Journal of Computer Science 5 (11): 811-816, 2009
ISSN 1549-3636

Application of a Database in the Monitoring of Workstations in a Local Area Network

[1]Eyo O. Ukem and [2]Buba G. Bajoga
[1]Department of Physics, Electronics and Computer Technology Unit,
University of Calabar, Calabar, Nigeria
[2]Department of Electrical Engineering, Ahmadu Bello University, Zaria, Nigeria

Abstract: Problem statement: Computer hardware fault management and repairs can be a big challenge, especially if the number of staff available for the job is small. The task becomes more complicated if remote sites are managed and an engineer or technician has to be dispatched. **Approach:** Availability of relevant information when needed could ease the burden of maintenance by removing uncertainties. Such required information could be accumulated in a database and accessed as needed. **Results:** This study considered such a database, to assist a third party hardware maintenance firm keep track of its operations, including the machines that it services, together with their owners. A software application was developed in Java programming language, in the form of a database, using Microsoft Access as the database management system. It was designed to run on a local area network and to allow remote workstations to log on to a central computer in a client/server configuration. With this application it was possible to enter fault reports into the database residing on the central computer from any workstation on the network. **Conclusion/Recommendations:** The information generated from this data can be used by the third party hardware maintenance firm to speed up its service delivery, thus putting the firm in a position to render more responsive and efficient service to the customers.

Key words: Software application, database, fault logging, fault management, client/server

INTRODUCTION

Since the advent of computers, virtually every facet of human endeavour has been greatly influenced. Computers have made inroads into virtually every area of human activities, and their use has become so widespread that it is rather difficult to identify an area where computers have not yet been used to improve processing. The easy availability of computers for application in almost all areas of human endeavour has resulted in computers being found in most establishments. The use of the machine has developed so much that users do not now require deep knowledge of the inner workings of the systems to be able to use them for their daily and routine businesses. As a consequence, there is now a proliferation of computer systems, even in developing countries such as Nigeria. However, even though the operations of the systems have been made as simple as possible, systems still do break down. Unfortunately, the process of repairs has not seen a corresponding simplification. This is due to the very complex nature of the hardware. Apparently, in a bid to simplify the operations of the machines for the benefit of users, the machines have invariably become a lot more complex. This complexity is transparent to the user, but when faults occur, only the most basic of such faults can be cleared by the regular user. The rectification of other faults requires personnel with some reasonable measure of expertise.

Since it is not practicable, for various reasons, for all establishments that use computers to have on their payroll personnel with the required level of expertise to handle all of their computer system problems, the practice is for a specialist firm to be established specifically to render these services to computer user-organizations. Ideally, such a third-party service firm would need to keep track of its operations.

The objective of this study is to describe a software application package, developed in the form of a database that can be employed by the third-party service firm to improve its services to the customers. With the data held in the database, and the information generated from it, the firm should be in a position to render more efficient service to the customers.

Corresponding Author: Eyo O. Ukem, Department of Physics, Electronics and Computer Technology Unit, University of Calabar, PMB 1115, Calabar, Nigeria Tel: +2348063382514

MATERIALS AND METHODS

All software applications can be collectively called data processing. Data Processing refers to the process of producing meaningful information by collecting all items of data together and performing required operations on them to extract the required information. The data items are the raw material while information is the end product. The methods of data processing generally range from those that are almost entirely manual to those that rely heavily on the use of computers, but, as in virtually all other areas of human endeavour, the use of computers has become prevalent. Software is built to process data, that is, to transform data from one form to another. This implies to accept input, manipulate it, and produce output. This fundamental statement of objective is true whether the software being built is a batch software or a real-time embedded software[1]. Whatever methods of data processing are employed, the process goes through the same basic stages, which can be identified as origination of data, preparation of data, input of data, processing of data, and output of information. The data (both the maintained data and input data) are mostly held in files of various structures. Files are thus the framework around which data processing revolves. Depending on the data processing strategy employed, the files may be held separately for different systems or grouped into an integrated file system, or the data may be held in a database (a single organized collection of structured data). The database approach has tended to overshadow the flat file approach, due to the advantages that the database system has, which include data consistency, data integration, data sharing, data independence, and data control. Of the four basic types of database systems, namely Hierarchical, Network, Relational, and Object-Oriented Database Systems, the Relational system is the most widely used, although the Object-Oriented system is showing signs of superiority and great potential, as it can store more types of data than all others[2]. The Relational Database Management System was selected for this research.

Every software project is prompted by some business need, such as the need to correct a defect in an existing application, the need to adapt a legacy system to a changing business environment, the need to extend the functions of an existing application, or the need to create a new product, service, or system[3]. For this work, the prompting was the need to provide a new service and thus remedy a situation (according to Eyo Ukem in an unpublished work: "On-Line Capture and Monitoring of the Profile and Maintenance Status of Computer Workstations in a Local Area Network"). The problem to be solved was expressed in the following problem statement:

> "The third-party hardware maintenance firm desires to implement a fault logging system that will maintain a database of all the computers (workstations) that the firm maintains (or services), together with their respective owners. The database is to contain such information about the machines as serial number, manufacturer, make, model, date of manufacture, date of purchase, owner, and location. Information held about machine owners should include name, address, telephone number, and type of client. The system is to maintain a record of the occurrences of faults to enable the firm track the trend of events and react speedily and on a timely basis to situations. The system should be capable of transmitting faults information across the network from client to server. When a fault report is received, either on-line or in batch, the system accepts the report and logs it. The system should record all the faults and the action taken in each case, including the cost of the remedial action, the personnel carrying out the action, and spare parts used, if any. The system should generate periodic reports on the faults and the affected machines"

The requirements identified for this software are summarized below.

The third-party hardware maintenance firm requires to:

- Keep track of the customers it serves
- Maintain a detailed record of every machine on its contract
- Keep and track record of all spare parts in stock
- Identify parts with their machines
- Monitor the status of client machines
- Maintain fault history of each machine, including repair costs and fault types
- Maintain vital information on each machine -such as date of purchase, make of machine, model, serial number
- Maintain service record, including service personnel, fault rate or frequency
- Highlight customer credit rating and payment fault rate
- Obtain on-line, as much as possible, information on the operational status of the workstations, so as to be able to improve response time
- Generate periodic reports

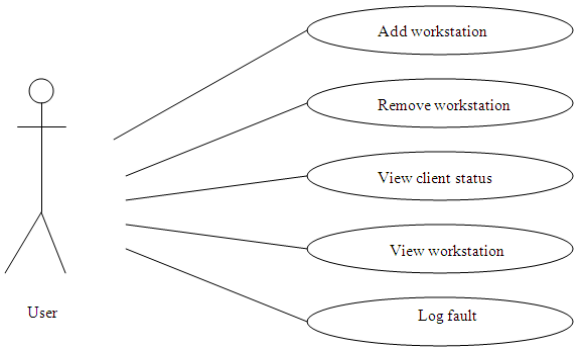

Fig. 1: Use case diagram for the fault logger application

Fig. 2: Some of the design classes with their attributes and methods

Based on the product description the candidate classes were extracted, from which use cases were modeled. Each use case describes a typical scenario for which the user uses the system[4]. Some of the use cases are identified as:

- Add workstation to the database
- Remove workstation from the database
- View client's credit status
- View status of workstation
- Log fault report

The use cases are shown in Fig. 1 using Universal Modeling Language (UML) representation. The design classes that were identified, together with their attributes and methods, are shown in Fig. 2, also in UML notation.

Implementation: The application is implemented in Java, in a client/server configuration, with Microsoft Access as the database management system. On execution, the server section of the application presents an opening screen which includes a menu that permits the selection of options to execute. Each option has its own screen accordingly. The Java code for the first screen of the application is shown below:

```
import java.net.*;
public class FRMainClass{
        static FRServer MyServer;
        static FRMainScr MyMainScr;
        static ServerAlertMsgReciever smr;

        public static void main(String s[]){
                MyMainScr = new FRMainScr();
                smr = new
                ServerAlertMsgReciever();
                smr.start();
                MyServer = new FRServer();
                MyServer.start();
        }
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class FRMainScr extends JFrame implements
        ActionListener{
        JButton ClientInfoButton;
        JButton MachineInfoButton;
        JButton FaultsButton;
        JButton ConnectionsButton;
        JButton RefreshButton;
        JButton HelpButton;
        JButton ExitButton;
        // The Buttons on the left hand side of the
        screen
        JSplitPane MySplitPane;
        // Seperates the left pane from the right pane
        JPanel LeftPanel;
        JPanel RightPanel;
        public static FRDatabase FRD;// create the
        database class object
        public static ClientInfoPanel cip;
        public static ConnectionsPanel cp;
        public static FaultsPanel fp;
        public static MachineInfoPanel mip;
        public static HelpPanel hp;

        public FRMainScr(){
        LeftPanel = new JPanel(new
        GridLayout(15,1));
```

```
//GridLayout, a method for laying out GUI components on screen
RightPanel = new JPanel(new BorderLayout());
RightPanel.setBackground(Color.WHITE);
Icon pix = new ImageIcon("FCpic.jpg");
RightPanel.add(new JLabel("", pix, SwingConstants.CENTER), BorderLayout.CENTER);
ClientInfoButton = new JButton("Client Info"); // Create Client Info button
ClientInfoButton.addActionListener(this);// make button to recieve click events
MachineInfoButton = new JButton("Machine Info");
MachineInfoButton.addActionListener(this);
FaultsButton = new JButton("Faults");
FaultsButton.addActionListener(this);
ConnectionsButton = new JButton("Connections");
ConnectionsButton.addActionListener(this);
RefreshButton = new JButton("Refresh");
RefreshButton.addActionListener(this);
HelpButton = new JButton("Help");
HelpButton.addActionListener(this);
ExitButton = new JButton("Exit");
ExitButton.addActionListener(this);
LeftPanel.add(new JPanel());
LeftPanel.add(ClientInfoButton);
LeftPanel.add(new JPanel());
LeftPanel.add(MachineInfoButton);
LeftPanel.add(new JPanel());
LeftPanel.add(FaultsButton);
LeftPanel.add(new JPanel());
LeftPanel.add(ConnectionsButton);
LeftPanel.add(new JPanel());
LeftPanel.add(RefreshButton);
LeftPanel.add(new JPanel());
LeftPanel.add(HelpButton);
LeftPanel.add(new JPanel());
LeftPanel.add(ExitButton);
LeftPanel.add(new JPanel());
MySplitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, LeftPanel, RightPanel);
MySplitPane.setDividerSize(2);
getContentPane().setLayout(new BorderLayout());
//BorderLayout, a method of laying out GUI components on screen
getContentPane().add(MySplitPane, BorderLayout.CENTER);
setSize(600,600);
setResizable(false);
setVisible(true);
setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
setDefaultLookAndFeelDecorated(true);
FRD = new FRDatabase();
cip = new ClientInfoPanel();
cp = new ConnectionsPanel();
fp = new FaultsPanel();
hp = new HelpPanel();
mip = new MachineInfoPanel();
}

public void actionPerformed(ActionEvent e){
    //Find out what button was clicked
if (e.getSource() == ClientInfoButton){
        MySplitPane.setRightComponent(cip);
}
if (e.getSource() == FaultsButton){
        MySplitPane.setRightComponent(fp);
}

if (e.getSource() == MachineInfoButton){
        MySplitPane.setRightComponent(mip);
  }
  if (e.getSource() == ConnectionsButton){
        MySplitPane.setRightComponent(cp);
  }
  if (e.getSource() == RefreshButton){
      //FRMainClass.MyServer.checkForDisconnection();
  MySplitPane.setRightComponent(RightPanel);
  }
  if (e.getSource() == ExitButton){
  FRMainClass.MyServer.closeConnections();
        System.exit(0);
  }
  if (e.getSource() == HelpButton){
      MySplitPane.setRightComponent(hp);
      }
  }
}
```

The Java code for the various options and other parts of the software application (going up to over one thousand lines of code) are left out of this write-up for want of space.

RESULTS

On completion, the database application was operated across a local area network put together specifically for the purpose of testing it. The network is comprised of three computers (an IBM ThinkPad laptop,

Fig. 3: Opening screen of the fault logging software

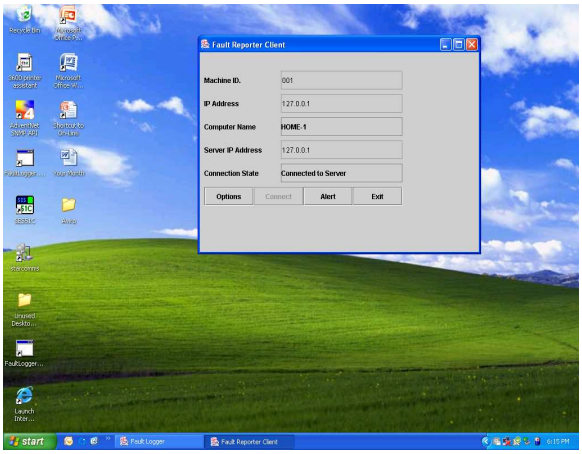

Fig. 5: Client screen, showing connection to server

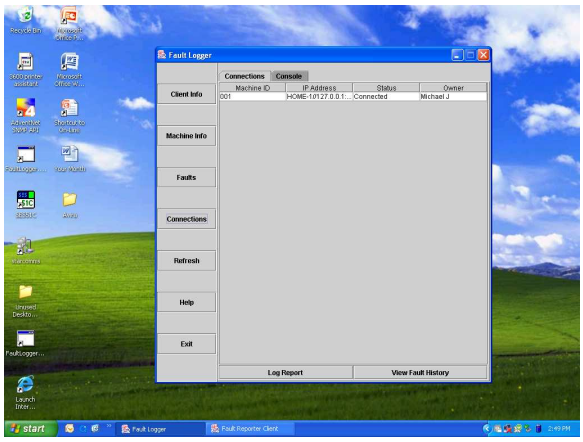

Fig. 4: Server screen with "connections" option selected, showing one client connected

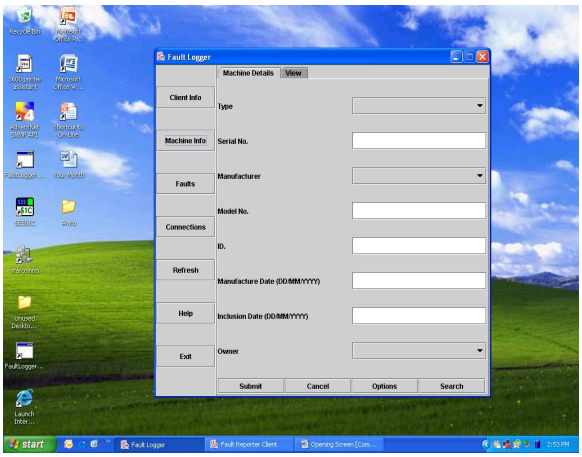

Fig. 6: Server screen with "Machine Info" option selected

a Brian Pentium 4 desktop, and a Gateway Pentium 4 desktop) in a star topology and client/server configuration. Once the network was properly set up, it was possible to log simulated faults manually from any client workstation unto the database on the server. Some of the screens presented by the application are shown in Fig. 3-6. The opening screen of the application (fault logging software) is shown in Fig. 3. Figure 4 shows the server screen when the "Connections" option is selected, and there is a client connected (logged on) to the server. The screen of the client when connected to the server is shown in Fig. 5.

In Fig. 6 is shown the server screen with the "Clients Info" option selected. Every option that is selected has its own screen, with provision for entering data or for querying the system. Some options are provided with sub-menus which allow for deeper and more detailed interaction with the system. This multi-layered arrangement makes it easier to navigate and use the system, and makes the system generally more user-friendly.

DISCUSSION

With this facility, a user at a remote location can send in a fault report, so long as the remote workstation is not totally down and can still communicate. The accumulated information can be used by the engineer at the maintenance company's office to predict the health of the remote workstation and plan in advance for its maintenance. However, the main purpose of the database is not to receive faults on-line, although that facility is useful for speeding up proceedings. Fault reports and remedial actions taken are entered into the database. The information forms a data bank to be used

to track the well-being of the dispersed workstations. The fault history of each machine would help engineers arrive at decisions rapidly concerning the current reported problem. The spare parts information held in the database would help speed up the process of securing any needed spare parts for the current job. The information on the track record of the machine owners, regarding their failure rate in the payment of legitimate charges, would be readily available to assist management decide the priority level to assign to such a machine owner. The information available about the physical locations of the machines would help engineers plan more effectively the logistics of getting to the machine as might be necessary. The accounting information held in the database would be very valuable in determining if the maintenance firm is gaining or losing in the entire endeavour, which is a core concern for any business venture. From the database it would be possible to identify profitable customers and the not so profitable ones, in order to direct the efforts of the organization appropriately. The database also gives an overall picture of the types and numbers of machines maintained by the organization, thus providing a good guide for decision on stock holding of spare parts. All these contribute to the efficient operations of the third party firm, to render quality service to the customers, and advance the cause of engineering practice.

CONCLUSION

This study has presented the Java developed application that is capable of assisting third party hardware maintenance firms keep track of their operations, including machines serviced, locations, and owners. The application, which makes use of Microsoft Access database and facilitates the logging of fault reports from clients unto the server, was operated successfully on a local area network. The application thus effectively enabled the transmission of faults information from remote workstations to the central computer. With proper Web enablement and adequate Internet availability the third party firm would be able to handle widely dispersed systems, and with the data held in the database, decisions on faults would be attended to more speedily. The firm would then be in a position to render more efficient service to its customers.

ACKNOWLEDGEMENT

The researchers wish to acknowledge Dr. M.B. Mu'azu and Dr. D.D. Dajab, both of the Department of Electrical Engineering, Ahmadu Bello University, Zaria, Nigeria, for their valuable assistance, and the Department itself for facilitating the study. Special thanks are extended to Prof. M.U. Onuu and Dr. S.O. Udoh, both of the Department of Physics, University of Calabar, Calabar, Nigeria, for their interest and encouragement, and to Mr. Cyril N. Mkpang for his interest in the Java adventure.

REFERENCES

1. Roger S. Pressman, 2001. Software Engineering: A Practitioner's Approach. 5th Edn., McGraw-Hill, Singapore, ISBN: 0073655783, pp: 860.
2. Williams, B.K. and S.C. Sawyer, 2004. Using Information Technology: A Practical Introduction to Computers and Communications. McGraw-Hill Irwin, Montreal, ISBN: 0071113789, pp: 512.
3. Roger S. Pressman, 2005. Software Engineering: A Practitioner's Approach. 6th Edn., McGraw-Hill, Boston, ISBN: 0072853182, pp: 912.
4. Deitel, H.M. and P.J. Deitel, 2007. Java How to Program. 7th Edn., Pearson Education Inc., Prentice-Hall, Upper Saddle River, New Jersey, ISBN: 0132222205, pp: 1596.