Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DESIGN AND IMPLEMENTATION OF ON - LINE METRICS

by
Ashok Vantipalli

The performance of a software product plays an important role in the software industry to survive the tough competition given by the other software vendors. The real test for a software product starts after its release when it is extensively used by the end users and in different environments. And the software goes through a number of tests by the end users which can be very different and insufficient from the regression tests the software vendor performed on it. This critical issue, requires each vendor to have a department mainly dealing with the testing and analysing the performance of a software product to improve future releases and support the current release. However, the performance analysis becomes very difficult for a human when there is lot of data over a period of time. So the ideal case for performance analysis would be a tool which could be fed with data and it does the dirty job of analysis and presents the software tester with easily understandable information in the form of graphs, pie charts etc. The requirments of the performance tool are essentially to give a graphical representation of different kinds of metrics which makes it easier for the software tester to draw conclusions about the performance of a particular software product and provide the software development team with important feedback which could lead to a better software product for the new release. On-Line Metrics, is a performance analysis tool which implements metrics to support the software testing team to derive better conclusions about a software product.

DESIGN AND IMPLEMENTATION OF ON - LINE METRICS

by
Ashok Vantipalli

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

May 1997

APPROVAL PAGE

DESIGN AND IMPLEMENTATION OF ON - LINE METRICS

Ashok Vantipalli

Dr. Franz J. Kurfess, Thesis Advisor — Date
Assistant Professor of Computer and Information Science, NJIT

Dr. Ajaz Rana, Committee Member — Date
Assistant Professor of Computer and Information Science, NJIT

Leon Jololian, Committee Member — Date
Director of Computer and Information Science Computing, NJIT

BIOGRAPHICAL SKETCH

Author: Ashok Vantipalli

Degree: Master of Science in Computer Science

Date: May 1997

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1997
- Bachelor of Science in Applied Mathematics and Statistics,
  State University of New York at Stony Brook, Stony Brook, NY, 1995

Major: Computer Science

Presentations and Publications:

Franz Kurfess, Lonnie Welch, Mrinalini Lankala, Ashok Vantipalli, "A Toolset for the Reengineering of Complex Computer Systems," International IEEE Symposium and Workshop on Engineering of Computer Based Systems(ECBS '97), pp 97-104,Mar 1997.

This work is dedicated to
my family

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his advisor, Professor Franz J. Kurfess, who not only provided valuable and countless resources, insight, and intuition, but also constantly gave him support, encouragement, and reassurance. Special thanks to Professor Ajaz Rana and Leon Jololian for serving as members of the committee. The author is grateful to Manager Edward Gold, for giving him an wonderful opportunity to work as an intern at Mentor Graphics Corporation and develop this tool. The author appreciates the timely help and suggestions from Paul Yao and Elizabeth Johnson during design of the project. And finally, a thank you to Ram Reddy for his help in technical aspects.

TABLE OF CONTENTS

Chapter		Page
1	INTRODUCTION	1
	1.1 Metrics	1
	1.2 Software Metrics	2
	1.3 Process Improvement	2
	1.4 On - Line Metrics	2
	1.5 Objective	3
	1.6 Background Information	3
2	THEORETICAL AND CONCEPTUAL BACKGROUND	4
	2.1 What are Defects ?	4
	2.2 Open Defects vs Time	4
	2.2.1 Severity	4
	2.3 Arrival vs Defects over Time	5
	2.4 Java vs CGI	5
3	USER REQUIREMENTS	6
	3.1 Functional Specification	6
4	ON - LINE METRICS	7
	4.1 Design Overview	7
	4.2 Applet Window	7
	4.2.1 Select Metric Button	7
	4.2.2 Example Metric Button	8
	4.2.3 Close Button	8
	4.3 Select Metric Window	9
	4.4 Defects vs Time Window	9
	4.4.1 Date List	9

Chapter Page

4.4.2 Gantt Chart 9

4.4.3 Text Fields 10

4.4.4 Pie Chart 10

APPENDIX A SOURCE CODE LISTING 13

REFERENCES 42

LIST OF FIGURES

Figure		Page
4.1	Design of the Tool	8
4.2	Applet Window	11
4.3	Select Metric	12
4.4	Defects vs Time	12

CHAPTER 1

INTRODUCTION

1.1 Metrics

Traditional engineering disciplines are marked by the availability of precise and well understood parameters of measurements. They have evolved to the present stage through a long term process of scientific examination and investigation and in a process of dynamic evolution with new metrics being proposed. Measurement techniques and parameters are desperately needed for assessing the quality and reliability of software as well as for the prediction and measurement of software production. Sixty percent of the software budget for all organizations using computers is being devoted to maintenance.

The importance for software metrics has been realized as early as 1972 and code metrics were proposed.A metric is a quantitative indicator of an attribute of a thing. A software metric with an associated theory or model, is one of the most important aspects of software engineering. Without measurement, we cannot understand the complex process that make up a software project and cannot make any objective statement concerning the quality of the software product.It is essential to identify or define indices of merit that can support quantitative comparisons and evaluations of software and of the process associated with its design, development, use, maintenance and evolution.

Numerous reports document the observation that, while the cost of hardware to perform a given function continues to decrease dramatically, the cost of software required for that function has continued to increase.In addition to the higher cost of software, schedule slippage is the rule rather than the exception.

1.2 Software Metrics

Software metrics are often classified as either process metrics or product metrics and are applied to either the development process or the software product developed. For the metrics to be useful they should be objective. An objective or algorithmic, measurement is one that can be computed precisely according to an algorithm. Its value does not change due to change in time, place or observer.

A Metric can be defined as form of measurement. It is basically, an algorithm to measure or calculate and provide with some kind of numbers which makes it easier for the human to draw conclusions based on the results. Software Metrics is basically an implementation of such algorithms.

1.3 Process Improvement

Software Process improvement implies the existence of a process to be improved. Processes are like programs. One must have the specification and design right before worrying about optimization. This is basically, a top-down process, guided by what you know you can implement efficiently but starting at the top, with requirements.

What are requirements ? Software developement is a engineering process, developing a complex set of rules to serve some useful purpose. Real business and industrial software has a set of complex requirements that must be implemented accurately and in great detail. In the later stages, the requirements will evolve and the software must be able to accommodate change without losing its integrity.

1.4 On - Line Metrics

A tool which is system independent, and can be accessed on the WWW.

1.5 Objective

The main objective of this thesis is to implement a tool which gives a complex graphical analysis of different types of products in the form of metrics. This project is a part of an ongoing effort by the Builder group at the Mentor Graphics Corporation for ISO9000 certification. The key features include not only a tool which monitors the builder products performance but also analyzes the company's performance in fixing the bugs in the released software versions for the customer support group.

1.6 Background Information

One of the main tasks of the Quality Control group at Mentor Graphics Corporation is to do a complex analysis of each builder product to ensure better improvements for the future releases. After the release of each product any bugs in the software are reported to the customer support group and they in turn assign a unique defect number for each in coming defect. These defects are entered in a database and forwarded to the corresponding department for fixing the bugs, and once these bugs are fixed they are entered back into the database as closed defects with date stamp. The quality control group tries to analyze each product by looking at the data in the database. There is no particular tool or an algorithm to do the analysis. This tool provides all the functionalities for a complex analysis.

CHAPTER 2

THEORETICAL AND CONCEPTUAL BACKGROUND

2.1 What are Defects ?

After the release of a software product any bugs in the product are reported to the customer service and they are logged in as "Open defects" in a database with time, day stamp and once these bugs are fixed they are entered into the database as "Closed defects" with time day stamp. So this project is an extensive analysis of the performance of a particular software product in the form of metrics. The following metrics are implemented:

- Number of Open defects Vs Time.
- Arrival Vs Resolution of defects over time.

2.2 Open Defects vs Time

This metric calculates the number of open defects on a monthly basis and displays in the form of Gantt chart. Each defect can then be characterized with a severity level and a priority level. The pie chart gives the distribution on the basis of severity of each defect. Each defect with certain severity can have a different priority level.

2.2.1 Severity

A defect can be characterized based on severity. The higher the severity worse the impact of the software bug. Having a lot of high priority bugs is considered bad by the quality control group. Severities are categorized into three different forms.

- High
- Medium
- Low

2.2.1.1 Priority Each defect can be categorized with different priority levels. The priority levels are assigned by the quality control group depending on the urgency to close each defect. Priorities are categorized in four different forms namely 1, 2, 3, 4.

2.3 Arrival vs Defects over Time

This metric displays the distribution of number of open defects and closed defects over time in the form of a graph. Ideally the resolution of defects should be greater than or equal to the arrival of defects.

2.4 Java vs CGI

This project could also be implemented in CGI/Perl 5.0, however the graphical capabilities are very limited. The user can enter all the information about the product, type of metric, starting and the ending date in the form of a HTML form and a backend perl script would take the input query the database, get the set of values and convert these values in the form of a plot and display the ".gif" file on the WWW. Even though it dynamically analyzes the data it has a user interface which does not have any user interactivity. The plot would be a static picture. Whereas in Java the user can get into the minute details of each month and can see the user interface change at run-time. And one of most important advantages of Java over CGI/Perl 5.0 is that it is compiled and it has a better performance.

CHAPTER 3

USER REQUIREMENTS

3.1 Functional Specification

1. The Following metrics are provided by this software tool:

   - Number of Open Defects over Time.
   - Arrival Vs Resolution of Defects over Time.

2. A graphical User Interface which would allow the user to select product, type of metric, starting and ending date during which the user wants to analyze the data.

3. A Window "METRICS" displaying the selected data in the form of Gantt charts, Pie charts and graphs.

4. The METRICS window should contain the range of all the dates the user selected, Gantt chart, Pie chart, Text fields containing the name of the product, total number of defects categorized into severity and priorities When the user selects one of the dates in the "METRICS" window , the bar n the Gantt chart corresponding to the date should be highlighted all the text fields and the Pie chart should change dynamically.

5. To retrieve the user specified data, query the database "SCOPUS" and format the data in a form which is readable by the Java code. The retrieval and data format is done is Perl 5.0

6. Finally this tool will be in the form of a Java applet, so that it is platform independent and can be accessed by any Java enabled browser.

CHAPTER 4

ON - LINE METRICS

4.1 Design Overview

The Metrics tool can be accessed by any Java enabled browser. The client basically loads the defect file through the URLConnection Class and runs the tool on its own machine. The defects file for each product is available on the server and when the user selects a particular product the corresponding defect file is loaded in a linked list. This particular tool cand also be implemented using Java Network Package using sockets. However the server has to be running all the times and any server breakdown would result in no access to the tool by any client. The above design would require lot of maintenence and hence this tool has been implemented using a design which is machine and server independent. The design is included as shown in fig 4.1

4.2 Applet Window

The Metrics tool is implemented in the form of a Java applet and can be accessed "http://hertz.njit.edu/ axv4610/gui.html". A applet window appears as shown in fig 4.2. It contains the following three buttons.

- Select Metric
- Example Metric
- Close

4.2.1 Select Metric Button

When a user clicks on the "Select Metric" button a new window appears which basically has the capability to select items from the select lists. The first list consists of the type of products which are as follows and the figure is shownin fig 4.3

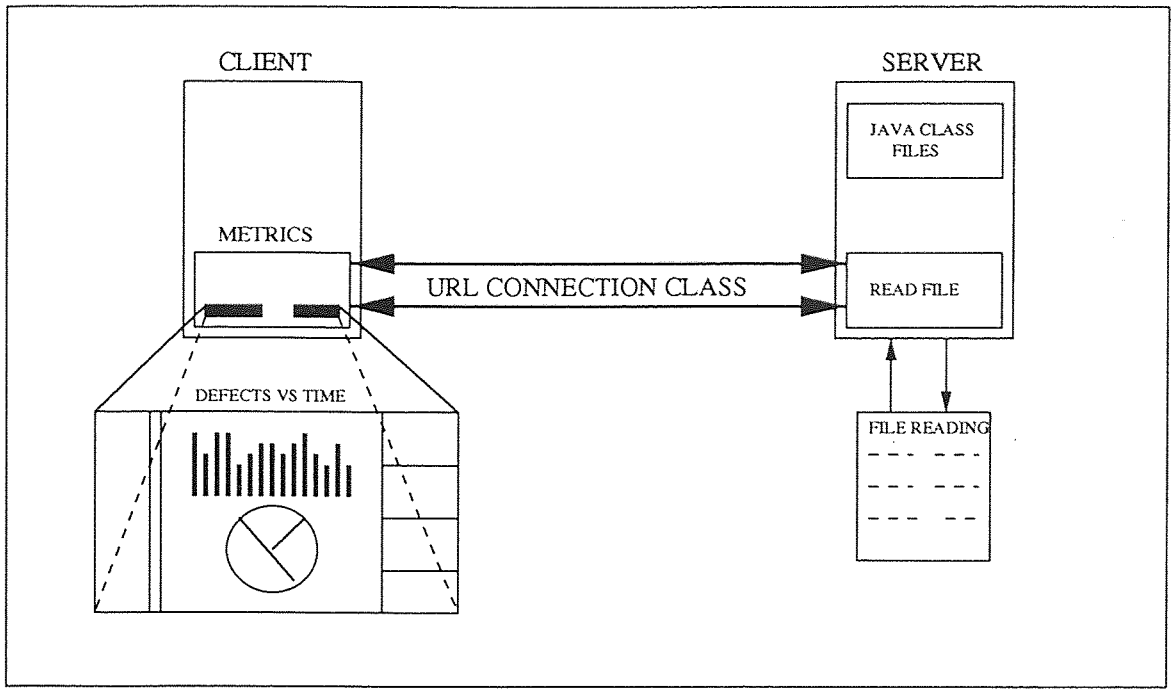

Figure 4.1 Design of the Tool

- Cell Builder
- Memory Builder
- IC Station
- Timing Builder

4.2.2 Example Metric Button

When the user clicks on the "Example Metric" button a new window appears which gives an example of metrics. It gives a model graph for defects over time.

4.2.3 Close Button

When the user clicks on the "Close" button, it closes the whole java application

4.3 Select Metric Window

This window consists of list items, which can be chosen. There are four productsfor which one could plot a Defects vs Time graph. Once a product is selected and starting and the ending dates are also selected and submitted the defect file from the server is accessed and calculates the metric values. The Select Metric Window is shown in fig 4.3

4.4 Defects vs Time Window

This window consists of the following objects, which perform different functionalities.

- Date List
- Gantt Chart
- Text Fields
- Pie Chart

4.4.1 Date List

This is list of the dates which fall in the range of dates the user selected. Each date in the list corresponds to a bar in the Gantt chart. If the user clicks on a particular date the corresponding bar in the gantt chart gets highlighted and the text fields change dynamically and also the Pie chart is redrawn based on the distribution.

4.4.2 Gantt Chart

This is basically a plot of defects over a period of time, where in this case the time is taken on a monthly basis. Each bar in the chart implies the total number of defects in a particular month. It is not possible to get the number of defects in a particular month just by looking at the bar, however the text fields give the exact count of the defects.

4.4.3 Text Fields

There are five text fields which control different feauters, The text field "Total" gives the total number of defects in the selected, highlightened month. The severity fields give the count of defects based on severities.

Total Defects = Severity1 + Severity2 + Severity3

Each severity is given a different color and severity3 defect is considered to be a bad defect. Ex: In a particular month if there were 4 total defects and out of them three were severity3, it is considered to be a bad month, when considering the performance of the software.

4.4.4 Pie Chart

The Pie chart calculates the percentage of each type of defect in a particular month. The events that foollow a select of month in the Date List are the corresponding date in the Gantt Chart gets highlighted gives the total and severity distribution of defects and finally calculates the percentage of each severity defect in a selected month using the Pie chart. The Defects vs Time Metric could be seen in fig 4.4

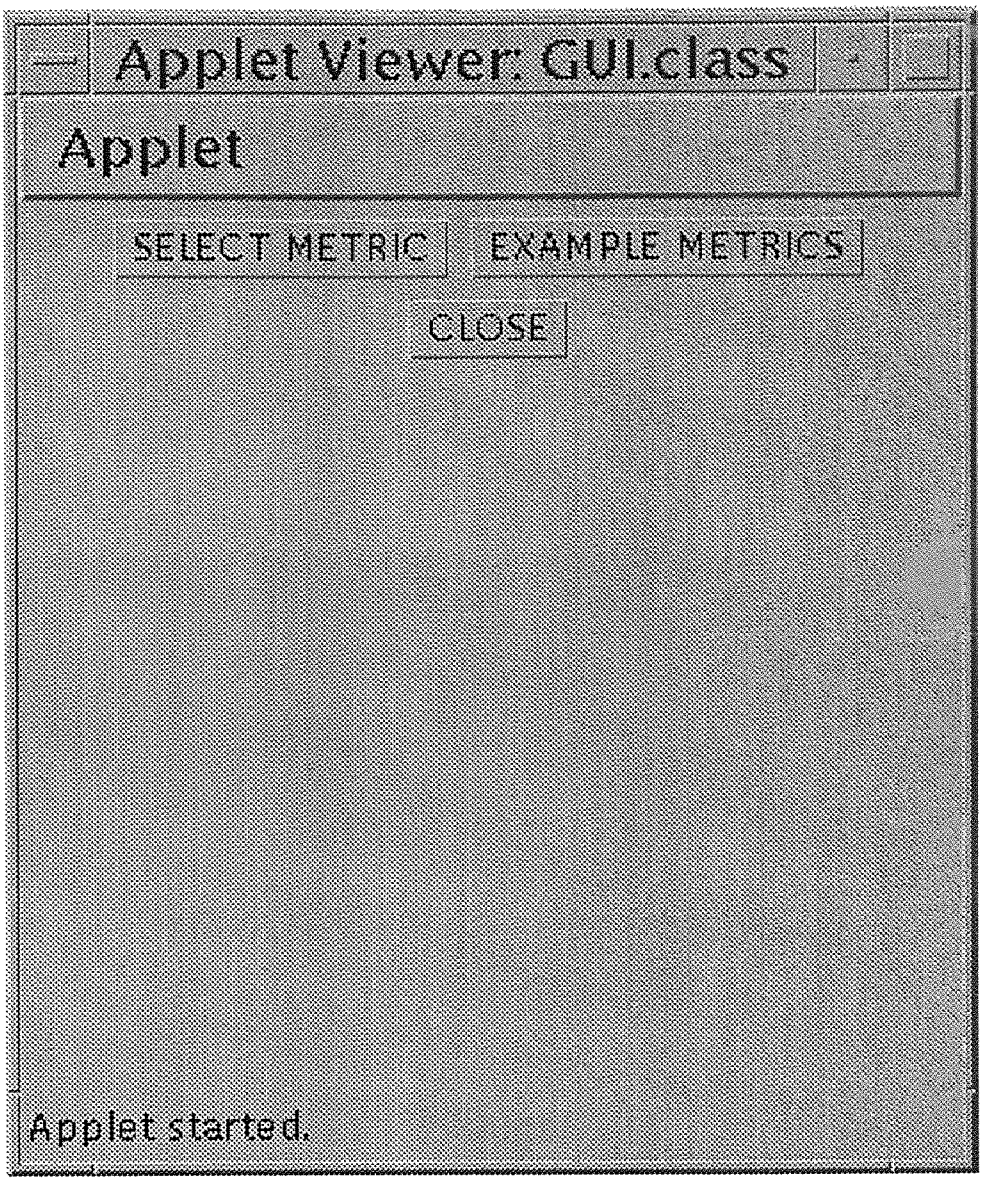

Figure 4.2 Applet Window

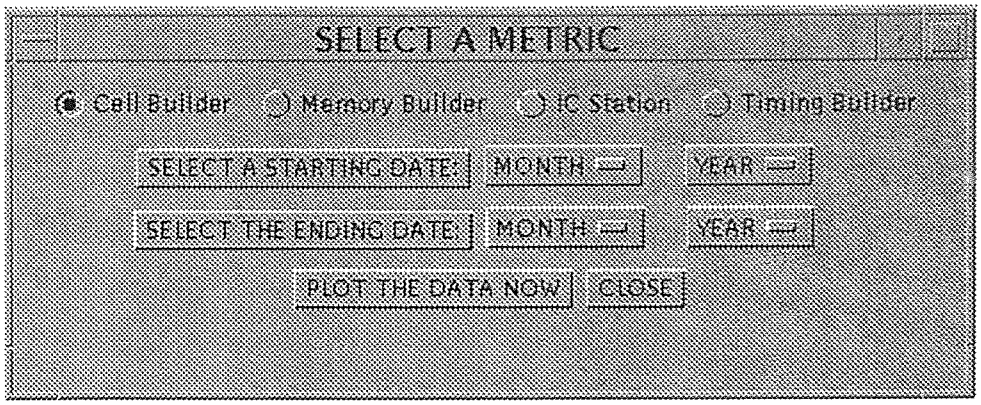

Figure 4.3 Select Metric

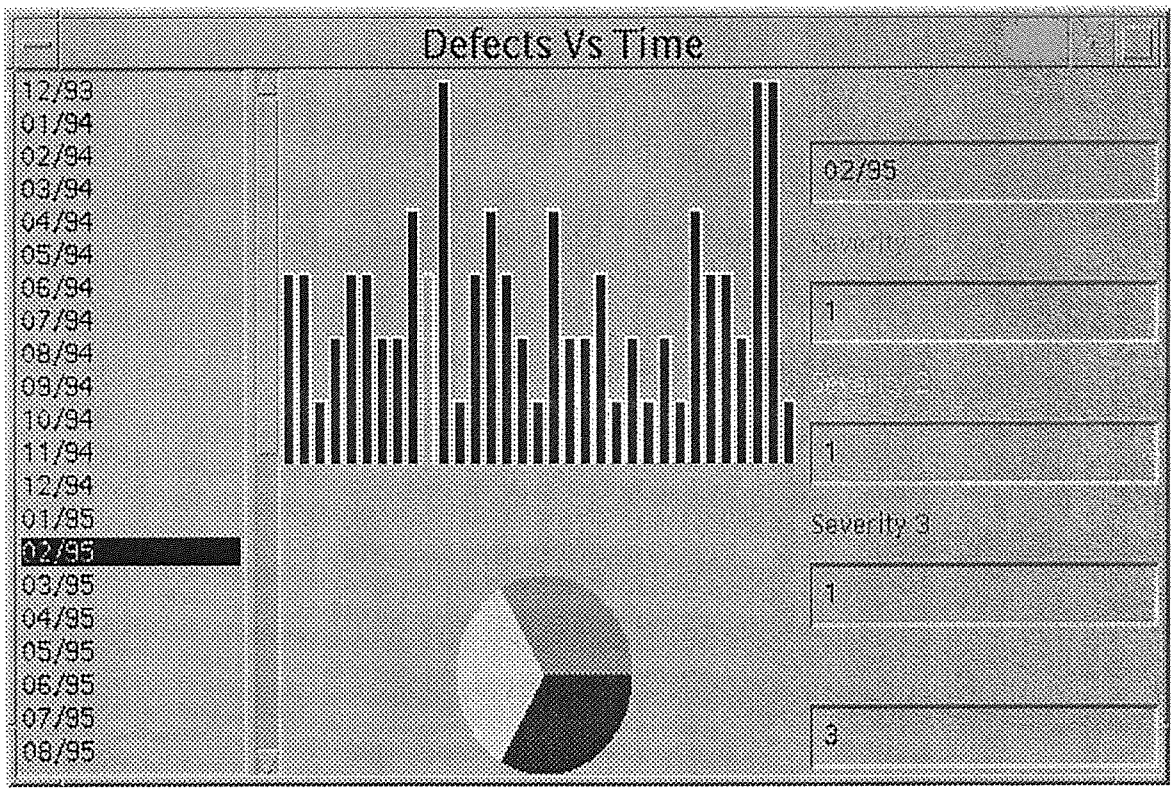

Figure 4.4 Defects vs Time

APPENDIX A

SOURCE CODE LISTING

```
********************** APPLET WINDOW ********************************
    import java.applet.*;
    import java.awt.*;


    public class GUI extends Applet {

        Frame window, window1;
        //inputFile appFile = new inputFile();
        //StringBuffer buf = new StringBuffer();

        public void init() {

            //buf = appFile.readFile();
            add(new Button("SELECT METRIC"));
            add(new Button("EXAMPLE METRICS"));
            add(new Button("CLOSE"));
            window = new MyFrame("A MODEL CHART");
            window.resize(500, 500);
            Point p = window.location();
            window.move(p.x+200, p.y+250);
            window1 = new Metric("SELECT A METRIC");
            window1.resize(500,500);
            Point q = window1.location();
```

```
                window1.move(q.x+200, q.y+250);
        }

        public boolean action(Event evt, Object arg) {

             if (evt.target instanceof Button) {
                String label = (String)arg;
                if (label.equals("SELECT METRIC")) {
if (!window1.isShowing())
                        window1.show();
                }
                if (label == "EXAMPLE METRICS") {
                        if (!window.isShowing())
                        window.show();
                }
                else if (label == "CLOSE")
                     //   if (window.isShowing())
                        //window.hide();
                          System.exit(0);

                return true;
             }
                else return false;
        }

}
```

```
class MyFrame extends Frame {

        Label L;
        Dialog d;
        Frame f;
        TextField tf1, tf2;
        inputFile appFile = new inputFile();
        StringBuffer buf = new StringBuffer();

        MyFrame(String title) {
        super(title);

 MenuBar mb = new MenuBar();
        Menu m = new Menu("METRICS");
        m.add(new MenuItem("Defects/Time"));
        m.add(new MenuItem("DEFECTS(Arrival vs Closure)"));
        m.add(new MenuItem("XXX"));
        mb.add(m);
        Menu c = new Menu("CLEAR");
        c.add(new MenuItem("CLEAR"));
        mb.add(c);
        Menu q = new Menu("QUIT");
        q.add(new MenuItem("QUIT"));
        mb.add(q);
        setMenuBar(mb);
        setLayout(new FlowLayout(FlowLayout.CENTER));
```

```
        L = new Label();
        //add(L, "South");
//      Frame f = new Frame("DateDialog Test");
//      f.resize(250,150);
        //f.show();
        //DateDialog d = new DateDialog(this);
        //d.show();
        }

public boolean action(Event evt, Object arg) {

        if (evt.target instanceof MenuItem) {
           String label = (String)arg;
           if (label.equals("Defects/Time")){
           buf = appFile.readFile();
           Date d = new Date(this);
           d.show();
 }
          else if (label.equals("DEFECTS(Arrival vs Closure)"))

           buf = appFile.readFile();
           Date d = new Date(this);
           d.show();
           }
        else if (label.equals("QUIT")) this.hide();
        else if (label.equals("CLEAR")) setBackground(Color.cyan);
        }
```

```
        return true;
        }
//      else return false;

public void paint(Graphics g) {

        g.setColor(Color.magenta);
        g.fillRect(30,150,30,100);
        g.setColor(Color.red);
        g.fillRect(60,175,30,75);
        g.setColor(Color.blue);
        g.fillRect(90,190,30,60);
        g.setColor(Color.yellow);
        g.fillRect(120,80,30,170);
        g.setColor(Color.black);
        g.fillRect(150,100,30,150);
        g.setColor(Color.orange);
        g.fillRect(180,100,30,50);
        g.setColor(Color.green);
        g.fillRect(180,150,30,50);
        g.setColor(Color.pink);
        g.fillRect(180,200,30,50);
 g.setColor(Color.cyan);
        g.fillRect(210,200,30,50);


        }
```

```
}

**************************** SELECT METRIC ****************************
import java.awt.*;

class Metric extends Frame {

        Button b1, b2, d1, d2;
        Panel P1;
        Label P2, P3, P4, P5;
        inputFile appFile = new inputFile();
        StringBuffer buf = new StringBuffer();
        Checkbox cb1, cb2, cb3, cb4;
        CheckboxGroup cbg;
        Choice choice, choice1, ch, ch1;
        Frame win, window1;


        Metric(String title) {
        super(title);

        win = new BarChart("BAR CHART");
        win.resize(500, 500);
        Point b = win.location();
        win.move(b.x+200, b.y+250);

        // Start checkboxes
```

```
        cbg = new CheckboxGroup();
        cb1 = new Checkbox("Cell Builder", cbg, false);
        cb2 = new Checkbox("Memory Builder", cbg, false);
        cb3 = new Checkbox("IC Station", cbg, false);
        cb4 = new Checkbox("Timing Builder", cbg, false);

        P1 = new Panel();
P1.setLayout(new FlowLayout());
        P1.add(cb1);
        P1.add(cb2);
        P1.add(cb3);
        P1.add(cb4);
        setLayout(new FlowLayout());
        add(P1);

        // CHOICE MENU

        d1 = new Button("SELECT A STARTING DATE:");
        //add(d1);
        choice = new Choice();
        choice.addItem("MONTH");
        choice.addItem("1");
        choice.addItem("2");
        choice.addItem("3");
        choice.addItem("4");
        choice.addItem("5");
        choice.addItem("6");
```

```
        choice.addItem("7");
        choice.addItem("8");
        choice.addItem("9");
        choice.addItem("10");
        choice.addItem("11");
        choice.addItem("12");
        P2 = new Label();
        //setLabelText(choice.getSelectedIndex(), choice.getSelectedItem());

        choice1 = new Choice();
        choice1.addItem("YEAR");
choice1.addItem("91");
        choice1.addItem("92");
        choice1.addItem("93");
        choice1.addItem("94");
        choice1.addItem("95");
        choice1.addItem("96");
        // SECOND DATE
        d2 = new Button("SELECT THE ENDING DATE:");
        //add(d2);
        ch = new Choice();
        ch.addItem("MONTH");
        ch.addItem("1");
        ch.addItem("2");
        ch.addItem("3");
        ch.addItem("4");
        ch.addItem("5");
```

```
        ch.addItem("6");
        ch.addItem("7");
        ch.addItem("8");
        ch.addItem("9");
        ch.addItem("10");
        ch.addItem("11");
        ch.addItem("12");
        P4 = new Label();
        //setLabelText(choice.getSelectedIndex(), choice.getSelectedItem());

        ch1 = new Choice();
        ch1.addItem("YEAR");
        ch1.addItem("91");
        ch1.addItem("92");
        ch1.addItem("93");
ch1.addItem("94");
        ch1.addItem("95");
        ch1.addItem("96");
        P3 = new Label();
        P5 = new Label();
        // BUTTONS

        b1 = new Button("PLOT THE DATA NOW");
        b2 = new Button("CLOSE");
        setLayout(new FlowLayout());
        add(d1);
        add(choice);
```

```
        add(P2);
        add(choice1);
        add(P3);
        add(d2);
        add(ch);
        add(P4);
        add(ch1);
        add(P5);
        add("South", b1);
        add("South", b2);
        validate();
        this.pack();
        }
        public boolean action(Event evt, Object arg) {

                if (evt.target instanceof Button) {
                String label = (String)arg;
                if (label == "PLOT THE DATA NOW") {
                        buf = appFile.readFile();
if (!win.isShowing())
                        win.show();
                }
                else if (label == "CLOSE") {
                        if (win.isShowing())
                        win.hide();
                }
                return true;
```

```
        }
        else return false;

        }

}

************************** DEFECTS VS TIME ************************
import java.io.*;
import java.util.*;
import java.lang.*;
import java.awt.*;
import java.net.URL;

class Address {
   protected String name;
   protected String date;
   protected
 ques1;
   protected int ques2;
   protected int ques3;
   protected int ques4;

   Address(String n, String d, int o1, int o2, int o3, int o4) {
      name = n;
      date = d;
```

```
      ques1 = o1;
      ques2 = o2;
      ques3 = o3;
ques4 = o4;
  }

  public String getName() {
     return name;
  }

  public String getDate() {
    return date;
  }

  public int getques1() {
     return ques1;
  }

  public int getques2() {
     return ques2;
  }
  public int getques3() {
     return ques3;
   }

  public int getques4() {
     return ques4;
```

```
    }


}


class SelectList extends List {
    public boolean handleEvent(Event evt) {
        List list = (List) evt.target;

        switch (evt.id)
            case Event.LIST_SELECT:
                return selected(evt,
                    list.getSelectedIndex(),
                    list.getSelectedItem());
            case Event.LIST_DESELECT:
                return deselected(evt,
                    list.getSelectedIndex());
            default:
                return super.handleEvent(evt);
        }
    }

    public boolean selected(Event evt, int index, String item) {
        return false;
```

```
   }


class AddressList extends SelectList {
   //static  protected Address list[] =
   static  Address list[] =
                    new Address[100];
   static int barlist[] = new int[100];
   static int curr;
   private InfoPanel info;
   private GradeSCanvas sgrade;
   private GradeCanvas ngrade;
   static List wist;

   AddressList(InfoPanel infoPanel,GradeSCanvas sgradeCanvas, GradeCanvas
ngradeCanvas) {

      Address newAddr;
      info = infoPanel;
      sgrade = sgradeCanvas;
      ngrade = ngradeCanvas;

      wist = new List(25,true);

      String is = null;

int i = 0;
```

```
 try {
      FileInputStream fin = new FileInputStream("cb.defect");
      DataInputStream din = new DataInputStream(fin);
      is = din.readLine();
  while (is != null) {
StringTokenizer st = new StringTokenizer(is);
  int slot = 0;
  while (st.hasMoreTokens()) {
      String uname = st.nextToken();
      String dname = st.nextToken();
      int ques1 = Integer.parseInt(st.nextToken());
      int ques2 = Integer.parseInt(st.nextToken());
      int ques3 = Integer.parseInt(st.nextToken());
      int ques4 = Integer.parseInt(st.nextToken());
     //   System.out.println(uname+home+shell);
  newAddr = new Address(uname,dname,ques1,ques2,ques3,ques4);
     list[i++] = newAddr;
      addItem(newAddr.getDate());
     wist.addItem(newAddr.getDate());
     barlist[i-1] = ques1;
        System.out.println(barlist[i-1]);
      }
    is = din.readLine();
   }
  }
  catch(Exception e) { }
```

```
    // wist = new List(5,true);
    //  wist.addItem(list[0].getName());

}

    public boolean selected(Event evt,
>                    int index,
                    String item) {

      Address addr = list[index];
      //static grade = list[index].ques0;

     // System.out.println(index+addr.name+addr.phone+addr.address);
      info.setDate(item);
      info.setques1(addr.getques1());
      info.setques2(addr.getques2());
      info.setques3(addr.getques3());
      info.setques4(addr.getques4());
      sgrade.setpie1(addr.getques1());
      sgrade.setpie2(addr.getques2());
      sgrade.setpie3(addr.getques3());
      int total = addr.getques1() + addr.getques2() + addr.getques3()
+
addr.getques4() ;
      sgrade.setpieT(total);
      //ngrade.setindex(addr.getques0());
```

```
        curr = addr.getques1();

        sgrade.repaint();
        ngrade.redraw(this);
        repaint();
 return true;
    }
}


class InfoPanel extends Panel {
    final int TextWidth = 15;
    final int TextHeight = 7;
    TextField nameT;
    TextField ques1T,ques2T,ques3T,ques4T;

    InfoPanel()
    {
        Label nameL = new Label("Date");
        nameT = new TextField(TextWidth);
        nameT.setEditable(false);
        Label ques1L = new Label("Severity 1");
        ques1L.setForeground(Color.green);
        ques1T = new TextField(TextWidth);
        ques1T.setEditable(false);
        Label ques2L = new Label("Severity 2");
        ques2L.setForeground(Color.yellow);
```

```
        ques2T = new TextField(TextWidth);
        ques2T.setEditable(false);
        Label ques3L = new Label("Severity 3");
        ques3L.setForeground(Color.red);
        ques3T = new TextField(TextWidth);
        ques3T.setEditable(false);
       Label ques4L = new Label("Total");

ques4L.setForeground(Color.black);
        ques4T = new TextField(TextWidth);
        ques4T.setEditable(false);
        // lots of geometry stuff here
        setLayout(new BorderLayout());
        Panel p2 = new Panel();
        p2.setLayout(new GridLayout(11,2,0,1));
        //p2.setLayout(new FlowLayout());

        add("East",p2);
        p2.add(nameL);
        p2.add(nameT);
        p2.add(ques1L);
        p2.add(ques1T);
        p2.add(ques2L);
        p2.add(ques2T);
        p2.add(ques3L);
        p2.add(ques3T);
        p2.add(ques4L);
```

```
        p2.add(ques4T);

    }

    public void setDate(String date) {
        nameT.setText(date);
    }


    public void setques1(int ques1) {

ques1T.setText(String.valueOf(ques1));
    }

    public void setques2(int ques2) {
        ques2T.setText(String.valueOf(ques2));
    }

    public void setques3(int ques3) {
        ques3T.setText(String.valueOf(ques3));
    }

    public void setques4(int ques4) {
        ques4T.setText(String.valueOf(ques4));
    }
```

```
}

 class GradeCanvas extends Canvas
{
   int curr1;
   int i;
   int v[] = new int[100];
   public GradeCanvas()
   {
     resize(250,175);
   }

   public void setindex(int ques) {

  curr1 = ques;
   }

   public void redraw(AddressList list)
   {
     nlist = list;
      for (i = 0; i < 40; i++)
      {  v[i] =  nlist.barlist[i];
         curr1 = nlist.curr;
      }
     repaint();
   }
```

```
    public void paint(Graphics g)

     {
       int minValue = 0;
       int maxValue = 0;
      int y1 = 0;
       int x1;
       int yOrigin;
       int height;
       //for (i = 0; i < nlist.list.length; i++)
       for (i = 0; i < 40; i++)
       {
          if (minValue > v[i]) minValue = v[i];
          if (maxValue < v[i]) maxValue = v[i];
       }
       if (maxValue == minValue) return;

    Dimension d = size();
       int barWidth = d.width / 30;
       double scale = (double)d.height
          / (maxValue - minValue);

  //The values are reversed because of origin being at top. Hence
yOrigin
  variable is used.

       yOrigin = (int)(maxValue * scale);
```

```
      for (i = 0; i < 30; i++)
      {   x1 = i * barWidth + 1;
//       System.out.println("Start V ="+v[i]+"\n");

          y1 = (int)( v[i] * scale);
          height = yOrigin - y1;


        if (v[i] < curr1)
           g.setColor(Color.blue);
        else if (v[i] > curr1)
           g.setColor(Color.red);
        else
        g.setColor(Color.green);
        g.fillRect(x1, height, barWidth - 2, yOrigin);
        g.setColor(Color.black);
        g.drawRect(x1, height, barWidth - 2, yOrigin);
      }

 }


  private AddressList nlist = null;

}
```

```
class GradeSCanvas extends Canvas {
       int i;
      int pieT;
      int pie[] = new int[10];
  public GradeSCanvas()

   {  resize(250,175);
     for (i = 0; i < 10; i++)
     pie[i] = 36;
     pieT = 360;
   }


  public void setpie1(int ques) {
      pie[1] = ques;
   }

   public void setpie2(int ques) {
      pie[2] = ques;
   }


}

  public void setpie3(int ques) {
      pie[3] = ques;
```

```
    }

    public void setpie4(int ques) {
       pie[4] = ques;
    }
>


  public void setpieT(int ques) {
       pieT = ques;
    }

    public void paint(Graphics g)
     {
       int r = 45;
       int cx = 125;
       int cy = 90;
       int temp = 0;
       int endangle,startangle;
     //   System.out.println("Startangle="+pie[1]+"angle = "+pie[2]+"\n");
       for (i = 0; i < 4; i++)
       {
          int v = (int) (pie[i] * 360 / pieT);
          startangle = temp;
         // System.out.println("Startangle="+startangle+"angle =
"+v+"\n");
          if (i < 1) {
```

```
            g.setColor(Color.blue);


g.setColor(Color.blue);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 0 && i < 2) {
            g.setColor(Color.white);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 1 && i < 3) {
            g.setColor(Color.magenta);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 2 && i < 4) {
            g.setColor(Color.black);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 3 && i < 5) {
            g.setColor(Color.cyan);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 4 && i < 6) {
            g.setColor(Color.red);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 5 && i < 7) {
            g.setColor(Color.orange);
```

```
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 6 && i < 8) {
            g.setColor(Color.pink);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);

g.setColor(Color.pink);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else if (i > 7 && i < 9) {
            g.setColor(Color.green);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, v);
             }
            else  {
            g.setColor(Color.yellow);
            g.fillArc(cx-r, cy-r, 2 * r, 2 * r, startangle, 360-startangle);
             }
            temp = startangle+v;
         }

/*       for (i = 0; i < AddressList.list.length; i++)
         {
            System.out.println("List "+i+" = "+AddressList.list[i].ques0+"\n");
         }
*/

   }
```

```
}


class AddressBook extends Frame {
   public static void main(String argv[])
   {
      new AddressBook();
   }
   AddressBook()
{

      InfoPanel infoPanel = new InfoPanel();
      GradeSCanvas  sgradeCanvas = new GradeSCanvas();
      GradeCanvas  ngradeCanvas = new GradeCanvas();
      AddressList list = new AddressList(
                                 infoPanel,sgradeCanvas,ngradeCanvas);

      // some geometry stuff, then:
      Frame f = new Frame("Defects Vs Time");

//       TextField workT;
 //      setLayout(new BorderLayout());
  //     Panel p3 = new Panel();
   //    p3.setLayout(new GridLayout(3,1,0,0));
    //    add("Center",p3);
     //   p3.add(list.wist);
```

```
/*     Label workL = new Label("Working Dir");
      workT = new TextField(20);
      p3.add(workL);

      try
         {
           String collect = null;
           String command1 = "/usr/openwin/bin/shelltool pwd ";
           Runtime r;
           r = Runtime.getRuntime();
//         collect = System.in.readLine(r.exec(command1));
           workT.setText(collect);

 }
      catch(Exception e)
         {
          System.out.println("Exception detected Hello Again");
         }
      workT.setText("Never works");
      p3.add(workT);
*/
>
      f.add("West",list);
      f.add("East",infoPanel);
      setLayout(new BorderLayout());
      Panel p3 = new Panel();
      p3.add("North",ngradeCanvas);
```

```
        p3.add("South",sgradeCanvas);
        f.add("Center",p3);
        f.resize(700,350);
        f.show();

    }
}

\end{verbatim}
```

REFERENCES

1. G. Cornell and C. S. Horstmann *Java in a Nutshell, second edition* O'Reilly Associates, Sebastopol, CA, USA 1996.

2. D. Flanagan *Core Java, second edition* Sunsoft Press, Englewood Cliffs, NJ, USA 1996.

3. D. M. Geary and A. L.McClellan *Graphic Java, Mastering the AWT, first edition* Sunsoft Press, Englewood Cliffs, NJ, USA 1996.

4. L. Lemay and C. L.Perkins *Teach Yourself Java in 21 Days, second edition* Sams Publishing, Indianapolis, Indiana, USA 1995.

5. M. Shepperd, *Software Engineering Metrics Volume 1: Measures and Validations,* McGraw-Hill Book Company, Berkshire, England 1993.