyze the large size of the average dataset. The desire to extract knowledge efficiently motivates the need for an effective visualization system. A dataset’s size is made up of three related properties:
- Number of elements stored in the dataset
- Number of attributes represented within the dataset
- Range of values possible for each attribute

Visualization tools are used to visualize these properties with effective representation (easy to perceive by viewer). These properties are represented in visualization known as information content and it is a combination of following properties:
- Number of elements
- Number of attribute values per element
- Range of different attribute values being visualized

The new technique described in this project seeks to increase information content by focusing on the last two properties, dimensionality and range [11].

1.4.4. Statistical Contents
In this thesis, we have projects with different versions, like Table 1.1 shows different classes and their metrics like LOC, WMC, TCC, NAM, NOM, MPC, LOD and different values against each Class, similarly Table 1.2, Table 1.3 shows the same project but as an improved version of the project.
### Table 1.1: [Version-1 of Sample Project] Project.V1

<table>
<thead>
<tr>
<th>Classes</th>
<th>LOC</th>
<th>WMC</th>
<th>TCC</th>
<th>NAM</th>
<th>NOM</th>
<th>MPC</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>1.2</td>
<td>1.5</td>
<td>1.2</td>
<td>2.5</td>
<td>2.6</td>
<td>2.8</td>
<td>3</td>
</tr>
<tr>
<td>Print</td>
<td>0.99</td>
<td>1.42</td>
<td>1.8</td>
<td>2.3</td>
<td>2.5</td>
<td>2.6</td>
<td>2.9</td>
</tr>
<tr>
<td>Compute</td>
<td>0.6</td>
<td>1.3</td>
<td>1.4</td>
<td>2</td>
<td>2.1</td>
<td>2.2</td>
<td>2.6</td>
</tr>
</tbody>
</table>

### Table 1.2: [Version-2 of Sample Project] Project.V2

<table>
<thead>
<tr>
<th>Classes</th>
<th>LOC</th>
<th>WMC</th>
<th>TCC</th>
<th>NAM</th>
<th>NOM</th>
<th>MPC</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>1</td>
<td>1.5</td>
<td>1</td>
<td>2.5</td>
<td>2.6</td>
<td>2.8</td>
<td>3</td>
</tr>
<tr>
<td>Print</td>
<td>0.99</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2.6</td>
<td>1</td>
</tr>
<tr>
<td>Compute</td>
<td>0.6</td>
<td>1.3</td>
<td>1.4</td>
<td>2</td>
<td>2.1</td>
<td>1</td>
<td>2.6</td>
</tr>
</tbody>
</table>

### Table 1.3: [Version-3 of Sample Project] Project.V3

<table>
<thead>
<tr>
<th>Classes</th>
<th>LOC</th>
<th>WMC</th>
<th>TCC</th>
<th>NAM</th>
<th>NOM</th>
<th>MPC</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>1.2</td>
<td>1.5</td>
<td>1</td>
<td>2.5</td>
<td>2.6</td>
<td>2.8</td>
<td>3</td>
</tr>
<tr>
<td>Print</td>
<td>0.99</td>
<td>2</td>
<td>1.8</td>
<td>1</td>
<td>2.5</td>
<td>2.6</td>
<td>2</td>
</tr>
<tr>
<td>Compute</td>
<td>2</td>
<td>1.3</td>
<td>1.4</td>
<td>2</td>
<td>2.1</td>
<td>1</td>
<td>2.6</td>
</tr>
</tbody>
</table>

1.5. **Outline**

Chapter 2 is about research work already done related to visualization of statistical contents of software projects. Chapter 3 contains the research and development methodology has been used in this project and analysis and design of the proposed system which also includes function and nonfunctional requirements, use case diagram of our visualization system. It also includes the test cases and database design of our application. In Chapter 4, we describe our implemented visualization techniques. It contains detailed overview of our software interaction and utilities too. Chapter 5 is complete explanation of our software implementation. Chapter 6 contains some case studies related to our research. Chapter 7 describes our conclusion and future work related to this research work.
2. Visualization and Proposed Techniques
This chapter includes some relevant research and literature covering the different visualization techniques in general. We also used a book of Colin Ware [12] that covers the many topics of information visualization (texture, glyph, animation).

2.1. Related Work
We will discuss previous work in short that is directly related to our proposed techniques, and highlight areas where we may possibly able to offer improvements over existing methods.

2.1.1. Charts
Charts are used to visually represent data, in which data is represented by symbols such as bar chart or line in a line chart [13]. A chart can represent tabular numeric data, functions or some type of quantitative structures [14].

The term “chart” has multiple meanings.
- Data chart is a type of diagram or graph that organizes a set of numerical data.
- Maps that are adorned with extra information (nautical chart or aeronautical chart) for a specific purpose is also known as charts.
- Music notation chart or record charts for album popularity, which are domain specific, are also known as charts.

Charts are used to ease understanding of large quantity of data and representing the relationship between those data. Charts can be read and understandable more than raw data. Charts are used in wide variety of application either they are design by hand or through computer applications.

Feature of Charts
Chart covers a large variety of forms. Some common features of charts having ability to extract meaning from data. A chart is a graphical, containing very little text like displaying values, title and description about the axis data [14].

Dimensions in the data are often displayed on axes. Horizontal and vertical axis are commonly used and are referred to as the X-axis and Y-axis respectively which are scale properly in intervals and also accompanied some suitable labels describing the dimension represented [13].

In charts grid can be used to align data into lines and is enhance able to a regular graduations. Chart data can be appear in different manner, it can be visualize without any proper labels displaying the contents, It may appear in dots or other shape, different patterns, colors, connected or unconnected line [14].

Data appearing in a chart contains multiple data sets, which visualize the difference and association among them [14].
Types of Charts

*Most common types of charts are as follows*

- **Histogram**: Shows the quantity of point falls within numeric ranges
- **Pie Chart**: Shows percentage values as a slice
- **Bar Chart**: Show values for different categories
- **Line Chart**: A two dimensional scatter plot of ordered observations

*Less common types of charts are as follows*

- **Bubble chart**: Two dimensional scatter plot where point size the represent the third value
- **Polar Area Diagram**: An enhanced form of pie
- **Radar Chart**: Three or more quantitative variables represented on axes starting from the same point
- **Water fall Chart**: Special type of floating-column chart
- **Candlestick Chart**: It is also a type of bar chart which describes the movements of prices over time
- **Kagi Chart**: Time independent stock tracking chart
- **Spakline Chart**: This kind of chart is used, where less detail required and chart size is paramount [14]

2.1.2. **Glyphs**

A Glyph is a graphical unit used to visualize the data of any type. It is a graphical symbol that provides appearance to data or form a shape for an existing data. A glyph may have properties of orientation, scaling, translation, deformation, size, placement, and color, etc. These properties are used to represents the input data. Glyphs are prejudiced by attributes of the current dataset. Different glyphs based visualization techniques are discussed here.

Chernoff developed iconic representation of the human face [15], [16]. The face elements like nose, eyes, eyebrows, mouth, and jowls are changed according to input data values. Foley and Ribarsky have created a visualization tool called Glyphmaker that is used to create visualization of multivariate datasets [17]. This tool uses a glyph editor and a glyph binder to create glyphs, to arrange them spatially, and to bind attributes to their visual properties. Levkowitz represented a visualization of colored squares to produce patterns to represent multivariate datasets [18].

In our thesis we find Glyphs are most suitable to visualize software metrics. Where they used to improve the visualization for a larger dataset, for example, scaling shows the attribute strength, or coloring to identify attribute or objects.

2.1.3. **Java Open Source Charting Tools**

In this thesis, we explore different charting and reporting tools in Java and few of them are mention below, which facilitate different visualization techniques and there features [30].
**JFreeChart**
It is a free Java class library used to generate chart like Pie Charts (2D and 3D), bar charts (regular and stacked, with an optional 3D effect), line and area charts, scatter plots and bubble charts, time series, high/low/open/close charts and candle stick charts, combination charts, pareto charts, gantt charts, wind plots, meter charts and symbol charts, wafer map charts [30], [31].

**Prefuse**
Java based toolkit to build interactive information visualization application. It supports different set of features for data modeling, visualization, and interaction. It provides optimized data structures for tables, graphs, and trees, a host of layout and visual encoding techniques, and support for animation, dynamic queries, integrated search, and database connectivity [30], [31].

**JOpenChart**
It is an open source Java library and toolkit for creating different kinds of charts and embedding them into web applications or Swing applications. [30]

**Chart2D**
Chart2D is a library written in Java for adding 2D charts to Java programs [30].

**JChart2D**
A Java Swing Widget (JComponent) for precise runtime dynamic display tupels in form of a stripe chart. Key future of JChart2D is minimal configuration effort, automatic scaling and labeling, thread safeness, a clean and extendible API and extensive documentation [30].

### 2.2. Visualization Approaches
In our research work, our visualization approach based over the user requirements and to meet the requirements we have studied few existing software visualization techniques to display a multivariate dataset. We found some of existing visualization tools that have been used as a different type of visualization features like, texture, animation, charts and flow visualization in 2D or 3D environment [23] [24].

#### 2.2.1. Line Charts
Line graph or chart is created by connecting a series of data points together with a line. Line graph or charts are a great choice and commonly used for data visualization. A line chart is often used to visualize a trend in data over intervals of time [22]. It is easy to understand and is great way to compare different data series which are change able with time. Especially it is a nice when comparing two or more datasets. Different options are available in line chart or graph to display/setting up information, we can set them up at x-axis and y-axis relate to version or values [20] [21] [22].
2.2.2. 2D Environment
We analyzed 3D visualization [25], [26] that is more complex and expensive to render the resulting image or display. We therefore concluded to use 2D (Two Dimensional) environment for the visualization tool to display software statistical contents.

2.2.3. Proposed Line Chart Visualization
After analysis and discussion of different related topics, we consider the following computational and visualization approach in this project.

![Proposed Visualization: Chart Visualization](image)

**Figure 2.1. Proposed Visualization: Chart Visualization**

In the graphical representation, shown in Figure 2.1, we have a 2D-Chart that contains a number of Poly Lines with Circle Objects. Where, each Line is drawn with help of different metric values, and it represents with unique Color (where, Color of line represents a single Project), and each Circle object identifies the single Software Metric.

2.3. Summary
In this chapter, we studied and discuss different visualization techniques by different researchers and their opinions about different techniques. After comprehensive research of software visualization, we reached at a point where we are able to finalize our proposed techniques (Section 2.2.3) for visualization of statistical contents. Our next step is to analyze the requirements of proposed system and then we will design the proposed visualization tool.
3. Requirement Analysis and Design
In this chapter, we will discuss and describe the development methodology and requirements of our proposed system and interaction between user and system, using UML diagrams.

3.1. Methodology
This section includes software development methodology which we have used in our research work. After considering the nature of our project and goal we have to attain, we chose Iterative and Incremental Development Methodology to develop our software.

3.1.1. Iterative and Incremental Process Model
In an iterative and incremental lifecycle (Figure 3.1), is a rework strategy in which various part of the system is revised and improved at different rates, with the intention for software project to understand the system it is building [29]. Iterative development is a rework scheduling strategy in which time is set aside to revise and improve parts of the system. It does not assume incremental development, but it works well with it. The basic idea behind iterative enhancement is to develop a software system incrementally, allowing the developer to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system.

![Figure 3.1. System Development Life cycle [28]](image)

The Rational Objectory Process defines the control for an iterative and incremental lifecycle. It defines an extensive set of guidelines that address the technical aspects of software development focusing on requirements analysis and design [30].

“Objectory is an iterative process. Given today’s sophisticated software systems, it is not possible to sequentially first define the entire problem, design the entire solution, build the software and then test the product at the end. An
iterative approach is required that allows an increasing understanding of the problem through successive refinements, and to incrementally grow an effective solution over multiple iterations. An iterative approach gives better flexibility in accommodating new requirements or tactical changes in business objectives, and allows the project to identify and resolve risks earlier” [30].

This process is structured along two dimensions:

- time and,
- process

The Time dimension is structured as follow:

- **Inception:** It is a specification of the project vision. In this phase, we have to focus on the actual problem, requirements gathering, and research work.
- **Elaboration:** Planning the necessary activities and required resources; specifying the features and designing the architecture.
- **Construction:** Building the product as a series of incremental iterations.
- **Transition:** Supplying the product to the user community (manufacturing, delivering, and training).

The Process dimension includes the following activities:

- **Project Management:** It includes planning of project development, development cases, description and guidelines.
- **Requirements:** Requirement specification of proposed system.
- **Analysis & design:** Analyze and describe the main functionalities of the proposed system, and also design the system for implementation phase.
- **Implementation:** Code generation and programming that will result in an executable system.
- **Test:** Verification of the entire system.
- **Deployment:** Deliver to end user in an operating environment.

Each activity of the process component dimension is applied to the each phase of the time based dimension. *Figure 3.2* shows how the process components are applied to each time based phase [28], [29], [30].
As we described the life cycle and the Rational Objectory Process of the Iterative and Incremental process model, we used the same development methodology in our research project. Similar to the proposed model, we

- get the requirements at first,
- analyze the old visualization research of software metrics,
- design the architecture of proposed system,
- develop the code, and
- test the entire program

All above steps are described in this report in detail. In the next section, we will define the requirement specification and use cases, which help us further to design the proposed system [30].

### 3.2. Requirements

Requirements analysis is the first stage in the systems engineering process and software development process [33]. A requirement specification helps us to define the scope and boundaries of the proposed system which leads to a better understanding of overall system. Two types of requirements: functional requirements and non-functional requirements.

#### 3.2.1. Functional Requirements

Now, we are having a view over the functional requirements of proposed system. It defines the main functions or components of a software system, and each of them includes the required function of the proposed system, its importance, and a brief description.
<table>
<thead>
<tr>
<th>FR-01</th>
<th>Security Certificate Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>To access applet user must accept security certificate authentication, so that applet can upload in browser.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FR-02</th>
<th>Execute The Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>User must run application to interact with system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FR-03</th>
<th>Upload Tool Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>Upload tool data before uploading metric data. Tool</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FR-04</th>
<th>Upload Metric Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>Upload metric data which contain project, its version information and metric information (LOC, NOM… WMC).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FR-05</th>
<th>Enter Criteria and Display Project / Metric list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>Enter information which kind of analysis tool, its version and type of analyze like class level, method level and package level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FR-06</th>
<th>Display Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>To visualize software metrics and projects in charts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FR-07</th>
<th>Zoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>Zoom in or Zoom out to view clearly all metrics and projects more clearly.</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FR-08</td>
<td>Print Drawing</td>
</tr>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>Printing drawing on paper, Print Drawing button allows printing through printers.</td>
</tr>
<tr>
<td>FR-9</td>
<td>Export Drawing Image</td>
</tr>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>Save drawing as an image file, for later use.</td>
</tr>
<tr>
<td>FR-10</td>
<td>Download Drawing Data</td>
</tr>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>Download drawing data (metrics) in a text file.</td>
</tr>
<tr>
<td>FR-11</td>
<td>Mouse Over On Data Points Display Metric</td>
</tr>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>On mouse over data points it display project, its version and metric information</td>
</tr>
</tbody>
</table>

Table 3.1. Functional requirements

3.2.2. Non-Functional Requirements
In this section, we will describe and discuss non-functional requirements with their importance and brief description, which also describe as usability, reliability, performance, and substitutability requirements.

<table>
<thead>
<tr>
<th>N-FR-01</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>The system should work fast.</td>
</tr>
<tr>
<td>N-FR-02</td>
<td>Efficient Display of Statistical Contents</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>To compare and manipulate the metric values in currently displaying data objects.</td>
</tr>
<tr>
<td>N-FR-03</td>
<td>Efficient Data structure</td>
</tr>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>Data structure must be efficient to get fast results.</td>
</tr>
<tr>
<td>N-FR-04</td>
<td>Application Maintenance</td>
</tr>
<tr>
<td>Importance</td>
<td>Essential</td>
</tr>
<tr>
<td>Description</td>
<td>Application can be maintainable, well documented.</td>
</tr>
<tr>
<td>N-FR-05</td>
<td>Application Scalability</td>
</tr>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>The system should be scalable. It should be helpful for developers to add the new features and functionalities.</td>
</tr>
<tr>
<td>N-FR-06</td>
<td>Platform Independent</td>
</tr>
<tr>
<td>Importance</td>
<td>Desirable</td>
</tr>
<tr>
<td>Description</td>
<td>The system should work on all computer machines.</td>
</tr>
</tbody>
</table>

*Table 3.2. Non-Functional requirements*
3.2.3. Use Case Diagram

![Use Case Diagram](image)

*Figure 3.3. Use Case Diagram*

3.3. Test Cases

In this section, we will discuss about our test scenario, how the contents can be visualize in a useful way? We are displaying metrics values in charts, where X-axis displays the Scaled Project Versions and Y-axis displays Scaled Metric values, where circles with different colors displays metric with respect to version. Our System allows to draw conclusions about software application related standards, conventions, and regulations in laws and similar prescription. Metrics are classified as supporting mainly statements on,

- Software Complexity,
- Software Architecture and Structure, and
- Software Design and Coding.
3.3.1. Software Complexity
Complexity metrics refer to the static, i.e., structural size and complexity of software. Complexity of software entities can be assessed by the following properties.

- Size
- Interface Complexity
- Structural Complexity

Here, we can analyze the complexity of given software entity through,

**LOC (Size):**
LOC is used to count the line of source code. LOC is a powerful software metrics entity to compute / assess the complexity of software entities. It is an important measure; “More lines of code mean more bugs” debugging is the most expensive and time consuming part of the software development. Here one can assess complexity with respect to size.

![LOC Chart](image)

**Figure 3.4. Test Case: Visualizing complexity of different projects by LOC**

As shown in Figure 3.4, displaying LOC in a chart by using software metric LOC, used to compute the complexity of software entities by their versions orders. Three software project (*Jedit, PMD, Xerces2*) having about 50 versions are shown in Figure 3.4, where the trend of line to upwards displays the increasing size of the source code (Complex) for the corresponding version and trend to tend to down displays the decreasing size of the source code.
Analysis by Visualizing the Size Complexity of given Projects
User can access and analyze the common software qualities through our system, which ensure the different characteristics Maintainability, Re-Usability, reliability and efficiency of a software system. Here, we will describe our analysis of three projects PMD, JEdit and Xerces2 of Figure 3.4.

- **Functionality:** User can assess the functionality of the projects with respect to their versions as PMD is having less functionality then other projects and also different versions have higher and lower functionality. “Might increase with increasing LOC”.

- **Reliability:** displays three projects PMD, JEdit and Xerces2, User can analyze the reliability of the projects with respect to their versions as PMD might be less reliable then other projects and also different versions might be higher and lower reliability. “Might increase with increasing LOC”.

- **Re-Usability:** User can analyze the re-usability of the projects with respect to their versions as PMD might be highly re-usable then other projects and also different versions might be higher and lower re-usability. “Might decrease with increasing LOC”.

- **Efficiency:** User can analyze the efficiency of the projects with respect to their versions as PMD is highly efficient then other projects and also different versions might be higher and lower efficiency. “Decline with increasing LOC”.

- **Maintainability:** User can analyze the maintainability of the projects with respect to their versions as PMD is highly maintainable then other projects and also different versions might be higher and lower maintainability. “Decline with increasing LOC”.

- **Portability:** User can analyze the portability of the projects with respect to their versions as PMD is highly portable then other projects and also different versions might be higher and lower portability. “Decline with increasing LOC”.


**NOM (Interface Complexity):**
We can compute interface complexity through NOM. Number of local Methods measures the number of methods locally declared in a class, where inherited methods are not included. It is the size of the interface of a class and allows conclusions on its complexity.

![Figure 3.5. Test Case: Visualizing complexity of different projects by NOM](image)

The above properties well describes the results asses from the Figure 3.5.

**Analysis by Visualizing the Interface Complexity of given Projects**
- **Functionality:** jEdit and PMD are less functional then Xerces2.
- **Reliability:** jEdit and PMD are less reliable then Xerces2.
- **Efficiency:** jEdit and PMD are more efficient then Xerces2.
- **Maintainability:** jEdit and PMD are easy to maintain then Xerces2.
**WMC (Structure Complexity):**
Assessing Structure complexity using Weighted Method Count (WMC), WMC is the sum of the McCabe's Cyclomatic Complexity values of all methods in the package.

![Graph](image_url)

**Figure 3.6. Test Case: Visualizing complexity of different projects by WMC**

**Analysis by Visualizing the Structural Complexity of given Projects**
Above properties defines the system behavior about the structure complexity.

- **Functionality:** Xerces2 is more functional than the jEdit and PMD
- **Reliability:** Xerces2 is more reliable than the jEdit and PMD
- **Efficiency:** Xerces2 is less efficient than the jEdit and PMD
- **Maintainability:** Xerces2 is difficult to maintain than the jEdit and PMD
3.3.2. Architecture and Structure:

- Inheritance
- Coupling
- Cohesion

**DIT (Inheritance):**
Depth of Inheritance Tree measures how many super classes can affect a class, which is applicable in Object Oriented Systems. It is said that classes with big depth of inheritance tree (DIT) metric are difficult to understand and to maintain.

![Figure 3.7. Test Case: Visualizing structural analysis of projects by DIT](image)

As shown in Figure 3.7, Displaying DIT in a chart by using software metric DIT, used to compute the complexity of software entities by their versions orders. Four software project (*Jedit, PMD, Xerces2 and tomcat*) having about 50 versions are shown in Figure 3.7.

**Analysis by Visualizing the Inheritance Architecture of given Projects**
Above given properties well define the inheritance.

- **Functionality:** Large number of Xerces2 versions are more functional the other software projects.
- **Reliability:** As we can see in Figure 3.7, project Xerces2 is more fault tolerance and recoverable then other projects but maturity level of Xerces2 is less then other projects.
- **Efficiency:** Xerces2 is less efficient then other software projects.
• **Maintainability:** Xerces2 is difficult to maintain than other Software projects in the figure.

**CBO (Coupling):**
Coupling between Objects is the number of other classes that a class is coupled to. CBO is only applicable to object-oriented systems.

![Coupling Architecture Diagram](image)

**Figure 3.8. Test Case: Visualizing architectural analysis by CBO**

**Analysis by Visualizing the Coupling Architecture of given Projects**
Above given properties are well define the coupling.

- **Functionality:** As we can see in **Figure 3.8**, project jEdit operate-ability is lesser but security is higher.
- **Reliability:** jEdit is reliable than other project in the figure.
- **Re-Usability:** jEdit is easy to re-us-able than other project in the figure.
- **Efficiency:** jEdit is more efficient than other project in the figure.
- **Maintainability:** jEdit is easy to maintain than other project in the figure.
**LCOM (Cohesion):**
The Lack of Cohesion in Methods metric is a measure for the number of not connected method pairs in a class representing independent parts having no cohesion. It represents the difference between the number of method pairs not having instance variables in common, and the number of method pairs having common instance variables.

![Diagram]

**Figure 3.9. Test Case: Visualizing architectural analysis by LCOM**

**Analysis by Visualizing the Cohesion Architecture of given Projects**
- **Re-Usability**: PMD project some version are more re-usable than other project versions, from scaled project version 2 they are less re-usable than other projects due to increase of LCOM.
- **Efficiency**: PMD project some version are more efficient than other project versions, from scaled project version 2 they are less efficient than other projects due to increase of LCOM.
- **Portability**: In PMD project, some versions are more portable than other project versions, from scaled project version ‘2’ they are discourage portability than other projects due to increase of LCOM.
3.3.3. Design Guidelines

**LOD:**
How many comments are lacking in a class, considering one class comment and a comment per method as optimum. Structure and content of the comments are ignored.

![Figure 3.10. Test Case: Visualizing architectural analysis by LOD](image)

**Analysis by Visualizing the Design Guideline of given Projects**
*Figure 3.10,* present better assessment of Design guidelines of software projects.

- **Reliability:** Xerces2 is more reliable than other projects in figure.
- **Re-usability:** Xerces2 is more suitable to re-use then other projects.
- **Maintainability:** Xerces2 is easy to maintain then other projects in figure.

3.4. Database Schema
Tables below shows the database schema used for the raw data after information extraction. We have seven tables and our system supports to add more tables according for different type metrics. In this section we will describe and discuss our database schema, which describe relation, attribute, data type of attributes and association between relations. I will discuss database schema into two different parts Tij and Sij.
### Database Name: \textit{statistics}

#### Table: tabanalysistools

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>NULL</th>
<th>Default</th>
<th>Extras</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool_ID</td>
<td>int(10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telephone</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 3.3. Table tabanalysistools Fields*

#### Table: tabanalysistoolversions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>NULL</th>
<th>Default</th>
<th>Extras</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool_Versio n_ID</td>
<td>int(10)</td>
<td></td>
<td></td>
<td>Auto increment</td>
<td>PK</td>
</tr>
<tr>
<td>Tool_ID</td>
<td>int(10)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td>FK</td>
</tr>
<tr>
<td>Version</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 3.4. Table tabanalysistoolversions Fields*

#### Table: tabanalysistype

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>NULL</th>
<th>Default</th>
<th>Extras</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type_ID</td>
<td>int(10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type_Name</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table_Name</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 3.5. Table tabanalysistype Fields*

#### Table: tabmetrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>NULL</th>
<th>Default</th>
<th>Extras</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric_ID</td>
<td>int(10)</td>
<td></td>
<td></td>
<td>Auto increment</td>
<td>PK</td>
</tr>
<tr>
<td>Type_ID</td>
<td>int(10)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td>FK</td>
</tr>
<tr>
<td>Tool_Version_ID</td>
<td>int(10)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td>FK</td>
</tr>
<tr>
<td>Name</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default_Value</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 3.6. Table tabmetrics Fields*

Table 3.3 to 3.10 displays the different tables belongs to tool and keep raw data of a tool and analysis type. Tij, with i tool and j version are stored in table
tabanalysistoolversion. Tij (Mm,An), with metric are stored in a table tabmetrics. Table typeanslysisstype store information which type of analysis either class, method level or other and in Table tabmetrics it defines the software quality model supports a list of metrics {LOC,WMC,TCC….WMC} or some different metrics. Database is more general and supportive to include a new Software quality model.

Table: tabclasses

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>NULL</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class_ID</td>
<td>int(10)</td>
<td>NULL</td>
<td>PK</td>
<td></td>
</tr>
<tr>
<td>Project_Version_ID</td>
<td>int(10)</td>
<td>NULL</td>
<td>NULL</td>
<td>FK</td>
</tr>
<tr>
<td>FullQualifiedName</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>ClassName</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_CBO</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_CYC_Classes</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_DAC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_DIT</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_ILCOM</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_LCOM</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_LD</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_LEN</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_LOC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_LOD_Class</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_MPC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_NAM</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_NOC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_NOM</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_RFC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_TCC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CMM_1_0_WMC</td>
<td>Double</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.7. Table tabclasses Fields

Table: tabprojects

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>NULL</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project_ID</td>
<td>int(10)</td>
<td>NULL</td>
<td>PK</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Unix_Name</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Project_Admins</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Topic</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>User_Interfaces</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Translations</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Programming_</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
Table 3.8. Table tabprojects Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Operation_System</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>License</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Intended_Audience</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Development_Status</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Database_Environment</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Bugs</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Patches</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Feature_Requests</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>To_Do</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Backports</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Public_Forums</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Mailing_List</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CVS</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>SVN</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Date_Stamp</td>
<td>varchar(255)</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.9. Table tabprojectversions Fields

Table 3.11 to 3.16 displays the different tables belongs to System and keep raw data of a system. $S_{ij}$, with $i$ System and $j$ version are stored in table tabprojectversion. The version string is not always sufficient to bring the systems into a correct timely order; therefore we have an additional integer attribute versionorder.

Table tabclasses contains separate analysis one or more datasets for every class $C_{ij,k}$ of a system. The attributes correspond to the absolute metrics values $M_l$ ($C_{ij,k}$) extracted for the metrics $l \in \{\text{Maintainability, LOC,} \ldots \text{WMC}\}$ as defined in software quality model.

3.5. Summary
In this Chapter, we defined the functional and nonfunctional requirements which help us to design the use cases and also database. We included database information and its design as well.
4. Implementation
In this chapter, we will discuss about our implementation aspects, tools, major components and environment in which codes are written.

4.1. Eclipse Development Environment
Eclipse provides a common user interface (UI) model for working with tools. It is designed to run on multiple operating systems while providing robust integration with each underlying OS. Plug-ins can program to the Eclipse portable APIs and run unchanged on any of the supported operating systems [34].

At the core of Eclipse is architecture for dynamic discovery, loading, and running of plug-ins. The platform handles the logistics of finding and running the right code. The platform UI provides a standard user navigation model. Each plug-in can then focus on doing a small number of tasks well [34].

As Eclipse platform has shown in Figure 4.1, it includes the Eclipse Java Development Tools, offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards [35].
4.2. **AWT**
The Standard Widget Toolkit (SWT) is a graphical widget toolkit used for the Java platform. It was developed by IBM and now Eclipse Foundation is maintained for Eclipse IDE [36]. IBM has created a new GUI library, called SWT, which solves the problems seen with the AWT and the Swing frameworks. The SWT framework accesses native widgets through JNI. If a widget is not available on the host platform, SWT emulates the unavailable widget [36] [37].

4.3. **Swing**
Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT).

**Platform Independence:** Swing is platform independent both in terms of its expression (Java) and its implementation [36] [37] [38].

**Extensibility:** Swing is a highly partitioned architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces. Users can provide their own custom implementation(s) of these components to override the default implementations. Swing users can extend the framework by extending existing (framework) classes or providing alternative implementations of core components to modify components according to their requirements [36] [37] [38].

**Component Oriented:** Swing is a component-based framework. The distinction between objects and components is a fairly subtle point, concisely; a component is a well-behaved object with a known/specified characteristic pattern of behavior. Swing objects asynchronously fire events, have "bound" properties, and respond to a well-known set of commands (specific to the component.) Specifically, Swing components are Java Beans components, compliant with the Java Beans Component Architecture specifications [36] [37] [38].

**Customizable:** Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible in Swing. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a "border", "inset", decorations, etc. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific Borders, Colors, Backgrounds, opacities, etc., as the properties of that component. The core component will then use this property (settings) to determine the appropriate renderers to use in painting its various aspects. However, it is also completely possible to create unique GUI controls with highly customized visual representation [36] [37] [38].

**Configurable:** Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at runtime to fundamental changes in its settings. For example, a Swing-based application can change its look and feel at
runtime. Further, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code [36] [37] [38].

Light Weight User Interface: Swing's configurability is a result of a choice not to use the native host OS's GUI controls for displaying itself. Swing "paints" its controls programmatically through the use of Java 2D APIs, rather than calling into a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics APIs [36] [37] [38].

4.4. Applets
An Applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page. When you use a Java technology-enabled browser to view a page that contains an Applet, the applet's code is transferred to your system and executed by the browser's Java Virtual Machine (JVM) [39][40]. Swing provides a special subclass of Applet, called javax.swing.JApplet, which should be used for all applets that use Swing components to construct their GUIs. [40]

4.5. Visualization System Components
In this section, we will discuss some major components of our implemented visualization system. In Figure 4.2, we can see the main components linked with visualization system,

Figure 4.2. Components of Visualization System
We have used a Java Applet in our visualization system for Web Users. JAVA core API’s are used as a main language of our software development. Database is main source of data for our software; we have used MySQL database as an input source (dataset) for our developed visualization tool (CV2DChart).

### 4.6. Software Development
Our software development contains number of classes like,

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorManager</td>
<td>This class is to generate the color code for each attribute, and it is manage by Map, we can also access the color of each attribute directly by calling getColor()</td>
</tr>
<tr>
<td>CV2DChartPanel</td>
<td>This class is used to get information to display chart</td>
</tr>
<tr>
<td>CVApplet</td>
<td>Main client-side, visualization class extends JApplet placed all visual components in this class, and it provides all visual interaction for web user.</td>
</tr>
<tr>
<td>CVColorDialog</td>
<td>Choosing colors with different way to visualize metrics information</td>
</tr>
<tr>
<td>CVDb</td>
<td>This Class contains different methods to connect with MySql Database and as well different methods to perform DML and DDL queries and return result set</td>
</tr>
<tr>
<td>CVFormComponents</td>
<td>Contains all form component, Which will be use to interact with users like Toolbar, JPanel, Combobox and Text etc</td>
</tr>
<tr>
<td>CVMetricComponent</td>
<td>Contains JPanel, adding pan, trees displaying projects and metrics</td>
</tr>
<tr>
<td>CVProjectComponent</td>
<td>Maintain Project information</td>
</tr>
<tr>
<td>CVTabs</td>
<td>Designing Tabbed Pane and adding tabs</td>
</tr>
<tr>
<td>CVToolBar</td>
<td>Design toolbar adding buttons and action listener for those buttons</td>
</tr>
<tr>
<td>ReadTextFile</td>
<td>Read Metrics Text File and Form Query to insert data into database</td>
</tr>
<tr>
<td>ReadToolTextFile</td>
<td>Read tool text file and form query to update database</td>
</tr>
<tr>
<td>WriteTextFile</td>
<td>Write Text file exporting metrics data from database to a text file over local system.</td>
</tr>
<tr>
<td>CheckNode</td>
<td>Nodes containing check boxes</td>
</tr>
<tr>
<td>CheckNodeTree</td>
<td>A tree containing nodes with check boxes</td>
</tr>
<tr>
<td>CheckRenderer</td>
<td>Managing and manipulating values tree layout and settings</td>
</tr>
</tbody>
</table>

*Table 4.1. List of Classes*
4.6.1. Database Class Diagram

*Figure 4.3* describes the database interaction from different objects of visualization system.

*Figure 4.3. Class Diagram: Database Interaction System*
4.6.2. Visualization Class diagram
The class diagram in Figure 4.4, describes the complete interaction of visualization system with other objects of implemented application.
4.6.3. User Interface Class Diagram
This class diagram in Figure 4.5, is explaining the Web Application Interface of implemented visualization application.

Figure 4.5. Class Diagram: Web Application
4.6.4. Color Manager
This class is used to generate the color code for each attribute, and it is manage by Map, we can also access the color of each attribute directly by calling getColor().

![ColorManager Diagram]

Figure 4.6. ColorManager: Class Design Diagram

client

Class ColorManager

```java
public class ColorManager extends java.lang.Object {
    // Field Definitions
    Map<Integer, Color> metricsColorMap;
    Map<Integer, Color> projectsColorMap;

    // Constructor
    public ColorManager() {
        // Default constructor
    }

    // Method Definitions
    public void createMetricsColorMapExperimentA(Set<Integer> set) {
        // Random color generation
    }

    public void createMetricsColorMapExperimentB(Set<Integer> set) {
        // This method allows the computation of Experiment-B
    }

    public void createMetricsColorMapExperimentC(Set<Integer> set) {
        // This method allows the computation of Experiment-C
    }
}
```

Field Summary
- metricsColorMap
  - Contains set of metric ids along their colors code
- projectsColorMap
  - Contains set of project ids along their colors code

Constructor Summary
- ColorManager()
  - Default constructor

Method Summary
- createMetricsColorMapExperimentA(Set<Integer> set)
  - Random color generation
- createMetricsColorMapExperimentB(Set<Integer> set)
  - This method allows the computation of Experiment-B
- createMetricsColorMapExperimentC(Set<Integer> set)
  - This method allows the computation of Experiment-C
<table>
<thead>
<tr>
<th>Void</th>
<th>createMetricsColorMapExperimentD(Set&lt;Integer&gt; set)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method allows the computation of Experiment-D</td>
</tr>
<tr>
<td>Void</td>
<td>createProjectsColorMap(Set&lt;Integer&gt; set)</td>
</tr>
<tr>
<td></td>
<td>get the attributes list from MainController class, and generate the unique color code for each attribute</td>
</tr>
<tr>
<td>Color</td>
<td>getMetricColor(Integer metricId)</td>
</tr>
<tr>
<td></td>
<td>to get the color of Metric</td>
</tr>
<tr>
<td>Color</td>
<td>getProjectColor(Integer projectId)</td>
</tr>
<tr>
<td></td>
<td>to get the color of Project</td>
</tr>
</tbody>
</table>

*Table 4.2. Class Description: ColorManager*

### 4.6.5. *CV2DChartPanel*

This class is used to display line charts (visualization).

```java
CV2DChartPanel

△ analysis_type: String
△ analysis_type_id: Integer
△ analysis_table: String
△ chart: ZoomableChart
△ cm: ColorManager
△ experimentType: String
△ frame: JPanel
△ metricsMap: Map<Integer, String>
△ projectList: Map<Integer, List<Integer>>
△ projectsMap: Map<Integer, LinkedHashSet<Integer>>
△ tool_id: Integer
△ tool_name: String
△ tool_version: String
△ tool_version_id: Integer
```

*Figure 4.7. CV2DChartPanel: Class Design Diagram*
## Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>analysis_type</td>
<td>store analysis type</td>
</tr>
<tr>
<td>Integer</td>
<td>analysis_type_id</td>
<td>store analysis type id</td>
</tr>
<tr>
<td>String</td>
<td>analysis_table</td>
<td>store analysis type table</td>
</tr>
<tr>
<td>ZoomableChart</td>
<td>chart</td>
<td>object of zoomable chart class</td>
</tr>
<tr>
<td>ColorManager</td>
<td>cm</td>
<td>object of color manager class</td>
</tr>
<tr>
<td>String</td>
<td>experimentType</td>
<td>store experiment type</td>
</tr>
<tr>
<td>JPanel</td>
<td>frame</td>
<td>JPane Object</td>
</tr>
<tr>
<td>Map&lt;Integer,String&gt;</td>
<td>metricsMap</td>
<td>maintain metric list</td>
</tr>
<tr>
<td>Map&lt;Integer,List&lt;Integer&gt;&gt;</td>
<td>projectList</td>
<td>maintain projects list</td>
</tr>
<tr>
<td>Map&lt;Integer,LinkedHashSet&lt;Integer&gt;&gt;</td>
<td>projectsMap</td>
<td>maintain project and version list</td>
</tr>
<tr>
<td>Integer</td>
<td>tool_id</td>
<td>store tool id</td>
</tr>
<tr>
<td>String</td>
<td>tool_name</td>
<td>store tool name</td>
</tr>
<tr>
<td>String</td>
<td>tool_version</td>
<td>store tool version</td>
</tr>
<tr>
<td>Integer</td>
<td>tool_version_id</td>
<td>store tool version id</td>
</tr>
</tbody>
</table>

## Constructor Summary

```java
CV2DChartPanel(String experimentType)
```

default constructor
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void addVisualizationMethod1()</td>
<td>this method is used for, Experiment: A, B, D, F</td>
</tr>
<tr>
<td>Void addVisualizationMethod2()</td>
<td>this method is used for, Experiment: C, E</td>
</tr>
<tr>
<td>Integer getAnalysis_type_id()</td>
<td>get the type id of analysis (either class, method, package, etc...) used for this visualization</td>
</tr>
<tr>
<td>String getAnalysis_type()</td>
<td>get the analysis type</td>
</tr>
<tr>
<td>String getAnlysis_table()</td>
<td>get the name of analysis table in database</td>
</tr>
<tr>
<td>Map&lt;Integer,String&gt; getMetricsMap()</td>
<td>get the map of metrics id's with their names</td>
</tr>
<tr>
<td>Map&lt;Integer,LinkedHashSet&lt;Integer&gt;&gt; getProjectsMap()</td>
<td>get the map&gt; of selected projects with their versions list</td>
</tr>
<tr>
<td>Integer getTool_id()</td>
<td>get the analysis tool id of this visualization chart</td>
</tr>
<tr>
<td>String getTool_name()</td>
<td>get the name of analysis tool</td>
</tr>
<tr>
<td>Integer getTool_version_id()</td>
<td>get the version number of analysis tool</td>
</tr>
<tr>
<td>String getTool_version()</td>
<td>get the version name of analysis tool</td>
</tr>
</tbody>
</table>

*Table 4.3. Class Description: CV2DChartPanel*

#### 4.6.6. CVApplet

It is an applet class use to instantiate the application, manage application with web browser. Main client-side, visualization class extends JApplet placed all visual components in this class, and it provides all visual interaction for web user.

*Figure 4.8. CVApplet: Class Design Diagram*
Class CVApplet

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Panel
        java.applet.Applet
          javax.swing.JApplet
            client.CVApplet

Field Summary
protected JPanel frame

Constructor Summary
CVApplet()
Default Constructor

Method Summary
Void init()
initializing the application

Table 4.4. Class Description: CVApplet

4.6.7. CVColorDialog
Helper class is displaying the Metrics and Projects Color information.

Figure 4.9. CVColorDialog: Class Design Diagram

Class CVColorDialog
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          client.CVColorDialog
public class CVColorDialog
extends javax.swing.JPanel
implements javax.swing.ListCellRenderer

Field Summary
protected Color m_c
    color’s information
Protected JLabel m_l
    label object

Constructor Summary
CVColorDialog()
default constructor

Method Summary
Component getListCellRendererComponent(JList list, Object obj, int row, boolean sel, boolean hasFocus)
Void paint(Graphics g)
    Graph drawing

Table 4.5. Class Description: CVColorDialog

4.6.8. CVDb
This Class contains different methods to connect with MySql Database and as well different methods to perform DML and DDL queries. This Class contains different methods to connect with MySql Database and as well different methods to perform DML and DDL queries and return result set.

Figure 4.10. CVDb (Database Handler): Class Design Diagram
### Class Description: CVDb

**public class CVDb**

extends java.lang.Object

#### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection</td>
<td>MySql connection object</td>
</tr>
<tr>
<td>database</td>
<td>store database name</td>
</tr>
<tr>
<td>password</td>
<td>store password</td>
</tr>
<tr>
<td>user</td>
<td>store user name</td>
</tr>
</tbody>
</table>

#### Constructor Summary

**CVDb(String database, String user, String password)**

Establish a connection passing database name, user name and password

#### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect()</td>
<td>Establish connection with database</td>
</tr>
<tr>
<td>disconnect()</td>
<td>Disconnecting with database</td>
</tr>
<tr>
<td>queryDDL(String)</td>
<td>Passing DDL query and returns true if successfully</td>
</tr>
<tr>
<td>queryDML(String)</td>
<td>Passing DML Query and returns a result set</td>
</tr>
</tbody>
</table>

*Table 4.6.*
4.6.9. CVFormComponent
Contains all form component, which will be use to interact with users like Toolbar, JPanel, Combobox and Text etc.

![CVFormComponent Class Design Diagram](image)

**Figure 4.11. CVFormComponent: Class Design Diagram**

### Class CVFormComponents
```
java.lang.Object
  \- client.CVFormComponents

public class CVFormComponents
  extends java.lang.Object
```

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>db</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVDb</td>
<td>Object for class cvdb</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVFormComponents()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPanel</td>
</tr>
<tr>
<td>designForm()</td>
</tr>
<tr>
<td>designing overall components over a form</td>
</tr>
<tr>
<td>static Integer</td>
</tr>
<tr>
<td>retrieving analysis type id from the table</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>retrieving analysis type name</td>
</tr>
<tr>
<td>static String</td>
</tr>
<tr>
<td>retrieving analysis type name from the table</td>
</tr>
<tr>
<td>static Integer</td>
</tr>
<tr>
<td>retrieving tool id from tool table</td>
</tr>
</tbody>
</table>
### Table 4.7. Class Description: CVFormComponent

#### 4.6.10. CVMetricComponent
Contains JPanel, adding pan, trees displaying projects and metrics

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client</td>
<td>CVMetricComponent</td>
</tr>
<tr>
<td>java.lang.Object</td>
<td>client.CVMetricComponent</td>
</tr>
</tbody>
</table>

#### Constructor Summary

**CVMetricComponent(JPanel panel)**

#### Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td>addPane()</td>
</tr>
<tr>
<td></td>
<td>Adding pane</td>
</tr>
<tr>
<td>static Map&lt;Integer,String&gt;</td>
<td>getSelectedMetrics()</td>
</tr>
<tr>
<td></td>
<td>this method returns the selected metrics with metric-id</td>
</tr>
<tr>
<td>static void</td>
<td>removePane()</td>
</tr>
<tr>
<td></td>
<td>remove Metrics tree</td>
</tr>
</tbody>
</table>

### Table 4.8. Class Description: CVMetricComponent
4.6.11. CVProjectComponent

Maintain Project information

![Class Design Diagram](image)

**Figure 4.13. CVProjectComponent: Class Design Diagram**

client

Class CVProjectComponent

```java
java.lang.Object
    client.CVProjectComponent
```

```java
public class CVProjectComponent
    extends java.lang.Object
```

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVProjectComponent(JPanel panel)</td>
<td><strong>addPane()</strong></td>
<td>Adding pane</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static void getProjectName(Integer projectId)</strong></td>
<td>Get project name from the project table</td>
</tr>
<tr>
<td><strong>static String getProjectVersionName(Integer projectVersionId)</strong></td>
<td>Get project version name from table project version</td>
</tr>
<tr>
<td><strong>static Integer getProjectVersionOrderNumber(Integer projectVersionId)</strong></td>
<td>Get project version number</td>
</tr>
<tr>
<td>Field Summary</td>
<td>Method Summary</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>static javax.swing.JTabbedPane pane</td>
<td>Void designTabs()</td>
</tr>
<tr>
<td>static CVFormComponents startTab</td>
<td>Adding tabs and labels</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVTabs()</td>
</tr>
</tbody>
</table>

4.6.12. CVTabs
Designing Tabbed Pane and adding tabs in this class.

Table 4.9. Class Description: CVProjectComponent

Table 4.10. Class Description: CVTabs
4.6.13. CVToolBar
In this class, design toolbar adding buttons and action listener for those buttons.

![Class Design Diagram](image_url)

**Figure 4.15. CVToolBar (Tool Bars): Class Design Diagram**

**Class CVToolBar**
java.lang.Object
  - java.awt.Component
    - java.awt.Container
      - javax.swing.JComponent
        - javax.swing.JPanel
          - client.CVToolBar

**Direct Known Subclasses:**

```
public class CVToolBar
extends javax.swing.JPanel
implements java.awt.event.ActionListener
```
### Nested Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>CVToolBar.ToolBarA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>CVToolBar.ToolBarB</td>
</tr>
<tr>
<td>Class</td>
<td>CVToolBar.ToolBarC</td>
</tr>
</tbody>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected int</td>
<td>int</td>
<td>buttonSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>store button size</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVToolBar()</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td>actionPerformed(ActionEvent arg0) event handler</td>
</tr>
<tr>
<td>CVToolBar</td>
<td>addToolObject(CV2DChartPanel chartData)</td>
</tr>
<tr>
<td></td>
<td>adding tool object chart etc</td>
</tr>
<tr>
<td>CVToolBar</td>
<td>addToolObject(int key)</td>
</tr>
<tr>
<td></td>
<td>adding toolbar objects</td>
</tr>
<tr>
<td>protected JButton</td>
<td>makeNavigationButton(String imageName,String actionCommand,String toolTipText,String altText)</td>
</tr>
<tr>
<td></td>
<td>handling navigation of JButton</td>
</tr>
</tbody>
</table>

*Table 4.11. Class Description: CVToolBar*
4.6.14. ReadTextFile
In this class, Read Metrics Text File to insert appropriate data into database.

```java
public class ReadTextFile
extends java.lang.Object
{
  // Field Summary
  Connection conn; // MySQL connection object
  List<Integer> countList; // List use to count
  CVDb Db; // Object of a CVDb class
  String projectID; // Store project id

  // Constructor Summary
  ReadTextFile();

  // Method Summary
  String buildFieldsString(String line); // Build Field String for Query eg.
  String buildValuesString(String line); // Build Value String for query eg.
}
```

Figure 4.16. ReadTextFile: Class Design Diagram
### Table 4.12. Class Description: ReadTextFile

#### 4.6.15. ReadToolTextFile

In this class, reading tool text file and form query to update database.

![ReadToolTextFile](image)

**Figure 4.17. ReadToolTextFile: Class Design Diagram**

```java
public class ReadToolTextFile

extends ReadTextFile
```
**Constructor Summary**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadToolTextFile()</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containRecord</td>
<td>Return true if a table contains record at given criteria else return false</td>
</tr>
<tr>
<td>getAnalysisTypeID</td>
<td>Return Type_ID if it contains else return null</td>
</tr>
<tr>
<td>getToolId</td>
<td>Return tool Id if it contains else return null</td>
</tr>
<tr>
<td>getToolVersionId</td>
<td>Return tool_version_id if it contains else return null</td>
</tr>
<tr>
<td>readToolFile</td>
<td>Read a text file from a given directory and update database while reading</td>
</tr>
</tbody>
</table>

Table 4.13. **Class Description: ReadToolTextFile**

**4.6.16. WriteTextFile**

In this class, writing Text file exporting metrics data from database to a text file over local system.

![WriteTextFile](image)

**Figure 4.18. WriteTextFile: Class Design Diagram**

```plaintext
client
Class WriteTextFile
    java.lang.Object
        └── client.WriteTextFile

extends java.lang.Object
```

**Field Summary**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>writer object</td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WriteTextFile(java.io.File file, CV2DChartPanel chartData)</code></td>
<td>create a text file in given directory and write tool, project and metrics information</td>
</tr>
</tbody>
</table>

*Table 4.14. Class Description: WriteTextFile*

4.6.17. CheckNode

This class, create check nodes in a tree list.

*Figure 4.19. CheckNode: Class Design Diagram*

```java
client.util.checkbox
Class CheckNode
java.lang.Object
   \--- javax.swing.tree.DefaultMutableTreeNode
   \--- client.util.checkbox.CheckNode

public class CheckNode
extends javax.swing.tree.DefaultMutableTreeNode
```

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>protected Boolean</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>protected int</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Constructor Summary

- **CheckNode()**
  Check Node keep the record of all nodes added as parent and child nodes
- **CheckNode(Object userObject, boolean allowsChildren, boolean isSelected)**
- **CheckNode(Object userObject, int id)**

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>getID()</td>
<td>Get id of the node</td>
</tr>
<tr>
<td>Int</td>
<td>getSelectionMode()</td>
<td>Get Selection mode</td>
</tr>
<tr>
<td>Boolean</td>
<td>isSelected()</td>
<td>Return true or false if a node is selected or not</td>
</tr>
<tr>
<td>Void</td>
<td>setSelected(boolean isSelected)</td>
<td>Setting a node to be selected</td>
</tr>
<tr>
<td>Void</td>
<td>setSelectionMode(int mode)</td>
<td>Set Selection Mode</td>
</tr>
</tbody>
</table>

*Table 4.15. Class Description: CheckNode*

### 4.6.18. CheckNodeTree
This class manages checkNode with tree check box system.

![CheckNodeTree](image)

*Figure 4.20. CheckNodeTree: Class Design Diagram*
public class CheckNodeTree
extends java.lang.Object

Nested Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckNodeTree.ButtonActionListener</td>
<td>Action listener for check node tree button</td>
</tr>
<tr>
<td>CheckNodeTree.NodeSelectionListener</td>
<td>Selection listener for check node tree</td>
</tr>
</tbody>
</table>

Constructor Summary

CheckNodeTree()

Method Summary

JScrollPane
    addNode(CheckNode root)
    Adding list of CheckNode to JTree

Table 4.16. Class Description: CheckNodeTree

4.6.19. CheckRenderer
This class manages and manipulates values tree layout and settings.

Figure 4.21. CheckRenderer: Class Design Diagram
### Nested Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>CheckRenderer.TreeLabel&lt;br&gt;Renderer tree label</th>
</tr>
</thead>
</table>

### Field Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected JCheckBox</td>
<td>check</td>
<td>JCheckBox object</td>
</tr>
<tr>
<td>protected CheckRenderer.TreeLabel</td>
<td>label</td>
<td>CheckRenderer tree label class</td>
</tr>
</tbody>
</table>

### Constructor Summary

**CheckRenderer()**

### Method Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td>doLayout()</td>
<td>Setting layout for label and check box</td>
</tr>
<tr>
<td>Dimension</td>
<td>getPreferredSize()</td>
<td>GetPreferredSize()</td>
</tr>
<tr>
<td>Component</td>
<td>getTreeCellRendererComponent(JTree tree, Object value, boolean isSelected, boolean expanded, boolean leaf, int row,boolean hasFocus)</td>
<td>Manipulating Tree and setting values and layout</td>
</tr>
<tr>
<td>Void</td>
<td>setBackground(Color color)</td>
<td>Setting background color</td>
</tr>
</tbody>
</table>

*Table 4.17. Class Description: CheckRenderer*
4.6.20. Collaboration Diagram of Visualization System

Figure 4.22. Collaboration Diagram: Visualization System

Figure 4.22 shows the internal implementation of components, drawing canvas and operation over drawing like zooming, exporting image and zoom options as well.
4.6.21. Collaboration Diagram of Tool Input File

Figure 4.23. Collaboration Diagram: ReadToolTextFile

Figure 4.23 shows the implementation of reading tool text file and building the fields and value to update database.
4.6.22. Collaboration Diagram of Metrics Input File

Figure 4.24. Collaboration Diagram: Read Metrics Dataset

Figure 4.24 shows the implementation of reading metric text file and building the fields and value to update database.
5. User Interaction and System Dialogs
In this Chapter, we will discuss how user can interact with system. These techniques allow the user to compare and analyze large projects having different improve versions. Below, we will explain our system with the help figure taken from the system.

5.1. Main Controller Window
This is the main interface for users to operate with the complete application. It contains a toolbar having buttons upload tool data and import metric data.

![Screen Shorts: Main Window](image)

*Figure 5.1. Screen Shorts: Main Window*

5.2. Upload Tool Information
The *Figure 5.2* represents format of our text file to import data from user end to server side which will be manage by DBMS. From start, we write some tag in following format `<.>`, to identify the sequence of data, second line represents fields that are identifiable by the system. The following text file displays the contents of a tool text file which is acceptable by the system.
Figure 5.2. Screen Shorts: Three steps to store tool data

To upload tool data, following steps are necessary to perform and follow the text file pattern given below. User must provide tool information as shown in a text file.

1. From main application toolbar, click on upload tool data button.
2. Select a text file containing tool data
3. Successfully upload message appears

Figure 5.3. Screen Shorts: Tool File Format
5.3. **Upload Metric Data**

![Image](image.png)

**Figure 5.4. Screen Shorts: Metric data format**

The *Figure 5.4* represents format of our text file to import *Metrics* data from user end. From start we write some tag in a < >, to identify the sequence of data, second line represents fields that are identifiable by the system.

Following steps are helpful to upload metric data into the system

1. From main application toolbar, click on upload Metric Data button.
2. Select a text file containing metric data
3. Successfully upload message appears

![Image](image.png)

**Figure 5.5. Screen Shorts: Three steps to store metrics data**
5.4. Make Selection
To visualize the project first of all user must enter some detail about tool and analysis type while selecting data from combo boxes on start pane. Clicking on next will provide some more details displayed in next figure.

![Main Form](image)

*Figure 5.6. Screen Shorts: Main Form*

After clicking next user will be able to see to check box trees, on left hand it displays number of projects available in information system and in other hand metrics supported by the selected tool is also available.

![List of Projects and Metrics](image)

*Figure 5.7. Screen Shorts: List of Projects and Metrics*
User must select the projects and metrics of his choice by clicking over the checkboxes in the tree. All check boxes can be select by clicking over the root check box. Project Metrics panel also contain a tool bar where only one button describing the visualizing.

Figure 5.8. Screen Shorts: Make selection for visualization

After clicking on visualization button, user can view the projects and metrics in form of 2D-Charts. As Figure 5.9 describes the best,

Figure 5.9. Screen Shorts: Visualization result
5.5. Exporting Metric Data

Exporting metric data of visualization (Figure 5.10), steps are written below

1. Click on the current visualization toolbar and click export drawing data button from toolbar.
2. Provide Path and folder name.

Figure 5.10. Screen Short: Exporting Metric Data

Figure 5.11. Screen Short: New directory
In Figure 5.11, jEdit (Project) metric data files display in explorer window starting with project id, and ending with version id.

5.6. Data Points
Visualization metric can be display in Figure 5.12,

![Figure 5.12. Screen Short: Data Point](image)

5.7. Zoom Out

![Figure 5.13. Screen Short: Zooming data point](image)
User can zoom out drawing by selecting specific area, it expands the selected area and user can visualized data points more clearly. It is useful when lines are much closer with each other.

Figure 5.14. Screen Short: Zooming result of selected point

To shrink drawing to its original size, click on zoom button available in toolbar as displayed in Figure 5.14.

5.8. **Export Drawing as Image**
User can save image data and as well it can also save current image while clicking on the button given in the toolbar.

Figure 5.15. Screen Short: Export chart as a image format

5.9. **Print Drawing**
This option allows user to get drawing hard copy, provides additional feature to support user and ease of the using the system.
Figure 5.16. Screen Short: Print chart

Summary
In this chapter, we discuss all possibilities how user can interact with the system and how he/she can visualize projects and there metrics information, which are supportive to assess and reach over a final decision.
6. Case Study
In this chapter we explain few possible scenarios, how we can visualize quality metrics value with respect to project version using different techniques and methods? As, all scenarios are based on 2D-Charts, where Average Value of software metric is along Y-Axis and Project Version is along X-Axis. Case studies A to F contains detail explanations.

6.1. Case Study A
In Case Study A, we have focus on colors for Metrics. Therefore we choose random color of circle (glyph object) to identify different Quality Metrics (LOC, NOM, WMC …).

![Figure 6.1. Case Study-A: Visualizing LOC, CYC Metrics](image)

Figure 6.1, displays project tomcat with two different metrics LOC, CYC. In this visualization each metric has different color. In this Case Study, colors of metrics look similar and it is very difficult to differentiate different metrics due to minor changes in their colors.

6.2. Case Study B
The main task in Case Study B is to display circle colors uniquely, as we used RGB colors, for example Red=1, then ‘1’ will never repeat again for Red next time.
6.3. Case Study C
In this Case Study, we compute the data points by calculating the average values of all selected Metrics.

6.4. Case Study D
In Case Study D, we try to avoid the colors that are almost same with minor difference. We are dealing with RGB colors in our code and to make some
difference among the colors. We reserve 10 from left and 10 from right to avoid similar color perception. For example, R=100 then we reserve R from 91 to 100 and 100 to 110.

Figure 6.4. Case Study-D: Visualizing LOC, CYC Metrics

The Figure 6.4 has better results than Case Study A and B. User can easily perceive the both Metrics by their Colors.

6.5. Case Study E

In Case Study E, we combined metrics by computing average value of selected metrics with respect to complexity, structure and maintainability. Each measurement is shown by single line for each project.

Figure 6.5. Case Study-E: Average data points of LOC and CYC

In Case Study E, idea is useful when we want to visualize large number of projects and user is unable to assess the desire results then the following Case Study is useful. In other scenario when user requirement is limited and want
assess metrics individually then other approaches are useful for user then the approach explain and shown in Figure 6.5.

6.6. Case Study F
In Case Study F, display metrics and project version details by text instead of using circle. In this approach circle is removed and text is displayed as shown in Figure 6.6.

![Case Study-F: Visualizing text glyph LOC, CYC Metrics](image)

Case Study F is an alternative technique using text nodes instead of circles. But in this technique visualization is not better than the previous one. Circle fill with color is a better technique as used in previous Case Study.

6.7. Summary
We discussed different case studies of visualization of statistical contents of software projects. Conclusion and final results of above Case Studies will be explained in the following chapter (Section 7.1).
7. Conclusion

Our intent in this thesis was to research and develop visualization techniques to visualize statistical content of software’s. In this chapter, we will discuss our achievement of this research project. We will also explain future work that is related to our proposed visualization techniques.

7.1. Results

In this thesis, we have presented few case studies (Chapter 6) for analysis of software projects with different software quality metrics, i.e., a statistical contents are the main input of these visualizations. All case studies are useful in different perspectives, but our focus is to find out the best visualization approaches to display analysis of software projects. We have introduced a 2D chart for the visualization of software metrics along respective projects versions, and it is used for all test cases. Here we will describe the solution of already defined objectives (Section 1.3) of this project.

- Design graphical representation of statistical contents using 2D-Graphics.
- To implement a visualization technique that is more user friendly and time efficient for a large dataset.
- Develop a Web-Service that provides the interface between online Visualization-Tool and Database.
- Construct the new dataset by computing the average values of all attributes of each project.
- Develop the Web-Based visualization tool on client-side.
- Develop the server-side tool to upload the metrics data of projects along their version information into database.

7.1.1. Graphical Representation

A very first goal of this project was to implement a graphical representation of statistical contents using 2-D Graphics. As in Chapter 2, we discussed previous work of Chart Graphics for visualization of statistical contents of software’s. In our study, we found a research work related to visualization of software projects through different chart graphical representation, and we were decided to develop Line Chart graphical representation for visualization of statistical contents of software projects.

7.1.2. Visualization Techniques

Here, we have used different visualization techniques to analyze our visualization tool (ChartVizz2D). Our main focus in this comparison test is perception and better understanding of visualization. As the ChartVizz2D is a visualization tool for statistical contents of software projects, we should also analyze the performance of our system by direct human interaction. In this case, I have chosen three friends of mine from the group of Software Technology, MSI, Vaxjo University to compare the visualization performance of all approaches (Chapter
6). They have analyzed different case studies, and finally all of them agreed that our visualization Case Studies D (Section 6.4) and E (Section 6.5), both are easy to perceive and better to understand. Case Study D is useful to compare multiple Software Metrics of either same Project or multiple Projects. And Case Study E is useful to compare Combined Software Metrics (Collection of Software Metrics) of different Projects.

As we have introduced a zooming technique to visualize the data points more clearly. This technique makes user friendly and also this is time efficient access to get the any data point in constant time $O(1)$.

Our implemented web application is also time efficient for large dataset. As we tested our tool on real dataset, as explain below.

**Testing Platform**

*Operating System:* Windows Xp Professional  
*Database:* MySQL  
*Machine:* Toshiba 2.0 G, Intel inside  
*RAM:* 1 G

**Given Input Dataset-1**  
*Project:* Tomcat  
*Versions:* 18  
*Quality Metrics:* 3  
*Approximate dataset size:* 30 000 Records in MySQL database  
**Application takes 4.82 seconds to execute.**

**Given Input Dataset-2**  
*Project:* Tomcat  
*Versions:* 18  
*Quality Metrics:* 17  
*Approximate dataset size:* 30 000 Records in MySQL database  
**Application takes 28.2 seconds to execute.**

As we can see in above two datasets, dataset (1) with low number of Metrics is faster than other dataset (2).

### 7.1.3. Database Interface
This is important objective of our application; CVDb is a class to establish a connection between Web Applet and MySQL database. This interface can be use for both DML (Data Manipulation Language) and DDL (Data Definition Language) queries.
7.1.4. **Web Application**
We designed and developed the *Web* application for visualization of statistical contents of software projects. We had used *Java Applets* to develop our proposed visualization system.

7.1.5. **Utilities**
In this section, we cover other utilities or facilities for our application. These include following,
- download the statistical contents of software project.
- upload of projects dataset into database.
- download and upload of analysis tool information.
- zooming option in *2D Chart Panel*.
- exporting visualization results into *jpeg* format.
- printing the visualization results.

7.2. **Future Work**
This thesis presented a visualization of statistical contents of software projects through different case studies. After the completion of this project, we conclude with the following future work:
- Improve the data structure to accommodate the dataset.
- To load large datasets into the program memory faster than now.
- More work on zooming feature in our visualization application.
- Extend to flexible displays, especially for large screens, projectors etc.
- To compute input dataset with other statistical formulas (like, mean, standard error, median, modus, standard deviation, kurtosis, skewness, range, minimum, maximum, sum, count).
8. Glossary

2D 2 Dimension
ChartVizz2D: 2 Dimensional Charts Visualization
API: Application Programming Interface
AWT: Abstract Window Toolkit
EMF: Eclipse Model Framework
GEF: Graphical Editing Framework
GUI: Graphic User Interface
IDE: Integrated Development Environment
JDK: Java Development Kit
JDT: Java Development Tools
LWS: Light Weight System
MVC: Model-View-Controller
OMT: Object Modeling Technique
OOSE: Object-Oriented Software Engineering
PDE: Plug-in Development Environment (PDE)
SDK: Software Development Kit
UML: Unified Modeling Language
JVM: Java Virtual Machine
9. References


[10] *Visualizing Multidimensional Query Results Using Animation*, Amit P. Sawanta and Christopher G. Healey, North Carolina State University, Department of Computer Science, Raleigh, NC, USA.


