| Serial No. | ELE, SYLVESTER IGBO  
| Author 1 | PG/M.Sc./08/49326  
| Author 2 |  
| Author 3 |  
| Title: | DESIGN OF AN EXPERT SYSTEM FOR COMPUTER FAULT DIAGNOSIS AND TROUBLESHOOTING  
| Keyword: |  
| Description: | DEPARTMENT OF COMPUTER SCIENCE  
| Category: | FACULTY OF PHYSICAL SCIENCES  
| Publisher: |  
| Publication Date: |  
| Signature: | Digitally Signed by: Content manager’s Name  
| | DN : CN = Webmaster’s name  
| | O= University of Nigeria, Nsukka  
| | OU = Innovation Centre  
| | Ameh Joseph Jnr |
DESIGN OF AN EXPERT SYSTEM FOR COMPUTER
FAULT DIAGNOSIS AND TROUBLESHOOTING

BY

Ele, Sylvester Igbo
PG/M.Sc/08/49326

Being an M.Sc project report submitted in partial fulfillment of the requirement for
the award of a Master of Science degree in Computer Science of the University of
Nigeria.

DECEMBER, 2012
CHAPTER ONE
INTRODUCTION

1.0 Background

Computers are machines and the more we rely on them the more vulnerable we find ourselves when they fail. The consequences are damage to important data, waste of resources and frustration.

According to Ikekeonwu (2003), before one can use a computer, the computer must be operating. One sets the computer operating by starting (or booting) it. Many things can obstruct the operation and performance of the computer.

Troubleshooting and diagnosing a computer system is a knowledge-intensive task. Depending on the experience of the technician, a simple problem could take hours or even days to solve. An Expert System such as developed in this study offers a viable solution to the problem. According to Pomykalski, Truszkowski and Brown (1999), an Expert System is a computer program that is designed to imitate the decision-making ability of a decision maker in a particular narrow field of expert knowledge or skill.

The acquisition of knowledge in this research work is conducted through interviews with technicians in the computer repairs workshop and the Technical staff of Cross River State ICT development department (maintenance and engineering section), Calabar.

The objective of this research is to transform the knowledge of these experts into a knowledge base computer process that will provide recommendations to computer technicians and users base on the nature of the faults.
1.1 Statement of Problem
The following factors initiated this research work.

i. Computer boot failure due to virus attack, error in reading data from CD ROM, delay in loading programs, blank screen, keyboard error or even failure to access hard drive, communication errors, example, Bluetooth, Internet etc.

ii. End-users, vendors, maintenance personnel, etc. need a computer-based aid for the diagnosis of computer faults.

1.2 Objective of the study
The main objective of this research work is as follows:
To develop an expert system for diagnosing and troubleshooting personal computers (PC) faults that can assist PC end-users in dealing with their PC problems and also assist PC technicians in accurate diagnosis of PC fault by providing a systematic and step-wise analysis of failure, possible cause(s) of the failure and offer maintenance recommendations.

The specific objectives to meet the above are:

i. Examine the situation base on the user’s input to the system.

ii. Identify the problem and provide a systematic and step-by-step analysis of the causes of the problems

iii. Provide maintenance recommendations to users, and also guides them to get help from a more technical expert in situations which are less clear.

1.3. Scope of the study
This research work covers the design of an Expert System for diagnosing and troubleshooting PC faults using the Maintenance and Engineering section of the Cross River State ICT Development Department (ICTDD) as a case study. The
target is to develop a working model of personal Computer (PC) faults troubleshooting system (CFTS). The model will have as many contexts as possible, organize into problems, symptoms, possible causes and resolutions. The model software only diagnoses and offer maintenance advice to users and technicians, and does not heal the identified faults. Its conclusions are dependent on the evidence of the symptoms.

1.4 **Significance of the study**

- On completion, the application designed in this research work will be a helpful tool that will assists maintenance personnel and end-users for diagnosing computer faults.
- This work will be a benchmark upon which further research aimed at developing a more sophisticated computer-based aid for computer faults diagnosis can be carried out.

1.5 **Limitation of study**

Finance is a constraint that challenged the scope of this research work. Poor financial resources limit the number of research tools employed in this research, thereby restricting the case study to the maintenance department of Cross River State Information and Communication Technology Development Department (ICTDD).

There was also the challenge of designing a computer program that can perform at human level of competence.

Time factor also limited the scope to which the study could be conducted.
CHAPTER TWO
LITERATURE REVIEW

This chapter focuses on the review of the related literatures on Expert System, knowledge representation, and fault management.

2.0 Expert Systems

The specific task of an expert system is to be an alternative source of decision-making ability for organization to use instead of relying on the expert knowledge or skill of few people or just one person. The focus in the development of expert system is to acquire and represent the knowledge and experience of a person(s) who have been identified as possessing the special skill or mastery. (Pomykalski, Truszkowski and Brown, 1999:3). The primary intent of expert system technology is to realize the integration of human expert knowledge into computer process. This integration allows humans to be freed from performing the more routine activities that might be associated with a computer-base system.

Expert systems have caused revolution in the way we think about work, skill and their possibilities for computerization. Expert system addresses real needs (Doyle, 1984:1, 5). Knowledge is of central importance to expert system. Data, facts and information are terms used with the meaning of knowledge. The process of building an expert system is commonly known as knowledge engineering. Knowledge engineering implies acquisition of knowledge from a human or other source and coding it into the knowledge base of the expert system. (Giarratano and Riley, 1998:6-7).

According to Jones and Barrett (1989), Expert systems are not suited for all types of problems. Initially, many developers actively sought problems amenable to expert system solution or try to solve all problems encountered
using Expert system. Expert systems are verified specifically. Petrovic also noted that expert systems are tailored-made for specific and narrowly defined problem area. For example, a diagnostic expert system for troubleshooting computers must actually perform all the necessary data manipulation as a human expert would. The developer must limit his or her scope of the system to just what is needed to solve the target problems. Duke and Regenie (1985:1) admitted that expert system technology has been identified as a potential solution to some of these problems.

Several general approaches for developing expert system have been proposed. Waterman, (1986) has provided the most widely accepted approach: identification, conceptualization, formalization, implementation and testing. Despite its potential for bringing automation and computer aided decision making into areas that are time and information demanding, Tavama noted that expert systems are complex information systems that are expensive to build and difficult to validate. According to the author, several knowledge representation approaches such as rules, semantic networks, frames, objects and logical expressions have been developed to provide high-level concept of a system. Gathering, analyzing and modeling of knowledge are activities necessarily undertaken when developing expert system.

An expert system is usually designed to have the following general characteristics:

- The expert system must be reliable and not prone to crash.
- The system should be able to explain the steps of its reasoning while executing so that it is understandable.
- The system must be capable of responding at a level of competency equal to or better than that of an expert in the field (Lucas: 23).
Petrovic identified three major human elements in expert systems. These are the expert, the knowledge engineer and the user. According to him, expert system has two major functions:

- To draw conclusions and
- To explain its reasoning.

According to Lucas, one of the first systems with which expert system has been associated is DENDRAL. The DENDRAL project commenced in 1965 at Stanford University. J. Lederberg, an organic chemist, in conjunction with E. A. Feigenbaum and B. A. Buchanan, developed it. DENDRAL helps in interpreting the patterns in a spectrogram to identify chemical constituents.

The work of DENDRAL led to many other successful application of the new technology known as expert system. The best-known expert system in this early period of the fields was MYCIN. Bruce Buchanan and Dr. Edward Shortliffe developed MYCIN in the late 1970s to diagnose blood infections (Pomykalski, Truszkowski and Brown, 1999: 6-9). Using about 450 rules, MYCIN was able to perform as well as some experts, and considerably better than some junior doctors were. MYCIN is one of the most widely known expert system applications developed. In addition, other significant expert system applications were developed in the early days of expert system. These systems include DIPMETER which helps in analyzing geological data for oil, PROSPECTOR-for analyzing geological data for oil and minerals, PUFF, which used MYCIN in the field of Pulmonary disorders, DELTA/CATS, which was developed at GENERAL Electric Company to assist railroad personnel in the maintenance of GE’s diesel electric locomotives. John McDermott at CMU developed XCON, originally titled R1, for aiding in the configuration of VAX and PDP-II computer systems at DEC (Pomykalski, Truszkowski and Brown, 1999: 6-9).
Diagnosis involves the correlation of information from a variety of sources. Most of the approaches to correlation involve matching gathered information with known patterns or model of faults. The technique base on artificial intelligence to bring out facts is called knowledge acquisition. Knowledge acquisition is the process by which the knowledge of a human expert is being extracted by a knowledge engineer.

(Bassey, 2009:36)

Fault diagnosis is very important in the field of computer engineering and information technology, especially in personal computer (PC) troubleshooting. Today, the use of computers is widespread. However, the knowledge in computer troubleshooting is limited, and this poses difficulties among organizations when faced with computer problems. This expert system is developed to assist and advise computer users in the diagnosis and troubleshooting of computer system.

A literature survey/review was carried out to look into specific applications of expert systems which bear similarities with the “computer fault diagnosis and troubleshooting system” under this research. The specific applications found are:

I. The work by Goh Wee Leng and Lau Kim Teen on ESPCRM, an expert system for personal computer repair and maintenance, 2003. This study describes the design and implementation of an expert system for personal computer repair and maintenance. It provides consultation for the repair and maintenance of the whole series of IBM/IBM compatible PCs from the XT to 486-based machines. The study bears some similarities to the “fault diagnosis and troubleshooting system” developed in this research work in that the database includes problems, symptom and diagnostic method and expert information.
II. The work of NarendraDev and Bart Anderson on Pimtool, an expert system to troubleshoot computer hardware failure, 1997 (conference proceeding).

This study describes a tool to diagnose the cause of failure of HP computer server. They do this by analysing the PIM (Processor Internal memory) dump and maximising the leverage expert learning from one hardware failure to another. The troubleshooting information is called PIM.

2.1 Application Areas

There are two different ways developers look at application areas for expert/knowledge-based systems. First, they look at the functional nature of the problem. Secondly, they look at the application domain. This study reviews both of these ways to get a better understanding for the application of expert/knowledge-based systems to “real-world” problems. In 1993, John Durkin (in Pomykalski, et al, 1999) published a catalogue of expert system applications that briefly reviews a number of applications of expert/knowledge based system technology and categorizes each of the nearly 2,500 systems. Both MYCIN and XCON point out two different functions that are viewed as highly favourable for expert/knowledge-based system development. MYCIN mainly deals with the diagnosis of a disease given a set of symptoms and patient information. XCON, on the other hand, is a synthesis-based (design) configuration expert system. It takes as its input the needs of the customer and builds a feasible arrangement of components to meet the need. Both of these systems solve different generic “types” of problems. An expert system may have many differing functions. It may monitor, detect faults, isolate faults, control, give advice, document, assist, etc. The range of applications for expert system technology ranges from highly embedded turnkey expert systems
for controlling certain functions in a car or in a home to systems that provide financial, medical, or navigation advice to systems that control spacecraft. The table below lists the ten different types of problems generally solved by expert/knowledge-based systems. Within each problem type expert perform a generic set of tasks, such as diagnosis or planning.

Table 2.1 Expert/Knowledge-Based System Application Areas

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>Governing system behaviour to meet specifications</td>
</tr>
<tr>
<td>Design</td>
<td>Configuring Objects under constraint</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Inferring System Malfunction from observables</td>
</tr>
<tr>
<td>Instruction</td>
<td>Diagnosing, debugging, and repairing student behaviour</td>
</tr>
<tr>
<td>Interpretation</td>
<td>Inferring situation description from data</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Comparing observations to expectations</td>
</tr>
<tr>
<td>Planning</td>
<td>Designing actions</td>
</tr>
<tr>
<td>Prediction</td>
<td>Inferring likely consequences of a given situation</td>
</tr>
<tr>
<td>Prescription</td>
<td>Recommending solution to system malfunction</td>
</tr>
<tr>
<td>Selection</td>
<td>Identifying best choice from a list of possibilities</td>
</tr>
</tbody>
</table>

2.3 Expert System Architecture

The architecture of an expert system comprised of a kernel augmented by knowledge acquisition tools, user interface and explanation facility.

Figure.1.1 displays the architecture of expert system.
2.4 **Elements of an Expert System**

The elements of a typical expert system are shown in figure.1.1. From existing literatures, it is clear that an expert system consists of the following components.

a. **User interface**: Communication between the user and the system is done through the user interface (Al-Taani, 2005:5).

b. **Explanation sub-system**: it is used by the expert system to explain its reasoning to user. It explains its advice or recommendations, and justify why a certain action was recommended (Adhikari*et al.*, 2008).

c. **Knowledge acquisition sub-system**: it is used by the expert system to acquire new facts and rules. New rule can be added to the knowledge base.
base by using the knowledge acquisition sub-system (Adhikari et al., 2008).

d. **Inference engine:** it is the part of the expert system that support reasoning about the environment by proper management of its rules and facts. The inference engine matches facts in the working memory against rules in the rule base and it determines which rules are applicable according to the reasoning method adopted by the engine (Adhikari et al., 2008).

e. **Knowledge base sub-system:** the knowledge base sub-system of the expert system stores the extensive knowledge gathered from experts, historical data and books regarding the application in the form of rules (Babita et al., 2009). It is a declarative representation of expertise, often in IF THEN rules (Adhikari et al., 2008).

2.5 **What is Fault Management?**

Fault management has conventionally been defined as the detection, diagnosis and correction of a fault or problem. Typically, the system is monitored to enable automatic detection (Su Myat may, 2008; 34). Identifying a problem is a very difficult task because of complexity of the computer system and the constant change. The systems are functionally complex as a whole, and additional complexity is introduced through the interactions of different components.

2.6 **Knowledge Acquisition**

Acquisition of field knowledge is a major problem in the development of an expert system. Every expert system program requires an evaluation of the existing data and information that form the knowledge base for the exact problem. The task of acquiring engineering knowledge includes the following
aspects: decomposition of the main field problem, integration of different data, use of different sources of knowledge, combining of doubtful information, organization and formulation of extracted knowledge, etc.

Computer based tools are available to assist in knowledge based construction, maintenance and documentation. For difficult problems where variety of sources and types of data and information are involved, computer based systems are very useful for organizing existing data and information. Consequently, knowledge acquisition techniques are tools used for bringing out field knowledge from an expert (Barai and Pandy, 2004:171).

2.7 **Knowledge Representation**

There are several ways to store knowledge for later use. Naturally, though, we wish to use computers to operate on the knowledge. Natural language is poor choice of doing this due to its lack of procedures and its implied syntax and semantics. A formal language which contains mappings from syntax to a computer recognizable symbol as in programming languages is best for this purpose.

Knowledge has different forms. It can be certain or uncertain, structured or unrelated. It can be found in formulas, tables, statements or well established traditional practices (Barai& pandy, 2004: 172). The issues that affect the selection of knowledge representation are efficiency of storage, consistency and naturalism of representation.

In the field of artificial intelligence, common approaches to knowledge representation include:

- Frames
- Production rules
- Predicate logics.
Frames

A frame is a structure that represents a concept. It can have any number of attributes or properties attached to it. Some of the properties can be relationship. An attribute may have any number of values. Other features of a frame include:

i. A frame contains information about prototypical instances
ii. Reasoning with frames is a process of matching prototypes against specific individuals.

Reasoning involves drawing conclusion from known knowledge. To automate reasoning one need to have a language for representing knowledge and a mechanism for manipulating knowledge in order to draw a conclusion. One powerful way for representing and manipulating, and thus reasoning about knowledge is predicate logic and theorem proving.

ProductionRule

In a journal published in 2004, Barai and Pandy referred to Hayes-Roth et al., 1983, which said that “rule base systems are composed of rules, working memory and a rule interpreter. In their simplest form, rules are condition-action pairs. Each rule represents a portion of knowledge”. Production rules are the most widely knowledge representation formalism available in expert system shells (Barai and Pandy, 2004:173).

PredicateLogic

Reasoning involves drawing conclusion from known knowledge. To automate reasoning one need to have a language for representing knowledge and a mechanism for manipulating knowledge in order to draw a conclusion. One powerful way for representing and manipulating, and thus reasoning about knowledge is predicate logic and theorem proving.

Deciding on how knowledge will be represented depends on the task at hand. Production rules are the most widely knowledge representation techniques in expert systems.
This chapter presents the detailed development information of the computer diagnosis and troubleshooting system.

3.1 **Design Model**

The design model adopted in this work is the structured system analysis and design methodology (SSADM). The structure of the SSADM (waterfall model) is represented in the figure below.

*Figure 2.2* The structure of the waterfall model.

*Source: Pearson Education Limited, 2004*
3.2 **ObjectOrientedDesignwithUML**

UML (Unified Modeling Language) is used for the model design of this system. The UML is now the most widely used graphical representation scheme for modeling object systems. An attractive feature of the UML is its flexibility. UML modelers are free to use various processes in designing systems. The UML is a complex, feature-rich graphical language (Deitel, H.M. and Deitel, P.J., 2004).

The UML specifies diagram for documenting the system behaviour. In this research work, three UML diagrams (use case, class and activity diagrams) are used to explain the behaviour of the expert system as shown in the figures below.

### 3.2.1 **UML UseCaseDiagram of the Fault Diagnosis System**

The use case diagram model the interaction between the user and the system.

![User Interface Function](Figure 3.2 Use Case Diagram showing the interaction between the user and the system.)
3.2.2 UML Class Diagram of the Fault Diagnosis System

The classes required to build the package ‘fault diagnosis expert system’ are identified and are implemented using java language. These are mainframe (working memory), Problem_category, Problem_table, Database_operation, and the baseform (main screen) classes. These classes and their relationships are described using UML class diagram as shown in figure 1.2 below. The multiplicity value 0…* at the problem_category end of the association between class problem_category and class databaseoperation indicates that zero or more objects of class problem_category take part in the association. Class databaseoperation has a one-to-many relationship with class problem_category. Similarly, class problem_category has a many-to-one relationship with class databaseoperation. There can be many problem categories stored in database. The same relationships exist between class problem and class databaseoperation. See figure below.
Figure 3.3. Class diagram for the expert system model showing composition relationships.
3.2.3 **UML Activity Diagram of the Fault Diagnosis System.**

**Figure 1.3.** Activity diagram of the Computer Troubleshooting Expert System.

- Display main menu
- Prompt user to select problem category
- (No problem in problem category selected)
- (Problems in problem category selected?)
- Prompt user to select problem in problem category
- (No troubleshooting tips in database)
- (Troubleshooting tips for problem exist in database)
- Display troubleshooting tips to user
- Display empty screen
3.3 **The Database Design of the Fault Diagnosis Expert System**

The database used in this design is Mysql. The database has three tables, the problem, problem_category and problem tips tables.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>FIELD NAME</th>
<th>TYPE</th>
<th>WIDTH</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Problem</td>
<td>Varchar</td>
<td>80</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Problem_Category</td>
<td>Varchar</td>
<td>80</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Tips</td>
<td>Text</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.1 **ProblemTable**

The problem table has two fields.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>FIELD NAME</th>
<th>TYPE</th>
<th>WIDTH</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Problem_Category</td>
<td>Varchar</td>
<td>80</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Label</td>
<td>Varchar</td>
<td>80</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 3.2 **ProblemCategoryTable**  
The problem_category table has one field.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>FIELD NAME</th>
<th>TYPE</th>
<th>WIDTH</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Label</td>
<td>Varchar</td>
<td>80</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.3 **ProblemTipsTable**  
The problem tips table has two fields, the problem label and the body.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>FIELD NAME</th>
<th>TYPE</th>
<th>WIDTH</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Problem_label</td>
<td>Varchar</td>
<td>80</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Body</td>
<td>Text</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
CHAPTER FOUR

SYSTEM IMPLEMENTATION AND METHODOLOGY

This section presents the analysis and design of the new system, application objectives, program specifications, the choice and justification of the programming language platform, and architecture of the software.

4.1 Analysis of the Present System

It can be quite frustrating to have a computer that takes a longer time to complete even the simplest of tasks, especially if it had previously been a lot faster. The non-specialist field service engineers (that is, technicians) and application users are only trained to use existing component specific of the system. As a result, identification of single or multiple faults from general manifestations remain a difficult task. The inability of computer users to communicate the exact nature of their computer problems to the technicians or system engineers, and the inaccurate diagnosis of computer problems by most system engineers and technicians is a major set-back in the use of manual process in fault diagnosis. The model software developed in this work can only diagnoses and offer maintenance advice to users and technicians, and does not heal the identified faults. A more sophisticated computer-based fault diagnosis system with the ability to identify and fixed the problems should be developed using this work as a benchmark.

4.2 Design of the Expert System

The design and development of the expert system will utilize methodologies consistent with the design and development of conventional applications, modified for areas specific for expert systems such as knowledge acquisition, verification and validation. Research in computer fault management, expert
systems, knowledge acquisition, knowledge acquisition tools, and multiple domain experts will be used to provide a basis for the knowledge needed by the knowledge engineer to successfully complete the design and development of the expert system.

4.2.1 Application objectives
The everyday occurrence of loss of data due to frequent break down of computers, the inability of computer users to communicate the exact nature of their computer problems to the technicians or system engineers, and the inaccurate diagnosis of computer problems by most system engineers and technicians were the situations that initiated this research work. The main objective of the expert system developed in this work is to assist PC end-users in dealing with their PC problems personally, and also assist PC technicians in accurate diagnosis of PC faults by providing a systematic and step-by-step analysis of failure, possible cause (s) of the failure and offer maintenance recommendations.

4.2.2 Program Specifications
The software and development experiences along with testing provided by this work have been varied and interesting. It has given a taste of real world design and importantly, it has been learnt that attention to detail and systematic testing methods are the key to a successful project. While individual software modules (classes and GUI) are developed, unit testing is performed on these components using dummy called modules.
4.2.3 **Program Modules**

The main modules (classes) of this system are Administrator (working memory) module, the User Interface module, problem category module, database definition module, and database operation module (class).

4.2.3 **Input/output Forms**

4.3.3.1 **Input**

Communication between the user and the system is done through user interface. The user interface is represented as a menu which displays lists of PC problems for users to select based on the nature of their problems. The input will be entered using the keyboard as an input device.

4.3.3.2 **Output**

A user friendly interface was designed in the system. The user has launched the application after finishing all required installations. When the application has started, it displays a welcome message printed in English language and prompts user to ‘Click here to continue’. When the user clicked the ‘click here to continue’ button, system login form appears prompting the user to enter application code (user password). When user typed password wrongly, it responds with an error message.
Figure 4.1 System flowchart of the Diagnosis Expert System
4.3.0 **Definition of Algorithm Using Pseudo**

The algorithm for the modules (classes) identified is presented below using pseudo code.

**Administrator (WorkingMemoryClass)**

1. Application setup
   1.1 set up the application environment
   1.2 load the problem category and add troubleshooting tips to panel
   1.3 WHILE (problem categories and troubleshooting tips exist in database)
   1.4 Fetch the problem and its tips
   ELSE
   1.5 Display error message
   1.6 IF fetch is successful THEN
   1.7 Update the form control with output
   1.8 ELSE
   1.9 Display failure message

2. Add problem
   2.1 Start application
   2.2 IF problem is selected and new problem name entered THEN
   2.3 Enter problem label into the problem label field in panel
   2.4 IF (problem label is successfully added to working memory) THEN
   2.4 System.out.print (‘problem added successfully’)
   ELSE
   2.5 System.out.print (‘sorry, failed to add problem’)

3. Add new problem category
   3.1 Start application
   3.2 Enter new problem category name
3.3 IF problem category name is successfully added THEN
3.4 System.out.println(‘problem category is successfully added’)  
3.5 Reload problem category inform combo
3.6 IF problem is successfully loaded THEN
3.7 Update combo
3.8 ELSE
3.9 Show error message

4. Delete problem category
   4.1 To delete problem category
   4.2 IF problem category is selected THEN
   4.3 Enter problem category name and select delete
   4.4 Is the selected problem category deleted success from database? THEM
   4.5 System.out.println(‘problem category deleted successfully) 
   4.6 ELSE
   4.7 System.out.println(‘sorry, problem category failed to delete) 
   END

5. Add troubleshooting tips
   5.1 Select problem category
   5.2 Load the problem under problem category selected
   5.3 IF problem under problem category exist THEN
   5.4 Load problem combo with output
   4.5 ELSE
   5.5 Set problem combo = “reset” and Set up editor for tips
   5.6 IF tips exists in database THEN
   5.7 Filter all tips from database associated with selected problem ELSE
   5.8 Return empty editor and prompt user to enter troubleshooting tips for the selected problem
6. Save troubleshooting tips
   6.1 Startup application
   6.2 Enter problem name
   6.3 IF tips already exist for selected problem THEN
   6.4 Update database ELSE
   6.5 Enter tips and select ‘save’
   6.6 IF tips is successfully saved in database THEN
   6.7 System.out.print (‘this tips is successfully saved’) ELSE
   6.8 System.out.print (‘sorry, save is not successful’)
   END

**Baseform (User Interface) Class**
This class provides the interface from which the user interacts with the expert system.

1. Set up environment for baseform
   1.1 Set problem category method
   1.2 Declare an arraylist and connect to the database to fetch all problems category registered
   1.3 Assign database result to the arraylist
   1.4 Setup the problem categorytablemodel and pass the arraylist to it
   1.5 Get a reference to the form’s problem category table and pass the problemcategorytable to it

2. To get problem under problem category
   2.1 Get selected index from problemcategorytable
   2.2 Get selected value base on selected index
   2.3 IF value = ‘empty string’ THEN
   2.4 Halt execution
3. Connect to database and get all problems registered under selected problem category
   3.1 Put the result in an iterator
   3.2 Get a reference to the problem combo and add each item in the result to it
   3.3 WHILE (result iterator has more content)
   3.4 Add each item to the problem combo
   END

**DatabaseOperation module (Class)**

This class manages the data in the database.

1. Initialize the database operation and get problem category
   1.1 Set connection = null
   1.2 Set preparestamentpstr = null
   1.3 Call the Arraylist method
   1.4 Connect to database and fetch all from registered problem_category
   1.5 Get reference to pstr and initialize result set counter = 0
   1.6 WHILE (result set has next)
   1.7 Get result from problem_category and add it to the list ELSE
   1.8 Display error message
   1.9 Return the list to the Arraylist method

2. Get problems
   2.1 Call Arraylistgetproblem method
   2.2 Connect to database and fetch all problems under a specified problem_category
   2.3 Set pstr and get reference to result set
   2.4 WHILE (result set.next())
   2.5 Get result from problem and add it to Arraylist method ELSE
   2.6 Display error message
2.7 Return result to ArrayList method

3. Get Problem Tips
   3.1 Call ArrayList and getproblemtips
   3.2 Connect to database and fetch all problems under a specified problem_category
   3.3 Set preparedstatement and get reference to result set
   3.4 WHILE (result set has more items)
   3.5 Add item to list ELSE
   3.6 Display error message
   3.7 Return item in list to ArrayList method

1.2.6 ProblemcategoryTablemodule (Class).
   1. Initialize the class ProblemcategoryTablemodel
      1.1 Set string column = “empty”
      1.2 Set datatype = “string”
      1.3 Set the ArrayList for the class
      1.4 Set the problemcatmode
   2. Instantiate the problemcategoryTablemodel method
      2.1 Get data to prob
      2.2 fireTablestructurechanged
   3. set ArrayList method to getdetails
      3.1 Return the data to ProblemcategoryTablemodel method
      3.2 Enter integer getRowcount
      3.3 Return the data size
      3.4 Enter integer getColumncount
      3.5 Return Column length
      3.6 Return

END.

4.4.0 Data Dictionary
This presents the data dictionary of the data base design in chapter three above.

**VARCHAR**: A variable-length character data, with maximum size of 80 bytes. It is used to store alphanumeric data.

**NOTNULL**: It is an Integrity constraints defined for the columns of the table that absolutely requires values all the time.

**TYPE**: Datatype of the value in the table. Datatype of a column must be either the same as, or convertible to, the datatype of variable.
4.5.0 **Choice and Justification of Programming Language Platform Used**

Java programming language platform was used to implement this system. The choice of the programming language was as a result of features supported by the language. Java is a powerful programming language, it is portable, robust, multi-platform enabled, has rich library, simple, etc. (Su & May). The key features are inheritance, polymorphism, reusability, knowledge representation, integration or extensibility. Java support procedural programming. Java is used for opened distributed application, has rich type system and is an object-oriented language (Adams, D. et al).

Net Bean is used to draw the user interface (UI) design. It is efficient, effective and reduces time consuming to draw UI in writing desktop application with java.

4.5.1 **Development Tools**

<table>
<thead>
<tr>
<th>Tool/Language</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. JDK</td>
<td>Programming language</td>
</tr>
<tr>
<td>2. MYSQL</td>
<td>Database management tool</td>
</tr>
<tr>
<td>3. Net Bean</td>
<td>User interface (UI) programming</td>
</tr>
<tr>
<td>4. Window XP/Window 7</td>
<td>Platform.</td>
</tr>
</tbody>
</table>
4.6.0 **System Requirements**

System requirement refers the necessary constituents that are needed for a system to achieve its aims and objectives. For this software to run therefore, certain requirements must be met.

4.6.1 **Hardware Requirements**

- Pentium III 1500MHz and above
- 15 GB HD and above
- 128 MB RAM and above
- VGA Monitor and above
- Enhanced keyboard

4.6.2 **Software Requirements**

- Window XP Professional and any other version
CHAPTER FIVE

SUMMARY, RECOMMENDATION AND CONCLUSION

5.1 Summary

An expert system was developed to diagnose and troubleshoot computer fault. The primary objective of this study is to develop an expert system that will assist users in computer fault troubleshooting. End-users, vendors and maintenance personnel need a computer based aid for the diagnosis and troubleshooting of computer fault.

Chapter one presented the objective, significance of the study and statement of the problem. A detailed survey of related literatures of different author on expert system is also presented. This chapter also captured several different approaches for developing expert system.

Chapter two concentrated on the theoretical background of the study. The theory of the expert system explaining its core and major concepts, such as fault management, expert system, expert system architecture, computer based tools for knowledge acquisition is presented. The chapter also examined several ways to store knowledge for later use, and identified common approaches to knowledge representation, such as frames, production rules and predicate logics.

Chapter three presented the analysis and design of the fault diagnosis expert system. It surveyed in detail object oriented design model with UML, specified the UML diagrams (such as the Use case, activity and class diagrams) for documenting the expert system behaviour and described the relationship between classes in the system. The chapter also explain the database tool used in this research work.
Chapter four presents the architecture of the expert system software, development environment, deployment platform and screenshots of the output showing the working of the fault diagnosis expert system.
5.2 RECOMMENDATION

This research work developed an Expert System diagnosing PC problems. Due to the constraints that have been identified, the work done here cannot be validated to be exhaustive; as such it is recommended that more researches aim at designing a more superior application for diagnosing and troubleshooting PC faults should be carried out.

Findings in the area of knowledge acquisition and knowledge representation can be validated by either duplicating this work or utilizing it in similar studies.

User friendliness of the system and effectiveness of the explanation function can also be validated.
5.3 Conclusion
The design of an expert system for computer fault diagnosis and trouble shooting is presented in this work. A brief description of expert system architecture and issues involved in developing expert system shell and the technology used is discussed.

An expert system shell is a package that facilitates the building of a knowledge-based expert system by providing a knowledge representation scheme and an inference engine. According to Bohanec and Rajkovic, expert system shell provides tools for building and verifying a knowledge base, evaluating options and explaining the result. Classic examples are DEX, an expert system shell for decision support and EMYCIN (empty MYCIN) shell, which was made by removing the medical knowledge base of the MYCIN expert system. Associated with a shell is a prescribed method for building applications by configuring and instantiating these components. Some of the generic components of a shell are knowledge base, reasoning engine, knowledge acquisition system, explanation system and user interface, the core components of expert systems shell are the knowledge base and the reasoning engine (Giarratano and Riley, 1998).

The package is developed on the basis of experts experience in computer fault diagnosis and troubleshooting. The package is developed to have the characteristics of good expert systems, such as high performance and adequate response time. Thus rendering cost-effective solution in diagnosing fault that may occur. The package is user friendly.
References
Fundamental, 10g Release 1(10.1)”, 2003. Oracle Corporation.

World Academic of Science, Engineering and Technology, December, 2005.

for Fault Diagnosis and Control of Power System Equipment”,
International Conference on Condition Monitoring and Diagnosis, Beijing, China, April, 2008.

(A project proposal presented to Advanced College of Engineering and
Management, Tribhun University), September, 2008.

Barai, S. and Pandy P. C., ” Knowledge Based Expert System Approach TO

Fault Diagnosis”, A Research conducted under a join study Agreement
between international business machine corporation and Standford
University.

Deitel, H.M. and Deitel, P.J., “Java™ How to Program 6/ed”, Prentice Hall,
2004.

Dev, N. and Anderson, B. “Pimtool, An Expert System to Troubleshoot
Computer Hardware Failures”, Conference Proceeding, in The America


in The national Aeronautics and Space Administration”, Ames Research


Petrovic, S. “Decision Support Methodologies”, School of Computer Science, University of Nottingham.


APPENDICES

APPENDIX A: Program Codes

THIS CLASS HANDLES THE EXPERT SYSTEM ADMINISTRATOR (WORKING MEMORY) APPLICATION.

```java
package com.expert.system;

import java.util.Iterator;
import javax.swing.JOptionPane;
import javax.swing.JFrame;

public class MainFrame extends javax.swing.JFrame {

    public void setup() {
        Iterator iterate = DatabaseOperation.getProblemCategories().iterator();
        try{
            this.comboProblemCat.removeAllItems();
            this.comboProblemCatTips.removeAllItems();
            this.comboProblemCat.addItem("select");
            this.comboProblemCatTips.addItem("select");
            while(iterate.hasNext()) {
                String item = iterate.next().toString();
                this.comboProblemCat.addItem(item);
                this.comboProblemCatTips.addItem(item);
            }
        }catch(Exception e) {
            
        }
    }
}
```
public MainFrame() {
    initComponents();
    setLocation(20, 100);
    setup();
}

private void btnAddProblemCatActionPerformed(java.awt.event.ActionEvent e)
    if (this.txtProblemCat.getText().equals"")
        { JOptionPane.showMessageDialog(this, "Please type the problem category you want to add, thank you");
            return;
        }
        if (DatabaseOperation.addProblemCategory(this.txtProblemCat.getText()))
            { setup();
                JOptionPane.showMessageDialog(this, "Problem category has been added successfully");
                this.txtProblemCat.setText"");
                this.txtTipEditor.setText"");
                this.btnSaveTip.setEnabled(false);
            }
        else
            JOptionPane.showMessageDialog(this, "Sorry Failed to add problem category, Make sure the database connections are set.");
    }
    private void btnAddProblemActionPerformed(java.awt.event.ActionEvent e)
        { // TODO add your handling code here:
            if (this.comboProblemCat.getSelectedIndex() == 0)
                { JOptionPane.showMessageDialog(this, "Please select the category you want to add problems to.");
            }
return;
}
else if(this.txtProblem.getText().equals(""))
{
    JOptionPane.showMessageDialog(this, "Please type the label of the problem you want to add.");
    return;
}

if(DatabaseOperation.addProblem(this.comboProblemCat.getSelectedItem().toString(), this.txtProblem.getText()))
{
    JOptionPane.showMessageDialog(this, "Problem label has been added successfully.");
    this.txtProblem.setText("");
    this.comboProblemLabelTips.removeAllItems();
    this.comboProblemCatTips.setSelectedIndex(0);
}
else
JOptionPane.showMessageDialog(this, "Sorry Failed to add problem, Make sure the database connections are set.");

private void comboProblemCatTipsItemStateChanged(java.awt.event.ItemEvent evt) {
    try
    {
        if(this.comboProblemCatTips.getSelectedIndex() == 0)
        {
            this.comboProblemLabelTips.removeAllItems();
            this.txtTipEditor.setText("");
            this.btnSaveTip.setEnabled(false);
            return;
        }
    }
    else
    {
        // Code continues here
    }
}
Iterator iterate =
DatabaseOperation.getProblems(this.comboProblemCatTips.getSelectedItem() .toString()).iterator();
this.comboProblemLabelTips.removeAllItems();
this.comboProblemLabelTips.addItem("select");
while(iterate.hasNext())
{
    this.comboProblemLabelTips.addItem(iterate.next());
}
}
} catch(Exception xcp)
{
}
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
if(this.comboProblemCatTips.getSelectedIndex() == 0)
{
    JOptionPane.showMessageDialog(this, "Please select problem category!");
    return;
}
else if(this.comboProblemLabelTips.getSelectedIndex() == 0)
{
    JOptionPane.showMessageDialog(this, "Please select problem label!");
    return;
}
if(DatabaseOperation.checkProblemTip(this.comboProblemLabelTips.getSelectedItem().toString()))
{
    String tips = "";
    StringBuffertipsPack = new StringBuffer(tips);
    Iterator iterate =
DatabaseOperation.getTroubleshootTips(this.comboProblemLabelTips.getSelectedItem().toString()).iterator();
while(iterate.hasNext())
{
tipsPack.append(iterate.next()+"\n");
}
this.txtTipEditor.setText(tipsPack.toString());
this.txtTipEditor.setEditable(true);
this.btnSaveTip.setEnabled(true);
}
else
{
this.txtTipEditor.setText(""");
this.txtTipEditor.setText("Troubleshooting tips for
:"+this.comboProblemLabelTips.getSelectedItem().toString()+"\n\n");
this.txtTipEditor.setEditable(true);
this.btnSaveTip.setEnabled(true);
}
}

private void btnSaveTipActionPerformed(java.awt.event.ActionEventevt) {
if(this.comboProblemLabelTips.getSelectedIndex() == 0)
{
JOptionPane.showMessageDialog(this, "Please select problem label!");
return;
}
else if(this.txtTipEditor.getText().equals(""))
{
JOptionPane.showMessageDialog(this, "Please select problem label!");
return;
}
if(DatabaseOperation.addTroubleshootTip(this.comboProblemLabelTips.getSelectedItem().toString(), this.txtTipEditor.getText()))
{
JOptionPane.showMessageDialog(this, "Troubleshooting tip has been saved successfully");
}
else
JOptionPane.showMessageDialog(this, "Sorry, could not save troubleshooting tip!");
}
private void comboProblemCatTipsActionPerformed(java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
}
private void btnDeleteProblemCatActionPerformed(java.awt.event.ActionEvent evt) {
if(this.comboProblemCat.getSelectedIndex() == 0)
   return;

if(this.comboProblemCat.getSelectedIndex() != 0)
{
   int flag = JOptionPane.showConfirmDialog(this, "Deleting this category will result in the lost of all problem label and troubleshooting tip associated with them, click yes to continue","Confirm delete", 0);
   if(flag == 0)
   {
      if(DatabaseOperation.deleteProblemCat(this.comboProblemCat.getSelectedItem().toString()))
      {
         JOptionPane.showMessageDialog(this, "Problem category has been deleted sucessfully.");
         setup();
      }
      else
         JOptionPane.showMessageDialog(this, "Sorry the was a problem deleting the problem category you selected!");
   }
   }
private void btnDeleteProblemLblActionPerformed(java.awt.event.ActionEvent evt) {
if(this.comboProblemLabelTips.getSelectedIndex() == 0)
    {
    return;
    }
else if(this.comboProblemLabelTips.getSelectedIndex() != 0)
    {
    int flag = JOptionPane.showConfirmDialog(this, "Deleting this problem label will result in the lost of all troubleshooting tip associated with it, click yes to continue", "Confirm delete", 0);
    if(flag == 0)
        {
        if(DatabaseOperation.deleteProblem(this.comboProblemLabelTips.getSelectedItem().toString()))
            {
            JOptionPane.showMessageDialog(this, "Problem label has been deleted successfully.");
            setup();
            }
        }
    else
    JOptionPane.showMessageDialog(this, "Sorry the was a problem deleting the problem label you selected!");
    }
}
private void comboProblemLabelTipsActionPerformed(java.awt.event.ActionEvent evt) {
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        newMainFrm().setVisible(true);
    }
});

\"THIS CLASS HANDLES THE BASEFORM (USER INTERFACE) APPLICATION.\"

package com.expert.system;

import java.util.ArrayList;
import java.util.Iterator;

public class BaseFrm extends javax.swing.JFrame {

    static {
        try{
            javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.metal.MetalLookAndFeel");
        } catch(Exception e){}
    }

    public BaseFrm() {
        initComponents();
        setUp();
        this.setLocation(20, 20);
    }

    private void setUp() {
        ArrayList<ProblemCat> problemCatList = DatabaseOperation.getProblemCategories();
        ProblemCatTableModel tblModel = new ProblemCatTableModel(problemCatList);
    }
this.tblProblemCat.setModel(tblModel);
this.txtBody.setEditable(false);
this.tblProblemCat.setFillsViewportHeight(true);
}

private void tblProblemCatMouseClicked(java.awt.event.MouseEvent evt) {
    try {
        int selRow = this.tblProblemCat.getSelectedRow();
        String val = this.tblProblemCat.getValueAt(selRow, 0).toString();

        if(val.equals(""))
            return;

        Iterator iterate = DatabaseOperation.getProblems(val).iterator();
        this.comboProblem.removeAllItems();
        this.comboProblem.addItem("select");
        while(iterate.hasNext())
            this.comboProblem.addItem(iterate.next().toString());
    }
    catch(Exception xcp) {}
}

private void comboProblemItemStateChanged(java.awt.event.ItemEvent evt) {
    try {
        if(this.comboProblem.getSelectedIndex() == 0)
            this.txtBody.setText(""),
            return;
    }
    catch(Exception xcp) {}
}

private void comboProblemItemStateChanged(java.awt.event.ItemEvent evt) {
    try {
        if(this.comboProblem.getSelectedIndex() == 0)
            this.txtBody.setText(""),
            return;
    }
    catch(Exception xcp) {}
}
Iterator iterate = DatabaseOperation.getTroubleshootTips(this.comboProblem.getSelectedItem().toString()).iterator();
this.txtBody.setText("");
while(iterate.hasNext())
{
    this.txtBody.setText(iterate.next().toString());
}
}
catch(Exception xcp)
{
    //JOptionPane.showMessageDialog(this, "There was an error loading data from the net.");
}

/**
 * @param args the command line arguments
 */
public static void main(String args[])
{
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            newBaseFrm().setVisible(true);
        }
    });
}

\\ THI CODE CREATE THE TABLE IN THE MYSQL DATABASE

DROP DATABASE IF EXISTS `expertSysDb`;

CREATE DATABASE `expertSysDb`;
USE `expertSysDb`;

CREATE TABLE IF NOT EXISTS `problem_category` (  
  `label` varchar(80) NOT NULL,  
  PRIMARY KEY (`label`)  
);

CREATE TABLE IF NOT EXISTS `problem` (  
  `problem_category` varchar(80) NOT NULL,  
  `label` varchar(80) NOT NULL  
);

CREATE TABLE IF NOT EXISTS `tips` (  
  `problem_label` varchar(80) NOT NULL,  
  `body` TEXT NOT NULL  
);

GRANT INSERT, SELECT, DELETE, UPDATE ON expertSysDb.* TO  
'root'@localhost identified by '';
THIS CLASS HANDLES ALL DATABASE OPERATIONS OF THE EXPERT SYSTEM APPLICATION

package com.expert.system;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

public class DatabaseOperation {
    static Connection conn = null;
    static PreparedStatement pstr = null;
    // <editor-fold defaultstate="collapsed" desc="Get all registered problem categories">
    public static ArrayList<String> getProblemCategories() {
        ArrayList<String> list = new ArrayList<String>();
        String sql = "SELECT * FROM problem_category";
        try {
            conn = DatabaseConnector.getConnectionString();
            pstr = conn.prepareStatement(sql);
            ResultSet rset = pstr.executeQuery();
            int count = 0;
            while (rset.next()) {
                ProblemCatModel pcModel = new ProblemCatModel();
                pcModel.setProblemCategory(rset.getString(1));
                list.add(pcModel);
                count++;
            }
        } catch (ClassNotFoundException cne) {
            // Handle ClassNotFoundException
        } catch (SQLException sqle) {
            // Handle SQLException
        }
    }
    // </editor-fold>
}
System.out.println(cne.getMessage());
cne.printStackTrace();
} catch (SQLException sqle) {
sqle.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
} finally {
try {
pstr.close();
conn.close();
} catch (SQLException sqle) {
sqle.printStackTrace();
}
}
return list;
}
// <editor-fold defaultstate="collapsed" desc="Get all problem under a speifed problem category">
public static ArrayList getProblems(String problemCategory)
{
ArrayList list = new ArrayList();
String sql = "SELECT * FROM problem WHERE problem_category= ?";
try {
conn = DatabaseConnector.getConnectionString();
pstr = conn.prepareStatement(sql);
pstr.setString(1, problemCategory);
ResultSet rset = pstr.executeQuery();
while (rset.next()) {
list.add(rset.getString(2));
}
} catch (ClassNotFoundException cne) {
System.out.println(cne.getMessage());
cne.printStackTrace();
} catch (SQLException sqle) {
sqle.printStackTrace();
}
public static ArrayList getTroubleshootTips(String problem) {
    ArrayList list = new ArrayList();
    String sql = "SELECT * FROM tips WHERE problem_label = ?";
    try {
        conn = DatabaseConnector.getConnectionString();
        pstr = conn.prepareStatement(sql);
        pstr.setString(1, problem);
        ResultSet rset = pstr.executeQuery();
        while (rset.next()) {
            list.add(rset.getString(2));
        }
    } catch (ClassNotFoundException cne) {
        System.out.println(cne.getMessage());
        cne.printStackTrace();
    } catch (SQLException sqle) {
        sqle.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            pstr.close();
            conn.close();
        } catch (SQLException sqle) {
            sqle.printStackTrace();
        }
    }
    return list;
}
// <editor-fold defaultstate="collapsed" desc="Get all problem under a
// specified problem category">
pstr.close();
conn.close();
    } catch (SQLExceptionsqle) {
sqle.printStackTrace();
    }
}
return list;
}
THIS CLASS HANDLES THE DATABASE CONNECTION TO THE EXPERT SYSTEM APPLICATION

package com.expert.system;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnector
{
    public DatabaseConnector()
    {}
    public static Connection getConnectionString() throws ClassNotFoundException, SQLException
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/expertSysDb","root","")
        ;
        return conn;
    }
}
Figure 5.1: The System initializes as the user launched the application
Figure 5.2: Introductory page
Figure 5.3: System prompt user to enter application code as user click to login
Figure 5.4: Administrator (Working Memory) module of the Expert System
Figure 5.5: Administrator (Working Memory) module showing how the system admin update the knowledge base of the expert system shell.
Figure 5.6: User Interface modules showing how the user interacts with the Expert System.
Figure 5.7: A User Interface module showing how the system responds to user’s request.