Sudan University of Science and Technology

College of Graduate Studies

Biomedical Engineering Department

Evaluation of Tuberculosis Screening Test using Fuzzy Expert System

تقييم مسوحات السل الرئوى بإستخدام نظام خبير غامض

A thesis Submitted to Fulfillment of Requirement for M.Sc. Degree in Biomedical Engineering (BME)

Submitted by:

Sabah Abdeen Ali Mohammed

Supervisor:

Dr.Eltahir Mohammed Hussein

January 2018

الآيـه

قال الله تعالي :

(إِنَّا عَرَضْنَا الأَمَانَةَ عَلَى السَّمَاوَاتِ وَالأَرْضِ وَالْجِبَالِ فَأَبَيْنَ أَن يَحْمِلْنَهَا وَأَشْفَقْنَ مِنْهَا وَحَمَلَهَا الإِنسَانُ إِنَّهُ كَانَ ظَلُومًا جَهُولا)

صدق الله العظيم

سوره الاحزاب أيه رقم (72)

Dedication

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, Abdeen and Halima whose words of encouragement and push for tenacity ring in my ears.my sisters and my brothers have never left my site are very special.

I also dedicate this dissertation to my husband who have supported me throughout the process. I always appreciates all they have done, for helping me develop my technology skills and many hours for proofreading.

Acknowledgement

This thesis constitute the efforts of eight months of deliberations and consultations by the author and the supervisor Dr. Altahir, who together formed the starting working group to pursue the recommendations of the healthcare expert system that will be held in in next future in Sudan University of Science and Technology.

Special thanks are due to Dr. Altahir for his patience and valuable assistance in the preparation and formatting of the text. Mr. Hassan further contributed in developing the systems code using programming language and also for designing the database.

Abstract

Tuberclause TB remains one of the world's most deadly infectious diseases and arguably, the greatest menace to modern society in terms of morbidity and mortality. To choose the right treatment and to ensure a quality of life suitable for a specific patient condition, early and accurate diagnosis of TB is essential. It reduces disease and prevents deaths. It also contributes to reducing TB transmission. In recent times. Computer Science has played a significant role in the task of medical diagnosis. This thesis work focused on Fuzzy Expert System for TB diagnosis. It is simple to use, portable, low cost and makes TB diagnosis more rapid and accurate. It supports medical practitioners and assists researchers to deal with the vagueness, imprecision and time-consuming found in traditional laboratory diagnosis of TB, and provide accurate output based on the input data. This thesis, is a software system for use in the diagnosis of TB is an expert system with a database containing an expert knowledge. The user only uses it to determine whether he or she has any of the diseases within its domain, this software Systems "Tuberculosis diagnosis expert system" TB-DES was designed using OOP design pattern and implemented using Microsoft Visual IDE Tools. The various modules as described are integrated and implemented using server based database by demonstrating MS-SQL Server as backend data storage.

Contents

List of Tables

List of Figures

CHAPTER ONE

INTRODUCTION

1.1. General view

Tuberculosis is a chronic bacterial infection that primarily affects lungs, but may also spread to other organs. The infection is transmitted by respirable droplets generated during forceful expiratory manoeuvres such as coughing. Tuberculosis infection can be either active or latent. People with active infection have signs or symptoms caused by actively replicating tubercle bacilli. If this is in
the lungs they are potentially contagious and usually have symptoms such as cough, chest pain, shortness of breath, fatigue, weight loss, fever and night sweats. With progress of pulmonary TB, symptoms of coughing, fever, and bloody sputum are seen. Those with latent infection have previously been infected but have no symptoms or evidence of disease and are not contagious. However, they remain at risk of developing active tuberculosis (reactivation) during their lifetime.

Various factors are associated with an increased risk of becoming infected and subsequently developing disease. Transmission is most efficient in poorly ventilated, crowded environments [1]. Physician or the expert person suspects tuberculosis through taking medical history, physical examination, laboratory tests, radiography, and in some cases tuberculin test [2].

Recent practice for medical treatment make it mandatory that patients consult specialists for further diagnosis and treatment. Other medical practitioners may not have adequate expertise or experience in handling certain high-risk diseases. Nonetheless, typical waiting time for treatments may be few days, weeks or even months. Possibly, by the time the patients consult the specialists the diseases may have already spread out. Since the majority of the high-risk disease could only be cured at the early stage, the patients may have to endure for the rest of their life, due to which new approaches with the support of computer technology for the diagnosis of diseases is essential. The mortality rate and the waiting time to see the specialist could be reduced by employing the computer technology or computer program or software developed by emulating human intelligence which supports the doctors in making decisions without the direct consultation with the specialists. It is possible to shortlist the patients with high-risk factors or symptoms or predicted to be highly effected with certain diseases or illness to see the specialist for further treatment [3].

"Computer Decision Support Systems" are designed and used in various fields of science. "Expert Systems", as a sub-branch of Artificial Intelligence, play a central role. In Expert Systems, decisions are made by the computer. Expert Systems are knowledge-based systems that initially acquire knowledge from an expert person, and then transfer it to the computer, so that by making decisions based on the data collected, they can be of help to humans. So far, many Expert Systems have been presented in various fields like medical, industry, control, space, financial decisions, etc. [2]

The aim is to present an Expert System for diagnosing Tuberculosis TB. A medical Expert System is a computer program that helps with decision making for diagnosis of diseases and suggests treatment methods. It includes sub-systems of database management, user interface management, and inference (search) engine. Once patient's data are entered, the system performs diagnosis of disease, predicts potential complications and diagnoses, and then suggests treatment. These systems are distinct from other medical software, since, to arrive at a medical result, they imitate the rationale of an expert physician. Expert systems require many rules and facts of medical science about diseases and condition of the disease to be able to make decisions and present accurate results.

One of the most common methods of designing Expert Systems is prototype method. I used prototype method which consists of three phases of analysis, design, and implementation. An Expert System uses knowledge and inference procedures to solve problems that are sufficiently difficult.

2. Problem statement

Tuberculosis TB remains one of the world's deadliest communicable diseases. In 2013, an estimated 9.0 million people developed TB and 1.5 million died from the disease. TB is slowly declining each year and it is estimated that 37 million lives were saved between 2000 and 2013 through effective diagnosis and treatment [4]. Tuberculosis TB is still a major public-health difficulty in the tropics. The combination of inadequate expertise in medical science at developing countries and sometimes the complexity of medical practices exponentially increases the morbidity and mortality rates of tuberculosis patients.

2.1. Objectives

The objectives are to:

1- Design and Develop expert system for diagnose and detect Tuberculosis T.B and providing decisions support platform to Tuberculosis T.B researchers, physicians and other health care practitioners in the tropics.

2- Minimize the spread of T.B disease.

2.2. Methodology

A rule-based approach has been used. Algorithmic rules in the Tuberculosis-Diagnosis Expert System necessitate decision coverage where tuberculosis is either suspected or not suspected. The architecture consists of a rule base, knowledge base, and patient database. These units interact with the inference engine, which receives patients' data through the special windows via a user interface.

To achieve the goal and complete task properly I:

- Acquired knowledge from activities of the physicians.
- Analysis - this gives a full description of diagnosis which involves direct contact with the medical doctor or personnel as the case may be.
- Design is based on analysis carried out in the previous step and information gathered in previous steps. Interface also created upon recommendations given in the previous stages.
- Systems implementation "Tuberculosis diagnosis expert system" TB-DES was implemented using Microsoft Visual. The various modules as described are integrated and implemented using C# MS-SQL Server,

  The three (3) modules (subsystems) of the system are: Patient information, Diagnosis and result

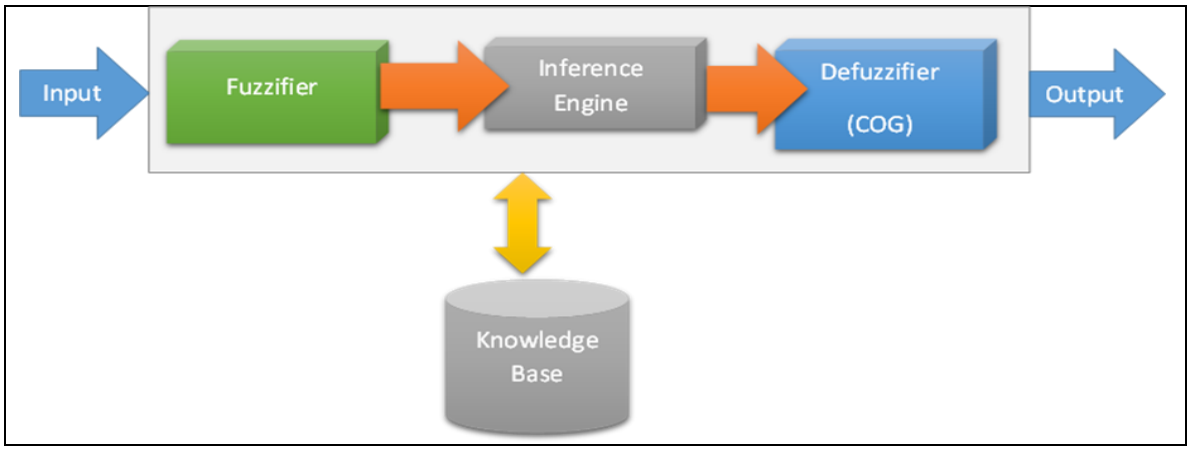

Figure (1.1) Proposed tuberculosis T.B Diagnosis System Using Expert System

2.3. The thesis layout:

This research consists of five chapters: chapter one is an introduction. The previous studies are given in chapter two. The concepts of ES techniques are presented at chapter three. Chapter four is about the proposed system. Chapter five presents result and discuss. Finally, conclusion and recommendation are given in chapter six

CHAPTER TWO

Previous studies

Several previous studies used expert systems in field of diagnosis diseases:

Fuzzy Expert Framework for Diagnosis of Typhoid Fever

Okpokpong Nathaniel Ntebong Celestine, Umar Farouk Ibn Abdulrahman, Itoro Akpabio Vol.2, No.9, pp: (13-16), October 2017 [5]

Typhoid fever is a disease that is caused by bacteria called salmonella typhi. It is also known as enteric fever. Typhoid fever is characterized by high fever, constipation, diarrhoea, abdominal pain, etc. It is very common in developing countries like Ghana and according to, it has caused an estimated 21.7million illnesses and 217,000 deaths. In 2013, there were 161,000 reported deaths from typhoid fever.

Typhoid fever is often treatable when diagnosed early, but if left untreated could lead to other medical complications like intestinal haemorrhaging which may require major surgeries and could even lead to death.

The cornerstone of any development project is a thorough understanding of the business requirements. Therefore, the first step was to explore and develop a good working knowledge of the problem which in this case is to develop a system which can diagnose Typhoid Fever accurately.

Typhoid fever, like most other fevers, has symptoms which are peculiar to itself and others which are shared across the different fevers and other ailments.

The goal is to build a system that can diagnose Typhoid Fever accurately. Fuzzy Logic helps us accomplish this by mimicking human reasoning with computer accuracy.

The system was initially modelled in MATLAB R2013a with 20 input members and 1 output member. The system is divided into three parts which are:

1. Inputs
2. Inference Rules
3. Output

It is based on this division that the entire system was modelled. The inference rules and the database. Input members can be described as the inputs of the system in this case, the input members are the symptoms the patient is experiencing. Usually these inputs are in crisp form (not discrete). Each input member has a membership function which in this case, is the severity of the symptom. Inference rules are a collection of IF-THEN rules that form the basis for decision making in the system.

The inference rules are what mimic the decision-making process in humans. The IF-Then statements have an "AND" connection. This implies that all conditions in each statement must be met for a specific result to be given as output. The Mamdani method was chosen due to the fact that it is well suited for human inputs.

Below are some inference rules used in the system:
1. If (Fever is Low) then (Prognosis is None)
2. If (Fever is Mild) then (Prognosis is FeverNoTyphoid)
3. If (Fever is Sever) then (Prognosis is None)
4. If (Fever is Low) and (StomachAche is Low) then (Prognosis is None)
5. If (Fever is Low) and (StomachAche is Mild) then (Prognosis is None)
6 If (Fever is Mild) and (StomachAche is Mild) and (DryCaugh is Low) and (ColdChills is Low) then (Prognosis is FeverNoTyphoid)

The overall aim of the system was to develop a system that uses fuzzy logic to diagnose typhoid fever. The system was able to take patient information and symptoms as input, process the said information and then provide the user with a diagnosis as an output.

The system proved to be reliable with an accuracy of 97.5% which is very impressive and thus making it one of the most reliable systems for the diagnosis of Typhoid Fever. The system made use of twenty-one (21) membership functions as inputs and has over 200 inference rules making it one of the most robust systems in the diagnosis of Typhoid Fever.

In conclusion, the research has demonstrated that fuzzy logic can be used to diagnose typhoid fever and that a more robust inference engine leads to greater accuracy in the diagnostic process

Diagnostic Expert System: architecture for translating patients -Tuberculosis to automate information from the web for use in tuberculosis diagnosis

Victor C Osamor, Ambrose A Azeta and Oluseyi O Ajulo(Jan 21, 2014) [6]

The pre-laboratory screening process against tuberculosis infection to aid diagnosis and make it fast and accessible to the public via the Internet. The expert system we have built is designed to also take care of people who do not have access to medical experts, but would want to check their medical status.

A rule-based approach has been used, and unified modeling language and the client–server architecture technique were applied to model the system and to develop it as a web-based expert system for tuberculosis diagnosis. Algorithmic rules in the Tuberculosis–Diagnosis Expert System necessitate decision coverage

where tuberculosis is either suspected or not suspected. The architecture consists of a rule base, knowledge base, and patient database. These units interact with the inference engine, which receives patient' data through the Internet via a user interface. l

The basic model of a rule-based expert system is composed of sets of rules that are regularly applied to the sets of facts in the database. While facts represent circumstances that describe a certain situation in the real world, rules consist of an "if" and a "then" portion or "if-then" statement which represents heuristics that define a set of actions to be carried out in a given situation or circumstance.16,17 The "If" portion is on the left-hand side (LHS) and is called premise or predicate, while the "then" portion is on the right-hand side (RHS) and is either a conclusion or action.

- Condition or set of conditions is expressed as a (simple or complex) Boolean expression which represents the situation under which the actions of the rule are enabled, in case the specified events occur.19

Algorithmic rules in the TB–Diagnosis Expert System (TB-DES)

Where TB is suspected. The presence of cough with bloody sputum and swollen lymph is probably an indication of Mycobacterium infection, which is captured in the following algorithmic rules.

If <symptom is cough AND
(NOT symptom is headache)
AND (symptom is bloody sputum)
AND symptom is swollen lymph/neck/joint>
Then <Notify (Patient), "Tuberculosis is very likely, please go and see your doctor">.

Where TB is not suspected. When cough is not observed and sputum occurrence is none or minimal, and where it occurs, it is not bloody, then TB is not likely. This is depicted in the following algorithmic rules.

If <symptom is cough AND
(NOT Symptom is bloody sputum)
AND (NOT symptom is swollen lymph/neck/joint>
Then <Notify (Patient), "Tuberculosis is not likely but ordinary cough">.

To support the design of TB-DES, the unified modeling language (UML) was applied to model the system and facilitate easy implementation.18 Use case (Figure 1), and Sequence diagrams (Figure 2) were created to describe the system. The architecture of TB-DES shown in Figure 3 consists of a rule base, knowledge base, and patient database. Patient data is acquired over the Internet via a user interface.

TB-DES was implemented using Java NetBeans Integrated Development Environment (IDE) version 6.0 hosted on the Windows Vista operating system.

NetBeans was chosen because it is a robust IDE suitable for rapid and reliable development of enterprise Java web applications. Java was chosen because of its multiplatform and concurrency support.
The various modules as described in Figure 3 are integrated and implemented using Java Server Pages (JSP), Apache Tomcat web server, Apache Derby Version 2.0, and Java NetBeans IDE. JSP technology enabled us to put snippets of Servlet Code directly into a text-based document. A JSP page is a text-based document that contains wireless markup language (WML), extensible markup language (XML) and JSP elements, which determine how the page constructs dynamic content. Apache Tomcat is the Servlet Container used in the official reference implementation for the Java Servlet and JSP technologies. The Java Servlet and JSP specification are developed by Sun under the Java Community Process.
Apache Derby is an open source relational database implemented entirely in Java and available under the Apache license version 2.0. Derby provides an embedded JDBC driver that lets one embed Derby in any Java-based solution. Derby also supports the more familiar client/server mode with the Derby Network Client JDBC driver and Derby Network Server. In other words, JDBC is responsible for initiating and maintaining the connection between the interface client and the Derby relational database. Java NetBeans IDE provided the total programming environment where all development took place. The system was tested with varying symptomatic input. The system was able to navigate through various prompted questions relevant to the preclinical screening and finally terminated with a conclusion that defines the infectious or noninfectious status of TB. In some cases, it also includes additional advisory information to help the patient improve his or her health.
The minimum hardware requirements for the TB-DES to work at an optimal level are Pentium III (600 MHz) processor or higher, 128 MB of RAM or more, with hard disk size of 2 GB and above, and Operating System Software (Windows XP, 2000 and above).

An Adoptive Medical Diagnosis System Using Expert System with Applications

Gufran Ahmad Ansari, (KSA-2014) [7]

It is well known that developing countries are facing a lot of shortage of medical expertise in medical science. Due to this, they are unable to provide good medical services to their country people. Peoples are not doing proper care of their health they are getting many health disease problems. Patients also find a huge queue in

hospitals. Nowadays common diseases like Malaria, Typhoid, Plague, and Typhus etc. are becoming more dangerous problems for the people that living in this world. Now these problems become a big challenge for the developing countries that how to provide a good medical services their country people. Also how can avoids the patients queue from the hospitals. In this paper, author proposed a model of Adoptive Medical Diagnosis System (AMDS) using Expert System (ES). The AMDS using ES system is very helpful for the patients that infected with common diseases and this system will give a prescription as a medical expert also this system is very helpful for rural areas where we have young medical expert or don't have medical expert.

It is moral responsibility of a country to provide the good medical services to their country people because healthy people make healthy country. Medical science field is an enormous field that includes a large number of expertise, knowledge and information. In this paper, a model of Adoptive Medical Diagnosis System (AMDS) using Expert System (ES) is proposed and described. This model is a combination of AMDS and ES. This model can be used to provide the prescription for general health diseases like Malaria, Typhoid, Plague, and Typhus also it works as a human expert. To get the good medical treatment is a right of every country citizens. Due to the insufficient availability experts in a medical domain it is a big challenge for the country to provide good medical services to their country people. The major advantage of Adoptive Medical Diagnosis System using Expert System is that it gives the prescription as the doctor ordered to the patient. Also this system can be used anytime and anywhere. Also the proposed system can work as a human expert. When hospitals are facing the problem of general expert this system can work as a human expert. Expert systems technology has been widely adopted by many software development companies and industry. Expert System is a kind of software that runs in computer memory and works the same kind of interaction as a specialist work and gives prescription to the patient as a specialist. Also you can say it is a kind of program that asks you about your symptoms disease and provide possible prescription matching of the symptoms like human expert. Expert system is a part of Artificial Intelligence. The main purpose of expert systems is to provide expert advice if the person seeks advice from System. Also it is a computer program that can reproduce thinker activities.

The structure of knowledge or facts is called the domain of ES. There are three major components of Expert System which are Knowledge base (KB),
Inference Engine (IE) and User Interface (UI).

Knowledge base holds the domain knowledge which is employed by the inference engine to draw conclusions.

The inference engine is responsible to assume and get the solutions from relationships of knowledge base and facts that provide by the user. It also decides

which rules should be implemented in a given situation, and in which order it will use information in the knowledge base. Medical science field is a very active and challenging field to implement the above techniques. For the medical diagnosis and prescription, doctors ask the question to the patient and try to find out the possible disease on the basis of interview. Doctors then write the prescription for the patient. Similarly, in medical treatment we use Expert System for retrieving the patient history from the database and current symptoms for the patient data and prescribe treatment as a human expert.

CHAPTER THREE

Expert System techniques

3.1 Expert Systems (ES)

It is a branch of **Artificial Intelligence** (AI) [8]. AI is the capability of a device such as a computer to perform task that would be considered intelligent if they were performed by a human [9]. An **ES** is a computer program that attempts to replicate the reasoning processes of expert and can make decisions and recommendations, or perform tasks, based on user input

Rule-based systems (also known as production systems or expert systems) are the simplest form of artificial intelligence. A rule based system uses rules as the knowledge representation for knowledge coded into the system [10].Rule-based ESs should contain, at the very least, the three components of an AI production system: the knowledge base; the database; the rule interpreter. You can see that structure at (fig. (3.1)).

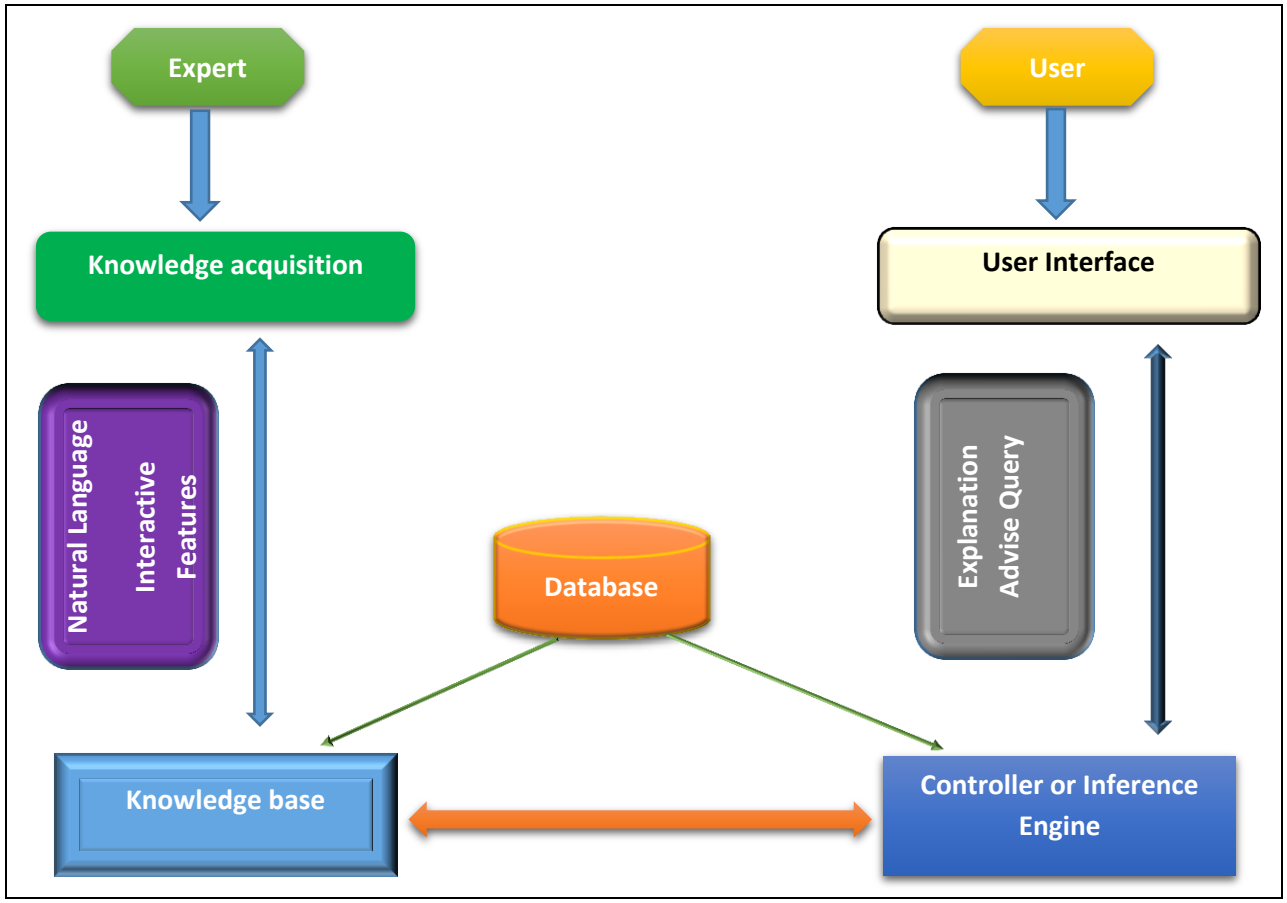

Figure (3.1) shows the structure of the Expert System

3.2 Fuzzy Logic:

In this research, I designed a fuzzy expert system for the diagnosing tuberculosis, Fuzzy logic is the science of reasoning, thinking and inference that recognizes and uses the real world phenomenon that everything is a matter of degree [11], In the simplest terms, Fuzzy Logic (FL) is a branch of machine intelligence that helps computers paint pictures of uncertain world [12]. It presents an inference morphology that enables appropriate human reasoning capabilities to be applied to knowledge-based systems[13]. The theory of fuzzy logic encompasses a mathematical strength to capture the uncertainties associated with human cognitive processes[14]. Fuzzy logic is a methodology that captures and uses the concept of fuzziness in a computationally effective manner (Lofti,1994)[16]. Fuzzy logic form of many-valued logic in which the truth values of variables may be any real number between 0 and 1. By contrast, in Boolean logic, the truth values of variables may only be 0 or 1, fuzzy logic Deals with noisy, imprecise, vague, ambiguous data. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions [17]

In this medical fuzzy expert system design ,the first step is determination of input variables.and a disease is usually characterized by directly observable symptoms that prompt the patient to visit a physician, this observable symptoms defined as veriables . A series of clinical observations are undertaken to detect the presence of Tuberculosis TB. There are eight input symptoms (variables) of the Tuberculosis TB:

1. Fever and night sweats
2. A bad cough lasting longer than two weeks
3. Chest pain
4. Coughing up blood or sputum
5. Weakness or feeling very tired
6. Weight loss
7. Lack of appetite
8. Chills

3.3 Algorithm of the Fuzzy Expert System for TB Diagnosis

1. Input signs and symptoms of patient complaint into the system. Where m = number of signs and symptoms.

2. Search in knowledge-base for the disease, which has the signs and Symptoms identified.
3. Apply *fuzzy* rules.
4. Map fuzzy inputs into their respective weighing factors to determine their degree of membership.
5. Determine the rule base evaluating (non-minimum values).
6. Determine the firing (conclusion) strength of the rules *R*.
7. Calculate the degree of truth *R*, of each rules by evaluating the nonzero Minimum value.
8. Compute the intensity of the disease.
9. Output fuzzy diagnosis.

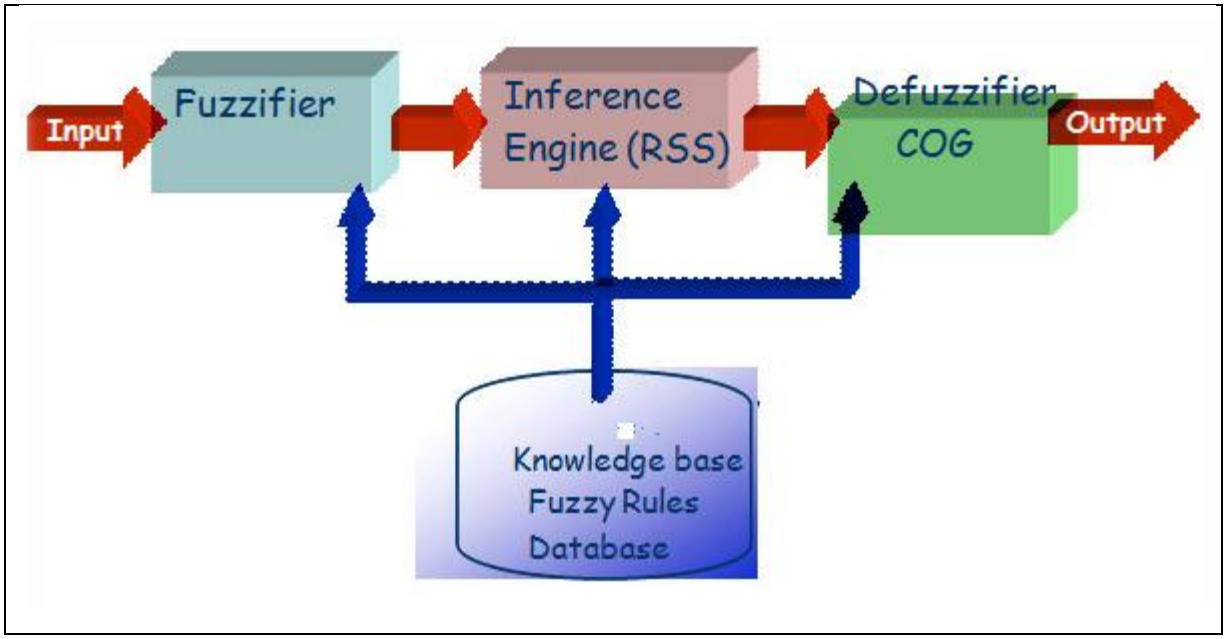

Figure (3.2) shows the Architecture of Fuzzy Expert System for T.B

3.4 Knowledge Base

Knowledge is a key factor in the performance of intelligent systems. Knowledge base is a special kind of database for knowledge management which provides a means for information to be collected, organized, shared, searched and utilized [17]. In another term, the knowledge base acts as a repository for information in the expert system. The knowledge base of the fuzzy expert system for the diagnosis of TB is composed of structured information. The structured knowledge is concerned with facts, rules and events of tropical disease and fuzzy rules which will be used to determine the rate of the disease.

3.5 Membership Functions

Membership function is a graphical representation of "magnitude of participation" of each input (symptom) to determine output response [19].

The membership function $\mu_A(x)$ describes the membership of the elements x of the base set X in the fuzzy set A, whereby for $\mu_A(x)$ a large class of functions can be taken. Shape of membership function is usually triangle (but could also be trapezoidal or other)

The height usually normalized to 1 (so in total: [0...1]) and Width of the base of function depends on number of functions.

The Sum of all membership values for a certain input value is always 1.0 and Membership values (antecedents) are evaluated for the produced conclusion (consequent) [20]

3.6 Fuzzification

This is the process of transforming crisp input into linguistic variables using the membership function in the fuzzy knowledge base. There are three types of Fuzzifiers: the trapezoidal fuzzifier, triangular fuzzifier, and Gaussian fuzzifier. This work concentrates on using triangular fuzzifier for changing of scalar value to fuzzy set which is in the range of 0 and 1[21]. In another term fuzzification can be viewed as the operation of transforming a crisp set to a fuzzy set, or a fuzzy set to a fuzzier set." The crisp input (that is, the measured value) is translated into linguistic variable.

3.7 Linguistic Variables

Linguistic variables mean variables whose values are words or sentence in a natural or artificial language [22]. The linguistic variables used in this work are, none, mild, moderate, severe, very severe.

To obtain the degree of symptom, we use the formula (1):

$$x_i/x_n \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$x_i \equiv$ Number of the linguistic variable

$x_n \equiv$ Total number of linguistic variables

This formula is used to prepare the triangular fuzzy table. For instance, if a patient complains of severe chest pain, as it gotten from the expert equal to 0.75.
Fuzzification begins with the transformation of the parameters (linguistic variables) to fuzzy sets by Equations (2) to (5) below. On the basis of domain experts' knowledge, the range of fuzzy value for each linguistic is shown in table (3.1) below

Table (3.1): shows ranges of Fuzzy value

Linguistic Variable	**Fuzzy Value**
None	0
Mild	$0.1 \leq x < 0.3$
Moderate	$0.3 \leq x < 0.6$
Severe	$0.6 \leq x < 0.8$
Very Severe	$0.8 \leq x \leq 0.1$

During the process, linguistic variables are evaluated using triangular membership function (2) and are accompany by associated degree of membership ranging from 0 to 1 as shown in equations (3) to (6) below. These formulas are determined by aid of both the expert doctors in the field of tropical medicine and literature

$$\begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{c-a} & \text{if } x \epsilon [a,c] \\ \frac{b-x}{c-b} & \text{if } x \epsilon [b,c] \\ 0 & \text{if } x \geq c \end{cases} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\mu_{Mild}(x) = \begin{cases} 0 & if \quad x \leq 0.1 \\ \dfrac{x - 0.1}{0.1} & if \quad 0.1 < x < 0.2 \\ \dfrac{0.3 - x}{0.1} & if \quad 0.2 < x < 0.3 \\ 0 & if \quad x \geq 0.3 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$\mu_{Modrate}(x) = \begin{cases} 0 & if \quad x \leq 0.3 \\ \dfrac{x - 0.3}{0.15} & if \quad 0.3 < x < 0.45 \\ \dfrac{0.6 - x}{0.15} & if \quad 0.45 < x < 0.6 \\ 0 & if \quad x \geq 0.6 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

$$\mu_{Severe}(x) = \begin{cases} 0 & if \quad x \leq 0.6 \\ \dfrac{x - 0.6}{0.1} & if \quad 0.6 < x < 0.7 \\ \dfrac{0.8 - x}{0.1} & if \quad 0.7 < x < 0.8 \\ 0 & if \quad x \geq 0.8 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$\mu_{VerySevere}(x) = \begin{cases} 0 & if \quad x \leq 0.8 \\ \dfrac{x - 0.8}{0.1} & if \quad 0.8 < x < 0.9 \\ \dfrac{1.0 - x}{0.1} & if \quad 0.9 < x < 1.0 \\ 0 & if \quad x \geq 1.0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

By using these linguistic variables, fuzzy IF THEN RULES which are the main output of the fuzzy system would be set up and generally presented in the form of: If (x is α) Then y is β

The next step in the fuzzification process is the development of fuzzy rules. The fuzzy rules for this research were developed with the assistance of domain experts (medical doctors). Table (3.2) shows sample fuzzy rule base for TB. Fever and night sweats a bad cough lasting longer than two weeks, Chest pain, Coughing up blood or sputum, Weakness or feeling very tired, Weight loss, Lack of appetite and Chills, as input parameters to the system. From the expert knowledge, these input parameters were used to generate 256 rules for the rule base for TB. Table (3.2) shows sample fuzzy rule base for TB. Triangular fuzzy values for signs, symptoms and investigations of table (3.2) are shown in Table (3.3)

Table (3.2) Sample of Fuzzy rules structure for diagnosis for TB

Rule No	IF								THEN
	A bad cough lasting longer than two weeks	Chest pain	Coughing up blood or sputum	Weakness or feeling very tired	Weight loss	Lack of appetite	Chills	Fever and night sweats	Conclusion
1	Moderate	Moderate	Moderate	Very Severe	Severe	Severe	Mild	Very Severe	Severe
2	Very Severe	Moderate	Moderate	Severe	Very Severe	Mild	Severe	Severe	Severe
3	Mild	Mild	Moderate	Moderate	Very Severe	Severe	Moderate	Mild	Moderate
4	Very Severe	Severe	Severe	Moderate	Mild	Mild	Severe	Very Severe	Severe
5	Moderate	Moderate	Very Severe	Very Severe	Moderate	Severe	Very Severe	Severe	Severe
6	Very Severe	Very Severe	Very Severe	Mild	Mild	Moderate	Moderate	Very Severe	Severe
7	Very Severe	Very Severe	Severe	Severe5	Very Severe	Moderate	Very Severe	Severe	Very Severe
8	Moderate	Moderate	Moderate	Very Severe	Mild	Moderate	Very Severe	Mild	Severe
9	Very Severe	Very Severe	Very Severe	Severe	Moderate	Severe	Moderate	Moderate	Severe
10	Mild	Moderate	Moderate	Mild	Severe	Severe	Very Severe	Mild	Moderate
11	Very Severe	Very Severe	Very Severe	Moderate	Moderate	Very Severe	Moderate	Mild	Severe

12	Moderate	Very Severe	Very Severe	Very Severe	Moderate	Mild	Very Severe	Moderate	Severe
13	Very Severe	Moderate	Moderate	Mild	Mild	Very Severe	Mild	Severe	Moderate
14	Severe	Moderate	Moderate	Mild	Mild	Moderate	Mild	Moderate	Moderate
15	Very Severe	Severe	Mild	Severe	Moderate	Severe	Moderate	Severe	Severe
16	Severe	Severe	Moderate	Moderate	Moderate	Very Severe	Mild	Very Severe	Severe
17	Very Severe	Moderate	Very Severe	Very Severe	Moderate	Severe	Severe	Mild	Severe
18	Moderate	Moderate	Moderate	Moderate	Moderate	Very Severe	Severe	Moderate	Moderate
19	Mild	Severe	Moderate	Mild	Severe	Mild	Severe	Moderate	Moderate
20	Moderate	Mild	Severe	Very Severe	Severe	Moderate	Moderate	Mild	Severe

Table (3.3) Triangular Fuzzy Numbers for TB.

No	A bad cough longer than two weeks	Chest pain	Coughing up blood or sputum	Weakness or feeling very tired	Weight loss	Lack of appetite	Chills	Fever and night sweats	Con.
1	0.5	0.5	0.5	1.0	0.75	0.75	0.25	1.0	0.75
2	1.0	0.5	0.5	0.75	1.0	0.25	0.75	0.75	0.75
3	0.25	0.25	0.5	0.5	1.0	0.75	0.5	0.25	0.5
4	1.0	0.75	0.75	0.5	0.25	0.25	0.75	1.0	0.75
5	0.5	0.5	1.0	1.0	0.5	0.75	1.0	0.75	0.75
6	1.0	1.0	1.0	0.25	0.25	0.5	0.5	1.0	0.75
7	1.0	1.0	0.75	0.75	1.0	0.5	1.0	0.75	1.0
8	0.5	0.5	0.5	1.0	0.25	0.5	1.0	0.25	0.75
9	1.0	1.0	1.0	0.75	0.5	0.75	0.5	0.5	0.75
10	0.25	0.5	0.5	0.25	0.75	0.75	1.0	0.25	0.5
11	1.0	1.0	1.0	0.5	0.5	1.0	0.5	0.25	0.75
12	0.5	1.0	1.0	1.0	0.5	0.25	1.0	0.5	0.75
13	1.0	0.5	0.5	0.25	0.25	0.9	0.25	0.75	0.5
14	0.75	0.5	0.5	0.25	0.25	0.5	0.25	0.5	0.5
15	1.0	0.75	0.25	0.75	0.5	0.75	0.5	0.75	0.75
16	0.75	0.75	0.5	0.5	0.5	1.0	0.25	1.0	0.75
17	1.0	0.5	1.0	1.0	0.5	0.75	0.75	0.25	0.75
18	0.5	0.5	0.5	0.5	0.5	1.0	0.75	0.5	0.5
19	0.25	0.75	0.5	0.25	0.75	0.25	0.75	0.5	0.5
20	0.5	0.25	0.75	1.0	0.75	0.5	0.5	0.25	0.75

Some of the rules (Rules 20, Rules 30, Rules 60, Rule 80, Rules 110, Rule 140, Rules 160, Rules 200 and Rule 256) interpreted at Table (3.5)

Table (3.4) shows an interpretation of rules 1 to 4.

Rule No.	interpretation
1.	If (A bad cough lasting longer than two weeks = Moderate) AND (Chest pain = Moderate) AND (Coughing up blood or sputum = Moderate) AND (Weakness or feeling very tired = Very Severe) AND (Weight loss = Severe) AND (Lack of appetite = Severe) AND (Chills =Mild) AND (Fever and night sweats = Very Severe) Then TB =Severe
2.	If(A bad cough lasting longer than two weeks = Very Severe) AND (Chest pain = Moderate) AND (Coughing up blood or sputum = Moderate) AND (Weakness or feeling very tired = Severe) AND (Weight loss = Very Severe) AND (Lack of appetite = Mild) AND (Chills = Severe) AND (Fever and night sweats = Severe) Then TB = Severe

3.	If (A bad cough lasting longer than two weeks = Mild) AND (Chest pain = Mild) AND (Coughing up blood or sputum = Moderate) AND (Weakness or feeling very tired = Moderate) AND (Weight loss = Very Severe) AND (Lack of appetite = Severe) AND (Chills = Moderate) AND (Fever and night sweats = Mild) Then TB = Moderate
4.	If (A bad cough lasting longer than two weeks = very Severe) AND (Chest pain = Severe) AND (Coughing up blood or sputum = Severe) AND (Weakness or feeling very tired = Moderate) AND (Weight loss = Mild) AND (Lack of appetite= Mild) AND (Chills = Severe) AND (Fever and night sweats = Very Severe) Then TB = Severe

3.8 Fuzzy Inference Engine

An inference engine is a computer program that tries to derive answers from a knowledge base. It is the "brain" that expert systems use to reason about the information in the knowledge base for the ultimate purpose of formulating new conclusions [22]. In fuzzy inference engine, Fuzzy inputs are mapped into their respective weighting factors and their associated linguistic variables to determine their degree of membership. The aggregation operator is used to calculate the degree of fulfillment or firing strength of a rule.

For this work, the fuzzy logical AND is used to evaluate the composite firing strength of the rules. In practice, the fuzzy rules sets usually have several antecedents that are combined using fuzzy logical operators, such as AND, OR and NOT, though their definitions tend to vary: AND simply uses minimum weight of all the antecedents, while OR uses the maximum value. There is also the NOT operator that subtracts a membership function from 1 to give the "complementary" function. The degree of truth (R) of the rules are determined for each rule by evaluating the nonzero minimum values using the AND operator. The inference engine evaluates all the rules in the rules base and combines the weighted consequences of all the relevant (fired) into a single fuzzy set. The inference engine technique employed in this research is the *Root Sum Square (RSS).* RSS is given by the formula (7)

$$\sqrt{\sum R_i^2} = \sqrt{R_1^2 + R_2^2 + \cdots + R_n^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(7)}$$

The $R_1^2 + R_2^2 + \cdots + R_n^2$ are values of different rules which have the same conclusion in the fuzzy rule base, that is, R_i^2 is the value of firing rule. RSS combines the effects of all applicable rules, scales the functions at their respective magnitudes and compute the "fuzzy" centroid of the composite area.

3.9 Defuzzification

The defuzzification is the process of converting the fuzzy output from the inference engine to a crisp value [23]. That is, the output gotten form the inference engine in this work using root sum square is defuzzified to get the level of the illness. The input to the defuzzification process is a fuzzy set while the output of the defuzzification process is a single number (crisp output). There are five commonly used defuzzifying methods [24]

Center of Sums Method (COS) , Center of gravity (COG) / Centroid of Area (COA) Method , Center of Area / Bisector of Area Method (BOA) ,Weighted Average Method and Maxima Methods include First of Maxima Method (FOM) ,Last of Maxima Method (LOM) and Mean of Maxima Method (MOM)
In this work, the centroid of area also called center of area or center of gravity technique is used for defuzzification. This is the most commonly used technique because of its high level of simplicity and accuracy. The centroid defuzzification technique can be expressed as formula (8) below

$$G(Y) = \frac{\sum \mu_Y(x_i)x_i}{\sum \mu_Y(x_i)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(8)}$$

Where $G(Y)$ = is the defuzzified output,

$\mu_Y(x_i)x_i$ = Is the aggregated membership function and

x_i = center of membership function.

Table (3.5) below shows the fuzzy expert doctor symptoms for patient **SALIH**. For each of the linguistic variable the respective output membership function in the interval of [0, 1] are computed from the possible rules using root sum square. The outputs from the inference engine are defuzzified using centroid of area also called center of gravity to obtain a crisp value.

Table (3.5): Example of fuzzy expert doctor symptoms for patient Salih

Symptoms	**Linguistic Variables**	**Assigned Values**
A bad cough lasting longer than two weeks	Severe	0.75
Chest pain	Moderate	0.50
Coughing up blood or sputum	Very Severe	1.0
Weakness or feeling very tired	Very Severe	1.0
Weight loss	Mild	0.25
Lack of appetite	Mild	0.25
Chills	Very Severe	1.0
Fever and night sweats	Moderate	.50
Extreme tiredness or lack of energy	Moderate	.50
The Conclusion Patient is TB suspected	Moderate	.50

These values will result in the fuzzy transcript as shown in Table (3.5) below using the rule base for TB as presented in Table (3.2) above. An example for rule base evaluation for patient SALIH is presented in Table (3.5)

Table (3.6) shows the "17" rules base for patient SALIH

Rule No	IF								THEN
	A bad cough lasting longer than two weeks	Chest pain	Coughing up blood or sputum	Weakness or feeling very tired	Weight loss	Lack of appetite	Chills	Fever and night sweats	Con
1	0.5	-	-	-	-	-	0.25	Very Severe	0.25
30	-	0.25	0.25	-	-	0.25	0.25	Moderate	0.25
40	-	-	-	-	0.5	-	-	Very Sever	0.5
50	-	-	0.25	-	-	-	-	Very Severe	0.25
70	0.5	-	-	-	-	-	0.25	Severe	0.25
80	-	0.25	-	-	-	-	0.25	Very Severe	0.25
90	-	0.25	0.25	-	-	0.25	-	Severe	0.25
100	0. 5	-	0.25	-	-	-	-	Sever	0.25
110	-	0.25	-	-	-	0.25	-	Moderate	0.25
120	-	-	-	-	-	0.25	-	Mild	0.25
130	-	0.25	0.25	-	-	-	0.25	Moderate	0.25
140	0.5	-	0.25	-	-	-	-	Very Severe	0.25
150	-	0.25	-	-	-	-	0.25	Moderate	0.25
200	-	-	0.25	-	-	0.25	0.25	Moderate	0.25
240	0.5	-	-	-	-	0.25	-	Moderate	0.25
256	-	-	-	-	0.5	-	-	Sever	0.5

Mild $=\sqrt{R_{120}^2}$

$=\sqrt{.25^2}$

$=0.25$

Moderate $=\sqrt{R_{30}^2 + R_{110}^2 + R_{130}^2 + R_{150}^2 + R_{200}^2 + R_{240}^2}$

$=\sqrt{.25^2 + .25^2 + .25^2 + .25^2 + .25^2 + .25^2}$

$=0.6124$

Severe $=\sqrt{R_{70}^2 + R_{90}^2 + R_{100}^2 + R_{256}^2}$

$=\sqrt{.25^2 + .25^2 + .25^2 + .5^2}$

$=0.6614$

Very Severe $=\sqrt{R_1^2 + R_{40}^2 + R_{50}^2 + R_{80}^2 + R_{140}^2}$

$=\sqrt{.25^2 + .5^2 + .25^2 + .25^2 + .25^2}$

$=0.7071$

Crisp Output $=\frac{(0.25*0.2)+(0.6124*0.4)+(0.6614*0.65)+(0.7071*0.9)}{0.25+0.6124+0.6614+0.7071}$

$=0.61$

$=61\%$

The patient with certainty Factor CF of 61.8570 N(0.61)

The CF of this case is 0.61

CHAPTER FOUR

The Proposed system

4.1 User Interface

User interface provides interaction between user of the ES and the ES itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.

It explains how the ES has arrived at a particular recommendation. The explanation may appear in the following forms:

- Natural language displayed on screen.
- Verbal narrations in natural language.
- Listing of rule numbers displayed on the screen.
- The user interface makes it easy to trace the credibility of the deductions.

Requirements of Efficient ES User Interface:

- It should help users to accomplish their goals in shortest possible way.
- It should be designed to work for user's existing or desired work practices.
- Its technology should be adaptable to user's requirements; not the other way round.
- It should make efficient use of user input.

4.2 Development of Expert Systems:

The process of ES development is iterative. Steps in developing the ES include:

Identify Problem Domain: The problem must be suitable for an expert system to solve it .then find the experts in task domain for the ES project and Establish cost-effectiveness of the system.

Design the System: Identify the ES Technology **to** Know and establish the degree of integration with the other systems and databases. Then realize how the concepts can represent the domain knowledge best.

Develop the Prototype from Knowledge Base: The knowledge engineer works to acquire domain knowledge from the expert. And Represent it in the form of If-THEN-ELSE rules.

Test and Refine the Prototype: The knowledge engineer uses sample cases to test the prototype for any deficiencies in performance. And End users test the prototypes of the ES.

Develop and Complete the ES: Test and ensure the interaction of the ES with all elements of its environment, including end users, databases, and other information systems. Document the ES project well.at last Train the user to use ES.

4.3 Maintain the Expert Systems

- Keep the knowledge base up-to-date by regular review and update.
- Cater for new interfaces with other information systems, as those systems evolve.

4.4 Benefits of Expert Systems

Availability: They are easily available due to mass production of software.

Less Production Cost: Production cost is reasonable. This makes them affordable.

Speed: They offer great speed. They reduce the amount of work an individual puts in.

Less Error Rate: Error rate is low as compared to human errors.

Reducing Risk: They can work in the environment dangerous to humans.

Steady response: They work steadily without getting motional, tensed or fatigued.

4.5 Methodologies

Three methodologies are proposed to control non-linear multi-input and multi-output process. A rule-base construction system which is aimed at extracting control rules from the process of iterative learining is proposed. This means that the correct control actions are progressively learned by operating the system repeatedly. Clearly , this process is similar to the learning process processed by human bieng.

The number of rules does not depend on the complexity of process or on the number of inputs/outputs, it depend on the desired sampling time and time references . The rule-base can be initialy empty and is constructed online. The coupling between input and ouput variables is handeled through a number of rules that constitute defuzzification mechanism and resullts in a decoupled stategy . The fuzzyfication and inference stage use the concept of adaptive simalrity factor in multi-input and multi-ouptput framework.

Each algorithm is collection of learning and reasonable algorithms. The learning algorithm is designed as conventional learning algorithm and as a neural learning algorithm , while two kinds of reasoning algorithms are designed, the first is direct reasoning algorithm and the second is indirect reasoning algorithm.

The first algorithm use conventional learning and direct reasoning, the second use conventional learning and indirect reasoning, the third use neural learning and indirect reasoning. The overall system compose four functional models: the refernce model, the learning algorithm, the rule-base formation mechanism and the controlled process which is assumed to have two inputs and two outputs.

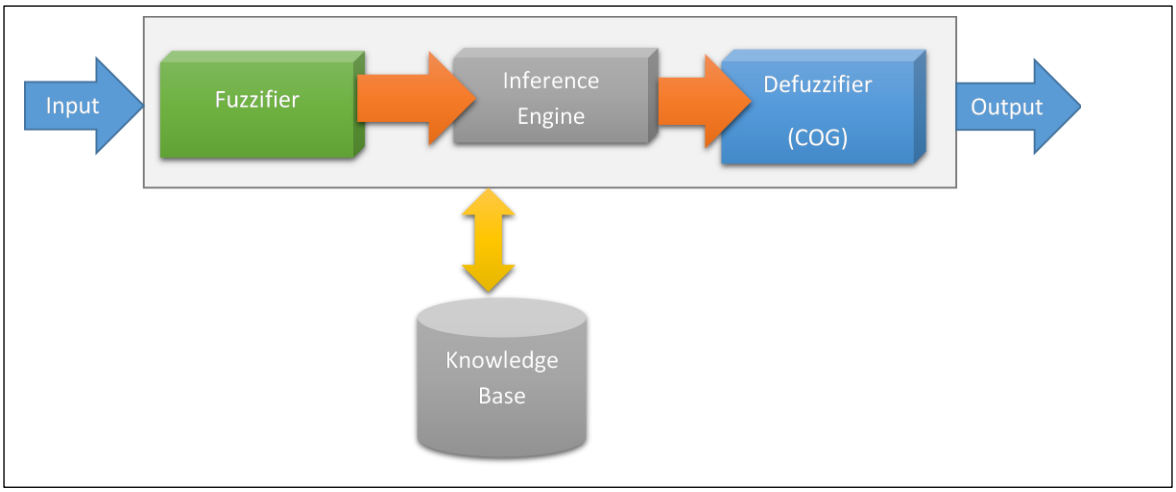

Figure (4.1) shows the TBES fuzzy system Architecture

4.6 The Expert System Application Software

This chapter will show up the interfacing software (act as medical discion assistant) as final stage of this thesis. TBES software is interface between knowldege base and doctor.

The TBES capture patient data (such as demographic and patient symptoms), analyzed and finally help the doctor to decide if the case is a TB suspected, and if so, how the percentage of the level of suspection.

A senior software engineer help me to acomplished this part and he developed the senario as I have drawn in this thises.

The TBES software was based on four core modules integrated to each other, and each of it has several submodules.

Patient management: This module is responsible of capturing patient data, in a very simple manner, by providing user friendly interface, which help the health care professional to get all information need by a doctor, including demographic data and clinical data such as patient's symptoms. And most importantly feature included is data validation tools, which validate the all information that has been entered by the user before it saved in the database. These validation tools are helping to reduce the number of exception that may be caused by these wrong data.

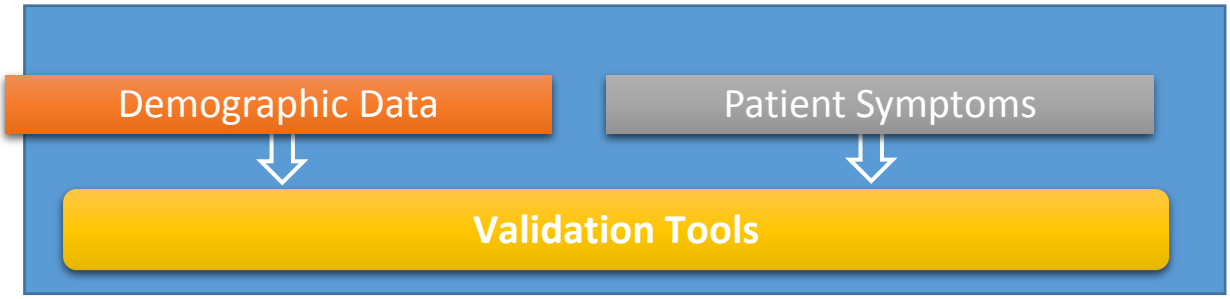

Figure (4.2) Depicts Patient Management Module

By considering the requirements for improving the health care professional decision about TB suspicion and quality of both demographic and medical data, key steps include:

Patient registration: The main purpose of this tool is to capture patient demographic information of a case, we can use some of this information later in the reporting to give a full picture about the TB disease and how it has been spread in a community or a specific region.

Case Module: An electronic form used to manage patient's clinical information, it provides a simple and easy to use interface to deal with all clinical activites done for patient care. This module has sub-modules to ensure scalability , reusability of TBES . It consists of a case tool used for case data acquisition and has additional features for entering data, tracking patient visits and other features.

Patient case: Prepare a case for medical consult visit (e.g. first visit or flow up visit) or close the case, and keep tracking of the patient medical history.

Patient symptoms: A weighted factors tool not only records the patient systems, also see the acuteness of the symptom (mild, moderate, severe or very severe), the ultimate decision about TB suspect ion depends many on this step.

TB disease risk factor: An optional tool could help the healthcare practitioner, to this tool work on the demographic data of the patient, it analyzes the data, then notify the immediately if the patient is close to the risk of been infected by TB disease.

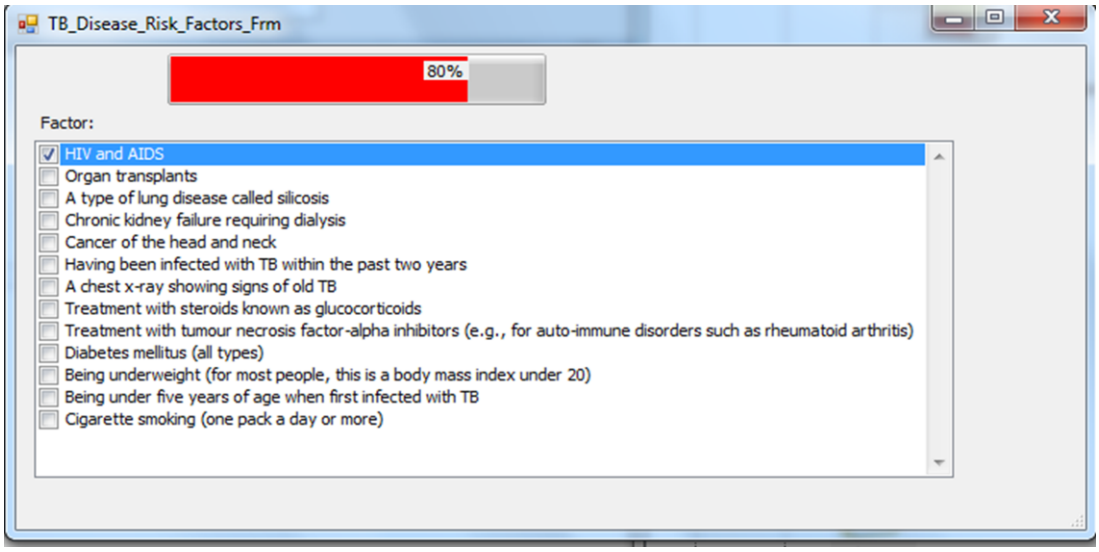

Figure (4.3) depicts the TB disease risk factor tool

TB infection risk factor: This tool work on the demographic data of the patient, it analyzes the data, then notify the healthcare practitioner immediately if the patient is close to the risk of been infected by TB disease.

Patient Index:

A searching tool allows users to search for identify and determine the patient master record, by providing a user friendly interface to make the searching process simple and easy to use.

Case evaluation: This module is the core of the application, because it evaluates the case regarding to his/her symptoms, infection risk factors, disease risk factors and or the clinical screening, and by applying some fuzzy assessments, then gives the probability of how patient is TB case?

System settings: Provide mechanisms to setup and configure the system and the knowledgebase. It has many forms that can help the system's user to customize his works, not once even regularly, depends on the clinical policies.

Rules Settings: After selecting the membership functions, the rules are also generated using the Rule-Editor. Then the rules can be viewed using the Rule-Viewer. In Fuzzy Inference Systems, based on the knowledge provided by the Domain Experts (Physician), decisions are made and outputs are generated. While collection of this type of knowledge generates a fuzzy knowledge base system,

which is basically collection of some fuzzy IF-THEN rules. In this proposed system, 1024 such type of fuzzy IF-THEN rules is generated by consulting various Domain Experts (Physician).

Rules: Rules are the popular paradigm for representing knowledge. A rule based expert system is one whose knowledge base contains the domain knowledge coded in the form of rules.

To get more felixble and more interactive TB_ES application, we develop a rules module which consists of two major parts, rule builder and rules interpreter,

Rules Builder: It's a very smart tool to formalizes and organizes knowledge, and as we know a knowldege is a set of rules, even simple rule or complex rule. A rules builder has two major components simple and complex rules builder.

Atomic rules builder: A user interactive tool that make the construction of simple rule easy, which conssits of feature (such feature of TB, like a known symptoms and screening for TB) and linguistic variable. Consequently a simple rule like to be in final form of "IF feature IS linguistic value, THEN …. " for example: " IF(A bad cough lasting longer than two weeks = Moderate " . Figure (4.4) is a screenshot that depicts how to build an atomic rule.

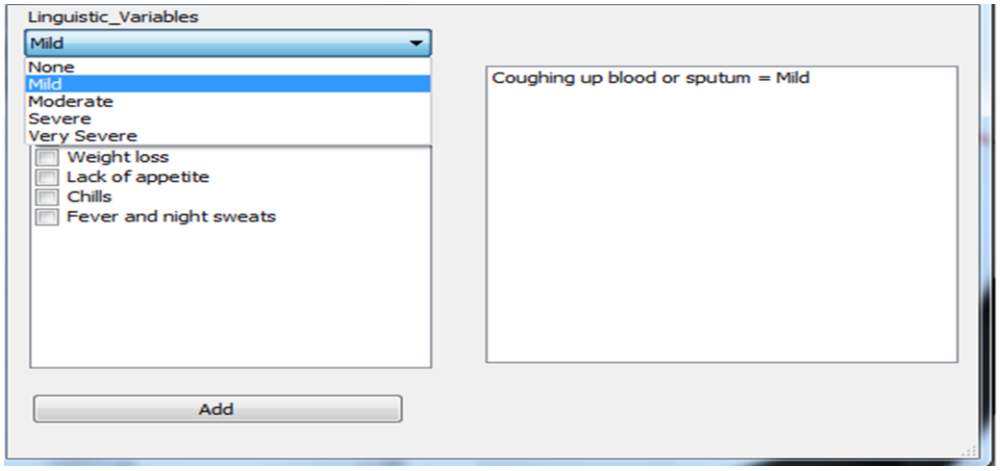

Figure (4.4): The interactive for constructing an atomic rule and stored in the knowledgebase

After the application user construct such a rule, rules builder esure the uniquness of the simple rule, and then store it in the database, for later use. Figure (4.5) shows how the rulse values are s tored in the system database.

	Code	Name	Value	Linguistic_Variable_Code
1	1	A bad cough lasting longer than two weeks	0.32	3
2	2	Chest pain	0.65	3
3	3	Coughing up blood or sputum	0.72	3
4	4	Weakness or feeling very tired	0.12	3
5	5	Weight loss	0.07	3
6	6	Lack of appetite	0.02	3
7	7	Chills	0.22	3
8	8	Fever and night sweats	0.12	3
9	9	A bad cough lasting longer than two weeks	0.72	4
10	10	Chest pain	0.72	4
11	11	Coughing up blood or sputum	0.72	4

Figure (4.5): a screen shot shows how simple rules stored in the database management system (MS-SQL Server in my case)

Complex Rules builder

Simply a complex rule consists of two or more simple rules, and it's a key part of core expert system knoldege base.

A Complex Rules builder is a very samart convenient and interactive tool to build, rebuild and validate , formulate and organize a complex rules . This tool captures the rules from the expert user and then formulated into a final form to be stored in the database.

After capturing a complex rule from the expert, the complex builder tool maps each simple rule into a number instead of text, for simplecity and programming purposes, and use a single chacter for joinig a rules together. Here is an example for rule format in database

33&2&3&4&5&6&15&8

Figure (4.6) show a screenshot of rules tables in the database.In that figure we use code for identifing the rule, and the rules field to stores the rules as it formatted in

the application, and finally the Linguistc Variable Code points to the code of linguistc variable table

	Code	Rules	Linguistic_Variable_Code
1	1	1&2&3&4&5&6&7&8	2
2	2	1&10&3&4&5&6&7&8	2
3	3	1&2&11&4&5&6&7&8	2
4	4	1&2&3&12&5&6&7&8	2
5	5	1&2&3&4&13&6&7&8	2
6	6	1&2&3&4&5&14&7&8	2

Figure (4.6): a screen shot shows how a complex rules stored in the database management system (MS SQL Server in my case)

Each number in this example represents a simple rule for instance 33 represents A " IF Bad cough lasting longer than two weeks is Mild"

So the a bove comples rule is transleated into coresponding text

(A bad cough lasting longer than two weeks = none)

AND (Chest pain = Mild)

AND (Coughing up blood or sputum = Mild)

AND (Weakness or feeling very tired = Mild)

AND (Weight loss = Mild)

AND (Lack of appetite = Mild)

AND (Chills = Moderate)

AND (Fever and night sweats = Mild)

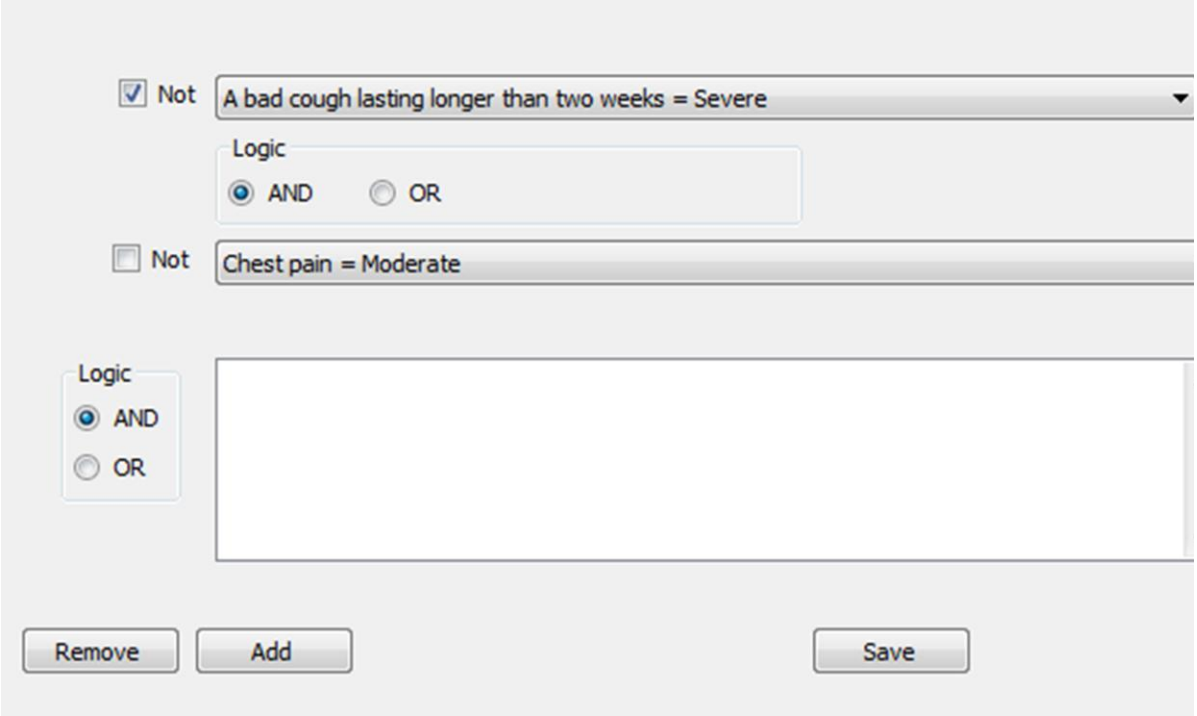

Figure (4.7): a screen shot shows how a user *constructing a complex rule*

Rules Interpreter: A dictionary tool that transelate the rule record in the database from symbolic form to understandable human language. It's purpose to makes a system administrator, a clinic researcher or an expert to understand what's the meaning and the structure of a specific rule. Figure (5) is a screen shot took from TB_ES, that shows the rule interpreter.

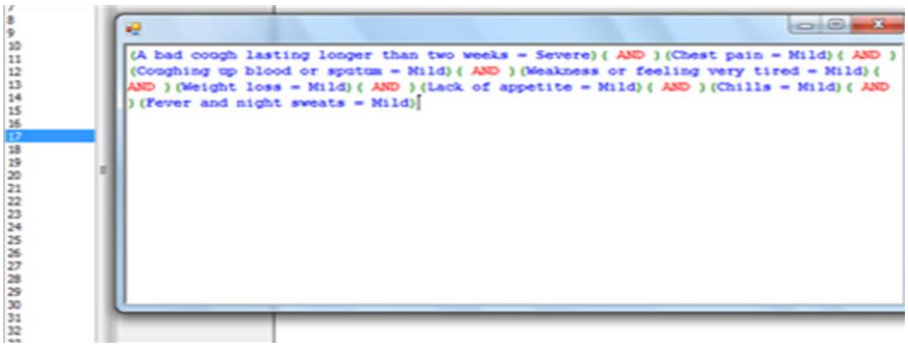

Figure (4.8) is a screen shot

4.7 Results Analysis

The output of this application which based on my thesis is coming out to be more reliable and dependable as I have used the fuzzy approach to diagnose the TB disease. Till date the best work done in this field was of "Fuzzy Expert System -Based Approach for TB Diagnosis" who had worked upon he fuzzy model to predict the level of suspection of TB for a patient based on the symptoms acquiered.

Chapter Five

Results and Discussion

5.1 Research Experiment

I designed a forms to collect information to get a full view of relationship between the TB symptoms and the degree of TB disease suspicion. The paper Filled carefully by physicians and registrar in specialist TB health care facilities in Khartoum state, including **Abu Angaa** quarantine in Omdurman besides taken data from hospital medical records, for both admitted and outpatients, has been followed up for their cases for about two years.

The main purposes of this paper, was to check accuracy of results that has come out by the expert system software.

After filling symptoms for each case in the expert system software screens, then the software completed its process as I described here in this chapter, we compare the final conclusion from the software with the decision of the consulting physician for the same case in the questioner. And here the summary of that comparison:

No of cases (questioners) = 60 distinct cases

The results for correspondence cases in the software and the actual physician final diagnosis as follows:

- 43 cases out of 60 cases the software gives exact diagnosis as the physician was.
- 12 cases out of 60 cases the software gives diagnosis to some extend the close the physician final diagnosis (e.g. the software concludes case as sever TB, and the actual diagnosis was moderate).
- 5 cases out of 60 cases software give totally different results than the actual final diagnoses.

That 5 cases are out may be due bad handwriting when I copied symptoms from the hospital records

My results based on real patient data confirms that the fuzzy logic expert system can represent the expert's thinking in a satisfactory manner in handling complex trade-offs. Fuzzy logic systems are excellent in handling ambiguous and imprecise information prevalent in medical diagnosis.

CHAPTER SIX

6-1 Conclusion

The using of fuzzy logic in medical diagnosis cannot be overemphasized. Fuzzy logic for medical diagnosis provides an efficient way to assist inexperienced physicians to arrive at the final diagnosis of TB more quickly and efficiently. The developed system provides decision support platform to assist TB researchers, physicians and other health practitioners in TB endemic regions.

6-2 Recommendation

I recommend:

1. If this thesis is fully implemented it will greatly aid the distribution of primary health care services
2. This system further aids in acquiring patients' medical information via web for use in TB diagnosis, thereby ensuring medical consultation without walls.

References

[1] Testing for tuberculosis Anastasios Konstantinos, Director of Queensland TB Control Centre (Specialised Health Services), Queensland Health, Brisbane 2013,

[2] TB FatemeKhodabakhshi1, SeyedMohammadShahraeini2 Social Development & Health Promotion Research Center Designing an expert system for diagnosis of Vol.3, No.1, 2013

[3] Dakshata Panchal and Seema Shah International Journal of Modeling and Optimization Artificial Intelligence Based Expert System for Hepatitis B Diagnosis, Vol. 1, No. 4, October 2011

[4] World Health Organization Global Tuberculosis Report 2014,

[5] Okpokpong Nathaniel Ntebong Celestine, Umar Farouk Ibn Abdulrahman, Itoro Akpabio Fuzzy Expert Framework for Diagnosis of Typhoid Fever Vol.2, No.9, pp: (13-16), October 2017.

[6]Victor C Osamor, Ambrose A Azeta and Oluseyi O Ajulo Diagnostic Expert System: architecture for translating patients -Tuberculosis to automate information from the web for use in tuberculosis diagnosis (Jan 21, 2014)

[7] Gufran Ahmad Ansari an Adoptive Medical Diagnosis System Using Expert System with Applications

, (KSA-2014)

[8] Author: Robin Artificial Intelligence Articles on Artificial Intelligence, October 17th, 2010 |

[9] Ayia Napa, Cyprus Wireless Mobile Communication and Healthcare: Second International ICST Conference, MobiHealth, October 18 - 20, 2010,

[10] Crina Grosan, Ajith Abraham, Intelligent Systems, Springer-Verlag Berlin Heidelberg 2011.

[11] Beth A. SprouleEmail the author Beth A. Sproule, Claudio A. Naranjo, I.Burhan Türksen Fuzzy pharmacology: theory and applications, Volume 23, Issue 9, p412–417, 1 September 2002

[12] Imianvan, A. A, Amadin, F. I... Prototype of a Neuro-fuzzy System for Detection of Environmental Induced Depression, Vol. 2 (2012), 79-90

[13] Robert Full´er Donner Visiting professor Abo Akademi University
Neural Fuzzy Systems °Abo 1995

[14] M.M. Gupta and D.H. Rao,On the principles of fuzzy neural networks, Fuzzy Sets and Systems 61 (1994) 1-18

[15] Ojeme Blessing Onuwa, Fuzzy Expert System for Malaria Diagnosis, Mathematics and Computer science Department Delta State University, Abraka

[16] Eddie C. L. Chan, George Baciu , and Introduction to Wireless Localization: With iPhone SDK Examples 2011.

[17] edited by Management Association, Information Resources Gamification: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications2012.
[18] Umoh, Uduak A. Ntekop, Mfon M. A Proposed Fuzzy Framework for Cholera Diagnosis and Monitoring, Volume 82 – No 17, November 2013
[19] Srinivasa Rao D Krishna Prasad MHM Seetha M ,Comparison of Fuzzy and Neuro Fuzzy Image Fusion Techniques and its Applications Volume 43 –No.20, April 2012
[20] Mobile Intelligent Autonomous System , edited by Jitendra R. Raol, Ajith K. Gopal
[21] X.Y. Djam1,*, G. M. Wajiga2, Y. H. Kimbi3 and N.V. Blamah4A Fuzzy Expert System for the Management of Malaria ,Int. J. Pure Appl. Sci. Technol., 5(2) (2011), pp. 84-108
[22] L. A. Zadeh ,The Concept of a Linguistic Variable and its Application to Approximate Reasoning (1975)
[23] Clarence W. de Silva ,Intelligent Machines: Myths and Realities, CRC Press, 22 Jun 2000
[24] B. V. Babu, Atulya Nagar, Kusum Deep, Millie Pant, Jagdish Chand Bansal, Kanad Ray, Umesh Gupta , Proceedings of the Second International Conference on Soft Computing for ,2014..

BIBLOGRAPHY

APPENDICES

Appendix A: System Source Code

```
namespace Expert_System
{
   public partial class MainFrm : Form
   {
      public MainFrm()
      {
         InitializeComponent();
      }
      private void mainTreeView_DoubleClick(object sender, EventArgs e)
      {
         switch ((((TreeView)sender).SelectedNode).Name)
         {
            case ("regNode"):
               {
                  (new Patient_Reg()).ShowDialog();
                  break;
               }
            case ("caseNode"):
               {
                  PatientIndex ptIx= new PatientIndex() ;
```

```
        if (ptIx.ShowDialog() == DialogResult.OK)
        {
            CaseFrm caseFrm = new CaseFrm();
            caseFrm.acc = ptIx.accession;
            caseFrm.ptName = ptIx.ptName;
            caseFrm.ShowDialog();
        }
        break;
    }
case ("diseaseFactorsNode"):
    {
        PatientIndex ptIx = new PatientIndex();
        if (ptIx.ShowDialog() == DialogResult.OK)
        {
            CaseDiseaseFactorsFrm diseaseFrm = new CaseDiseaseFactorsFrm();
            diseaseFrm.accBox.Text = ptIx.accession.ToString();
            diseaseFrm.nameBox.Text = ptIx.ptName;
            diseaseFrm.ShowDialog();
        }
        break;
    }
case ("infectionFactorsNode"):
    {
        PatientIndex ptIx = new PatientIndex();
```

```
        if (ptIx.ShowDialog() == DialogResult.OK)
        {
            InfectionFactorsFrm infectionFrm = new InfectionFactorsFrm();
            infectionFrm.accBox.Text = ptIx.accession.ToString();
            infectionFrm.nameBox.Text = ptIx.ptName;
            infectionFrm.ShowDialog();
        }
        break;
    }
case ("symptomsNode"):
    {
        PatientIndex ptIx= new PatientIndex() ;
        if (ptIx.ShowDialog() == DialogResult.OK)
        {
            SymptomsFrm symfrm = new SymptomsFrm();
            symfrm.nameBox.Text = ptIx.ptName;
            symfrm.accBox.Text = ptIx.accession.ToString();
            symfrm.ShowDialog();
        }
        break;
    }
case ("indexNode"):
    {
        (new PatientIndex()).ShowDialog();
```

```
        break;
    }
case ("finalNode"):
    {
        PatientIndex ptIx = new PatientIndex();
        if (ptIx.ShowDialog() == DialogResult.OK)
        {
            Final_DecisionFrm final = new Final_DecisionFrm();
            final.accBox.Text = ptIx.accession.ToString();
            final.nameBox.Text = ptIx.ptName;
            final.ShowDialog();
        }
        break;
    }
case ("rulesNode"):
    {
        (new RulesSettingsFrm()).ShowDialog();
        break;
    }
case ("featuresNode"):
    {
        (new Features_SettingsFrm()).ShowDialog();
        break;
    }
```

```
case ("diseaseNode"):
   {
      (new TB_Disease_Risk_Factors_Frm()).ShowDialog();
      break;
   }
case ("infectionNode"):
   {
      (new TB_Infection_Risk_FactorsFrm()).ShowDialog();
      break;
   }
case ("interpertationNode"):
   {
      (new Rules_InterperterFrm()).ShowDialog();
      break;
   }
case ("reportsNode"):
   {
        break;
   }
case ("sympSettNode"):
   {
      (new Sysmtoms_SettingsFrm()).ShowDialog();
      break;
   }          }       }    } }
```

```
namespace Expert_System.Inference
{
    public class TriFunction
    {
        double x;
        double a;
        double b;
        double c;
        double Value = 0;
        public TriFunction(double val)
        {
            x = val;
        }
        public TriFunction(double val, double a1, double b1, double c1)
        {
            x = val;
            a = a1;
            b = b1;
            c = c1;
        }
        public void SetRanges(double a1, double b1, double c1)
        {
            a = a1;
            b = b1;
```

```
    c = c1;
}
public double GetValue()
{
    if (x <= a)
        Value = 0;
    else
        if (x <= a)
            Value = 0;
        else
            if (x >= a && x <= b)
            {
                Value = (x - a) / (b - a);
            }
            else
                if (x >= b && x <= c)
                {
                    Value = (c - x) / (c - b);
                }
                else
                    if (x >= c)
                        Value = 0;
    return Value;
}   }}
```

```
namespace Expert_System.Interface
{
  public partial class Address
    {
        public decimal Code { get; set; }
        public decimal Accession_No { get; set; }
        public decimal State_Code { get; set; }
        public string Tel { get; set; }
        public string Street { get; set; }
        public decimal City_Code { get; set; }
        public string Other { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Case
    {
        public Case()
        {
            this.Case_Symptoms = new HashSet<Case_Symptoms>();
        }
        public decimal Code { get; set; }
        public decimal Accession_No { get; set; }
        public decimal Visit_No { get; set; }
```

```
        public System.DateTime Visit_Date { get; set; }
        public decimal Doctor_Code { get; set; }
        public decimal User_Code { get; set; }
        public byte Close_Case { get; set; }
        public byte Visit_Type { get; set; }
        public virtual ICollection<Case_Symptoms> Case_Symptoms { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Case_Disease_Risk_Factors
    {
        public decimal Code { get; set; }
        public decimal Case_Code { get; set; }
        public decimal Disease_Risk_Factors_Code { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Case_Screening
    {
        public decimal Code { get; set; }
        public decimal Created_By { get; set; }
        public System.DateTime Create_Time { get; set; }
```

```
        public decimal Modified_By { get; set; }
        public System.DateTime Modify_Time { get; set; }
        public decimal Case_Code { get; set; }
        public decimal Screening_Code { get; set; }
        public decimal Screening_Result_Code { get; set; }
        public string Screening_Result_Desc { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Case_Symptoms
    {
        public decimal Code { get; set; }
        public decimal Case_Code { get; set; }
        public decimal Symptom_Code { get; set; }
        public decimal Linguistic_Variables_Code { get; set; }
         public virtual Case Case { get; set; }
        public virtual Linguistic_Variables Linguistic_Variables { get; set; }
    }
}
namespace Expert_System.Interface
{
   public partial class Case_TB_Infection_Risk_Factors
   {
```

```
        public decimal Code { get; set; }
        public decimal Case_Code { get; set; }
        public decimal Infection_Risk_Factors_Code { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class CaseFrm : Form
    {
        public string ptName = "";
        public decimal acc ;
        public CaseFrm()
        {
            InitializeComponent();
        }
        private void CaseFrm_Load(object sender, EventArgs e)
        {
            nameBox.Text=ptName;
            accBox.Text = acc.ToString();
            GetCases(acc);
        }
        private void GetCases(decimal acc)
        {
            ESEntities Es = new ESEntities();
```

```
    var query = (from i in Es.Patients
                 where i.Accession_no == acc
                 select i).Single<Patient>();
    nameBox.Text = query.Name;
    Case ca = new Case();
    var ptCase = (from i in Es.Cases
                  join d in Es.Doctors on  i.Doctor_Code equals d.Code into codeDoc
            from c in codeDoc
                  join t in Es.Visit_Type on  i.Visit_Type equals t.Code into ty from y in ty
                  join s in Es.CaseStatus
                  on i.Close_Case equals s.Code
                  where i.Accession_No == acc
                  select new { i.Code, i.Visit_Date, i.Visit_No,y.Type, s.Status, c.Name });
    caseGrid.DataSource = ptCase.ToList();
}
private void searchButton_Click(object sender, EventArgs e)
{
    PatientIndex pIx = new PatientIndex();
    try
    {
       if (pIx.ShowDialog() == DialogResult.OK)
       {
          accBox.Text = pIx.accession.ToString();
          ESEntities Es = new ESEntities();
```

```
            var query = (from i in Es.Patients
                         where i.Accession_no == pIx.accession
                         select i).Single<Patient>();
            nameBox.Text = query.Name;
            Case ca = new Case();
            var ptCase = (from i in Es.Cases
                          join d in Es.Doctors on i.Doctor_Code equals d.Code into codeDoc
                          from c in codeDoc
                          join t in Es.Visit_Type on i.Visit_Type equals t.Code into ty
                          from y in ty
                          join s in Es.CaseStatus
                          on i.Close_Case equals s.Code
                          where i.Accession_No == pIx.accession
                          select new { i.Code, i.Visit_Date, i.Visit_No, y.Type, s.Status, c.Name });
            caseGrid.DataSource = ptCase.ToList();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error in searcht\n " + ex.Message);
    }
}

private void newButton_Click(object sender, EventArgs e)
```

```
{
   try
   {
      ESEntities Es = new ESEntities();
      decimal pIx = Convert.ToDecimal(accBox.Text);
      Case[] ptCase = (from i in Es.Cases
                  where i.Accession_No == pIx
                  select i).ToArray<Case>();
      NewVisitFrm vstfrm = new NewVisitFrm();
      vstfrm.name = nameBox.Text;
      vstfrm.cs.Accession_No = Convert.ToDecimal(accBox.Text);
      vstfrm.cs.Visit_Type = 2;
      vstfrm.cs.Visit_No = ptCase[ptCase.Length-1].Visit_No + 1;
      if (vstfrm.ShowDialog() == System.Windows.Forms.DialogResult.OK)
      {
         Case cs = new Case();
         cs.Visit_Type = vstfrm.cs.Visit_Type;
         cs.Visit_Date = vstfrm.cs.Visit_Date;
         cs.User_Code = vstfrm.cs.User_Code;
         cs.Accession_No = vstfrm.cs.Accession_No;
         cs.Doctor_Code = vstfrm.cs.Doctor_Code;
         cs.Visit_No=vstfrm.cs.Visit_No;
         cs.Close_Case = 1;
         Es.Cases.Add(cs);
```

```
                Es.SaveChanges();
                GetCases(pIx);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error in new vistit\n"+ex.Message);
        }
    }
  }
}
namespace Expert_System.Interface
{
  public partial class CaseStatus
  {
    public decimal Code { get; set; }
    public decimal Create_Code { get; set; }
    public System.DateTime Create_Time { get; set; }
    public decimal Modify_Code { get; set; }
    public System.DateTime Modify_Time { get; set; }
    public string Status { get; set; }
  }
}
namespace Expert_System.Interface
```

```
{
    public partial class City
   {
       public decimal Code { get; set; }
       public decimal State_Code { get; set; }
       public string Name { get; set; }
       public string Locale { get; set; }
       public byte Visible { get; set; }
   }
}
namespace Expert_System.Interface
{
   public partial class ComplexRule : Form
   {
       public ComplexRule()
       {
           InitializeComponent();
       }
       private void AddButton_Click(object sender, EventArgs e)
       {
           string Old = "";
           if (listBox1.Items.Count>0)
              Old=listBox1.Items[0].ToString();
           listBox1.Items.Clear();
```

```
    string R1 ="";
    string R2 = "";
   if(R1Box.SelectedIndex>0)
   R1 = R1Box.Items[R1Box.SelectedIndex].ToString();
   if (R2Box.SelectedIndex > 0)
   R2 = R2Box.Items[R1Box.SelectedIndex].ToString();
   if (R1 == R2)
      return;
   string Logic = "And";
   if(AndButton.Checked)
      Logic = "And";
   else
      if(OrButton.Checked)
         Logic = "Or";
   string value ="("+ R1 + ")" + Logic + "(" + R2 + ")";
   string cLogic = "";
   if (CAndButton.Checked)
      cLogic = "And";
   else
      if (COrButton.Checked)
         cLogic = "Or";
   listBox1.Items.Add(Old+" "+cLogic+" "+value);
}
private void ComplexRule_Load(object sender, EventArgs e)
```

```
        {
            R1Box.Items.Clear();
            R2Box.Items.Clear();
            ESEntities Es = new ESEntities();
            foreach (Rule r in Es.Rules)
            {
                R1Box.Items.Add(r.Name + " = " + r.Linguistic_Variables.Varialble);
                R2Box.Items.Add(r.Name+" = "+r.Linguistic_Variables.Varialble);
            }
        }
        private void RemoveButton_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
        }
    }
}
namespace Expert_System.Interface
{
    public partial class Country
    {
        public decimal ID { get; set; }
        public string Name { get; set; }
        public string Locale { get; set; }
        public byte Visible { get; set; }
```

```
    }
}
namespace Expert_System.Interface
{
    static class  DataBaseHandelrs
    {
            static public Patient SetPatient(string name, DateTime dt, byte sex, decimal nationality, decimal occupation)
        {
            Patient pt = new Patient();
            pt.BoD = dt;
            pt.Name = name;
            pt.Nationality = nationality;
            pt.Occupation = occupation;
            pt.Sex = sex;
            pt.Create_Time = DateTime.Today;
            pt.Modify_Time = DateTime.Today;
            ESEntities Es = new ESEntities();
            try
            {
                Es.Patients.Add(pt);
                foreach (System.Data.Entity.Validation.DbEntityValidationResult s in Es.GetValidationErrors())
                {
                }
```

```
        Es.SaveChanges();
        return pt;
    }
    catch (Exception ex)
    {
        MessageBox.Show("A problem caused in saving patient data process " + ex.Message);
        return pt;
    }
}
static public void SetAddress(Address add)
{
    ESEntities Es = new ESEntities();
    try
    {
        Es.Addresses.Add(add);
        Es.SaveChanges();
    }
    catch (Exception ex)
    {
        MessageBox.Show("A problem saving address data " + ex.Message);
    }
}
static public decimal GetLinguistic_Variable_Code(string c)
{
```

```
    ESEntities Es = new ESEntities();
    try
    {
        var query = (from i in Es.Linguistic_Variables
                     where i.Varialble == c
                     select i.Code).Single<decimal>();
        return query;
    }
    catch (Exception ex)
    {
        MessageBox.Show("A problem getting Linguistic Variables code " + ex.Message);
        return 0;
    }
}
static public decimal GetSymptom_Code(string c)
{
    ESEntities Es = new ESEntities();
    try
    {
        var query = (from i in Es.Symptoms
                     where i.Name == c
                     select i.Code).Single<decimal>();
        return query;
    }
```

```
    catch (Exception ex)
    {
        MessageBox.Show("A problem getting Symptom code " + ex.Message);
        return 0;
    }
}
static public decimal GetInfection_Code(string c)
{
    ESEntities Es = new ESEntities();
    try
    {
        var query = (from i in Es.TB_Infection_Risk_Factors
                     where i.Factor == c
                     select i.Code).Single<decimal>();
        return query;
    }
    catch (Exception ex)
    {
        MessageBox.Show("A problem getting Infection code " + ex.Message);
        return 0;
    }
}
static public void OpenConnection()
{
```

```
            try
            {
                MessageBox.Show ("Connection Open ! ");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Can not open connection ! "+ex.Message);
            }
        }    }}
namespace Expert_System.Interface
{
    public partial class Doctor
    {
        public decimal Code { get; set; }
        public string Name { get; set; }
        public string Speciality { get; set; }
    }
}
namespace Expert_System.Interface
{
    public class Evaluatation
    {
        decimal Case_No;
        double None = 0;
```

```
double Mild = 1;
double Moderate = 2;
double Severe = 3;
double Very_Severe = 4;
public double Crisp_Output = 0;
decimal None_Code = 2;
decimal Mild_Code = 3;
decimal Moderate_Code = 4;
decimal Severe_Code = 5;
decimal Very_Severe_Code = 6;
List<Case_Symptoms> Case_SymptomsList=new List<Case_Symptoms>();
public Evaluatation(decimal caseNo)
{
   Case_No = caseNo;
   Case_SymptomsList= GetCase_Symptoms(Case_No);
   Evaluate();
}
public List<Case_Symptoms> GetCase_Symptoms(decimal Case_No)
{
   ESEntities Es = new ESEntities();
   List<Case_Symptoms> query= (from i in Es.Case_Symptoms
                        where i.Case_Code==Case_No
                         select i).ToList<Case_Symptoms>();
   return query;
```

```
}
double GetSumSqRoot(decimal rcode,double val)
{
   ESEntities Es = new ESEntities();
   List<decimal> query = (from i in Case_SymptomsList
                          where i.Case_Code == Case_No
                    select i.Linguistic_Variables_Code).ToList<decimal>();
   int n = query.Count;
   double S = 0;
   S = Math.Sqrt(n * val*val);
   return S;
}
private void Evaluate()
{
   None = GetSumSqRoot(None_Code, None/4);
   Mild = GetSumSqRoot(Mild_Code, Mild/4);
   Moderate = GetSumSqRoot(Moderate_Code, Moderate / 4);
   Severe = GetSumSqRoot(Severe_Code, Severe / 4);
   Very_Severe = GetSumSqRoot(Very_Severe_Code, Very_Severe / 4);
   Expert_System.Inference.TriFunction mildFun=new Inference.TriFunction(Mild);
   mildFun.SetRanges(0.1,(0.3-0.1)/0.2,0.3);
   double mild = mildFun.GetValue();
   Expert_System.Inference.TriFunction modrarteFun = new Inference.TriFunction(Mild);
   modrarteFun.SetRanges(0.3, (0.6 - 0.3) / 0.2, 0.6);
```

```
            double modrarte = modrarteFun.GetValue();
            Expert_System.Inference.TriFunction severeFun = new Inference.TriFunction(Mild);
            severeFun.SetRanges(0.6, (0.8 - 0.6) / 0.2, 0.8);
            double severe = mildFun.GetValue();
            Expert_System.Inference.TriFunction verysevereFun = new Inference.TriFunction(Mild);
            verysevereFun.SetRanges(0.8, (1 - 0.8) / 0.2, 1);
            double verysevere = mildFun.GetValue();
            double S=0;
            S=Mild+Moderate+Severe+Very_Severe;
            Crisp_Output = (Mild * mild + Moderate*modrarte + Severe*severe + Very_Severe*verysevere) / S;
        }
    }
}
namespace Expert_System.Interface
{
    public partial class Feature
    {
        public decimal Code { get; set; }
        public byte Type { get; set; }
        public decimal Weight { get; set; }
        public decimal Feature_Code { get; set; }
    }
}
namespace Expert_System.Interface
```

```
{
    public partial class HIV_TB_Country_Risks
    {
        public decimal Code { get; set; }
        public decimal Country_Code { get; set; }
        public decimal Risk_Factor_HIV { get; set; }
        public decimal Risk_Factor_TB { get; set; }
    }
}
namespace Expert_System.Interface
{
     public partial class Linguistic_Variables
    {
        public Linguistic_Variables()
        {
            this.Case_Symptoms = new HashSet<Case_Symptoms>();
            this.Rules = new HashSet<Rule>();
            this.RulesBases = new HashSet<RulesBas>();
        }
      public decimal Code { get; set; }
        public string Varialble { get; set; }
        public string Variable_Locale { get; set; }
        public Nullable<decimal> Value { get; set; }
         public virtual ICollection<Case_Symptoms> Case_Symptoms { get; set; }
```

```
        public virtual Range_Of_Fuzzy_Values Range_Of_Fuzzy_Values { get; set; }
        public virtual ICollection<Rule> Rules { get; set; }
        public virtual ICollection<RulesBas> RulesBases { get; set; }
    }
}
namespace Expert_System.Interface
{
    public class ProgressBarEx : ProgressBar
    {
        private SolidBrush brush = null;
        public ProgressBarEx()
        {
            this.SetStyle(ControlStyles.UserPaint, true);
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            if (brush == null || brush.Color != this.ForeColor)
                brush = new SolidBrush(this.ForeColor);
            Rectangle rec = new Rectangle(0, 0, this.Width, this.Height);
            if (ProgressBarRenderer.IsSupported)
                ProgressBarRenderer.DrawHorizontalBar(e.Graphics, rec);
            rec.Width = (int)(rec.Width * ((double)Value / Maximum)) - 4;
            rec.Height = rec.Height - 4;
            e.Graphics.FillRectangle(brush, 2, 2, rec.Width, rec.Height);
```

```
        }
    }
}
Namespace Expert_System.Interface
{
    public partial class NewVisitFrm : Form
    {
        public Case cs = new Case();
        public string name = "";
        public NewVisitFrm()
        {
            InitializeComponent();
        }
        private void NewVisitFrm_Load(object sender, EventArgs e)
        {
            ESEntities Es = new ESEntities();
            var query = (from i in Es.Doctors
                         select i.Name).ToList();
            foreach (string s in query)
                doctorBox.Items.Add(s);
            accBox.Text = cs.Accession_No.ToString();
            nameBox.Text = name;
            doctorBox.SelectedIndex = 0;
        }
```

```
        private void selectButton_Click(object sender, EventArgs e)
        {
            cs.Doctor_Code = doctorBox.SelectedIndex+1;
            cs.Visit_Date = visitPicker.Value.Date;
            cs.Visit_Type = 2;
            Close();
        }
        private void visitPicker_ValueChanged(object sender, EventArgs e)
        {
            if (visitPicker.Value.Date < DateTime.Today)
            {
                MessageBox.Show("Error date must be greater that " + DateTime.Today.ToShortDateString(),"Wrong data entry");
                return;
            }       }    }
}
namespace Expert_System.Interface
{
     public partial class Patient
    {
        public decimal Accession_no { get; set; }
        public string Name { get; set; }
        public System.DateTime BoD { get; set; }
        public byte Sex { get; set; }
        public decimal Create_Code { get; set; }
```

```
        public System.DateTime Create_Time { get; set; }
        public decimal Modify_Code { get; set; }
        public System.DateTime Modify_Time { get; set; }
        public Nullable<decimal> Nationality { get; set; }
        public Nullable<decimal> Occupation { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Patient_Reg : Form
    {
        ESEntities Es = new ESEntities();
        public Patient_Reg()
        {
            InitializeComponent();
        }
        void Initialize_Lists()
        {
            CountryBox.Items.Clear();
            NationalityBox.Items.Clear();
            foreach (Country cnt in Es.Countries)
            {
                if (cnt.Visible == 1)
                {
```

```
            CountryBox.Items.Add(cnt.Name);
            NationalityBox.Items.Add(cnt.Name);
        }
    }
    foreach (Occupation occ in Es.Occupations)
    {
        if (occ.Visibile == 1)
        {
            OccupationBox.Items.Add(occ.Name);
        }          }
}
private void Patient_Reg_Load(object sender, EventArgs e)
{
    Initialize_Lists();
    GenderBox.SelectedIndex = 0;
    CountryBox.SelectedIndex = CountryBox.Items.IndexOf("Sudan");
    StateBox.SelectedIndex = StateBox.Items.IndexOf("Khartoum (Al Khartum)");
    CityBox.SelectedIndex = CityBox.Items.IndexOf("Khartoum ");
    OccupationBox.SelectedIndex = OccupationBox.Items.IndexOf("Unemployment");
    NationalityBox.SelectedIndex = NationalityBox.Items.IndexOf("Sudan");
}
private void button1_Click(object sender, EventArgs e)
{
```

```
        decimal acc = DataBaseHandelrs.SetPatient(NameBox.Text, BoDBox.Value.Date,
(byte)GenderBox.SelectedIndex,NationalityBox.SelectedIndex+1,OccupationBox.SelectedIndex+1).Acce
ssion_no;

        Address add = new Address();

        add.Accession_No = acc;

        add.City_Code=CityBox.SelectedIndex;

        add.Street = StreetBox.Text;

        add.Tel = TelBox.Text;

        add.State_Code = StateBox.SelectedIndex;

        add.Other = otherBox.Text;

        DataBaseHandelrs.SetAddress(add);

    }

    private void AgeBox_TextChanged(object sender, EventArgs e)

    {

        try

        {

          int age = 0;

           if (!(String.IsNullOrEmpty(AgeBox.Text) || String.IsNullOrWhiteSpace(AgeBox.Text)))

              age = Convert.ToInt16(AgeBox.Text);

           DateTime BoD = DateTime.Today;

           DateTime Today = DateTime.Today;

           BoD = Today.AddYears(-age);

           BoDBox.Text = BoD.ToShortDateString();

        }

        catch (Exception ex)
```

```
    {
        MessageBox.Show("Error Age:"+ex.Message);
    }
}       private void StateBox_SelectedIndexChanged(object sender, EventArgs e)
{
    CityBox.Items.Clear();
    string state =StateBox.SelectedItem.ToString();
    var stCode = from i in Es.States
                 where i.Name == state
                 select i.Code;
    decimal code = stCode.Single<decimal>();
    List<string> qury = (from i in Es.Cities
                         where i.State_Code == code
                         select i.Name).ToList<string>();
    foreach (string st in qury)
    {
        CityBox.Items.Add(st);
    }
}
private void CountryBox_SelectedIndexChanged(object sender, EventArgs e)
{
    StateBox.Items.Clear();
    string Cnt = CountryBox.SelectedItem.ToString();
    var cntCode = from i in Es.Countries
```

```
                where i.Name == Cnt
                select i.ID;
        decimal code = cntCode.Single<decimal>();
        List<string> qury = (from i in Es.States
                    where i.County_Code == code
                     select i.Name).ToList<string>();
        foreach (string st in qury)
        {
           StateBox.Items.Add(st);
        }       }    }
}
namespace Expert_System.Interface
{
   public partial class Visit_Type
   {
      public decimal Code { get; set; }
      public decimal Create_Code { get; set; }
      public System.DateTime Create_Time { get; set; }
      public decimal Modify_Code { get; set; }
      public System.DateTime Modify_Time { get; set; }
      public string Type { get; set; }
   }
}
namespace Expert_System.Interface
```

```
{
  public partial class TB_Infection_Risk_Factors
    {
        public decimal Code { get; set; }
        public string Factor { get; set; }
        public decimal Value { get; set; }
        public string Locale { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class TB_Disease_Risk_Factors
    {
        public decimal Code { get; set; }
        public string Factor { get; set; }
        public decimal Value { get; set; }
        public string Locale { get; set; }
    }
}
namespace Expert_System.Interface
{
     public partial class Symptoms_Class
    {
        public decimal Code { get; set; }
```

```
        public string Name { get; set; }
        public string Locale { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Symptom
    {
        public decimal Code { get; set; }
        public string Name { get; set; }
        public string Locale { get; set; }
        public decimal Class_Code { get; set; }
        public decimal Factor { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class State
    {
        public decimal Code { get; set; }
        public decimal County_Code { get; set; }
        public string Name { get; set; }
        public string Locale { get; set; }
        public byte Visible { get; set; }
```

```
    }
}
namespace Expert_System.Interface
{
    public partial class SimpleRule : Form
    {
        public SimpleRule()
        {
            InitializeComponent();
        }

        private void SimpleRule_Load(object sender, EventArgs e)
        {
            ESEntities Es=new ESEntities();
            foreach (var l in Es.Linguistic_Variables)
            {
                Linguistic_VariablesBox.Items.Add(l.Varialble);
            }
            RulesBox.Items.Clear();
            foreach (var x in Es.Rules_Data)
            {
                RulesBox.Items.Add(x.Data);
            }
        }
```

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        ESEntities Es = new ESEntities();
        foreach (int itm in RulesBox.CheckedIndices)
        {
            Rule r = new Rule();
            string S = Linguistic_VariablesBox.Items[Linguistic_VariablesBox.SelectedIndex].ToString();
            decimal code = (from i in Es.Linguistic_Variables
                            where i.Varialble == S
                            select i.Code
                           ).SingleOrDefault<Decimal>();
            r.Linguistic_Variable_Code = code;
            r.Name = RulesBox.Items[itm].ToString();
            r.Value = 0;
            Es.Rules.Add(r);
            string val = RulesBox.Items[itm] + " = " + Linguistic_VariablesBox.Items[Linguistic_VariablesBox.SelectedIndex];
            listBox1.Items.Add(val);
        }
        Es.SaveChanges();
    }
    catch (Exception ex)
```

```
        {
            MessageBox.Show("Simple Rules :\n"+ex.Message);         }       }    }
}
namespace Expert_System.Interface
{
   public partial class Screening
   {
      public decimal Code { get; set; }
      public string Name { get; set; }
      public string Locale { get; set; }
   }
}
namespace Expert_System.Interface
{
   public partial class RulesBas
   {
      public decimal Code { get; set; }
      public string Rules { get; set; }
      public decimal Linguistic_Variable_Code { get; set; }

      public virtual Linguistic_Variables Linguistic_Variables { get; set; }
   }
}
namespace Expert_System.Interface
```

```
{
    public partial class Rule
    {
        public decimal Code { get; set; }
        public string Name { get; set; }
        public decimal Value { get; set; }
        public string Comment { get; set; }
        public decimal Linguistic_Variable_Code { get; set; }
            public virtual Linguistic_Variables Linguistic_Variables { get; set; }
    }
}
namespace Expert_System.Interface
{
     public partial class Resident_Groups
    {
        public decimal Code { get; set; }
        public string Name { get; set; }
        public string Locale { get; set; }
        public decimal Risk_Factor { get; set; }
    }
}
namespace Expert_System.Interface
{
    public partial class Range_Of_Fuzzy_Values
```

```
    {
        public decimal Code { get; set; }
        public decimal Variable_Code { get; set; }
        public decimal Min { get; set; }
        public decimal Max { get; set; }
            public virtual Linguistic_Variables Linguistic_Variables { get; set; }
    }
}
```

Appendix B: paper

Hospital/Clinic Name: _____

Patient Number: _____

Age ☐ 1-10 years ☐ 10-20 years ☐ >20 years

Gender ☐ Male ☐ Female

Nationality

☐ Sudanese ☐ Other

Address _____

City _____ **State** _____

Symptoms:

***Please choose one option per symptoms**

A bad cough lasting longer than two weeks

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Chest pain

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Coughing up blood or sputum

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Weakness or feeling very tired

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Weight loss

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Lack of appetite

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Chills

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Fever and night sweats

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Extreme tiredness or lack of energy

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe

Final Diagnosis

Tuberculosis Result

☐ None ☐ Mild ☐ Moderate ☐ Severe ☐ Very Severe