Software Reliability Assessment using Neuro Fuzzy System

Thesis

SUBMITTED TO

BABASAHEB BHIMRAO AMBEDKAR UNIVERSITY,

(A CENTRAL UNIVERSITY)

LUCKNOW

BABASAHEB
BHIMRAO
AMBEDKAR
UNIVERSITY

LUCKNOW

प्रज्ञा शील करूणा

ESTABLISHED 1996

FOR THE DEGREE OF

Doctor of Philosophy

IN

INFORMATION TECHNOLOGY

BY

BONTHU KOTAIAH

DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL FOR INFORMATION SCIENCE AND TECHNOLOGY

BABASAHEB BHIMRAO AMBEDKAR UNIVERSITY

(A CENTRAL UNIVERSITY)

VIDYA VIHAR, RAEBARELI ROAD, LUCKNOW-226 025 (U.P.), INDIA

2015

DEDICATION

This thesis is dedicated to my Grand Mother Mrs. Bonthu Jayamani, for her inspiration, support at my childhood. Without her sacrifice, none of this would have been happened. Thanks for everything she provided me and thanks to God.

DECLARATION

I, Mr. Bonthu Kotaiah, student of Ph.D. in Information Technology hereby declare that the thesis titled "Software Reliability Assessment using Neuro Fuzzy System" which is submitted by me to the School for Information Science and Technology (SIST), Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow in fulfillment of the requirement for the award of the degree of Doctor of Philosophy has not previously formed the basis for the award of any degree, Diploma Associateship, Fellowship or other similar title or recognition. It is further declared that to the best of my knowledge and belief it has not been submitted earlier for the award of any degree.

Dated:

(Bonthu Kotaiah)
Research Scholar
Department of Information Technology
Babasaheb Bhimrao Ambedkar University
(A Central University)
Lucknow, India.

CERTIFICATE

It is certified that the thesis entitled "Software Reliability Assessment using Neuro Fuzzy System" being submitted by Mr. Bonthu Kotaiah is a record of bonafide work carried out by him. He worked under our guidance and supervision. He has fulfilled the requirements for the submission of thesis, which to my knowledge has reached the requisite standard. In view, the contents of this thesis and interpretations have subsequently added to the existing knowledge on the subject.

This thesis presented by him, to the best of our knowledge and belief, did not form the basis for the award of any degree earlier.

Dated:

Prof. R.A. Khan
(Supervisor)
Department of Information Technology
Babasaheb Bhimrao Ambedkar University
(A Central University)
Lucknow, India

Dr. Dhirendra Pandey
(Co-Supervisor)
Department of Information Technology
Babasaheb Bhimrao Ambedkar University
(A Central University)
Lucknow, India

ACKNOWLEDGEMENTS

All praise be to almighty God, our Lord, cherisher and sustainer.

First, I would like to thank almighty God who helped me at different critical stages during this work and shown me the right way when I was in trouble.

The completion of this thesis work gives me a much needed opportunity to express my thanks to all who have helped along the way.

Words do not suffice to express adequately my profound indebtedness and gratitude to my teacher and guide, my god father Prof. R.A. Khan, his able supervisions enabled me to undertake and complete this work. His knowledge, patience and ease of manner are paralleled.

I express my heartfelt gratitude and utmost respect and thanks to Dr.Dhirendra Pandey, Co-Supervisor for his invaluable guidance and constant support throughout my research work.

I express my sincere gratitude to my entire department teachers Mr.Pavan Kumar Chaurasia, Mrs. Rajshree for their encouragement, and consistent support throughout our research work.

Thanks are due to Dr.Suhel Ahmed Khan, Mr. Sandeep Singh, Dr. Amitabha , Dr.Shalini Chandra, Dr. Alka Agrawal my fellow research scholars for their help and support. Especially, I would like to thank Dr. Alka Agarwal from the bottom of my heart who support me for the successful completion of the report. I express my thanks to the technical and administrative staff of the department of Information Technology for their genuine help and cooperation, especially to Mr. Amar.

I express my heartful gratitude to Prof. Mohammed Miyan, Ex-Vice-Chancellor, Maulana Azad National Urdu University, Hyderabad Who has given me permission to continue my Ph.D. work in BBAU, Lucknow after my appointment as Assistant Professor and to Prof. Abdul Wahid, Dean, School of CS & IT, MANUU, Hyderabad who encourages, supports me a lot for the successful completion of this work.

Finally I am thankful to my family members for always being with me with their blessings and support.

Bonthu Kotaiah

ABSTRACT

Software reliability is defined as the probability of software to deliver correct service over a period of time under a specified environment. This is becoming more and more important in various software organizations to discover the faults that occur commonly during development process. As the demand of the software application programs increases the quality becomes higher and higher and the reliability of these software becomes more essential. Hence Software reliability is mentioned to be as the one of the important factor during development. Also it is a challenge for software companies to deliver good quality and error free software in right time. The impact of the failures produces severe consequences such as environmental impact, inconvenience, economical losses, loss of human life etc. Needless to say, the reliability of computer systems has become a major concern for our society.

Many analytical models are being proposed over the years for assessing the reliability of a software system and for modeling the growth trends of software reliability with different capabilities of prediction at different testing phases. But there is a need for developing such a single model which can be applicable for a relatively better prediction in all conditions and situations. Software quality and reliability assessment is very important factor for every software development process however having a product life cycle with complete product requirement documents, external and internal specifications, design reviews, code inspections and many; each product development group has a development organization and a Quality Assurance (QA) organization

In order to achieve desired reliability, different techniques for measurement have been developed and their main purpose is to test the software and measure the reliability according to a predefined criteria. However, the Neuro Fuzzy systems for the assessment of software reliability based on hybrid factors is not available. For the purpose, a Neuro Fuzzy based software reliability (SR)

model is presented to estimate and assess the reliability based on the combination of normalized MTBF and availability. Multiple datasets containing software failures are applied to the proposed model. These datasets are obtained from several software projects. It is observed that the results obtained indicate a significant improvement in performance by using neural fuzzy model over conventional statistical models (Fuzzy model) based on non homogeneous Poisson process.

Firstly, the need and necessity of reliability assessment, problem findings, and motivation for the research and its significance in research has been discussed. Secondly, the use of neural networks, fuzzy models, Neuro Fuzzy models and the earlier approaches for the effective assessment of Software Reliability has been explored. Thirdly, a model for Software Reliability Assessment based on Neuro Fuzzy systems is developed. Mathematical analysis of the proposed Neuro Fuzzy model is performed to arrive at the accuracy level of assessed software reliability. Finally, a validation has been done theoretically and statistically, in order to find out the level of significance of the proposed approach.

TABLE OF CONTENTS

ABBREVIATIONS

SDLC : Software Development Life Cycle

MTBF : Mean Time Between Failures

MTTR : Mean Time to Repair

MTTF : Mean Time to Failure

SR : Software Reliability

BPN : Back Propagation Neural Networks

FIS : Fuzzy Inference System

ANFIS : Adaptive Neuro Fuzzy Inference System

QA : Quality Assurance

ISO : International Standards Organization

IEEE : Institute of Electrical, Electronic Engineers

NRMSE : Normalized Root Mean Squared Error

RMSE : Root Mean Squared Error

MSE : Mean Squared Error

SRE : Software Reliability Engineering

AIAA : American Institute of Aeronautics and Astronautics

SRGM : Software Reliability Growth Model

FL : Fuzzy Logic

NFS : Neuro Fuzzy System

POFOD : Probability of Failure on Demand

EMR : Electronic Medical Records

ROCOF : Rate of Fault occurrence

CI **:** Computational Intelligence

AE **:** Average Error

GP : Genetic Programming

FIS : Fuzzy Inference System

GE : Glace EMR

NHPP : Non-Homogeneous Poisson Process

BPNN : Back Propagation Neural Network

WNN : Wavelet Neural Networks

GRNN : Generalized Regression Neural Network

MLR : Multiple Linear Regressions

TANN : Threshold Accepting Trained Neural Network

NFS : Neuro Fuzzy System

ANN : Artificial Neural Network

MF : Membership Function

T.F.N. : Triangular Fuzzy Number

GMDH : Group Method of Data Handling

RBF : Radial Basis Function

DENFIS : Dynamic Evolving Neuro Fuzzy Inference System

LIST OF TABLES

LIST OF FIGURES

1.1 Background

Dependency on computer aided systems is rapidly increasing day by day and the software systems operating on it. However this quality of service by the system is degraded by some software failures or faults to meet the required level of performance and this make many of the people to strike off these softwares. When the software functions are critical and the consequences of the problems are significant enough, the engineers has come forward to give the solutions. It depends upon the satisfaction on the application under certain situation. There are many definitions regarding the software quality.

Cross and Hoyer, R. W. and Hoyer [1] [2] explains his perspective on quality as a conformance to requirements. Khan R.A. made an attempt in his book[93] that has been made to describe various pertinent aspects of software quality from different points of view. Walter Edwards [3] states that quality must be defined in terms of customer satisfaction. Armand villain explains his perspective as "Quality is a measure determination, not an engineer's determination not a marketing determination nor a general determination. Feigenbaum A. V [4] proposed some techniques about how to control the quality. Many authors and people have given their own definition and one has to do with the quality as an objective. Finally the researchers can conclude that a quality software model which depends on the focused software is needed to be successfully applied for different systems. This model attempt to match product properties with the software quality attributes. There are three basic elements in this model such as product properties, quality attributes and linking product properties with quality attributes. Product properties are correctness, internal, contextual and descriptive. Functionality and reliability are the attributes which would contribute to the correctness product property.

The attributes of the internal product property are maintainability, efficiency and reliability. Maintainability, re-usability, portability and reliability are the attributes of contextual product property. The attributes which would contribute to descriptive product property are maintainability, re-usability,

portability and usability. Hence if a company is to develop high quality software, it is important to employ some efforts on Software Reliability and usability. This thesis focuses only on Software Reliability based models.

1.2 Software Quality

Quality is neither visible nor tangible and it is immeasurable. This is a physical feeling of the user which causes happiness or comfort with respect to a particular aspect. For instance if a user experience a trouble free use of the software ,then the quality aspect usability of that software is privileged. Software quality is very important factor for every software development process however having a product life cycle with complete product requirement documents, external and internal specifications, design reviews, code inspections and many; Each product development group has a development organization and a quality assurance (QA) organization.

The development organization is responsible for developing the software by performing unit tests while the QA is responsible for developing test software and performing the integration with system tests. When the development organization is satisfied with the functionality and quality of the product it delivers software to the QA. IEEE defines the software quality as the degree to which the software possesses a desired combination of attribute [5].ISO defines the so the software quality as: "*the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs*" [6]. Software quality is described in the means of models which are called software quality models and these have their own quality attributes [7], ISO 9126 defines software quality with six software quality attributes as functionality, reliability, usability, effectiveness, maintainability and portability [6].

The following abstractions may be listed as the essential features for designing the quality metrics: Compliance: The ability to cover all aspects of quality factors and the design characteristics.

- *Orthogonality*: The ability to represent different aspects of the system under measurement.
- *Formality*: The ability to get the same value for the same systems for different people at different times through precise, objective and unambiguous specification.
- *Minimality*: The ability to be used with the minimum number of metrics. Implement ability/Usability: The implementation technology independent ability.
- *Accuracy*: A quantitative measure of the magnitude of error, preferably expressed as a function of relative error.
- *Validity*: Validity refers to the degree to which a study accurately reflects or assesses the specific concept that the researcher is attempting to measure.
- *Reliability*: The probability of failure free software operation for a specified period of time in a specified environment. Interpretability: The ease with which the user may understand and properly use and analyze the metrics results [94].

The applicability of user-oriented reliability is not limited to software alone. Hardware reliability can also be estimated by the number of "users" being affected by a failure. [101][143]

1.3 Software Reliability

Software Reliability Engineering (SRE) is the discipline that helps the organizations to improve the quality of their products and processes. The American Institute of Aeronautics and Astronautics (AIAA) defines SRE as "the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems"[8]. Among the attributes of software quality, reliability is generally accepted as one of the major factor in software quality since it quantifies the failures.

There are many reasons why the organizations have to encourage this discipline and promote the usage of software reliable models. Finally the researchers can conclude that a quality software model which depends on the focused software is needed to be successfully applied for different systems. This model attempt to match product properties with the software quality attributes. There are three basic elements as such product properties, quality attributes and linking product properties with quality attributes in this model. Product properties are correctness, internal, contextual and descriptive. The mathematical expressions that specify the failure of software process is said to be Software Reliability estimation models or growth models (SRGMs) [150].

Organizations attain many advantages through SRGM using these developers and customers will have the continuity and determination what they tend to have. When the software system development is done through the agreement between vendor and customer, the reliability objective of the software should be either a pre agreed one of software quality metrics or it should be as a part of standard practice of the organization. By employing such reliability measures the validation and the quality of the product can be improved. Some of reliability issues during the requirement formulation focus on reducing the erroneous requirements in consideration, accounting of the risk of failure occurrences of each requirement, and the change management issues of future changes of the requirements. Designing and development phase is the most crucial and important phase and needed to be more reliable. According to [9] software companies recognize the need for systematic approaches using Software Reliability engineering techniques. There are three kinds of identifiers for Software Reliability.

1. Probability of failure free operation over a specified time interval.
2. Mean Time To Failure (MTTF) the predicted elapsed time between inherent failures of a system during operation.
3. Expected number of failures per unit time interval termed failure intensity.

In order to achieve effective reliability different techniques for measurement have been developed and their main purpose is to test the software and measure the reliability according to a predefined criteria. This process is clearly explained in the Figure 1.1.

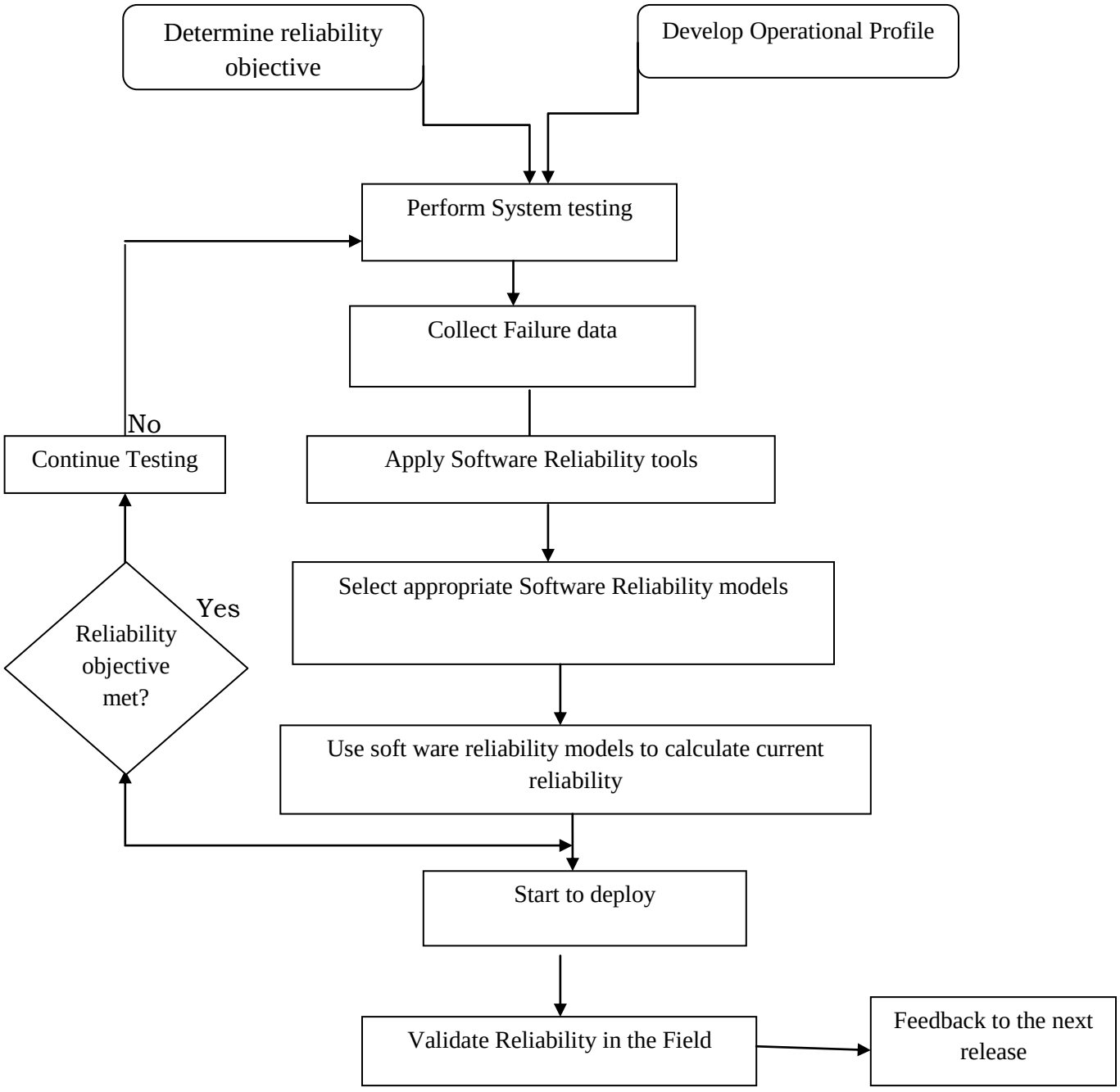

Figure 1.1: Software Reliability Engineering Process Overview [9].

In addition to measurement there are software growth models which define how the software should be developed while sustaining reliability such models are discussed in the section 1.4. For a software application, the objective of the optimization will depend on the phase of the software life cycle. During the design phase, reliability constrained cost minimization and cost constrained reliability maximization could be the two objectives. [105]**.** There is still a question mark on the practicing of Software Reliability in commercial development organizations. The reliability tasks consume a lot of time but the quality of the application will be very high this makes the user very comfortable and the software may reaches their requirements. Practicing this at industry level requires a huge economy for which the organizations are not willing. So there is an urgent need of more practical solutions for software quality guarantee for instance Software Reliability estimation at industry level.

1.4 Software Reliability Assessment

Software Reliability is expressed in execution time [102] [103]. There are three main factors on which the Software Relliability can be judged.

- *Fault:* The result that causes from the mistakes in the software is called as a fault (faulty instruction(s) or data).
- *Error:* When invocating a faulty instruction or a data by an appropriate input pattern, the fault produces an error
- *Failure:* If the erroneous data affect the delivered service (in value and/or in the timing of their delivery), failure occurs.

Wasserman in [10] gives a more accurate definition of reliability including execution time, satisfied conditions, and customer satisfaction before/during/ after the release of the software. Schneidewind, N. F., 2001[11] has given a model for the correction of faults in software during different phases of SDLC. What happens when a product cannot perform its intended function is called a failure. The following may some of the reasons of failures.

1. *Design Deficiencies*

- ❖ Omitting an important design feature
- ❖ Deficiencies in product design which lead to early failures
- ❖ The design of the process can also have deficiencies resulting in a defective product.

2. *Quality Control*

- ❖ Due to quality control problems, inefficient or unsuccessful products which lead to performance problems while being used
- ❖ Possible damage to products while handling or distribution

3. *Possible misuse of the product by the end user or during service.*

The possible failure outcomes define the reliability levels of the product, so for the different projects there might be different reasons and considerations can be applied.

Mathematically, the reliability function R(t) is the probability of a system performing its function without failure .

$$R(t) = P(T > t),\ t \geq 0$$

The failure probability can be defined as

$$F(t) = 1 - R(t) = P(T \leq t)$$

If the random variable T has the density function f(x) then R(t) can be calculated as

$$R(t) = \int_t^{\infty} f(x)dx$$

Then the expected time to next failure can be defined, in other words Mean Time To Failure (MTTF) is

$$MTTF = \int_0^{\infty} tf(t)ft = \int_0^{\infty} R(t)dt$$

In almost every reliability model these equations are used for assessment.

1.5 Neural Networks and Fuzzy Logic

a. Neural Networks

Artificial Neural Networks are generally known as “Neural Networks” act in a way similar to the human brain. Non linearity and complexity of the brain is very high and behaves like a parallel computer. It has the ability for organizing its structural constituents known as neurons; hence it performs certain computation very quickly than the fastest computer present on earth. The brain structure is very intense and it builds up its own rules through experiences. Experiences are built up over time with the development of the human brain through many stages. A developing neuron is as similar as a plastic brain. To adapt with the surrounding environment the developing nervous system has the property of plasticity. Plasticity appears to be essential to the functioning of neurons as information processing units in the human brain. Similarly this same thing happens with Neural Networks made up of artificial neurons. A neural network is a machine that is designed to model the way in which the brain performs a particular task [148]. To achieve good performance, Neural Networks should have a massive interconnection of simple computing cells referred to as neurons” or “processing units”. Neural Networks perform essential computations through a process of learning. The simple architecture of neural network was used in the model [73] [86] implemented for the prediction of software errors to maximize the software reliability.

Thus a neural network consists of simple processing units and big parallel distributed processors. The ability of storing experiential data and making it available for use comes naturally to it. The applicability of the connectionist approach is explored using various network models, training regimes, and data representation methods. [149]. Artificial neural network do not approach the complexity of the brain. It is similar to brain in two ways: 1.A learning process is used to acquire knowledge from its surrounding by the network. 2. The acquired knowledge is stored by the interneuron connection strengths known as synaptic weights. The procedure used to perform the process of learning is

called learning algorithm. Function of learning algorithm is to modify the synaptic weights of the networks in an orderly manner in order to attain a desired design objective. The general structure of neural network is shown in Figure 1.2.

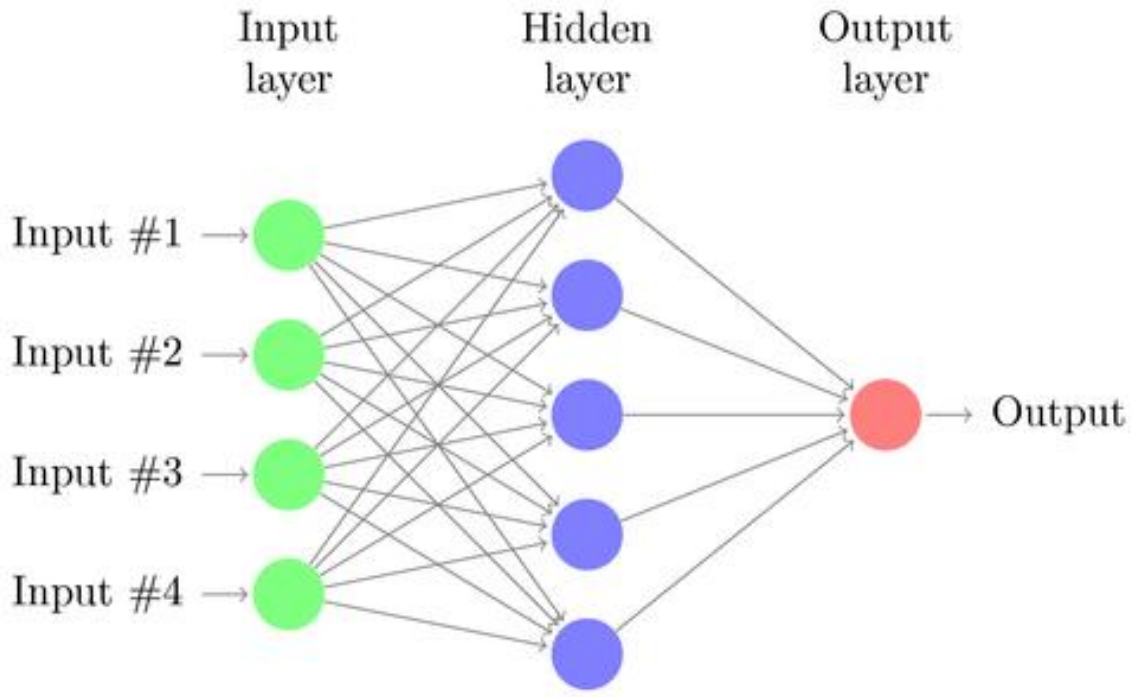

Figure 1.2: Generalized Block Diagram of a Neural Network

b. Fuzzy Logic

Fuzzy Logic has two different meanings. These are as follows:

1. Fuzzy Logic is a logical system, which is an extension of multivalued logic.
2. Fuzzy Logic (FL) is almost synonymous with the theory of fuzzy sets, a theory which relates to classes of objects with unsharp boundaries by using membership.

Fuzzy Logic Toolbox software with MATLAB technical computing software can be used as a tool for solving problems with Fuzzy Logic. It is a convenient way to map an input space to an output space. Mapping input to output is the starting point for everything. Fuzzy Logic is a logical system that generalizes

classical two valued logic for reasoning under uncertainty. It refers to all of the theories and technologies that employ fuzzy sets which are classes with unsharp boundaries. Fuzzy Logic has been viewed as a theory for dealing with uncertainty about complex systems. The distinguishing mark of Fuzzy Logic in rule-based systems is its ability to deal with situations in which making a sharp distinction between the boundaries of application in the use of rules or constraints is very difficult. The various applications of Fuzzy Logic include consumer products, automotive and power generation, industrial process control, robotics and manufacturing and Software Reliability.

Fuzzy Logic offers an alternative to the probability paradigm, possibility that is much more appropriate to Software Reliability when there is uncertainty in the data. Possibility mathematics allows for quantitative reliability calculations that preserve the uncertainty present in the original data, avoids making unwarranted assumptions and makes the consequences of the required assumptions clear throughout the analysis. Fuzzy Logic also improves reliability analysis through the concept of utility.

Standard reliability models usually assume a binary representation of failure, a system is working or it is failed. A more flexible and realistic model allows for easy representation of partial system failures. Thus, there is considerable motivation to adapt the traditional probability based reliability methods to the Fuzzy Logic context. A Fuzzy Inference System (FIS) can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value [132].

The Advantages of Fuzzy Logic are:

- Fuzzy Logic is conceptually easy to understand.
- The mathematical concepts behind fuzzy reasoning are very simple.
- Fuzzy Logic is flexible.
- With any given system, it is easy to layer on more functionality without starting again from scratch.
- Fuzzy Logic is tolerant of imprecise data.

- Everything is imprecise if it is looked closely enough, but more than that, most things are imprecise even on careful inspection.
- A fuzzy system to match any set of input-output data can be created. This process is made easy by adaptive techniques like Adaptive Neuro Fuzzy Inference Systems (ANFIS), which are available in Fuzzy Logic Toolbox software.
- Fuzzy Logic can be built on top of the experience of experts.
- In direct contrast to Neural Networks, which take training data and generate opaque, impenetrable models, Fuzzy Logic lets you rely on the experience of people who already understand the system.
- Fuzzy Logic can be blended with conventional control techniques.
- In many cases fuzzy systems augment them and simplify their implementation.

The Functionality of Fuzzy Logic Toolbox Software in MATLAB:

You can create and edit Fuzzy Inference Systems with Fuzzy Logic Toolbox software. You can create these systems using graphical tools or command-line functions. The interaction of Fuzzy Logic toolbox in MATLAB environment [151] is shown in the Figure 1.3.

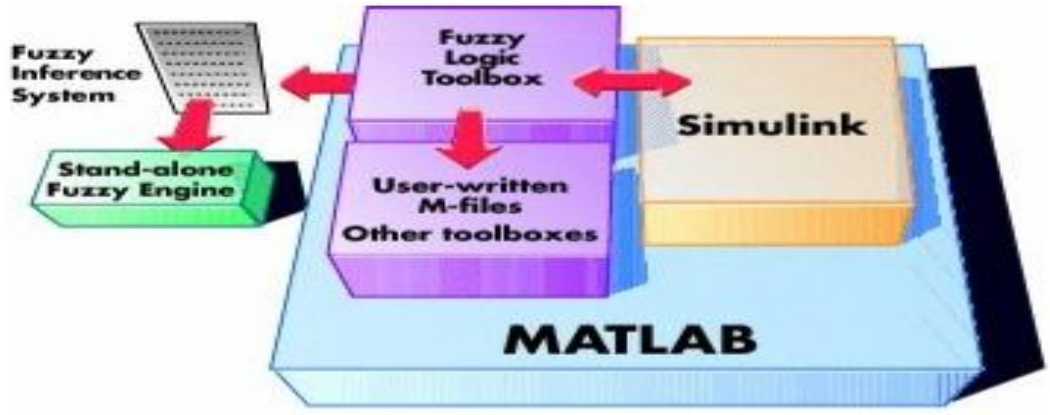

Figure 1.3: Interaction of Fuzzy Logic Toolbox with different other Components in the MATLAB environment.

The various Software Reliability models contain parameters which are uncertain in nature. To incorporate the uncertainty these parameters are modeled as fuzzy numbers. Failure and repair rates are taken as fuzzy variables which are defined in terms of fuzzy numbers with suitable presumption level and membership grades. Triangular or trapezoidal possibility distributions can be used in fuzzification process. The resulting fuzzified model offers more flexibility and accommodates uncertainties in failure and repair rates and other reliability measures. Fuzzified models offer several advantages to the software developer as they are more realistic and practical in nature than other models with rigid assumptions.

1.6 Neuro Fuzzy Models

The idea of a Neuro Fuzzy system is to find the parameters of a fuzzy system by means of learning methods obtained from Neural Networks. Here, the basic properties of Neuro Fuzzy systems are discussed. The learning techniques that can be used to create fuzzy systems for data; a common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural-network-like architecture. Then a learning algorithm – such as back propagation – is used to train the system. There are some problems, however. Neural network learning algorithms are usually based on gradient descent methods. They cannot be applied directly to a fuzzy system, because the functions used in the inference process are usually not differentiable. There are two solutions to this problem:

a) Replace the functions used in the fuzzy system (like min and max) by differentiable functions, or
b) Do not use a gradient-based neural learning algorithm but a better-suited procedure.

There are several different approaches which have much in common, but differ in implementation aspects. To stress the common features of all these

approaches, and to give the term Neuro Fuzzy system a suitable meaning, it is applied only to systems which possess the following properties:

(i) A Neuro Fuzzy system is a fuzzy system that is trained by a learning algorithm (usually) derived from neural network theory. The (heuristic) learning procedure operates on local information, and causes only local modifications in the underlying fuzzy system. The learning process is not knowledge-based, but data-driven.

(ii) A Neuro Fuzzy system can always (i.e. before, during and after learning) be interpreted as a system of fuzzy rules. It is possible both to create the system out of training data from scratch, and to initialize it from prior knowledge in the form of fuzzy rules.

(iii) The learning procedure of a Neuro Fuzzy system takes the semantical properties of the underlying fuzzy system into account. This results in constraints on the possible modifications of the system's parameters.

(iv) A Neuro Fuzzy system approximates an n-dimensional (unknown) function that is partially given by the training data. The fuzzy rules encoded within the system represent vague samples, and represent vague prototypes of the training data. A Neuro Fuzzy system should not be seen as a kind of (fuzzy) expert system, and it has nothing to do with Fuzzy Logic in the narrow sense.

(v) A Neuro Fuzzy system can be represented by a special three-layer feed forward neural network. This view of a fuzzy system illustrates the data flow within the system and its parallel nature. However, this neural network view is not a prerequisite for applying a learning procedure, it is merely a convenience.

The Neuro Fuzzy technique, then, is used to derive a fuzzy system from data, or to enhance it by learning from examples. The exact implementation of the Neuro Fuzzy model does not matter. It is possible to use a neural network to learn certain parameters of a fuzzy system, like using a self-organizing feature map to find fuzzy rules, or to view a fuzzy system as a special neural network and to apply a learning algorithm directly.

The Structure of Adaptive Neuro Fuzzy Inference System is shown in Figure 1.4.

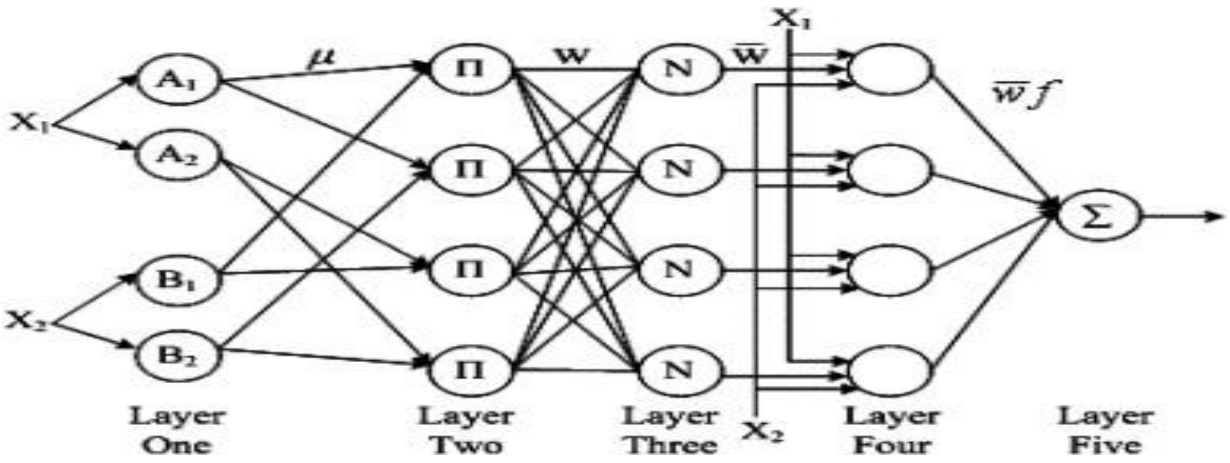

Figure 1.4: Structure of Adaptive Neuro Fuzzy Inference System.

The Neuro Fuzzy which enhances Fuzzy Logic gives the set values for the neural network controllers based on the process control expertise put in the Fuzzy Logic rules. The Neuro Fuzzy Logic technologies enable the efficient and transparent implementation of the software systems. Neuro Fuzzy approach allows for the automated generation and optimization of Fuzzy Logic based software systems based on training data. Many ways of integrating neural nets and Fuzzy Logic have been proposed in the scientific literature. Only a very few have already been successfully applied in software applications. The Neuro Fuzzy systems combine the design and optimization process of fuzzy controllers with learning capabilities derived from Neural Networks.

The Neuro Fuzzy technology is blessed due to following characterization:

1. Fast computation using fuzzy number operations.
2. Capable of handling any kind of information (numeric, linguistic, logical, etc.).
3. No need of prior knowledge of relationships of data can manage imprecise, partial, vague or imperfect information. Emirates mimic human decision making process.

4. Self-learning, Self-organizing and Self-tuning capabilities.

The rule structure is made up of Neuro Fuzzy commands of “IF”, “AND, “Continues” and “THEN” carefully connected together to arrive at the specified desired parameter (output). All the commands are in such a way that a computer program can be written on them for computerization. The Neuro Fuzzy commands have significant meanings that account for the effectiveness of the model. The command “IF” means if the outcome of the relationship between input and output parameters is this, the command “AND” means “and this outcome”, the command “continues” means “the outcome continues over time, finally the command “THEN” means if all the previous commands hold “then the system should prompt the specified desired output”. The Neuro Fuzzy models are gradually becoming established not only in the academia but also in the software applications. The tools for building Neuro Fuzzy models are based on combinations of algorithms from the fields of Neural Networks, pattern recognition and regression analysis.

1.7 Problem Definition

Developing high quality software products is a goal in many development projects. However, quality is a highly subjective term and depends on the goals and perceptions of stakeholders. To better reason on software product quality, software quality models have been suggested to describe and measure software quality [6] [7]. Software quality attributes (also called quality characteristics) are characteristics which provide the basis for evaluating quality of software systems (adapted from [13] pg. no 81)). Examples for software quality attributes of software systems are reliability, usability, and performance.

Software quality attributes are one of the influential factors to take into account when designing software architecture [14]. Relevant software quality attributes when designing software architecture are reliability, modifiability, performance, security, testability, and usability [14]. Jain, R.[15] demonstrated about how to manage the systems during different phases in his work “the Art

of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling". Smith, C. U. and Williams, L. G., proposed three new performance antipatterns that often occur in software systems, which affects its performance [16]. Immonen, A. and Niemela, E., discovered methods that are suitable for the reliability and availability prediction of today's complex systems [17]. For some software quality attributes, quantitative quality metrics are available to assess the level of quality achieved by a software system. Reliability is the capability of a system to provide functionality as expected for a specified period of time in the intended execution context. It is for example measured as the probability of failure on demand (POFOD). The notion of availability is closely related and focuses on the fraction of time that the system is available to serve requests [14]. For example, one may require that a system is available 360 days a year.

Depending on the goals of the system to be developed, additional software quality attributes may be relevant, such as portability or interoperability. Thereby, single software quality attributes cannot be considered in isolation, because improving a system with respect to one software quality attribute has an effect on other software quality attribute. Often, software quality attributes conflict: For example, security and reliability often negatively influence each other: While a system is secure if it offers few places that keep sensitive data, such an organization may lead to single points of failure and decreased reliability. Furthermore, almost all software quality attributes conflict with performance.

Software industries attempt to release software to grab the market as soon as it is ready. The software market is very competitive in this dynamic world. Software industries attempt to release software to grab the market as soon as it is ready. Now it is a challenge for software developers to rapidly design, implement, test, and maintain complex hardware or software systems as per the demands of the users. Also it is a challenge for software companies to deliver good quality and error free software in right time. The impact of the

failures produces severe consequences such as environmental impact, inconvenience, economical losses, loss of human life etc. Needless to say, the reliability of computer systems has become a major concern for the society. Software Reliability is an important facet of software quality characteristic. Many researchers have used Neural Networks to predict Software Reliability. Different Neural Networks with different learning methods also have been modeled. It is also observed that connectionist models perform better than the previous parametric models. Prediction of Software Reliability using computational intelligence (CI) can be very accurate and significant compared to traditional statistical methods. Computational Intelligence (CI) can offer promising approaches to Software Reliability prediction and modeling.

With the rapid development of technology people are tend to depend mainly on the computer aided systems in which the major role is plays by the Operating Softwares(OS) and the application softwares. There is dramatic change in the usage of softwares in the recent years and a wide variety of application softwares are developed to meet the requirements for every area of human life ranging from scientific researchers to kids at kindergarten. In comparison with advancements of hardware components, software achieves less progress caring a larger burden of the total system. The potential of integrating the independently developed software into such hardware system has enabled software designers to develop high complex systems. However, in comparison to the hardware technology, the software technology has not succeeded in keeping measures such as quality. Software reports major source of outages in many systems.

There are many consequences where the software failures have a huge impact. Below are few of them: In the NASA Voyager project, the Uranus encounter is in jeopardy because of late software deliveries and reduced capability in the Deep Space Network. Several Space Shuttle missions have been delayed due to hardware/software interaction problems. Software glitches in an automated baggage- handling system forced Denver International Airport

to sit empty more than a year after airplanes are to fill its gates and runways. The Hong Kong Airport experienced a similar problem [6].

Secondly, the massive Therac-25 radiation therapy machine have been enjoyed a perfect safety record until software errors in its sophisticated control systems malfunctioned and claimed several patients' lives in 1985 and 1986. On October 26, 1992, the Computer Aided Dispatch system of the London Ambulance Service broke down right after its installation, paralyzing the capability of the world's largest ambulance service to handle 5000 daily requests in carrying patients in emergency situations. In the recent aviation industry, although the real causes for several airliner crashes in the past few years remained mysteries, experts pointed out that software control could be the chief suspect in some of these incidences due to its inappropriate response to the pilots' desperate inquires during an abnormal flight condition [7].

From the two scenarios it is clear that the failures of software system carry huge disaster impacting a lot for human lives. Software Reliability is the science of the studying about the failures of these softwares and quantifies them. This thesis discuss about the stochastic Software Reliability modeling with Neuro Fuzzy approach that can be utilized at the time of development. Keeping in view of the above discussions, a Neuro Fuzzy based approach is designed in this thesis based on the evaluated metrics Normalized MTBF and Availability to arrive at the best assessment of the Software Reliability for a given dataset of Software observations, the Problem is defined under the title:

"SOFTWARE RELIABILITY ASSESSMENT USING NEURO FUZZY SYSTEM"

1.8 Objectives

The following objectives are set in the research study:

- To review and examine the literature on Software Reliability in relation with Availability, MTBF and reliability estimation models and the available metrics.
- To identify the factors that affect the Software Reliability at the design phase of SDLC(Software Development Life Cycle)
- To find out, analyze the effect of errors, bugs and failures in the software under the development.
- To collect the datasets of software programs from running software with the appropriate runtime errors that is useful for the assessment.
- To formulate a theoretical analysis for the evaluation of the metrics used for assessment and development of model.
- To identify how availability and MTBF relates with the Software Reliability.
- To calculate the metrics with the given datasets both analytically and programmatically.
- To develop a model for the assessment of Software Reliability based on Neuro Fuzzy Systems approach and to implement it.
- To train the neural network with some collected Software Reliability parameters (at design phase of SDLC) mapped to numerical data and are loaded into neural network at input layer.
- To assess and evaluate the performance of the trained network for Software Reliability at the design level with some numerically approximated values by using fuzzy membership function (sigmoid).
- To compare the approximated Software Reliability against the expected reliability approximation.
- To adjust at the input layer of Neuro Fuzzy model so as to minimize the difference between actual and expected values of reliability.

- To compare the performance of proposed model against conventional FIS (Fuzzy Inference System) models based on evaluation and validation metrics to prove that the proposed model is the promising one than the others.
- To validate the proposed model both theoretically and sttistically.

1.9 Significance of the Study

The Software Reliability Growth Models have been developing during the past thirty years. Based on different assumptions and philosophy of software phenomena various Software Reliability Growth Models are described. In general, these models are based on Non-Homogenous Poison Process (NHPP) and an analyst could choose the "best" model for his use. The optimal stopping formulation suggested may be helpful to determine the most favorable time to discontinue testing. The models developed in the present thesis exhibit the release options that can be chosen at any time during the software testing phase, in order to avoid wasting effort in the development process. To ensure ongoing software quality, the new release options of the software are suggested in different scenario. These release policies suggest improved and error-free versions and the process of providing new versions throughout the software life-cycle. Our research will provide an insight to the software developers and testers to agree with high quality software. Based on framework it can be chalked out during the development of the software life cycle. Software Reliability measures suggested may be embedded in an organizations and software engineering process to produce more improved software as far as quality is concerned under techno-economic and reliability constraints.

This study seems to have the following significance:

- Based on different assumptions and philosophy of software phenomena various Software Reliability Growth Models have been described. In general, these models are based on Non-Homogenous Poison Process (NHPP) and an analyst could choose the "best" model for his use. The

optimal stopping formulation suggested may be helpful to determine the most favorable time to discontinue testing.

- The models developed in the present thesis exhibit the release options that can be chosen at any time during the software testing phase, in order to avoid wasting effort in the development process.
- For ensuring ongoing software quality, the new release options of the software are suggested in different scenario. These release policies suggest improved and error-free versions and the process of providing new versions throughout the software life-cycle.
- The research will provide an insight to the software developers and testers to agree with high quality software.
- Software Reliability measures suggested may be embedded in an organizations and software engineering process to produce more improved software as far as quality is concerned under techno-economic and reliability constraints.
- It may help to understand the design aspect of the software system clearly, so that development of the system will be made easy by monitoring the reliability.
- It may help to develop different alternative designs of the software under development.
- It may help one to choose the best reliable design among different alternative designs.
- It may help to determine how to maximize availability to maximize reliability.
- It may help to provide best estimation of Software Reliability, so that developers can get clear estimation of total software development cost.
- It facilitates the planning of new activities.
- The proposed model for reliability assessment based on Neuro Fuzzy Systems helps to come out with good availability, MTBF and Reliability estimation models with wider acceptability.

- It also helps future researches in the Neuro Fuzzy based systems to develop new approaches.
- The proposed approach may find a place among the measurement tools for assessing the Software Reliability based on Neuro Fuzzy Systems.

1.10 Organization of Thesis

This thesis on Neuro Fuzzy model based Software Reliability assessment is organized as:

Chapter 2 explains vividly about the literature review on Software Reliability, Software Reliability Engineering process, Neural Networks, Fuzzy Logic, Neuro Fuzzy models and the earlier approaches that are present on the same context for the assessment of Software Reliability, its flaw and overcoming. It also stress on the reliability model used in the research. It mainly focus on the need of reliability, importance of reliability assessment, with existing literature and the research gaps in the existing literature and what needs to be done.

Chapter 3 mainly focuses on fuzzy modeling and Neuro Fuzzy modeling, relation between the findings and the approach used in the models. It also provides the mathematical analysis of the proposed approach of Neuro Fuzzy model for Software Reliability Assessment. It presents the model with five phases like Identification Phase, Quantification Phase, Measurement Phase, Verification and Validation Phase and Finalization Phase. The review and revision is also done at every phase of the proposed approach for the effective assessment of Software Reliability.

Chapter 4 shows the implementation of the model under different experimental datasets. Also, analyses the performance and evaluates the outcome of the conventional FIS based system of Software Reliability Assessment and the current approach. The implementation and validation is done using the MATLAB software environment.

Chapter 5 ends up with the conclusions, significance of the research, research findings and suggestions for future work and the limitations of the approach.

2.1 Background

A large number of mathematical models have been developed for reliability growth. Duane [50] in (1964) is first person to report the most commonly accepted pattern for reliability growth. Duane model is basically a graphical approach to perform analysis of reliability growth. After then since 1970's, the traditional way of predicting Software Reliability has been the use of Software Reliability Growth Models. Many research activities in Software Reliability engineering have been conducted and Software Reliability Growth Models in different frameworks have been proposed to assess the reliability of the software. During the last three decades, many Software Reliability Growth Models (SRGMs) have developed for measuring the growth of reliability of the software.

The most widely used models are developed by Jelinski and Moranda's (1972) [51], Shooman Model [52], Musa [53] [54] examined logarithmic poison execution time model for Software Reliability measurement. Time-dependent error detection rate model for Software Reliability and other performance indices are considered by Goel and Okumoto [55]. The objective of Software Reliability testing is to determine probable problems with the software design and implementation as early as possible to assure that the system meets its reliability requirements. Several hundreds of statistical models used in Software Reliability testing have been developed over the years [56] [57].

2.2 Literature Survey

In 2014 Kirti Tyagi, Arun Sharma [152] suggested An adaptive Neuro Fuzzy model for estimating the reliability of component-based software systems. In their work, they proposed a model for estimating CBSS reliability, known as an adaptive Neuro Fuzzy Inference System (ANFIS), that is based on these two basic elements of soft computing, and the research study compare its performance with that of a plain FIS (Fuzzy Inference System) for different data sets. In 2013 Hu.et.al [109] estimate the Software Reliability through testing,

an extended adaptive testing strategy, namely Modified Adaptive Testing (MAT). In the same year, Pooja Rani *et.al.* proposed a risk predicting tool based on Neuro Fuzzy approach for software Risk Prediction. Firstly Fuzzy Inference System is created and then Neural Network based three different training algorithms: BR (Bayesian Regulation), BP (Back propagation) and LM (Levenberg-Marquardt) are used to train the neural network. In [2013] Lance Fiondella et.al presents an efficient methodology based on the multivariate Bernoulli (MVB) distribution to analyze the reliability of a software application considering COCOF.Unlike the earlier techniques, the proposed methodology introduces only a quadratic number of parameters [104].

In 2012 Singh et al [136] explain that the transfer function is a function used to transform the activation level of a neuron into an output signal. The behavior of the ANN depends on both the weights and the activation function that is specified for the neuron. Neurons are structured depending on the learning algorithm used. Learning consists of adjusting the weight and threshold values until certain criteria are encountered for example the training error falls below some user-defined level, the number of training cycles or epochs exceeds a certain value or all the validation tests have been performed.

In 2011, we propose an adaptive framework of incorporating path testing techniques into the reliability estimation for modular software systems. Three estimated methods based on common program structures are presented to calculate the path reliability. The derived path reliabilities then are used as an approximation of software reliability. When more testing paths are available as testing proceeds, our proposed methods can then recalibrate the estimated accuracy [108]. In 2011 Bragina, T. & Tabunshchyk, G. Analysed the project risk, that depends on basic software risk level of projects with iterative lifecycle and software development project model [141]. Fuzzy model for the project risk are proposed by the authors. In 2011, Marko Palviainen et.al defines an approach for reliability evaluation that assists software developers in producing predicted reliability values for not-yet-implemented components, to measure

reliability values for components and to evaluate how different component selections affect the reliability of a software system.[116].

In 2011 Okubo *et.al* [126] proposed a new security impact analysis method for software enhancement. The method consists of two techniques: an analysis method of horizontal impacts using an extended misuse case. In 2010 SVM was introduced as a pattern classifier and it seeks to determine a separating hyperplane. Separating hyperplanes can differ in how large a margin of separation. For linearly separable training data there exists many hyperplanes that might classify the data. [115]. In 2010 Ongsakorn *et.al* [121] utilizes a threat representation structure called a Cyber Threat Tree. This idea was motivated from the ideas of fault trees, which were originally devised by Bell laboratories. Cyber threat trees have important differences from the fault trees in that many threat events are not statistically independent and that, unlike the fault tree model, we do not model threats as faults. In the fault tree model, a fault either exists or does not; hence, it is based on a binary Boolean logic switching function. In 2010 Hu, Y., Zhang, X., Sun, X., Zhang, J., Du, J. & Zhao,J. has established a framework unifying risk analysis and planning to maximize Software Reliability[140].

In 2010 Y. Singh and P. Kumar explored the applicability of neural network models for better prediction of reliability in a realistic environment and present an assessment method of software reliability growth using connectionist model. They applied feed forward back propagation algorithm and discuss the related issues of network architecture, method of data representation and some unrealistic assumptions incorporated with software reliability models [30]. In 2009, Lo [35] designed a model for Software Reliability prediction using artificial Neural Networks. This approach examines several conventional Software Reliability Growth Models without assuming some unrealistic things. In the same year Goswami and Acharya [107] proposed an approach to CBSS reliability analysis which takes into consideration the system's component usage ratio, calculated through mathematical formulas.

Due to the flexibility of the component usage ratio, this approach may be used for real-time applications. This approach quantified overall software system reliability based on the individual component reliabilities which are combined together to form the system. Moreover, the components which have longer execution time contribute more towards overall system reliability. Operational profile of a component is used for the calculation of component usage ratio, which is a very different function from the traditional use of operational profiles in Software Reliability engineering. In 2009 Vandana Gandotra, et.al [123] presented threat modeling process in order to identify various threats to the system. Threat modeling process involves understanding an adversary's goal in attacking a system based on system's assets of interest In 2008 work estimates the reliability of composite Web service by executing its sub-processes according to its control structures in the early stage. How to compute the reliability of the composite Web service by the eight control strictures is discussed [110].

In 2007 a SRGM describing the failure-occurrence or fault detection phenomena in the software testing phase is developed by Inoue and Yamada in [64]. In the same year, Zheng [147] used the ensemble of Neural Networks to modeling SRGMs.RajKiran et al. [34] implemented the use of wavelet Neural Networks (WNN) to predict Software Reliability. In this paper, the authors employed two kinds of wavelets i.e. Morlet wavelet and Gaussian wavelet as transfer functions. They made a comparison on test data with multiple linear regression (MLR), multivariate adaptive regression splines (MARS), back-propagation trained neural network (BPNN) and threshold accepting trained neural network (TANN), pi-sigma network (PSN), general regression neural network (GRNN) and found that its performance is better than others.

In 2005 Gong et al. [131] presented an implementation of GA (genetic algorithm) based approach to Network Intrusion Detection using GA and showed software implementation. The approach derived a set of classification rules and utilizes a support-confidence framework to judge fitness function. Ngai, E. W. T., & Wat, F. K. T. described the development of a fuzzy decision

support system (FDSS) for the assessment of risk in e-commerce (EC) development. A Web-based prototype FDSS is designed and developed to assist EC project managers in identifying potential EC risk factors and the corresponding project risks. A risk analysis model for EC development using a fuzzy set approach is presented by them [133]. In 2003 Cai, K., Bai, C., Zhong, X., provided an overview for the state of the art of Component Based Systems reliability estimation [100]. In 2002 Nikola et.al [43] introduces a new type of fuzzy inference systems, denoted as dynamic evolving neural-fuzzy inference system (DENFIS), for adaptive online and offline learning, and their application for dynamic time series prediction. DENFIS evolve through incremental, hybrid (supervised/unsupervised), learning, and accommodate new input data, including new features, new classes, etc., through local element tuning. In 2002, M. M. T. Thwin and T. S. Quah, Eds., presented the application of neural network for predicting software development faults including object-oriented faults. They used Object-oriented metrics for quality estimation [31]. In 1999 Sitte [33] analyzed two methods for Software Reliability prediction: 1) Neural Networks and 2) parametric recalibration models. These approaches differentiate the Neural Networks and parametric recalibration models in the context of Software Reliability prediction and conclude that Neural Networks are much simpler and better predictors. In 1999 Tian [63] described the recent work in establishing predictive linkage between Software Reliability and other indices that software developers can measure and control early in the development life-cycle of the software.

Most of the Software Reliability Growth Models have been proposed to estimate some important performance measures such as Mean Time To Failure (MTTF), Mean Time To Repair (MTTR), Mean Time Between Failures (MTBF), number of remaining faults, failure intensity, etc.. Besides these, Software Reliability Growth Models are also used to determine the behavior of fault detection and removal/correction process. It is assumed that detected faults

are removed or corrected immediately but in real life situations there is a time gap between detection and correction processes.

In 1997, Yager in [59] implemented Fuzzy Logic controllers using a neural network framework. In the same year Saileshwar Krishnamurthy et al.[113] reported an experiment to evaluate a method, known as Component Based Reliability Estimation (CBRE), for the estimation of reliability of a software system using reliabilities. CBRE involves computing path reliability estimates based on the sequence of components executed for each test input. In 1996, Kuo and Yang in [61] used non-homogeneous poison process in Software Reliability with Bayesian approach. The industrial applicability of SRGMs can be found in the literature by several case studies. Modeling the failure process in a piece of software is a very challenging exercise.

In 1994, Singpurwalla and Wilson in [62] analyzed the statistical methods in software engineering with reliability and risk. In 1993, Hossain and Dahiya [60] estimated the parameters of a non- homogenous Poisson process model for Software Reliability. In 1992, Yamada [58] did Software Reliability analysis based on a non homogenous error detection rate model. In 1992, Debar H et.al [129] present a model of the behavior of a user on a computer system using a neural network coupled with an expert system and showed a implementation for the neural network component inside an intrusion detection system, with the links with available security expertise and N. Karunanithi and D. Whitley, presented an adaptive modeling approach based on connectionist networks and demonstrate how both feedforward and recurrent networks and various training regimes can be applied to predict software reliability[32].

In 1992 T. M. Khoshgoftaar, A. S. Pandya and H. More, introduced a new approach for static reliability modeling and compares its performance in the modeling of software reliability from software complexity in terms of the predictive quality and the quality of fit with more traditional regression modeling techniques[29]. In 1991 Karunanithi et al. [28] predicted Software Reliability using feed forward network and recurrent network. The authors compared the result with 14 different literature representative data sets and

suggested that neural network produced better predictive accuracy compared to analytical models at end-point predictions. Boehm, B. W. tried to find out the software equivalents of Beauvais Cathedral, the S.S. Titanic, and the "Galloping Gertie" Tacoma Narrows Bridge. The frequency of these disaster projects is a serious concern, which is a survey of 600 firms done by him [139]. In 1990 Shadmehr *et.al* [27] estimated model parameters of pharmacokinetics system using feed forward multilayered network and predicted the noise resides in the measured data sample. The authors compared the results with that of the optimal Bayesian estimator and found the performance is better than the maximum likelihood estimator. The ANN tools and feed forward network using back propagation algorithm are applied for reliability and software quality prediction [68–70]. The authors developed a connectionist model and took failure data set as input to produce reliability as output. These papers describe network architecture, method of data representation and some unrealistic assumptions associated with Software Reliability models. In 1989, Hartler [57] discussed reliability growth models for hardware and software systems based on non homogenous Poisson processes. In 1988, Werbos [26] proposed back-propagation learning as an alternative to regression technique to identify sources of forecast in uncertainty in a recent gas market model. Thus it can be concluded that neural network models are very useful for regression techniques of forecasting in uncertainty of any data.

2.3 Software Reliability Engineering

Software Reliability Engineering is the discipline that helps the organizations to improve the quality of their products and processes. The American Institute of Aeronautics and Astronautics (AIAA) defines SRE as "the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems" [18]. To overcome the failures like design deficiencies, quality control, possible misuse of the product by the end user or

during service, among the attributes of software quality, reliability is generally accepted as one of the major factor to decide about software quality since it quantifies the failures. There are many reasons why the organizations has to encourage this discipline and promote the usage of software reliable models

2.3.1 Advantages and Usages

Organizations attain many advantages through SRGM using these developers and customers will have the continuity and determination what they tend to have. When the software system development is done through the agreement between vendor and customer, the reliability objective of the software should be either a pre agreed one of software quality metrics or it should be as a part of standard practice of the organization. By employing such reliability measures the validation and the quality of the product can be improved. Some of reliability issues during the requirement formulation focus on reducing the erroneous requirements in consideration, accounting of the risk of failure occurrences of each requirement, and the change management issues of future changes of the requirements. Designing and development phase is the most crucial and important phase and needed to be more reliable. Critical operations must be included to improve the quality and availability and the release time can be determined using SR during testing. Certain operations and principles pertaining to maintenance must be allocated which will increase the productivity.

2.3.2 Software Reliability Activities

SR comprises of three activities [19]

- Error Prevention
- Fault detection and removal
- Measurements to improve reliability

The errors prevention techniques used in the development are designed by contract, bug tracking systems, monitoring systems, coding standards,

definitive programming and culture of development. When employing more techniques in a project, that project is closer to achieve a comprehensive and coherent error prevention program. Testing is done for fault detection and removal, Integration of the modules is examined during the integration testing subsequent to the module testing. By encompassing the software requirements, the system testing is carried out; this will verify the faults in the complete system invocation. Finally the acceptance testing is done to ensure the requirements collected from the customer. Several strategies and techniques are used in test selection, test design and stop testing to high level fault detection and removal.

The process of Software Engineering evolves with a unique issue of testability. It is an external software attribute that assesses the complexity and effort required for testing software. The insight provided by testability is valuable during design, coding, testing and quality assurance [99].

2.3.3 Software Engineering Measurement

The importance of measurement in software engineering is widely acknowledged, especially in helping management in decision-making activities, such as estimating; planning; scheduling; and tracking. In formal terms, measurement is the process by which numbers or symbols are assigned to attributes of entities (e.g., elapsed time in a software testing phase) in the real world in such a way as to describe them according to clearly defined rules. The numbers or symbols thus assigned are called metrics that signify the degree to which a certain entity possesses a given attribute. The scope of measurement in software engineering can include several activities [20] [21].

- Cost and effort estimation.
- Productivity measures and models.
- Data collection.
- Quality models and measures.
- Reliability models

- Performance evaluation and models.
- Structural and complexity metrics.
- Capability-maturity assessment.
- Management by metrics.
- Evaluation of methods and tools.

These activities are supported by a range of software metrics; a common categorization is based on the management function they address, i.e., project, process or product metrics.

- Project metrics — Used on a project level to monitor progress, e.g., number of faults found in integration testing.
- Process metrics — Used to identify the strengths and weaknesses of processes, and to evaluate processes after they have been implemented or changed, e.g., system test effort.
- Product metrics—used to measure and assess the artifacts produced during the software life cycle. Product metrics can further be differentiated into external product metrics and internal product metrics. External product metrics measure what commonly refer to as quality attributes (behavioral characteristics, e.g., usability, reliability, portability, efficiency). Internal product metrics measure the software attributes itself, e.g., lines of code.

The Software Development Life Cycle phases are explained in detain in the following Figure 2.1.

Figure 2.1: Block Diagram of Software Development Life Cycle (SDLC)

There are two different ways to predict application performance and reliability.

• **Composite method:** The architecture of the application can be combined with the failure behavior of the components and the interfaces into a composite model which can then be analyzed to predict the performance and reliability of an application.

• **Hierarchical method:** The other possibility is to solve the architectural model and superimpose the failure behavior of the components and the interfaces on to the solution of the architectural model, to predict reliability. Solution of the architectural model provides the performance metrics for the application. [106]. Several techniques have therefore emerged to analyze the reliability of component-based applications. These fall into two groups:

System-level reliability estimation: reliability is estimated for the application as a whole.

Component-based reliability estimation: application reliability is estimated based on the reliability of the individual components and their interconnection mechanisms.[117].

2.3.4 Metrics to be Evaluated

Quantifying the Software Reliability is still a question mark for the researchers because of the lack of understanding the nature of software and its application in wide areas. There is no specific definition on how to use different aspects to relate the Software Reliability. Hence the study can be conclude that there is no suitable way measure the Software Reliability. It is important to have an understanding about the measurements that is related to reliability to reflect the characteristics, if reliability cannot be measured directly.

Product Metrics:

Software is reflected by its complexity, development effort and reliability. Complexity directly relates to Software Reliability, Complexity oriented metric is a method of determining the complexity of a program's control structure, by representing in graphical form. Representative metric is Mc Cabe's Complexity metric, test coverage metrics are a way of estimating fault and reliability on software products.

Project management metrics:

Researchers have realized that good management can result in better products. Research has demonstrated that a relationship exists between the development process and the ability to complete projects on time and within the desired quality objectives. Higher reliability can be achieved by using better development process, risk management process, configuration management process, etc.

Process metrics:

Based on the assumption that the quality of the product is a direct function of the process, process metrics can be used to estimate, monitor and improve the reliability and quality of software. ISO-9000 certification, or "quality management standards", is the generic reference for a family of standards developed by the International Standards Organization.

Fault and failure metrics:

The goal of collecting fault and failure metrics is to be able to determine when the software is approaching failure-free execution. Minimally, both the number of faults found during testing (i.e., before delivery) and the failures (or other problems) reported by users after delivery are collected, summarized and analyzed to achieve this goal.

2.3.5 Fault Tolerance & Testing

Software fault tolerance is a necessary part of a system with high reliability. It is a way of handling unknown and unpredictable software (and hardware) failures (faults), by providing a set of functionally equivalent software modules developed by diverse and independent production teams. The assumption is the design diversity of software, which itself is difficult to achieve. Software testing serves as a way to measure and improve Software Reliability. It plays an important role in the design, implementation, validation and release phases. It is not a mature field. Advances in this field will have great impact on software industry.

The defects in software are significantly different than those in hardware and other components of the system: they are usually design defects, and a lot of them are related to problems in specification. The unfeasibility of completely testing a software module complicates the problem because bug-free software cannot be guaranteed for a moderately complex piece of software. Bug-free software product cannot be achieved even with the hardest attempts during the development. Losses caused by software defects cause more and more social

and legal concerns. Guaranteeing not known bugs is certainly not a good-enough approach to the problem. Components act as transformers to the operational profile and an appropriate modeling formalism has to be found to describe this transformation maybe based on extended service effect automation [112]. The sequential flow of steps for Software Reliability Assessment are explained in Figure 2.2.

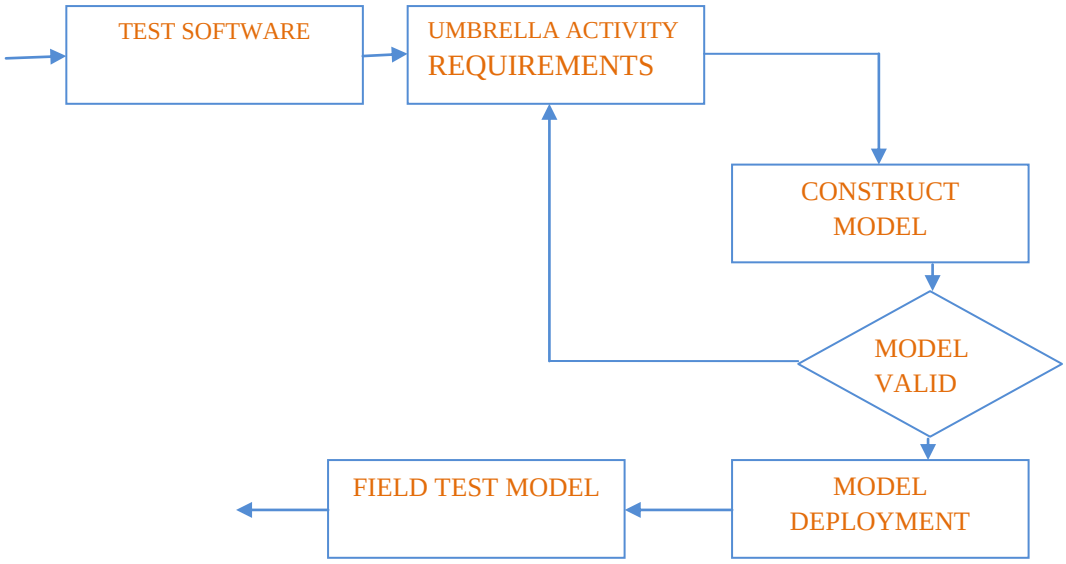

Figure 2.2: Block Diagram of Software Reliability Assessment

2.3.6 Software Reliability (SR) Estimation Models

The mathematical expressions that specify the failure of software process is said to be Software Reliability estimation models. This chapter discuss about the various types and methods for prediction and estimation of the software failures. A brief description about the existing methods and evaluation strategies, classifications and their assumptions is presented. According to [22] & [23] there are two models for Software Reliability estimation

a. Software Reliability Prediction Models

b. Software Reliability Estimation Models or Growth Models (SRGM)

a. SR Prediction Models

A fault (or bug) refers to the manifestation in the code of a mistake made by the programmer or designer with respect to the specification of the software. Activation of a fault by an input value leads to an incorrect output. Detection of such an event corresponds to an occurrence of a software *failure*. Input values may be considered as arriving to the software randomly. So although software failure may be not generated stochastically, it may be detected in such a manner. Therefore, this justifies the use of stochastic models of the underlying random process that governs the software failures. Two approaches are used in SR modeling. The prevalent is the so-called *black-box* one, in which only the interactions of the software with the environment are considered.

These types of models predict the reliability based on the reliability metrics measured or calculated during early stages of SDLC [18].This is done by comparing with a similar project in which the failure probability is known. The software with which it compared is said to be as proof program/software. The proof program reliability can be known at any instant of SDLC. The similarities are compared in terms of operational profile, service delivery and validity. Sleeping faults and changes in the operational profile may leads to behavioral change in SR. Operational profile changes occur when there is hardware ware out or some malicious program which affect the hardware resource usage which corresponds to the mismatch with the software design. In reality it is very difficult to find out exactly same software, so there is a alternative method included for SR prediction, it is measured using metrics which are possible to compute .The reliability metrics such as fault density, initial number of faults, fault exposure ratio and the time for prediction is to be valid, failure probability rate, and unit initial failure rate [18] [23]

Threat modeling: It is only one point on the broader risk management continuum. In one sense, it's a way of quantifying risk. A business assigns a risk rating to potential threats that the threat-modeling process has identified. The business can then prioritize actions on the basis of that assigned risk rating. Various groups in the security realm have given specific meanings to

the generic term threat modeling. [120] Extended Model Driven Architecture (MDA) approach is used with the addition of quantitative security assessment model that provides the feedback at every phase of Software Development Life Cycle that will help to identify security flaws as early as possible [124]

b. Software Reliability Growth Model (SRGM)/Estimation Models

SRGM estimates the reliability based on the observed data which are collected during testing. A wide variety of SRGM models can be found in literature and these models consider observed failure time as input. The observed failure time data is twofold as interval data and failure time data; interval data defines as the number of failures observed over a desired and a constant period where as the failure time data defines the time to occur a failure. When compared against the definition SR, estimation of time to next failure is more useful than estimating the number of failures over a period of time.

The reliability quantified against time can be categorized as execution time, calendar time, clock time. The actual CPU usage time by the software, from the start of the program to the end is the *execution time.* The time people normally experience which includes the time during which a computer may not be running is considered as *calendar time*. However, the elapsed time from the start to end of the software running is called as *clock time* which includes the time which CPU doesn't use for program execution.

However, finding the execution time is very difficult. The actual CPU usage time is not easy to measure as operating software and other auxiliary software are also executed in parallel to particular software execution. So most of the SRGM consider clock time rather than others. Many SRGMs proposed in the literature are mainly divided into two main categories: parametric models and non-parametric models. Most parametric models depend on priori assumptions about the probability of individual failures occurring, development environments and the nature of software failures [36].

c. SRGM Classifications

There are three classes of SRGMs. These are:

(i). Exponential NHPP models

(ii). Non-exponential NHPP models

(iii). Bayesian models

A Poisson probability distribution function takes the form $f(t) = \lambda e^{-\lambda t}$. The mean time function is $\mu(t) = \lambda$. Homogeneous Poisson Process models assume a constant mean time function while Non-Homogeneous Poisson Process models assume a mean time function to be non-liner.

(i). Exponential NHPP Models

Models in this type are based on Shooman's model, Musa's basic model, Jelinski and Moranda's model. Below is the probability distribution of these models

Shooman's model

$$f(t) = k[\frac{E_0}{I_T} - \epsilon_c(x)] \qquad (2.1)$$

Where E_0 is the initial number of faults in the program that will leads to failures

E_c is the number of faults in the program which have been found and corrected

K- is constant of proportionality

Musa's basic model

$$\mu(t) = \beta_0(1 - e^{-\beta_1 t}) \qquad (2.2)$$

Where β is Negative of derivative of failure rate divided by failure rate

Jelinski and Moranda's Model

$$\mu(t) = N(1 - e^{-\phi t}) \qquad (2.3)$$

Scheneidewind's models

$$\mu(t) = \frac{\alpha}{\beta}(1 - e^{-\beta i}) \qquad (2.4)$$

(ii). Non-Exponential NHPP Models

Duane's Model

$$\mu(t) = \alpha t^{\beta}) \quad (2.5)$$

Brook and Mohey's Poisson models

$$P(x = n_i) = \frac{(N_i \phi_i)^n \, e^{N_i \phi_i}}{n_i!} \quad (2.6)$$

Yamada's S-Model

$$\mu(t) = N \sum_{i=1}^{k} p_i [1 - e^{-\beta_i t}] \quad (2.7)$$

Musa and Okumoto model

$$\lambda_0 e^{-\phi \mu} \quad (2.8)$$

(iii). Bayesian Models

Little wood $\lambda_{linear}(t) = \frac{\alpha - 1}{\sqrt{\beta_0^2 + 2\beta_1 t(\alpha - 1)}}$

and $\lambda_{quadratic}(t) = \frac{v_1}{\sqrt{t^2 + v_2}} \left(\left(t + (t^2 + v_2)^{\frac{1}{2}} \right)^{\frac{1}{3}} - (t - (t^2 + v_2)^{1/2)^{\frac{1}{3}}} \right)$

(2.9)

The accuracy of the models in the same class is generally the same, as the general reliability function of them is same. Hence, it is enough to argue about the accuracy if at least one model in each class is considered.

d. Consideration for SRGM

Many considerations are associated with SRGM's. Some of these assumptions are tabulated below.

Table 2.1: Assumptions Vs Reality of SRGMs

S.No	Assumptions	Reality
1	Discovered defects are repaired immediately	Test time may be artificially accumulated if anon repaired defect prevents other defects from being found. These are not repaired immediately but practically accommodated
2	Defect repair is perfect	Repairing of defects creates more new defects and these are less likely to be discovered
3	No new code is introduced during QA testing	New code is introduced throughout the entire test period. there are techniques to account for new code introduction
4	Testing group will report the defects	Many groups of people will report the testing ,this can be accommodated by restricting defects those discovered by QA
5	Each unit of time is equivalent	This is not true for calendar time. For execution time "corner" tests some times are more likely defects. However as long as the test sequences are reasonably consistent from release to release, this can be accounted for lessons learned on previous releases.
6	Operational profile representation	Customers run many application under different configurations which is difficult to define an appropriate profile
7	Independency of failure	This is reasonably acceptable when there is section of code that has not been thoroughly tested .test run against this section of code may find misappropriate share of defects

Most of the models use the assumptions stated in the Table 2.1. Some of them are represented in the Table 2.2.

Table 2.2: Assumptions of SRGMs

Model	Assumptions
Jelinski-moranda model	1,2,3,4,5,6,7
Non homogeneous Poisson process	1,2,3,4,5,6
Scheidewind's	1,2,3,4,5,6
Musa's basic execution model	1,2,3,4,5,6
Hyper exponential	1,2,3,4,5,6
Weibull model	1,2,3,4,5,6
S-Shaped reliability growth model	1,2,3,4,5,6
Duane's model	1,2,3,4,5,6
Geometric model	1,2,3,4,5,6
Musa Okumoto Logarithmic model	1,2,3,4,5,6
LittleWood Verrall reliability growth model	1,2,3,4,5,6

e. Limitations or Issues concerned with existing SRGM's

Accuracy and time consumption are the two main issues related to SRGMs. Here, the description about the affect of accuracy is presented. The factors are: Uncertainty, Lack of flexibility, complexity of parameter estimation. All exponential distribution based models reflect finite failures and logarithmic distribution based model reflect infinite failures [95] [96].

(i). Uncertainty of Software Behavior

Behavior of software is uncertain [18] and it is not accurate that the failure data would not allow a particular distribution. Any SRGM implicitly expects a pattern for failure data and with respect to such distribution. Usually some parameters are used to estimate this distribution like mean value of time

to failure, total number of remaining failures are calculated. A mean time is used to represent the failure dataset and likely to be distributed around the mean time. To achieve the uncertainty assumptions like dataset are followed for a particular distribution. When a new estimation has to be made the parameters of SRGM should be estimated again and again using the failure data.

(ii) Lack of Flexibility

Software should be flexible and two folded as the code changes and the operational profile changes [24]. L. Tian and A. Noore, proposed network structure that has the capability of learning and recognizing the inherent internal temporal property of cumulative failure time sequence. Further, by adding a penalty term of sum of network connection weights, Bayesian regularization is applied to the network training scheme to improve the generalization capability and lower the susceptibility of overfitting [25]. During testing and maintenance phase bugs are discovered and are fixed while fixing bugs the software codes are normally changes as a result large number of code changes will change the behavior of the software. So the reliability calculated before the change is no longer valid and it is should be calculated again. From this study, the researcher can say that the software must be flexible enough to accommodate the code changes. When there is a change in the operational environment or changes in the hardware equipments then the software behavior is changes. To estimate the changed reliability accurately by SRGM, it should be capable of estimating reliability with a minimum number of failure data. This feature is used only in Scheniedewinds models.

(iii) Complexity of Estimation

Parameter estimation involves complex calculations and includes various numerical analysis such a s least square method, minimum mean square approach and many .when the calculation are complex enough and models are difficult to analyze then they are not preferable to use in the real time

environment. This might be one of the reason why Software Reliability is not practiced in many of the commercial environments. When designing SRGM care should be considered to include simple mathematical calculation based estimation approaches.

f. Characteristics Features of SRGM

Some of the listed features are discussed here, it is clear that any useful model should not follow any statistical distribution. Hence the usage of parametric statistical methods will not contribute to enhance the accuracy of reliability estimation. Similarly, the reliability is a random process. It is important to give a considerable contribution to achieve the randomness in the estimation process. Recent past failure data are only implied the exact estimate of the Software Reliability, so the researchers focus on employing the randomness, when estimating the reliability.

2.4 Neural Networks

Statistics are built up with integers, real numbers, vectors and matrices where as artificial Neural Networks are built up with directed graphs which are intended to model to some extent the neural mechanisms of the human brain. The nodes of the graphs are termed as inputs, processing units and the directional links are weighted input feeds and weighted connections between the processing units. Many network structures and theories are described so far but fundamentally any Neural Networks is a collection of computational units that are inter connected in some manner. The network architecture is specified by the arrangement of processing units and their interconnections.

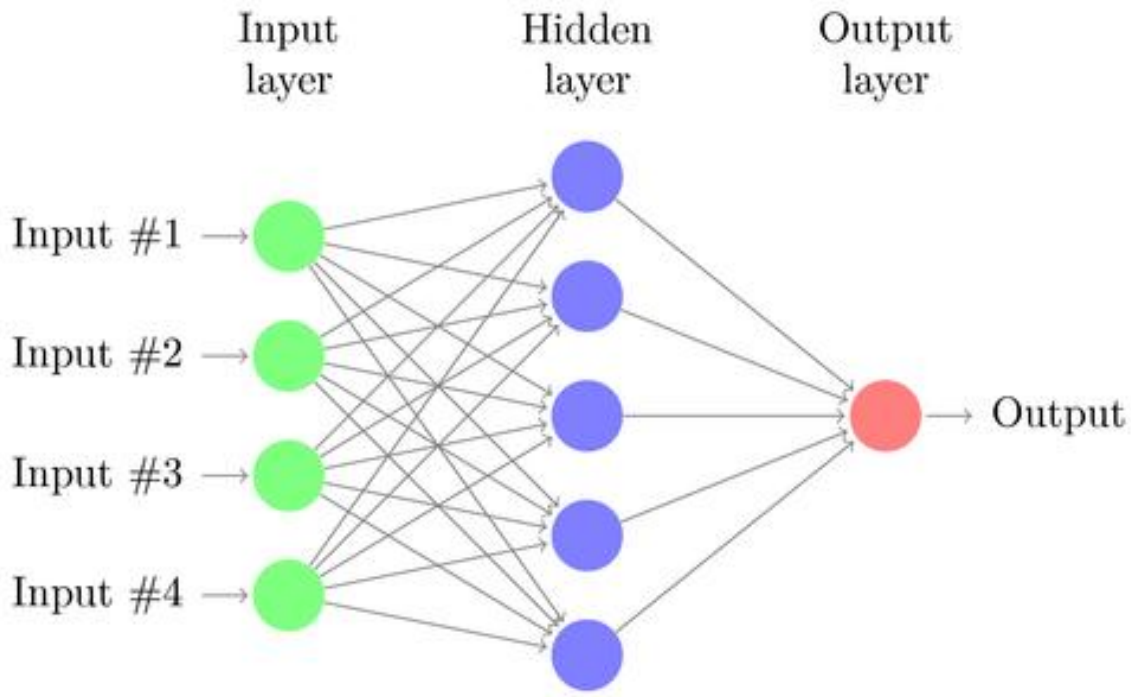

Figure 2.3: Generalized Block Diagram of a Neural Network

Figure 2.3 shows an example of a Neural Networks structure which has internal processing layer and two interfaces with the external world, an input layer, one or more hidden layers and an output layer. A Neural network can be seen as an array of layers of parallel processing units with the overall goal of mapping an input to output distributed among them in a fault tolerant manner. Neural Networks are made up of many artificial neurons. An artificial neuron is simply an electronically modeled biological neuron. How many neurons are used depends on the task at hand. It could be as few as three or as many as several thousand. There are many different ways of connecting artificial neurons together to create a neural network and most common is called a feed forward network.

2.4.1 Back Propagation Neural Network (BPNN)

The most commonly applied algorithm used in neural network is the back propagation network. This is very successful in obtaining the complex non linear mappings where an analytic representation is unknown. This

architecture of back propagation is multi layer, feed forward and each layer is usually connected to the succeeding layer in opposite direction. The main types of learning are supervised, reinforcement and unsupervised. Supervised learning is inductive learning from specific examples. Back propagation is a supervised learning algorithm which means that training involves comparing the output signals with the supplied targets.

In the training phase, the output state for all units in each layer is fed forward to the next layer until the output layer is reached; the error is computed and fed backward to the units in the preceding layers for the error distribution and weight readjustment. The extent of weight adjustment is controlled by the learning rate and the momentum, the former is a factor of controlling the magnitude of the current weight change and the latter is a factor governing the amount of the previous weight change to be added to the current weight change.

Rumelhart in [153] discussed the error as the most defined as the Euclidean distance from the target, the arithmetic difference between the desired to the target value $b_{k,j}$, for a given output unit and its actual value $Z_{k,j}$ where k is the index of training pattern and j is the index output unit .The error for the training pattern Is given as

$$E_k = \frac{1}{N_0}\sum_{j=1}^{N_0}(b_{k,j} - Z_{k,j})^2 \qquad (2.10)$$

Where N_0 is the number of output units.

An important factor in the back propagation is the choice of the activation function. The requirement for the learning algorithm is that the function be everywhere differentiable. A commonly used function is one of the sigmoid functions. The desirable properties of a sigmoid function are that it is continuous, differentiable monotonically increasing and has a finite maximum and a finite minimum value reaching asymptotically. The sigmoid function typically given as

(i) Log Sigmoid Function

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.11)$$

f(x) varies between [0 1]

(ii) Hyperbolic Tangent Transfer function

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (2.12)$$

F(x) varies between [-1 1]

Algorithm for Back propagation Neural Network

1. Initialize the weights for all connections in the network

2. For each Pattern, do the following

Each unit X_i, *i=1,…n* in the input layer receives the input signals X_i, the i^{th} component of the n- dimensional input vector and broad casts this signal to all units in the next layer

3. For each hidden layer, each hidden unit -Z_j, j =1,,,P sum of its weighted inputs

$$Z_{\_inj} = V_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (2.13)$$

Applies its activation function to compute its output signal

$$Z_j = f(Z_{inj}) \quad (2.14)$$

And sends this signal to all units in the next layer (Output Layer)

4. Each output unit (Y_k, i=1,..m) sum its weighted input signals

$$y\_in_k = w_{ok} + \sum_{j=1}^{p} Z_j W_{jk} \quad (2.15)$$

Applies its activation function to compute its output signal

$$y_k = f(y_{in_k}) \quad (2.16)$$

5. Each output unit (Y_K, k=1,..m) receives a target pattern corresponding to the input training pattern, computes its error information term

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \quad (2.17)$$

Calculates its weight correction term

$$\Lambda w_{jk} = n\delta_k Z_j + \alpha \Lambda w_{jk}(old) \quad (2.18)$$

Calculates its bias correction term

$$\Lambda w_{0k} = n\delta_k + \alpha\Lambda w_{0k}(old) \quad (2.19)$$

6. For each hidden layer each hidden unit (Z_j, j=1,...p) sums its delta inputs from the next layer

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.20)$$

7. Multiples by the derivate of its activation function to obtain its error information term .calculates its weight correction term for updating V_{ij}

$$\Lambda v_{ij} = n\delta_k + \alpha\Lambda vij(old) \quad (2.21)$$

8. For Each output unit (Y_k,k=1,...m) adjust its bias weights for p+1 input connections (j=0,...p)

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad (2.22)$$

9. For each hidden layer, hidden unit (Z_j,j=1,..p) adjusts its bias and weights for n+1 input connections

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \quad (2.23)$$

The algorithm based step by step procedure about the methodology of BPN is explained in the form of flow chart in Figure 2.4.

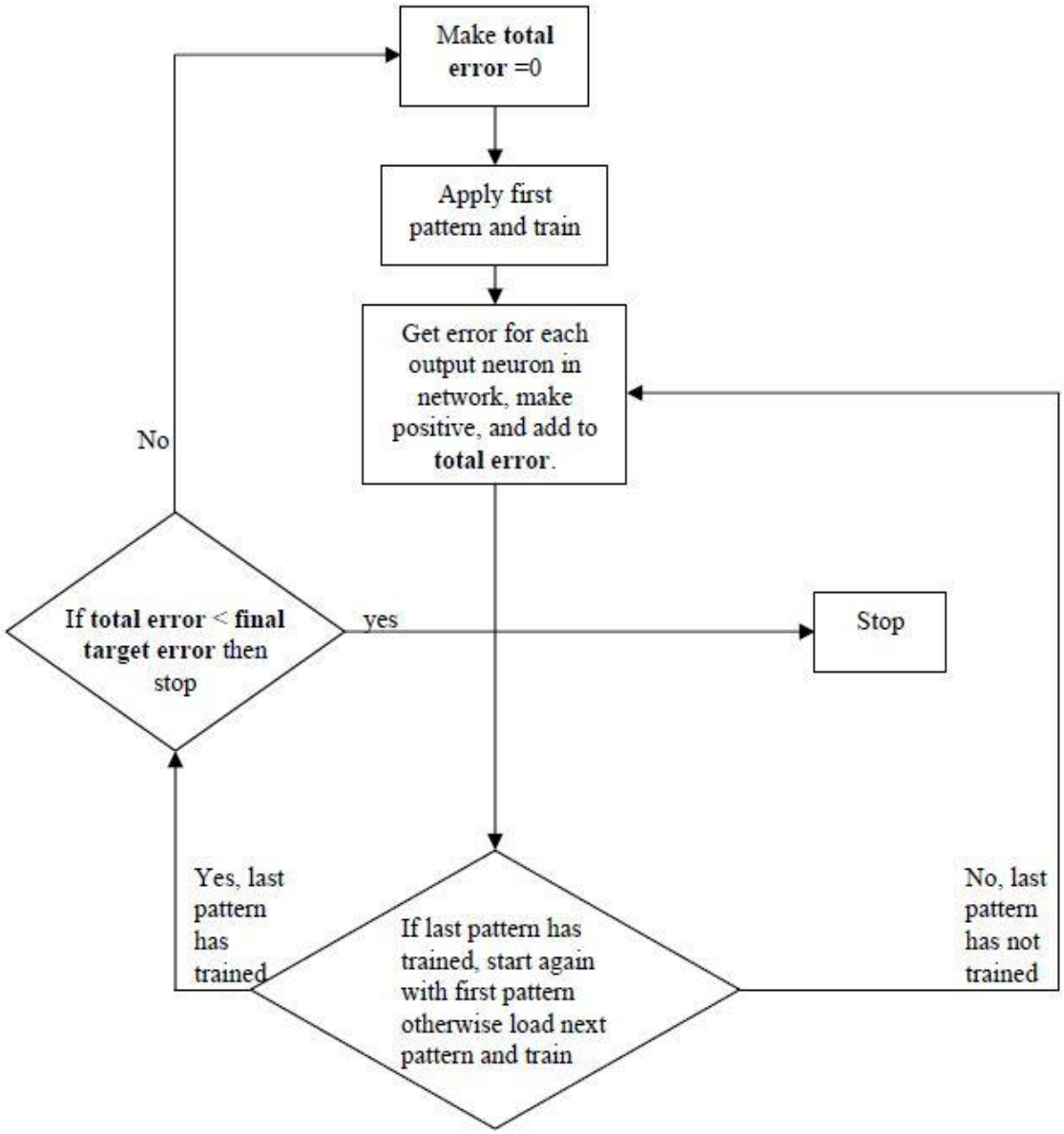

Figure 2.4: Flow Chart explaining briefly about the Methodology of BPN

2.4.2 SRGM with ANN

Here first consider the logistic growth curve model. This model simply fits the mean value function with a form of the logistic function. Its mean value function is given by:

$m(t) = \frac{a}{1+ke^{-bt}}$ a>0, b>0,k>0 (2.24)

Now replace k with e^{-c}

$$m(t) = \frac{a}{1+e^{-c}e^{-bt}} = \frac{a}{1+e^{-(bt+c)}} \qquad (2.25)$$

Assume $g(x) = bx+c$

There fore $m(x) = \frac{a}{1+e^{-(bx+c)}}$

This means that the mean value function of logistic growth curve model is composed of $g(x)$, $f(x)$, and $m(x)$. Subsequently, try to derive the compound functions from the viewpoints of neural network. Consider the basic feed-forward network shown in Figure 2.5. Note that the network has only one neuron in each layer and W^1_{11} , W^0_{11} are the weights and b_1, b_0 are the biases. When the input, $x(t)$, at time t is fed to the input layer, the following form can be derived.

The input of the $h_{in(t)} = w^1_{11}t + b_1$

The Output of the hidden layer is $h(t) = f(h_{in(t)})$

Where f(x) is the activation function in the hidden layer

The input of the output layer is

$y_{in(t)} = w^0_{11}h(t) + b_0$

The output of the output layer is $y(t) = g(y_{\_in(t)})$

Where g(x) is the activation function in the output layer

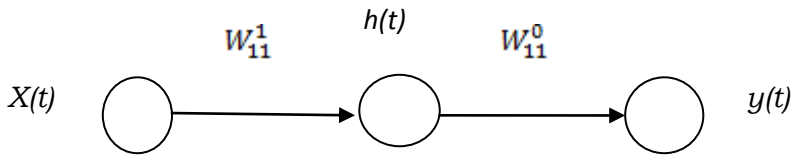

Figure 2.5: Feed Forward Neural Network with Single Neuron in each layer

Furthermore, remove the bias in the output layer and can consequently get

$$y(t) = W_{11}^{0} f\left(h_{in(t)}\right) = \frac{W_{11}^{0}}{1+e^{-(W_{11}^{1}+b_1)}}$$

From the above equation the Software Reliability growth model now integrated with ANN

2.4.3 Calculations of the SRGM

The basic feed forward neural network architecture comprises in two steps

- Feed forward neural Network
- Back propagation

The input vector is propagated through a weight layer. It is combined with the previous state activation.

The conventional feed forward neural network consists of two layered network, so it comprises of two step mapping

$$y(t) = G(F(x(t))$$

The back propagation learning techniques are used in the above equation to update the weights of the networks for training the feed forward back propagation network. The operation is restricted to hidden layer and output layer.

The input vector 'x' is propagated with a layer associated with weights 'V' as

$$y_j(t) = f\left(net_j\right)(t)$$

$$net_j(t) = \sum_i v_{ij}\left(x_i(t)\right) + \theta_i$$

Where 'n' is the number of input nodes θ_i is a bias 'f' is an activation function

The output of the network is calculated by state and weight 'W' associated with the output layer

$$y_k(t) = g(net_k(t))$$

$$g(net_k(t)) = \sum_j y_j(t) w_{kj} + \theta_k$$

Here sigmoid function is takes as activation function. The general flow of the Neuro Fuzzy based approach is depicted in Figure 2.6.

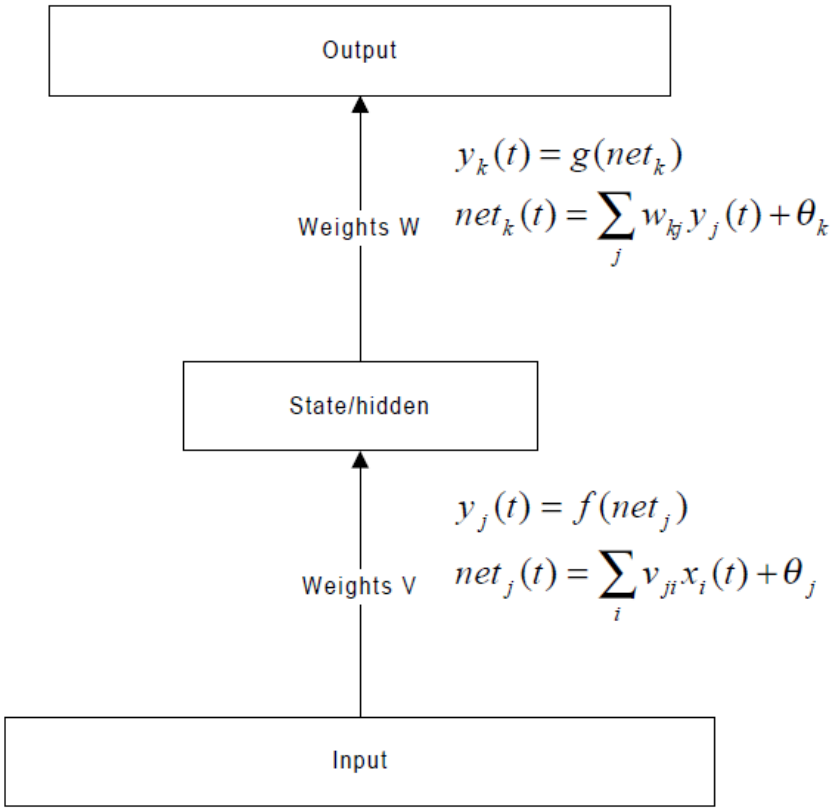

Figure 2.6: Process Flow of the Neuro Fuzzy Model

2.4.4 Tools to estimate Software Reliability

The following performance measures are being used to evaluate the proposed models

(i) Relative Error (%) $RE_i = (|P_i - A_i|)/A_i) * 100$

(ii) Average Relative Error (%) : $\frac{1}{n}\sum RE_i$

(iii) Root Mean squared error: $RMSE = \sqrt{[(\sum P_i - A_i)]^2 / n}$

(iv) Mean Absolute Error : $[\sum |P_i - A_i|]/n$

(v) Mean Error : $[\sum (P_i - A_i)]/n$

Where P_i= Predicted Value

A_i=Actual Value

N= total no of observations

2.4.5 Applications of Artificial Neural Networks (ANNs)

Software Community or industry also realized the advantages of Artificial Neural Networks. They used Artificial Neural Networks as a tool to solve problems in:

- Cost Estimation
- Software Reliability Engineering Strategy
- Software size estimation
- Software quality prediction

An interesting study is conducted on an open-issue of the comparative study for developing predictive models of software metrics for assessing the Software Reliability is provided (A. Gray and S. MacDonell 1997 in the Information and Software Technology Journal, Elsevier). This study suggested the following techniques for developing the Software Reliability Growth Models (SRGMs) for assessing the Software Reliability at different stages of Software Development Life Cycle (SDLC) [38] [39].

- Least square regression analysis
- Robust regression analysis
- Neural network
- Fuzzy systems
- Rule based systems
- Case-based reasoning

Most traditional growth models evaluated show poor directional change detectability, as evi- dent from the insignificance of the tests. This problem can be overcome with the use of neural network structures. [145].

2.5 Fuzzy Logic

Fuzzy Logic is a logical system that generalizes classical two valued logic for reasoning under uncertainty. It refers to all of the theories and technologies that employ fuzzy sets which are classes with unsharp boundaries. Fuzzy Logic has been viewed as a theory for dealing with uncertainty about complex systems. The distinguishing mark of Fuzzy Logic in rule-based systems is its ability to deal with situations in which making a sharp distinction between the boundaries of application in the use of rules or constraints is very difficult. The various applications of Fuzzy Logic include consumer products, automotive and power generation, industrial process control, robotics and manufacturing and Software Reliability. Fuzzy Logic offers an alternative to the probability paradigm, possibility that is much more appropriate to Software Reliability when there is uncertainty in the data.

Possibility mathematics allows for quantitative reliability calculations that preserve the uncertainty present in the original data, avoids making unwarranted assumptions and makes the consequences of the required assumptions clear throughout the analysis. Fuzzy Logic also improves reliability analysis through the concept of utility. Standard reliability models usually assume a binary representation of failure, a system is working or it is failed. A more flexible and realistic model allows for easy representation of partial system failures. Thus, there is considerable motivation to adapt the traditional probability based reliability methods to the Fuzzy Logic context.

2.5.1 Fuzzy Set and Membership Function

A set in a classical set theory has a sharp boundary because an element either completely belongs does not belong to a set at all. Fuzzy set directly addresses this limitation by allowing membership in a set to be a matter of degree. The degree of membership in a set is represented by a number between 0 and 1; 0 means entirely not in the set, 1 means completely in the set and a number in between means partially in the set. A fuzzy set is defined by a

function that maps objects in a domain of concern to their membership value in a set. Such function is called the membership function. The membership function of a fuzzy set A is denoted as μA, and membership value of x in A is denoted as μA (x).The domain of membership function, is called the universe of discourse.

2.5.2 Basic Operations on Fuzzy Sets

(i) Fuzzy Disjunction Operation

A fuzzy disjunction operator is the maximum operator defined as

$\mu_{AUB}(x)=\max\{\mu_A(x), \mu_B(x)\}$

(ii) Fuzzy Conjunction Operation

A fuzzy conjunction operator is the minimum operator defined as

$\mu_{A\cap B}(x)=\min\{\mu_A(x), \mu_B(x)\}$

(iii) Fuzzy Complement Operation

The complement of a fuzzy set A is defined by the difference between one and the membership degree in A.

$\mu_{Ac}(x)=1-\mu_A(x)$

2.5.3 Fuzzy Numbers

A fuzzy number is a number that is characterized by a possibility distribution or is fuzzy subset of real numbers (Guillermo and Manic, 2006).A fuzzy number is either a convex or a concave subset of the real line.

2.5.4 Types of Membership Functions

A membership function can be designed by interviewing those who are familiar with underlying concept and later adjusting the values or by constructing automatically from data or feedback from the system performance. The most commonly used membership functions are parameterizable membership functions as their use reduces the system design

time and it also facilitates the automated tuning of the system by making changes in the parameters.

(a) Triangular membership Function

A triangular membership function is specified by three parameters [a, b, c] where 'a' is the lowest value, 'b' is the nominal value and 'c' is the maximum value.

$$Triangle\ (X{:}a,b,c) = \begin{cases} 0 & x < a \\ \frac{(x-a)}{(b-a)} & a \le x \le b \\ \frac{(c-x)}{(c-b)} & b \le x \le c \\ 0 & x > c \end{cases} \qquad (3.1)$$

A triangular fuzzy number is a fuzzy number whose membership function is of triangular shape.

Triangular fuzzy number is used when fuzziness exists on both sides of the parameter.

Algebraic Operation of Triangular Fuzzy Number (T.F.N)

(i) The addition or subtraction of two T.F.N's will also be T.F.N.

(ii) If operations such as multiplication, inverse and division are performed on two T.F.N's, the resulting fuzzy numbers need not to be T.F.N.

(iii) If operations such as maximum and minimum are performed on two T.F.N's, the resulting fuzzy numbers need not to be T.F.N.

Consider two T.F.N.'s A and B defined by Triplets such as A= [a, b, c] and B= [d, e, f], the various arithmetic operations on A and B are defined as follows

(a) Addition

A (+) B = [a, b, c] + [d, e, f]

= [a + d, b + e, c + f] is also a T.F.N.

(b) Subtraction

A (-) B = [a, b, c] - [d, e, f]

= [a - d, b - e, c - f] is also a T.F.N.

(c) Symmetry

-A= [-a, -b, -c] is also a T.F.N.

(d) Multiplication, Inverse and Division

For other operations such as multiplication, inverse and division, T.F.N in the form of Triplet is not possible to be used.

Similarly there few other membership function available for fuzzy system which are discussed below

(b) Trapezoidal membership Function

A trapezoidal membership function is specified by four parameters [a, b, c, d] as follows:

$$Triangle\ (X{:}a,b,c) = \begin{cases} 0 & x < a \\ \frac{(x-a)}{(b-a)} & a \le x \le b \\ 1 & b \le x \le c \\ 0 & x \ge d \\ \frac{(d-x)}{(d-x)} & c \le x \le d \end{cases} \qquad (3.2)$$

(c) Gaussian membership Function

A Gaussian membership function is specified by two parameters $\{m, \sigma\}$ as follows:

$$Gaussian(X{:}\,m,\sigma) = \exp\left(\frac{(x-m)^2}{\sigma^2}\right) \qquad (3.4)$$

(d) Bell-shaped membership Function

A bell shaped membership function is characterized by three parameters {a,b,c} as follows:

$$Bell(X{:}\,a,b,c) = \frac{1}{1+\left(\frac{(x-c)}{a}\right)^{2b}} \qquad (3.5)$$

2.5.5 Mathematical Analysis of Fuzzy Sets

A set is a well-defined collection of objects. It has a sharp boundary to distinguish which element of the universe of discourse belongs to the set. In real life applications situations, it is not possible to distinct these elements by such a sharp layer due to the uncertainty involved. A fuzzy set is a set that consists of the elements having varying degrees of belongingness in the sets. So these situations may be better explained by the fuzzy sets, the set which

contains all the elements of the universe but with different degrees of membership.

A fuzzy set A may be defined over a universe *X* and may be characterized by its characteristic function χ_A as $= \{(x, \chi_A)\}$

Where $\chi_A(x) = \begin{cases} 1 & if\ x \epsilon A \\ 0 & if\ x \sim \epsilon A \end{cases}$

In analogy to the characteristic function, the study can define a function called membership function $\mu_A : X \rightarrow [0,1]$ to characterize a fuzzy set defined over the universe *X* as follows.

$\tilde{A} = \{(x_i, \mu_A(x_i)) : x_i \epsilon X\}$ Where $, \mu_A(x_i) \epsilon [0,1]$

If the universal set U on which a fuzzy set A is defined is a finite set having n elements. Then this fuzzy set A can be written as

$\tilde{A} = \frac{a_1}{x_1} + \frac{a_2}{x_2} + \cdots \ldots \ldots \ldots \ldots \frac{a_n}{x_n}$

Where slash is used to link the elements of universal set with their membership grade and the plus sign does not mean the usual meaning algebraic sum. But if the universal set is an infinite countable set then fuzzy set *A* may be written as

$A = \sum_i^{\infty} \frac{a_i}{x_i}$

In a similar manner, if the universal set is a interval of real numbers or an infinite uncountable set, the fuzzy set A can be written as

$A = \int \mu_A(x_i) / x_i$

Here also the integral sign does not have its usual meaning. Rather it indicates that all the pairs in x and $\mu_A(x)$ from the interval [0, 1] collectively form *A*.

2.5.6 Principle of Operation

Let f be a crisp function from a set X to the set Y. Then this function f will be fuzzified to act as a function from fuzzy sets A to fuzzy set B defined over X

and Y respectively. In other words, it is the generalization of a crisp function. Here the study shall use the same notation *f* for this new function that has the following form.

$$f: F(x) \to F(x)$$

The above principle that is used to fuzzify the crisp function *f* is called extension principle. In crisp set theory the extended functions are restricted to crisp sets *P(X)* and *P(Y)*. But in fuzzy set theory the extended version of a crisp function is a function from *P(X)* to *P(Y)* that for a fuzzy set $\tilde{A} \epsilon P(Y)$ is defined by

$$f(\tilde{A}) = \{y: y = f(x), \quad x \epsilon \tilde{A}$$

Here *P(X)* and *P(Y)* denotes the family of fuzzy sets defined over universal set *X* and *Y*. The inverse of this extended version is a function from *P(Y)* to *P(X)*, that for any $\tilde{B} \epsilon P(Y)$ is defined by

$$f^{-1}(\tilde{B}) = \{x: f(x) \epsilon \tilde{B}\}$$

In terms of membership function the fuzzy sets $f(\tilde{A}) and\ f(\tilde{B})$ can be expressed as below

$$\mu_{f(\tilde{A})}(y) = \sup(\tilde{A}(x))$$

$$\mu_{f^{-1}(B)} = \tilde{B}(f(x))$$

In fuzzy set theory, the researchers should be familiar to three set theoretic operations, complement of a fuzzy set, intersection and union of two fuzzy sets. In this theory these operations can be given by the following equations in the form of their characteristic function.

In crisp set theory, the researchers are familiar to three set theoretic operations, complement of a fuzzy set, intersection and union of two fuzzy sets. In this theory these operations can be given by the following equations in the form of their characteristic function

$$\chi_{A^c} = 1 - \chi_A(x)$$

$$\chi_{A \cap B}(x) = \min\{\chi_A(x), \chi_B(x)\}$$

$$\chi_{A \cup B}(x) = \max\{\chi_A(x), \chi_B(x)\}$$

Where $\tilde{A}$ *and* $\tilde{B}$ are two sets defined over universal set U. in the case of fuzzy sets the researchers can generalize these operations to form some new set theoretic operations in fuzzy set theory. Analogues to these three operations, if the researchers have two more fuzzy sets then complement, intersection and union of the above sets may e defined by the following equations

$$\mu_{A^c} = 1 - \mu_A(x)$$

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$$

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$$

Where $\tilde{A}$ *and* $\tilde{B}$ are two fuzzy sets and μ_{A^c} , $\mu_{A\cap B}$, $\mu_{A\cup B}$ are membership function for fuzzy complement and fuzzy set A , intersection and union of two fuzzy sets $\tilde{A}$ *and* $\tilde{B}$ respectively. But it is also true that they are not the only possible generalizations. Rather there are many other function that are qualify to be generalization the classical operations i.e., the complement, intersection and union are not unique in fuzzy set theory.

2.5.7 Fuzzy Complement

Let $\tilde{A}$ be a fuzzy set defined over U then $\mu_{\tilde{A}}(x)$ represents the degree of belongingness of x in $\tilde{A}$. Then $\mu_{\tilde{cA}}(x)$ means not only the degree to which x belongs $C\tilde{A}$ but also the degree to which x does not belongs to $\tilde{A}$. Then the complement of any fuzzy set $\tilde{A}$ may be defined by a function c: [0,1] that assigns a value $\mu_{\tilde{cA}}(x)$ to each membership grade $\mu_{\tilde{A}}(x)$ of any grade given fuzzy set $\tilde{A}$

All functions c : [0. 1] need not to be a fuzzy complement. Rather there are certain axioms those must be satisfied by the function c to be a fuzzy complement. Following are the axioms that are necessary for a function c to form fuzzy complement

Axiom c1: Boundary Conditions: c(0)=1 and c(1)=0., then

Axiom c2: Monotonicity: For all $a, b \in [0, 1]$, if $a \leq b$, *then* $c(a) \geq c(b)$

These two axioms are called Skelton axioms of fuzzy complement. There may be many functions those satisfy the axiom c1 and c2. These functions form a most general class of fuzzy complements. Except the skeleton axioms there are following two namely c3 and c4.

(i) Fuzzy Intersections

In a similar manner the intersection of two fuzzy sets $\tilde{A}$ and $\tilde{B}$ is defined bya binary operation on the unit interval, if intersection $\tilde{A}$ and $\tilde{B}$ is denoted by I then

I: X[0,1]-> [0,1]

Any function *i* of the above form will qualify as a fuzzy intersection if it possesses appropriate properties. Following are the properties those should be at least satisfied by the binary operation *i* on [0. 1].

(ii) Fuzzy Variables or Linguistic Variables

Fuzzy numbers like “about 2”, “near to 5” etc. are used to define the state of variables. A variable whose states are fuzzy numbers is called a fuzzy variable. In some cases, when the fuzzy numbers used to represent the state of fuzzy variables are linguistic terms such as “very small”, “small”, “medium” and so on, the fuzzy variable is termed as linguistic variable. Fuzzy variables are more significant than crisp variables because they show the gradual transitions between states and therefore possess a natural capability to deal with uncertainties. On the other hand the crisp variables do not have this capability.

Each linguistic (fuzzy) variable can be characterized by a quintuple (*v, T, U, g, m*), where *v* is the name of variable which is called base variable, *T* is the set of linguistic *terms* of base variable *v*. *U* is the universal set, over which the values of base variable *v* ranges, *g* is a syntactic (grammatical) rule used to arrange linguistic terms, and *m* is a function that maps the elements (linguistic

terms) of set *T* to the set. *F(U)*, where *F(U)* is the set of all fuzzy numbers defined over *U* i.e. $m : T \rightarrow F(U)$.

Figure 2.7 shows an example of fuzzy (linguistic) variables. The name of this fuzzy variable is *"temperature"*. It shows the temperature of an entity in a given context by few linguistic terms very *low, low, medium, high, and very high.* All these five linguistic terms are assigned of five fuzzy numbers by the function *m* (semantic rule). All these fuzzy numbers have the trapezoidal shape and defined on the interval [0, 200], which is also the range of the base variable.

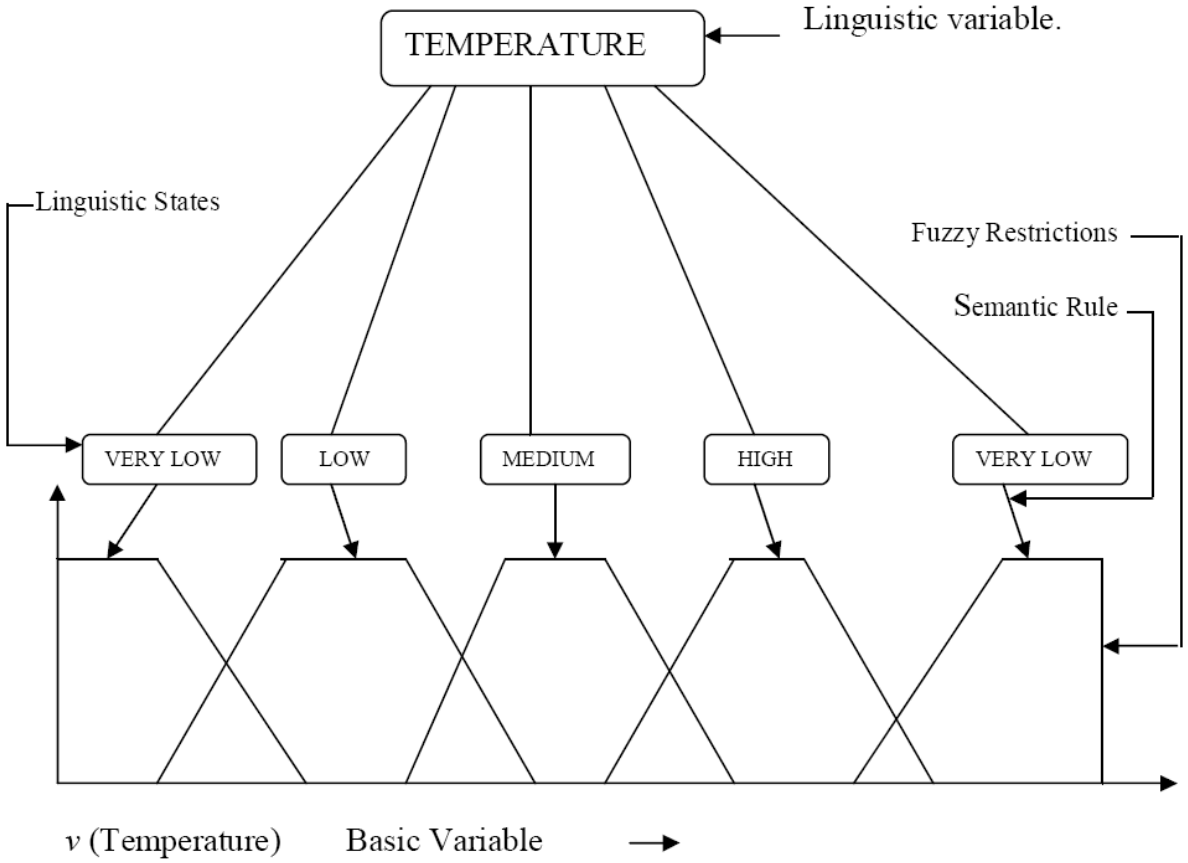

Figure 2.7: Example of Linguistic Variable

2.5.8 Construction of Fuzzy Sets

There is always a general scenario within which the researchers has to construct a fuzzy set for research purposes. This scenario is comprised of specific knowledge domain of interest, one or more experts in this domain and a knowledge engineer. The knowledge engineer works as a collector i.e. he /she has to collect the knowledge of interest provided by the experts and to express it in form of propositions involving linguistic terms. In the study, this knowledge is expressed in term of linguistic variables.

In first phase, the knowledge engineer tries to elicit the knowledge in the form of natural language. In second phase the knowledge engineer tries to understand the meaning of each linguistic term employed in these propositions. In other words the study can say that the fuzzy set is constructed in second phase. There are many methods in the literature, based on experts' judgment. These methods have been classified as direct methods and indirect methods. Indirect method is easier than direct method in the sense that, in direct method the expert has to answer some typical questions concerned to the constructed membership function. However, in indirect method, the questions to be answered by the experts are easier than the questions in direct method.

Further these two methods are classified into the methods involving one expert or more than one expert. Thus, there are four categories of these methods i.e.

(i) Direct methods involving one expert.

(ii) Direct methods involving more than one expert.

(iii) Indirect method with one expert.

(iv) Indirect method with multiple experts.

(i) Fuzzification

After taking the measurement of all variables involved in the analysis, convert them into an appropriate fuzzy set to express measurement uncertainties. In other way, a fuzzification function is introduced for each variable concerned to the study. Suppose there is a function *fe* known as a fuzzification function for a variable *e*. Then this fuzzification function has the form

$$f_e = [-a, a] \rightarrow R$$

Where *R* denotes the set of all fuzzy numbers and *x* takes values from the closed interval [-a, a]. In particular $f_e(x0)$ is a fuzzy number chosen by *fe as a fuzzy approximation measure e=x0*

(ii) Defuzzification

In certain situations one needs a crisp output when the input number is fuzzy. De-fuzzification is the tool that makes it possible. There are several methods of de-fuzzification in the literature. The centriod method, which is given by the following expression

$$X^* = \frac{\int \mu_A(x).x.dx}{\int \mu_A(x).dx}$$

Where *A* is a fuzzy set, which is the union of two or more fuzzy sets

Many problems and issues are reported over practicing SRGMs model structures that made analysis and design of such model a quite challenging process to complete. For example:

- Missing or incomplete data
- Large number of variables or unused extra variables.
- Strong co-linearity (cohesion) between/among the variables.
- Complex linear/non-linear relationships between model variables.
- Outliers and small sizes of the data sets used for the model evaluation.

Defuzzification is the process to calculate the output, after applying if-then rules. It refers the way in which fuzzy sets are transformed into numerical

value. Seventeen Input Parameters and Output parameter named Security Vulnerability [142].

2.5.9 Fuzzy Methodology in Software Reliability Analysis

(i) Probabilistic System

The central concept of reliability theory is a measure of the expected capability of an engineering product without failures under specific conditions for a given period of time. Generally, the classical reliability theory is based on two basic assumptions:

- Assumption of dichotomous states or Binary-state assumption: At any given time every system demonstrates two crisp states, one is fully working state and the other is fully failed state.
- Probability assumption: The system failure behavior with respect to the two crisp states (functioning and failed) can be fully characterized in terms of probability theory.

(ii) Profust Reliability

The binary state assumption on which the classical reliability theory is based is not acceptable in many practical situations. In order to cope with this unacceptability, there is a need to replace the binary state assumption by the fuzzy state assumption. Thus a invention of a new theory of reliability analysis, called Profust Reliability analysis. This theory is based on the following two assumptions.

- Fuzzy-state assumption: The behavior of a system cannot be characterized by the binary state assumption since the degradation of a system is gradual not a random phenomenon. That is at any time the system can be viewed as being in one of the two fuzzy states to some extent. In other words the system failure is not defined precisely, rather it is defined in a fuzzy way.

- Probability assumption: Similar to Probist Reliability theory. The system behavior can be characterized by the probability theory with respect these fuzzy states.

(iii) Posbist Reliability Theory:

The probability theory is a very powerful tool to measure the uncertainty. But it is not possible always that every situation can be characterized by probability theory. Therefore there is a need of an alternative of probability theory. Kauffmann (1988) [40] is first to introduce the term possibility for reliability and Cai (1996) [41] used the mathematical notion of possibility and fuzzy variables to develop a theory Posbist Reliability. The theory, is based on the following two assumptions

- Possibility assumption: This assumption states that the failure behavior of a system is fully characterized by the possibility theory.
- Binary–state assumption: This is same as, in Posbist reliability theory, that every system posses two states and at any particular time the system is either in functioning state or in failed state.

(iv) Posfust Reliability theory: As the name suggests the Posfust Reliability theory is based on two basic assumptions:

- Possibility assumption: In Posfust Reliability theory the possibility assumption is the same as in the Posbist Reliability. That is, the failure behaviour of a system with respect to fuzzy states "working" and "failed" can be characterized by possibility theory.
- Fuzzy-state assumption: In Posfust Reliability theory, the fuzzy state assumption has the same meaning as in Profust reliability theory.

Using the Posfust reliability theory the reliability of a system is interpreted as the possibility that a fuzzy system failure does not occur in a specified time interval under prescribed conditions. The system failure is a fuzzy event and not a crisp event.

2.6. Neuro Fuzzy based Approach

The idea of a Neuro Fuzzy system is to find the parameters of a fuzzy system by means of learning methods obtained from Neural Networks. In this chapter the basic properties of Neuro Fuzzy systems are discussed. The learning techniques that can be used to create fuzzy systems for data; a common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural-network-like architecture. Then a learning algorithm – such as back propagation – is used to train the system. There are some problems, however. Neural network learning algorithms are usually based on gradient descent methods. They cannot be applied directly to a fuzzy system, because the functions used in the inference process are usually not differentiable. There are two solutions to this problem:

- Replace the functions used in the fuzzy system (like min and max) by differentiable functions
- Do not use a gradient-based neural learning algorithm but a better-suited procedure.

There are several different approaches which have much in common, but differ in implementation aspects. To stress the common features of all these approaches, and to give the term Neuro Fuzzy system a suitable meaning, only apply it to systems which possess the following properties:

- A Neuro Fuzzy system is a fuzzy system that is trained by a learning algorithm (usually) derived from neural network theory. The (heuristic) learning procedure operates on local information, and causes only local modifications in the underlying fuzzy system. The learning process is not knowledge-based, but data-driven.
- A Neuro Fuzzy system can always (i.e. before, during and after learning) be interpreted as a system of fuzzy rules. It is possible both to create the system out of training data from scratch, and to initialize it from prior knowledge in the form of fuzzy rules.

- The learning procedure of a Neuro Fuzzy system takes the semantical properties of the underlying fuzzy system into account. This results in constraints on the possible modifications of the system's parameters.
- A Neuro Fuzzy system approximates an n-dimensional (unknown) function that is partially given by the training data. The fuzzy rules encoded within the system represent vague samples, and represent vague prototypes of the training data. A Neuro Fuzzy system should not be seen as a kind of (fuzzy) expert system, and it has nothing to do with Fuzzy Logic in the narrow sense [66] [67].
- A Neuro Fuzzy system can be represented by a special three-layer feed forward neural network. This view of a fuzzy system illustrates the data flow within the system and its parallel nature. However, this neural network view is not a prerequisite for applying a learning procedure, it is merely a convenience.

The Neuro Fuzzy technique, then, is used to derive a fuzzy system from data, or to enhance it by learning from examples. The exact implementation of the Neuro Fuzzy model does not matter. It is possible to use a neural network to learn certain parameters of a fuzzy system, like using a self-organizing feature map to find fuzzy rules [68], or to view a fuzzy system as a special neural network and to apply a learning algorithm directly [69]. A lot of Neuro Fuzzy approaches use a neural network-like graph to illustrate the data flow and the computations that are carried out in a fuzzy system [111].

This neural network representation is then used to formalize the application of a learning algorithm. Many Neuro Fuzzy approaches use a 5-layer feed forward or node-oriented architecture as it is shown in Figure 2.8 (a) below. This kind of representation is called node-oriented, because in this kind of representation all membership function parameters of the fuzzy system reside inside the nodes of the network and the connections are not needed to carry any parameters. This means, from a neural network point of view, this architecture is not adaptive as long as there are no weights attached to the connections. Such a node-oriented representation of a fuzzy system is therefore

often used for defining a neural network-like learning algorithm based on adaptive weights that are attached to some of the connections. The layers of the network are not fully connected, but the connections are selected such that they represent the rule base of the fuzzy system. The network in Figure 2.8 represents a fuzzy system with the following two fuzzy rules

R1: if x1 is small and x2 is small then y is small

R2: if x1 is large and x2 is large then y is large

The meaning of the layers of the network representation is in Figure 2.8.

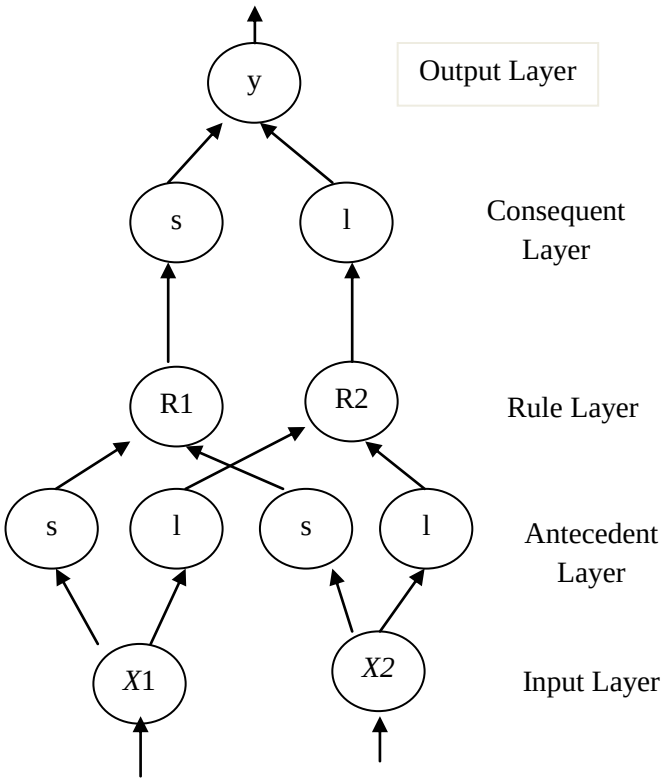

Figure 2.8 (a): Two Fuzzy Systems represented as 5-Layer Feed Forward Network.

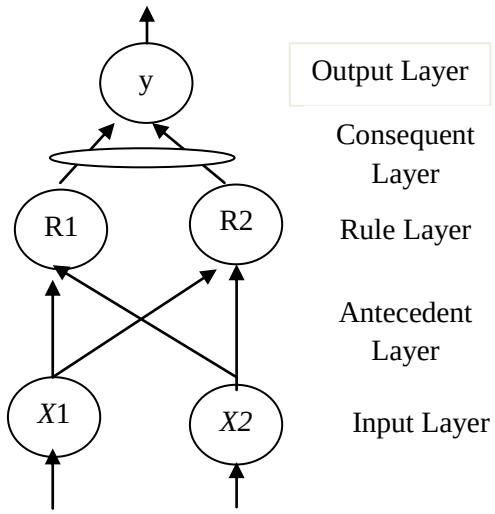

Figure 2.8 (b): Two Fuzzy Systems represented as 3-Layer Feed Forward Network with shared weights.

- antecedent layer: fuzzy sets used in linguistic terms of antecedents,
- rule layer: fuzzy rules,
- consequent layer: fuzzy sets used in linguistic terms of consequents,
- Output layer: output variables.

If a learning algorithm is applied to modify parameters of the fuzzy system, this would mean that parameters inside the nodes of the second and fourth layer are modified. By storing the membership functions inside these layers, there is a need to ensure that all fuzzy rules that are represented by the nodes of the third layer use the same set of membership functions to represent linguistic terms. Instead of the 5-layer network representation of a fuzzy system shown in Figure 2.8 (a), Generally, the researchers can prefer a 3-layer or connection-oriented representation as it is shown in Figure 2.8 (b), where the connections carry fuzzy sets as weights. This representation better corresponds with a neural network view where all parameters that can be changed by a learning algorithm are located at the connections. In order to ensure that each linguistic term is only represented by one fuzzy set, the study can use shared weights (coupled connections).

The fuzzy system in Figure 2.8 (b) consists of the two rules

if $x1$ is *large* and $x2$ is *small* then y is *large*

if $x1$ is *small* and $x2$ is *large* then y is *large*

In this example, the connections from the rule layer to the output layer share the same (fuzzy) weight – *large* – which is represented as a fuzzy set over the domain of the output variable *y*. A learning algorithm will recognize that the connections are coupled and make sure that the weight is always identical for both connections. The meaning of the layers of the network representation in Figure 2.8 (b) is equivalent to the layers in Figure 2.8 (a), with the same names. In the 3-layer representation the connections also encode parameters of a fuzzy system:

- The weights on the antecedent connections represent fuzzy sets used in linguistic terms of antecedents.
- The weights on the consequent connections represent fuzzy sets used in linguistic terms of consequents.

With the use of Fuzzy Logic secure software system (SSS) approach is introduced. It will help to avert the failed state of the system [134].

2.6.1 Validation of Neuro Fuzzy Interface model

The idea behind ensemble (Neuro Fuzzy) systems is to exploit each constituent model's unique features to capture different patterns that exist in the dataset. Both theoretical and empirical works indicate that ensembling can be an effective and efficient way to improve accuracies. Since then, many researchers worked on ensembling or combined forecasts.Makridakis.et.al in 1982 [44] reported that combining several single models has become common practice in improving forecasting accuracy. Then Pelikan.et.al in 1992 [45], proposed combining several feed-forward Neural Networks to improve time series forecasting accuracy. Some of the ensemble techniques for prediction problems with continuous dependent variable include linear ensemble, weighted average Perrone and Cooper [46] and stacked regression and nonlinear ensemble (e.g., neural-network-based nonlinear ensemble Yu.et.al [47] reported that the generalization ability of a neural network system could be significantly improved by using an ensemble of a number of Neural Networks.

The purpose is to achieve improved overall accuracy on the production data. In general, for classification problems, an ensemble system combines individual classification decisions in some way, typically by a majority voting to classify new examples. The basic idea is to train a set of models (experts) and allow them to vote. In majority voting scheme, all the individual models are given equal importance. Another way of combining the models is via weighted voting, wherein the individual models are treated as unequally important. This is achieved by attaching some weights to the prediction given by the individual models and then combines them. Olmeda and Fernandez [48] a genetic algorithm based ensemble system, where a GA determines the optimal combination of the individual models so that the accuracy is maximized. Zhou et. al [49] carried out a detailed study on ensembling Neural Networks and proposed that using a set of Neural Networks to form an ensemble is better than to use all the Neural Networks. They proposed an approach that can be used to select the Neural Networks to become part of the ensemble from the available set of Neural Networks. Genetic algorithm is used to assign weights to the constituent networks.

It is generally the case that for a given dataset one kind of intelligent technique outperforms the other and the results can be entirely opposite when a different dataset is used. In order not to lose any generality and also to combine the advantages of the intelligent techniques, an ensemble uses the outputs of all the stand-alone intelligent techniques with each being assigned a certain priority level and provides the output with the help of an arbitrator.

An ensemble uses the output obtained from the individual constituents as inputs to it and the data is processed according to the design of the arbitrator. Four different variants of ensembles are designed and employed as shown in figures below. These include (i) linear ensemble based on average, (ii) linear ensemble based on weighted mean, (iii) linear ensemble based on weighted median, and finally (iv) A non-linear ensemble based on BPNN. These ensembles are described briefly below. Two Generic designs of Linear and Non-Linear assemble is represented in Figure 2.9 (a) and Figure 2.9 (b) respectively.

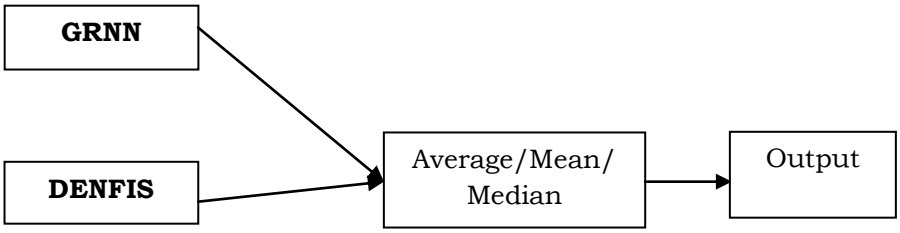

Figure 2.9: a) Generic Design of Linear Ensemble.

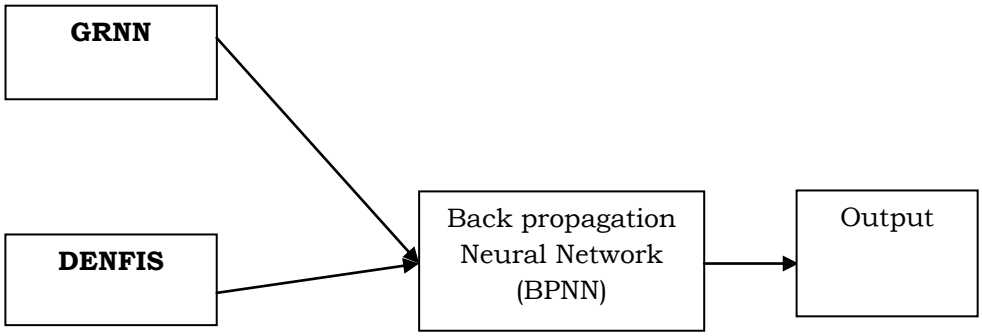

Figure 2.9 b) Generic Design of Non-Linear Ensemble.

Whenever it is finding that the proposed Framework to Requirement Defect Detection by some means unhealthy for identifying a particular defect, Requirement Inspection Participants must review the representation best fit for defect detection. The differences between two successive reliability degrees after defect mitigation may be minimized through proper & concrete introduction of mitigation variables and their implementation. In some cases it may be needed to introduce some additional defect classification, augment the detection processes or sometimes even recommend supplementary mitigation variables in Defect Data Dictionary with respect to specific defect mitigation. [97][98].

(i) Linear ensemble based on average

For each observation, the output values of the individual components are taken as the input to the ensemble and the average of these values is output by the ensemble. This is the simplest kind of ensemble one can imagine. Moreover those components having longer execution time contribute more towards overall system reliability [107].

(ii) Linear ensemble based on weighted mean

In this ensemble, the individual output values are not taken as they are but are given weights based upon certain criteria set by the user. In this case, the criteria of setting the weight ages are based on the mean of the normalized root mean square error (NRMSE) values over the individual lags on the test data. The lower the mean the higher the weight age with the condition that the sum of all the weights is equal to one. This helps in setting the priority towards a technique based on its performance.

(iii) Linear ensemble based on weighted median

It is similar to the linear ensemble based on weighted mean, except that the median of the NRMSE values of the individual techniques on the test data is considered in assigning the weight ages instead of the mean of the values.

(iv) Neural network based non-linear ensemble

Here, no assumptions are made about the input that is given to the ensemble. The output values of the individual techniques are fed into an arbitrator, which is a back propagation neural network (BPNN) which when trained, assigns the weights accordingly. Figure 2.10 shows the generalized structure of Neuro Fuzzy system , the inputs are faults for testing , Input MF is the multi layer feed forward network , Rules are the fuzzy rules (If-then rules) leading to the network and de-fuzzification.

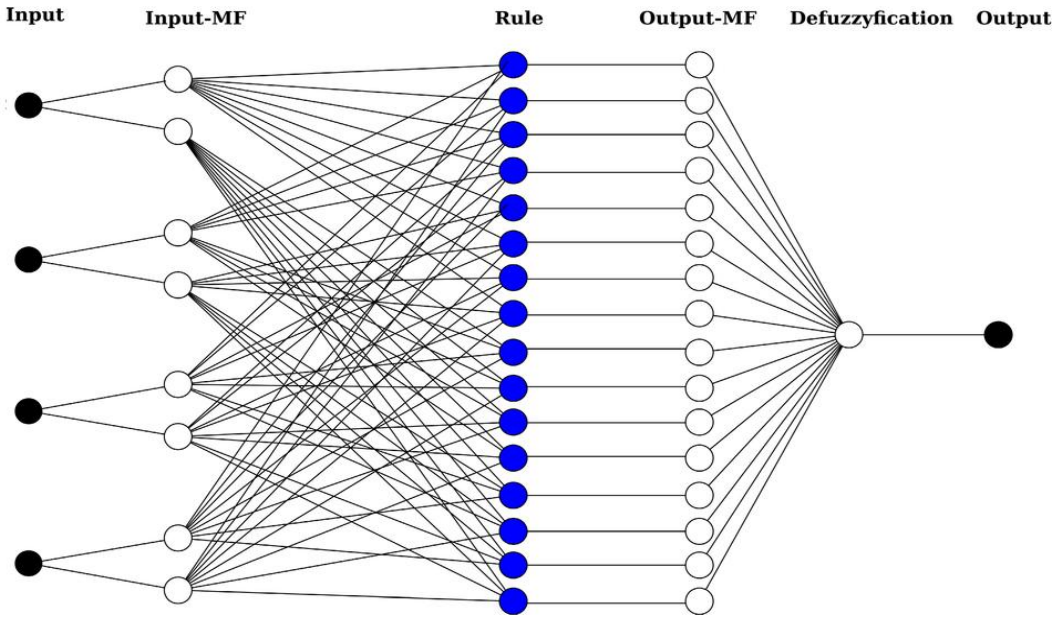

Figure 2.10: Structure of Neuro Fuzzy System

2.7 Linear Model Structure

By using the provided data sets and according to the experimental studies and/or results, it is found that the best Linear model structure which can be developed using Least-Square Regression is the Auto-regression model order 4 shown in Figure 2.11, which will take the 4 inputs (y(k-1), y(k-2), y(k-3), y(k-4)) and process them by using membership functions/activation functions and finally generates a single result(y(k)).

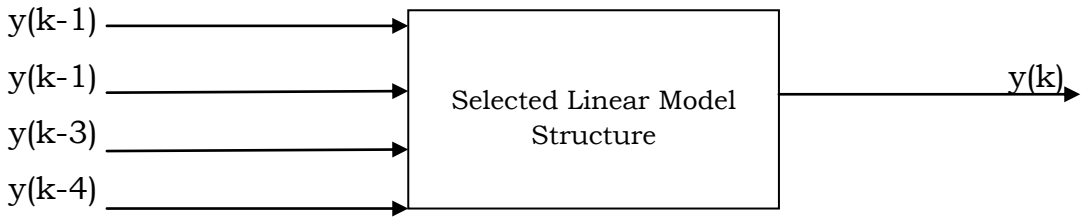

Figure 2.11: Linear Model Structure

It is observed and/or implemented many model structures to predict faults. But, according to experimental results and/or observations, it is found that a regression model of order 4 is the best in the study for assessing the Software Reliability effectively. This will be a foundation for making comparison of different models for assessing their effectiveness in assessing the Software Reliability.

2.8 Previous Whitley's Neural Network Model

- A single-input single-output model is proposed as shown in Figure 2.12. Means only input layer and one output layer for the neural network structure implementation.
- The model input is the execution time t=1,2,..., N. N is the number of tests conducted during the software testing process of the Software Development Life Cycle(SDLC) process.
- During the testing phase, measurements obtained during the execution of the application can be used to refine the architecture–based models further. In order to use the execution information generated during the testing of the application to refine the architectural models, techniques must be developed to record and process the execution information.[106]
- The model output is the number of the observed faults during the different phases of the Software Development Life Cycle (SDLC) process.

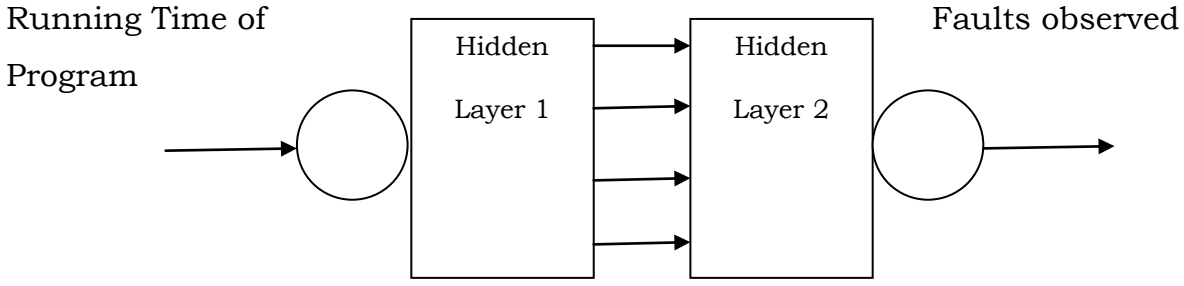

Figure 2.12: Whitley's ANN Model

Whitley's NNs Model is build with the use of traditional model structure in mind which counts on the running time of the program as a metric for predicting software faults at different stages of SDLC in cumulative manner.

2.9 Generalized Regression Neural Network (GRNN)

Specht in 1991 [42] introduced GRNN. It can be thought of as a normalized radial basis function (RBF) network in which there is a hidden unit centered at every training case. These RBF units are called "kernels" and are usually probability density functions such as the Gaussian. The hidden- to-output weights are just the target values, so the output is simply a weighted average of the target values of training cases close to the given input case. The only weights that need to be learned are the widths of the RBF units. These widths (often a single width is used) are called "smoothing parameters" or "bandwidths" and are usually chosen by cross-validation or by more esoteric methods that are not well known in the neural net literature; gradient descent is not used. GRNN is a universal approximate for smooth functions, so it should be able to solve any smooth function-approximation problem given enough data. The main drawback of GRNN is that, like kernel methods in general, it suffers badly from the curse of dimensionality. GRNN cannot ignore irrelevant inputs without major modifications to the basic algorithm.

2.10 Dynamic Evolving Neuro Fuzzy Inference System (DENFIS)

DENFIS is introduced by Kasabov in 2002. DENFIS evolve through incremental, hybrid (supervised/unsupervised) learning, and accommodate new input data, including new features, new classes, etc., through local element tuning. New fuzzy rules are created and updated during the operation of the system. At each time moment, the output of DENFIS is calculated through a Fuzzy Inference System based on most activated fuzzy rules, which are dynamically chosen from a fuzzy rule set. A set of fuzzy rules can be

inserted into DENFIS before or during its learning process. Fuzzy rules can also be extracted during or after the learning process.

2.11 Relevant Findings

From the above discussion, the researcher has found the following findings:

- The literature available discusses about analysis of traditional Software Reliability Growth Models.
- The relationship between the Neural Networks, Fuzzy Systems and Neuro Fuzzy systems with regards to the Software Reliability is also been discussed.
- Different Neuro Fuzzy models for the assessment of Software Reliability are studied and analysed.
- Mathematical analysis for the implementation of these models is also observed and discussed.
- The survey shows that there is no work done about the estimation of Software Reliability using Neuro Fuzzy system and that too based on sigmoid membership function at the hidden layer of neural network.
- The reliability estimation during design phase is either missing or ignored.
- On the basis of extensive survey, it can be said that the Software Reliability can be affected positively when the availability and MTBF are affected.

2.12 Conclusion

A brief explanation regarding the Neural Networks and Neuro Fuzzy systems is presented in this chapter. From the above discussion the study can say that the Neuro Fuzzy system is the advanced way of mathematical tool for the Software Reliability approximation. This chapter also discussed about the relation between the Software Reliability Growth Models and its relation with Neuro models and Neuro Fuzzy models. It is shown clearly how to model

Software Reliability based on traditional Software Reliability Growth Models and conventional Fuzzy Inference Systems(FIS).

A detailed observation and analysis done also on the Fuzzy Logic and its importance in dealing with multivalued logic based data for modeling Software Reliability. Fuzzy rules and different fuzzy membership functions are discussed to map them for the use of effective assessment of Software Reliability. Also, the detailed survey is done on the factors of Software Reliability to decide about the factors to be used for the assessment by using the proposed approach. The factors like MTTF, MTTR, MTBF, availability are studied in detail to establish the hybrid relationship between MTBF and availability for use in the proposed approach for the effective assessment of Software Reliability.

3.1 Background

To overcome the problems identified in literature survey, the research will use machine learning techniques [158] like Neural Networks, fuzzy-logic approaches for the effective assessment of the Software Reliability after the extensive survey of existing machine learing techniques for the improvement of Software Reliability [74][87][91]. In this research, a combined advantage of theoretical aspects of Artificial Neural Networks and Fuzzy Inference Systems for Software Reliability estimation and modeling is discussed and shown that it is a promising technique in all aspects when compared to all other existing techniques for assessing the Software Reliability [37].

Artificial Neural Networks approaches found to be useful in many applications. For example:

- Prediction purpose in the fields like weather forecasting.
- System Identification and Control mechanism implementation purpose.
- Computer Network Design and its implementation purpose.
- Image Processing and many others.

Whitley stated that," The influence of the external parameters and other peculiarities of a Software Reliability growth model(SRGM) can be eliminated if the study can have a system that can develop its own model from the past failure history of the software system" which can be developed very easily by using Artificial Neural Networks and Fuzzy Logic based systems. The research proposes an Artificial Neural Network based approach with Fuzzy membership functions as an activation functions for Software Reliability estimation and modeling in the design phase of Software Development Life Cycle (SDLC). A comparative study is done to assess the performance of some well known existing conventional FIS based Software Reliability Growth Models against the approach developed in this research from the three aspects: goodness of fit, prediction ability for short-term prediction and long-term prediction. The approach has been depicted in Figure 3.1. This is the methodology which is adopted in the research.

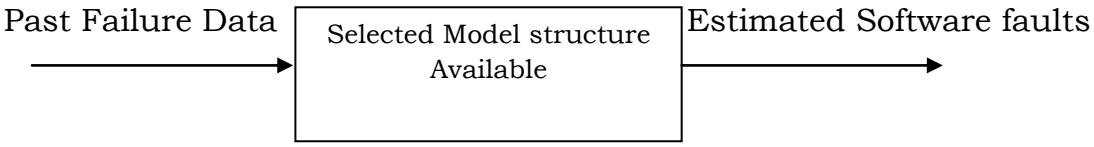

Figure 3.1: Artificial Neural Network based Approach for Assessing the Software Reliability

Where the activation function/processing function can be defined as

$$y(k) = f(y(k-1), y(k-1), \ldots\ldots\ldots y(k-m)$$

Where y(k) is the accumulated faults at given instance of time 'k' and 'm' is the selected number of delays of the chosen model.

3.2 Proposed Approach for Reliability Assessment

After observed different issues in predicting the Software Reliabity [75][90] and also different levels of severity faced by oraganizations in the case of reliability issues [80], An attempt has been made to implement a model based on the Neuro Fuzzy System. Here, the complex problem of Software Reliability modeling can be split to number of sub-problems:

- Selection of the Software Reliability model structure.
- Estimation of the Software Reliability model parameters like MTTF [78] (Mean- Time-To-Failure) and Mean- Time- To-Repair (MTTR) , MTBF and availability etc.,
- Evaluation of the model prediction capabilities.

The prelimilinary prediction of Software Reliability prediction using Neural Network Based Systems and the comparison of Software Reliability Assessment Methods [156] with Neuro Fuzzy Based Systems [84][85][89] was observed by the researcher to arrive at the conclusion for the development of the proposed approach [83][88]. Figure 3.3 shows the proposed model of the research analysis, where the parameters concerned with the reliability assessment are given as the inputs for network. Weights are been added at the consequent layer ad using the predefined rules a decision is obtained at the validation.

Based upon the outcome of validation the assessment will be finalized. A generalized block diagram of the proposed approach is shown in the Figure 3.3. In this research work a Neuro Fuzzy interference model is designed for the assessment of reliability of a software growth model, the algorithm mainly focuses on MTBF and Availability which is analyzed and calculated theoretically and practically. Fuzzy rules employed for the proposed model are:

- *If MTBF (Mean time Between Failure) >0.8 & availability >0.8 then reliability is very high*
- *If 0.7<MTBF <0.8 & 0.7< availability <0.8 then reliability is high*
- *If 0.6<MTBF <0.7 & 0.6<availability <0.7 then reliability is moderate*
- *If 0.5<MTBF <0.6 & 0.5< availability <0.6 then reliability is low*
- *If 0.4<MTBF <0.3 & 0.4<availability <0.3 then reliability is very low*

In order to assess the reliability of software during design phase using Neuro Fuzzy Logic, the model is divided into five phases. These phases are as follows:

3.2.1 Identification Phase

The objectives of identification phase are:

1. To identify the reliability factors.
2. To evaluate a mathematical analysis for the approximation constraints.

3.2.2 Quantification Phase

The objectives of the quantification phase are:

1. To identify the how the reliability factors are linked with assessment of reliability and quantify them.
2. To evaluate a mathematical analysis for the relationship.

3.2.3 Measurement Phase

The objectives of the measurement phase are

1. To assess the metrics for the estimation of reliability.
2. To conduct the experiments for the generation of results.

3.2.4 Verification and Validation Phase

The objectives of verification and validation phase are

1. To verify the model's effectiveness in the assessment of software reliability.
2. To compare and validate the reliability model with conventional models.

3.2.5 Finalization Phase

The objectives of finalization phase are

1. To incorporate the changes and suggestions.
2. To finalize the metrics and model for assessment.

3.2.6 Review & Revisions

There is a review and revision of output of every phase for the refinement of the study. The aim of this phase is to check whether the objectives of each phase are fulfilled or not. The review and review process is depicted in Figure 3.2.

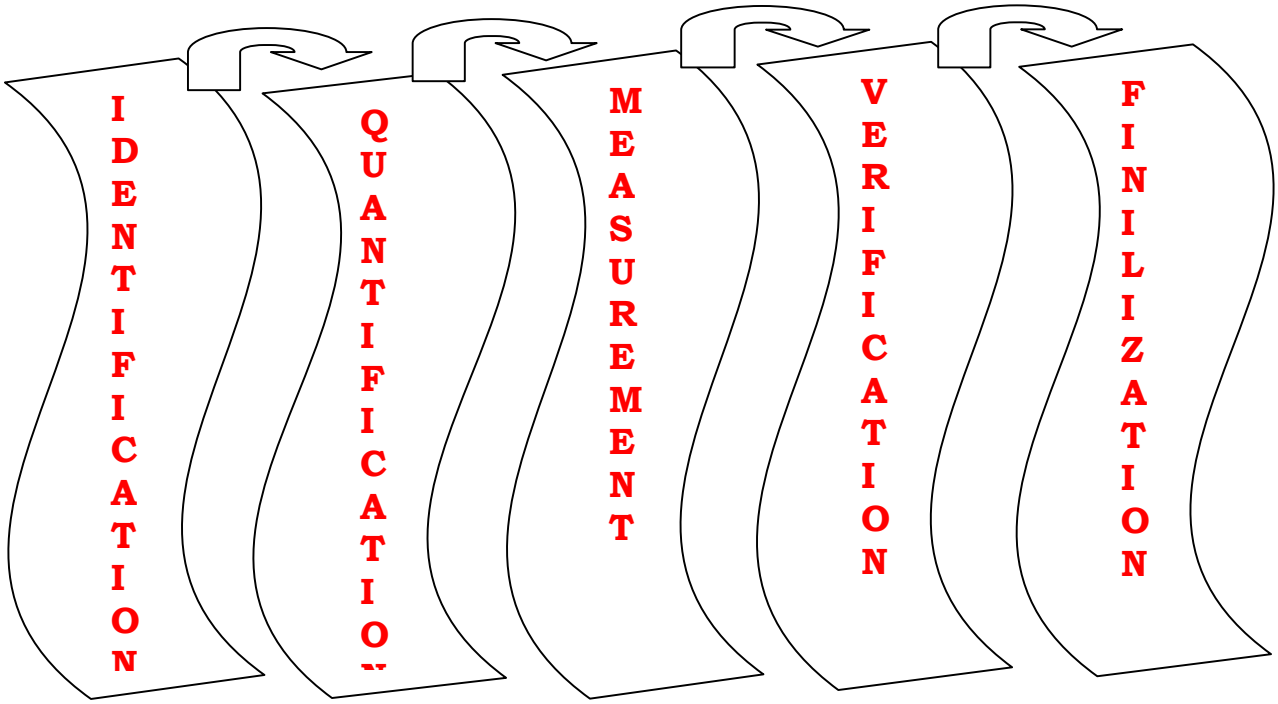

Figure 3.2: Process Flow of the Proposed Approach

3.3 Implementation of the Proposed Approach

The Figure 3.3 shows the steps to be followed for implementation of the approach [79]. The values of 2 parameters Normalized MTBF and availability are given as input at the input layer of Neural Network.Sigmoid fuzzy membership function at the hidden layer of neural network is applied and the Software Reliability approximated value is found out. During validation, the values assessed using conventional FIS system and the value obtained from the proposed Neuro Fuzzy systems based model will be compared against two evaluation criteria like MSE (Mean Squared Error) and Average Error (AE).

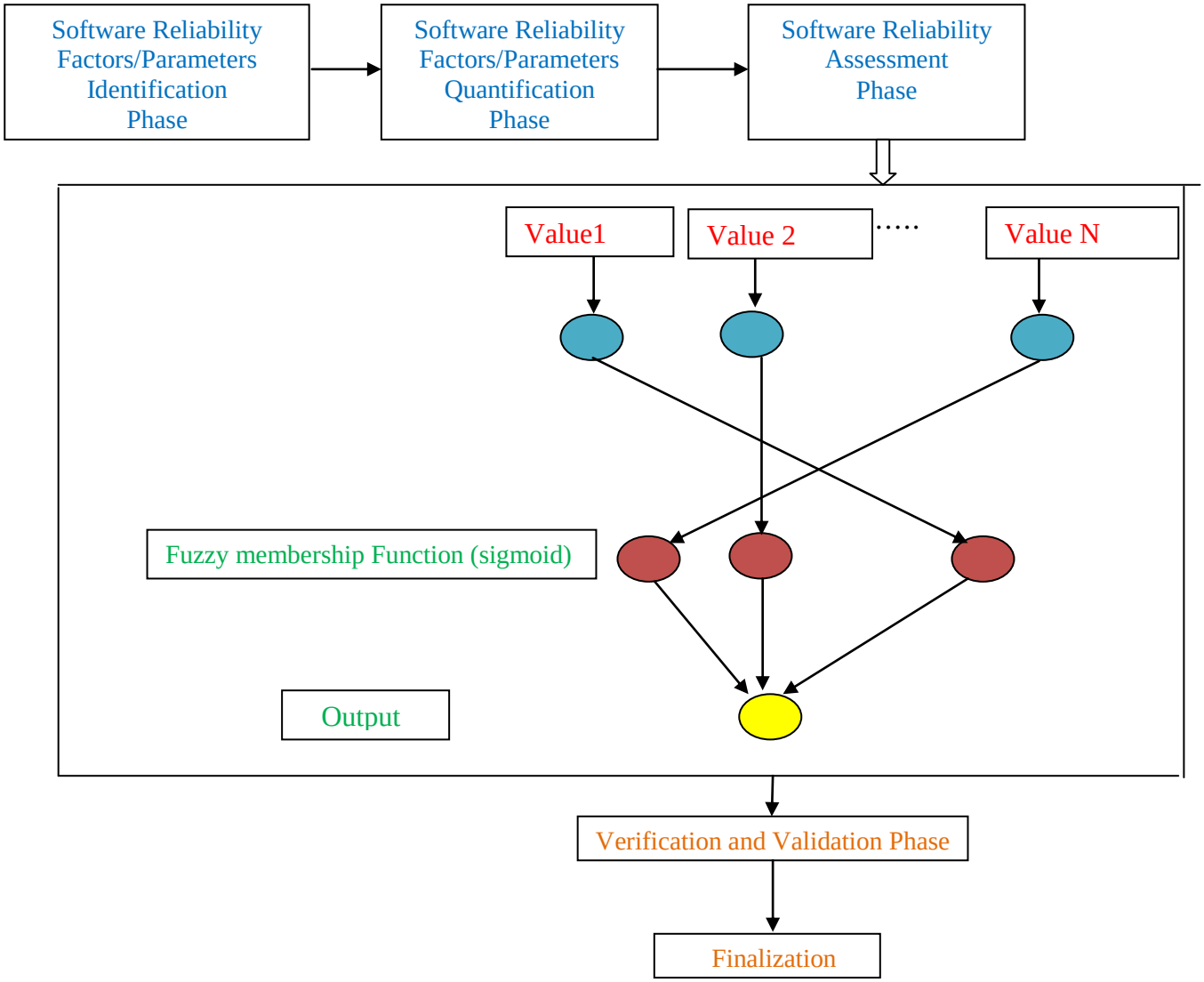

Figure 3.3: Proposed Neuro Fuzzy based Model of Software Reliability Assessment

3.4 Mathematical Analysis

A detailed study was done by the researcher about how we can mathematically implement, validate the proposed approach [157]. In this research sigmoid based fuzzy membership function is used for assessment.

A sigmoid MF is defined as

$$f(x,a,c) = \frac{1}{1+e^{-a(x-c)}}$$

Depending on the sign of the parameter *a*, the sigmoid membership function is inherently open to the right or to the left, and thus is appropriate for representing concepts such as "very large" or "very negative." More conventional-looking membership functions can be built by taking either the product or difference of two different sigmoid membership functions. In the context of the proposed approach the exponential relation between the assessed Software Reliability value and the membership function is shown in the Figure 3.4. The proposed model facilitates the results for multi stage process. Consider that there are three types of faults low, moderate, high then the membership function can be written as

$$m(t) = aq_1\left[1-\exp\left(-b_1 w(t)\right)\right] + aq_2\left[1-\left(1+b_2 w(t)\right)-\exp\left(-b_2 w(t)\right)\right]$$

$$+\square_3\left[1+{}_3() + \frac{({}_3())^2}{2!}\right)\exp\left(-{}_3()\right)\Big]$$

b1= [.2 .3 .4], b2=[.3 .4 .5] , b3=[.4 .5 .6]

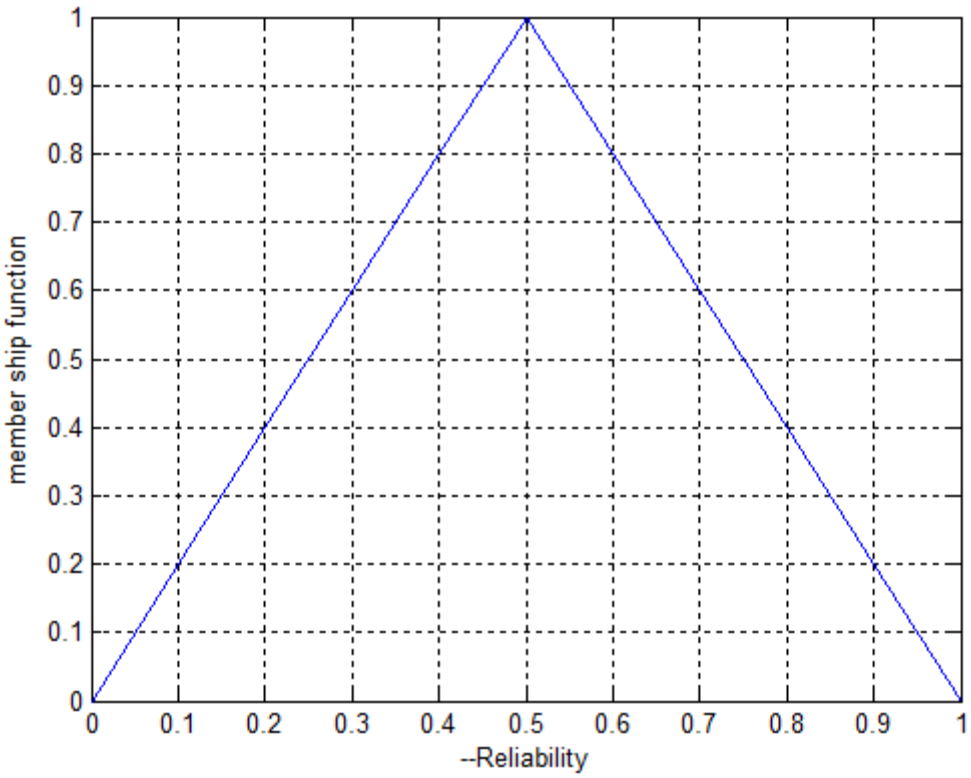

Figure 3.4: The Membership Function of the Proposed Model

The profile of the proposed model is shown in Figure 3.5.

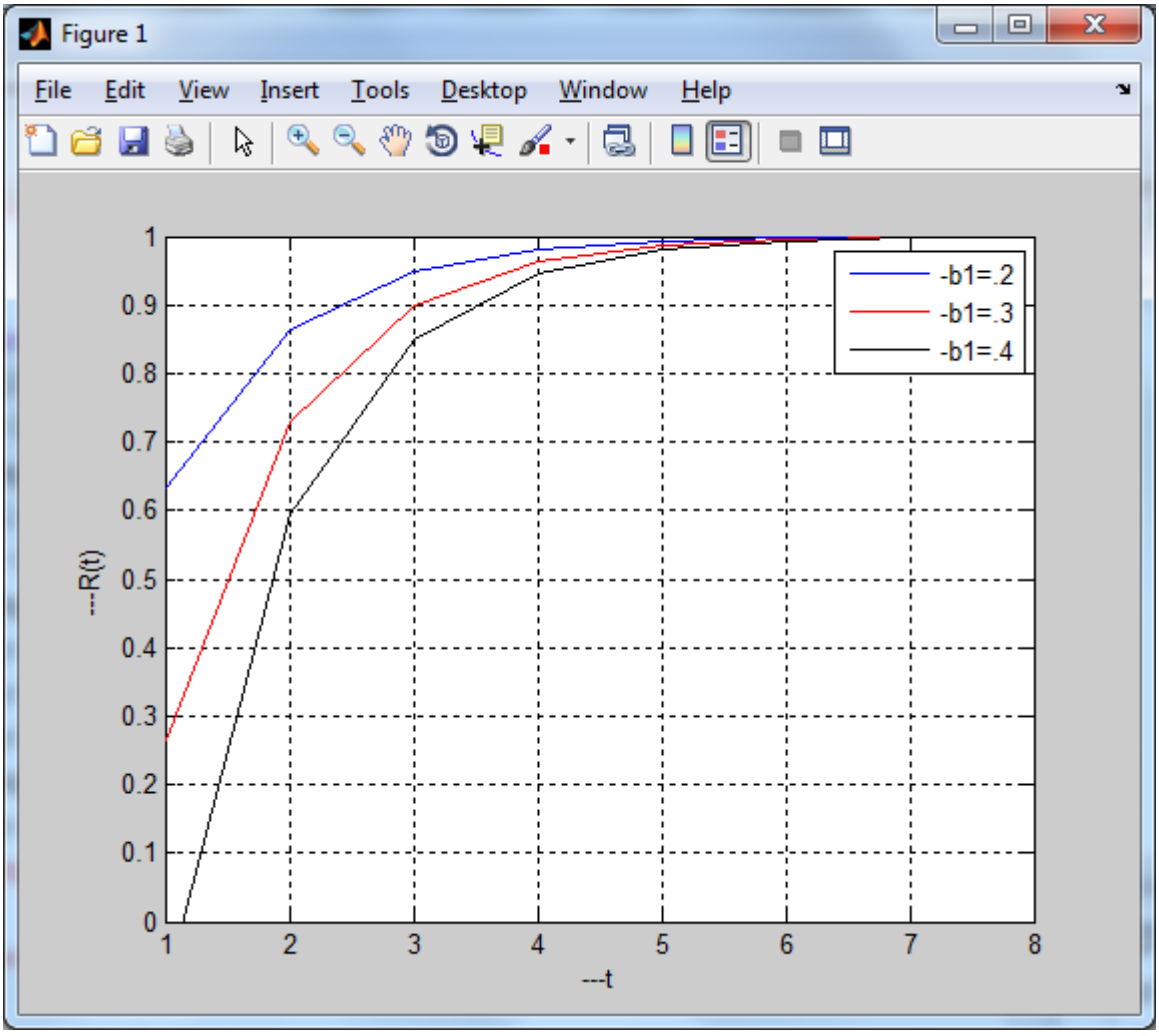

Figure 3.5: Profile of the Proposed Model

Mathematical approximation of proposal model metrics

For approximating the values of the proposed model metrics, a quantitative approach is adopted for calculating the appropriate results. The formula that has been used to calculate approximated values is defined as:

Formula: Ca (x_i) = C (a) - h X f(a), based on Euler's theorem

Where, C (a) = Set of Measured values.

'h' can be derived by,

$x_1 + x_0$ n h

Where, n= no. of values in the dataset. $x_0 = 0$ and $x_1 = 1$ (since the probability ranges from 0 to 1). Here 'x' is MTBF. f(a) can be function, denoted as,

f(a)=MTBF/(1+MTBF)

Ca (x_i) is the set of values to be approximated.

Procedure for 'h' Calculation

Let us take, $x_0 = 0$ and $x_1 = 1$ then, $1 = 0 + 17 * h$

$h = 1/17 = 0.058$

According to [146], Iterations are Performed at least 5 to 10 iterations to arrive at good approximated Software Reliability value. At every iteration, to calculate % of Reliability, use the following formula

% of Reliability = (Average of Approximated values)/ (Average of Measured values) * 100

At final iteration, if the approximated value falls above 99.00%, then the study can say that it is good approximation.

3.5 Conclusion

This chapter discussed clearly about the propsed model architecture, the steps involved in the model for the assessment and given a mathematical analysis for the calculations that are involved in the assessment. As it is stated there are five phases in the developed model; Identification, Quantification, Measurement, Verification and Finalization. Also, this chapter focuses on fuzzy modeling and Neuro Fuzzy modeling, relation between the findings and the approach used in the models. It also provides the mathematical analysis of the proposed approach of Neuro Fuzzy model for Software Reliability Assessment.

To develop the proposed approach, the detailed analysis on how to impement a model by using Neural Networks was done by the researcher [92]. The review and revision is also done at every phase of the proposed approach for the effective assessment of Software Reliability. The development of the

model goes in sequential in the form of above said five phases and the mathematican Euler's theorem is used as a base for getting the accurate approximated value of Software Reliability. Hybrid factors like normalized MTBF and availability are identified as key parameters for assessing the Software Reliability using the proposed approach. The refinement is done at every phase during review and revision throughout the study.

4.1 Background

Verification is the process of determining whether the output of one phase of Software Development confirms to that of its previous phase. Validation is the process of determining whether a fully developed system/model/framework confirms to its requirement specification or not. It is also carried out to analyze the validity of model as a whole and phase wise. Validation activities need to be done both theoretically and practically. In the thesis, validation is done to check about the functionality of the proposed model with respect to the given dataset. Validation also helps to find out whether the proposed model offers more advantages than the traditional models. An analysis of Software Reliabilty assessment based on Neuro Fuzzy Systems to arrive at the conclusion about how to perform validation process [77].

Metrics have always been used to help managers with decision about the working model. Software metrics is a quantitative measure of a degree to which a software system or process possess some property. In this chapter a practical calculations are done with the approximation discussed in section 3.4. This practical value consists of 17 readings taken from health software. Metrics like MTBF, MTRF, and availability are used to evaluate the percentage of reliability. Figure 4.1 shows the practical implementation of the FIS model in MATLAB software [151] tool using FIS. The NF system is trained using a hybrid learning algorithm using both least squares method and back propagation algorithm. In the forward pass the consequent parameters are identified using least squares and in the backward pass the premise parameters are identified using back propagation [12]. A Fuzzy Inference System (FIS) can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value. The derivation of if-then rules and corresponding membership functions depends heavily on the a priori knowledge about the system under consideration [138]. The trained NF system is then tested for the 17 inputs and the results are comparaed against test data and FIS output nand comparison plot is shown in Figure 4.2.

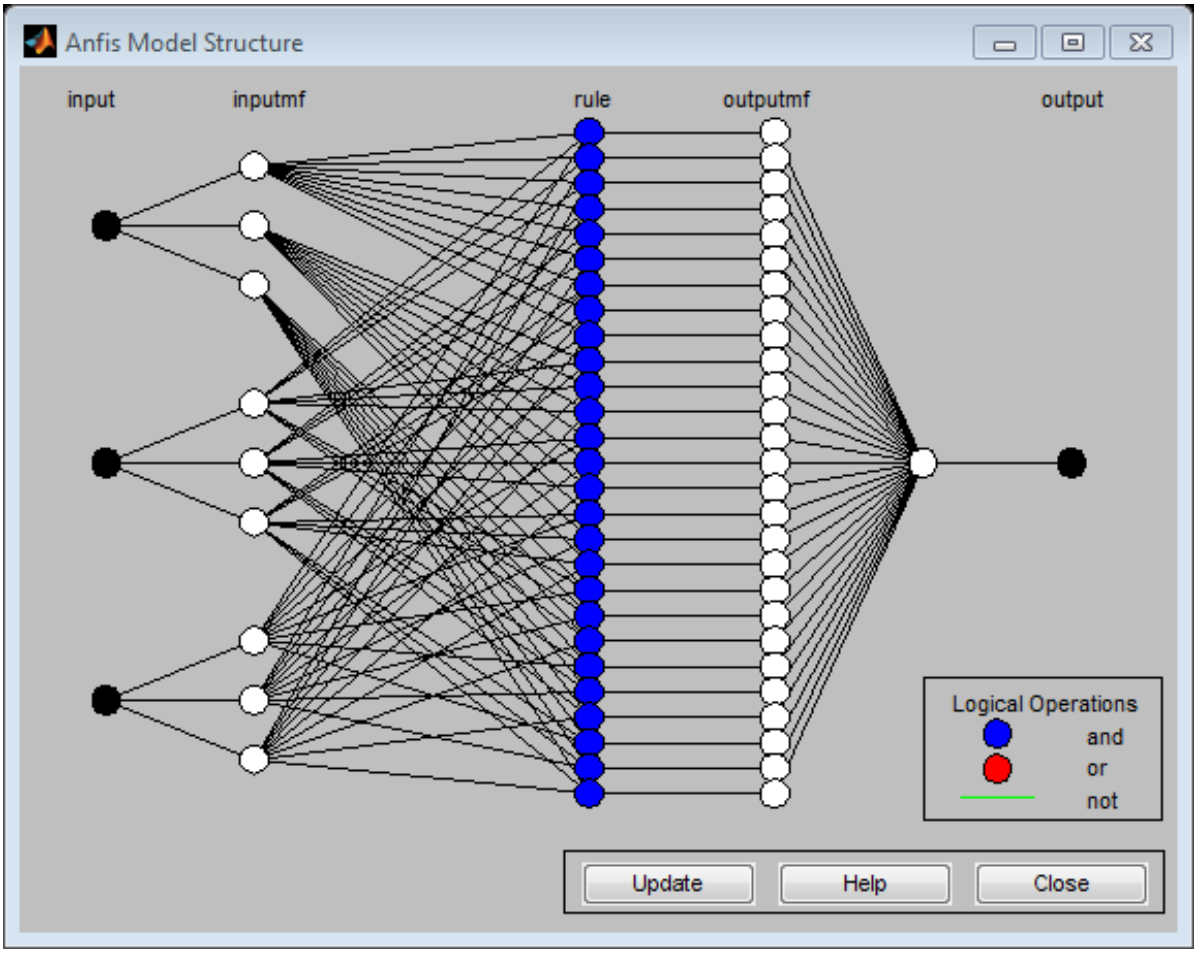

Figure 4.1: Real time Design of Neuro Fuzzy Structure

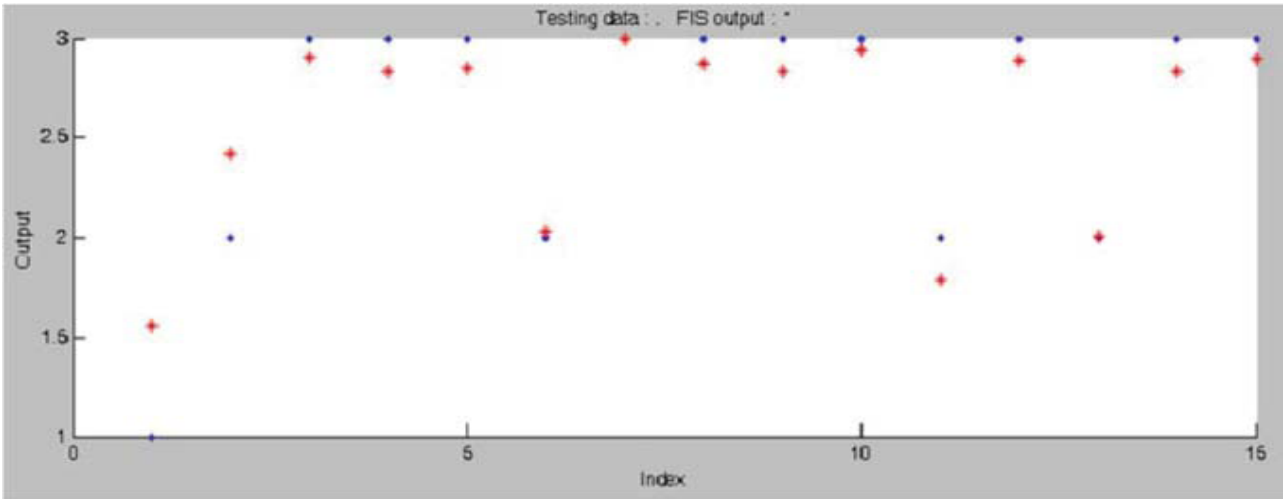

Figure 4.2: Test Data Vs FIS Output

It shows 0.1571, 0.2140 as NRMSE, RMSE values respectively. The plot of the expected and the output of the NF system for the different inputs are shown in Figure 4.2.

4.2 Data Collection

To validate the proposed model, 17 programs of Glace EMR Medical Billing Software are taken (on which researcher has worked previously as a Software Engineer at L Cube Innovative Solutions Pvt. Ltd.). The MTTF (Mean Time To Failure), MTTR (Mean Time To Repair), MTBR (Mean Time Between Repair) and Software Reliability Approximated value based on the program execution observations are calculated. The 3 values were put to the input layer of Neural Network. Sigmoid fuzzy membership function at the hidden layer of neural network was applied and the Software Reliability approximated value was found out. The previous values assessed using conventional FIS traditional Software Reliability Growth Models and the proposed Neuro Fuzzy systems based model is compared. It was found that the proposed model is the promising one. The amount an element is in a set is measured with a membership function. Membership functions range from 0 to 1. Membership functions are used to describe linguistic terms such as low, medium and high. There are various types of fuzzy membership functions such as triangular, trapezoidal, and Gaussian [137].

The dataset contains failure observations of 17 programs in Glace EMR Billing Software, in time series (i, Xi) and is used to predict the performance of the proposed model. Where, i = serial number of the program. The failure interval dataset with the purpose of helping software managers to monitor test status, predict schedules and help researchers to validate Software Reliability models are collected. The models are applicable to the area of Software Reliability engineering. Table 4.1 shows all the 17 projects and the information recorded. The data represents a variety of applications in Glace EMR Billing and is recorded in the 2013. The application types are Patient Registration,

Service Entry, Reports, Online Patient Insurance Verification applications. The attributes recorded for each software are Software Code, Type of Application, Size of Software (in Lines of Code (LOC)), Number of Failures.

Table 4.1: Software Reliability Data Project Information

Software Code	Type of Application	Size(LOC)	No. of Failures
GE01	Patient Registration	22,300	14
GE02	Patient Registration	10,500	25
GE03	Patient Registration	9,800	21
GE04	Patient Registration	31,870	39
GE05	Patient Registration	12,400	48
GE06	Service Entry	4,870	36
GE07	Service Entry	26,490	35
GE08	Service Entry	23,400	35
GE09	Service Entry	21,700	45
GE10	Reports	10,890	10
GE11	Reports	28,740	58
GE12	Reports	36,350	54
GE13	Online Patient Insurance Verification	61,800	32
GE14	Online Patient Insurance Verification	34,700	21
GE15	Online Patient Insurance Verification	39,800	52
GE16	Online Patient Insurance Verification	43,200	58
GE17	Online Patient Insurance Verification	44,600	56

4.3 Parameters used for Measurement and Validation

Reliability can be defined as the probability of failure free operation under stated conditions for specific period of time [70]. The researchers identified the parameters based on two observations through a revisit of hybrid combination of the parameters used for the assessment of Software Reliability [76]. Assessment of reliability performance for a component are usually defined for the expected input profile in actual operational use. The commonly used metric for assessment are, Mean time to time failure (MTTF), Mean Time Between Failures (MTBF) and robustness [71].In [72] storey has given the definition as a function of time *R(t)* at a constant failure rate of λ.

$$R(t) = e^{-\lambda t}$$

Where λ is the probability that there is no failure before time t

Then the MTTF can be given as

$$MTTF = \frac{1}{\lambda}$$

And $MTBF = MTTF + MTTR$

Where MTTR is the mean time of recovery defined as the average time a component takes to recover from a failure. The measures MTBF, MTTF and MTTR are usually considered to apply in the case of a system operating continuously; however for a system operating on demand as is the case here, equivalent definitions apply where time is treated in discrete units [7]. Reliability can be defined as the probability of failure free operation under stated conditions for specific period of time [70].Assessment of reliability performance for a component are usually defined for the expected input profile in actual operational use. Software Reliability is measured in terms of Mean Time Between Failures (MTBF). MTBF consists of Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR). MTTF is the difference of time between two consecutive failures and MTTR is the time required to fix the failure.

MTTF = Average time between 2 observed failures. i.e., average time it takes for a system to fail

MTBF = Average time between consecutive software system failures =MTTF+MTTR

MTTR = Average time taken to repair the system after the occurrence of failure.

Time intervals in the assessment of Software Reliability: The three types of time interval is taken for the assessment of reliability. The intervals are as follows:

1. **MTTF: Mean Time To Failure.**
2. **MTTR: Mean Time To Repair.**
3. **MTBF: Mean Time Between Failures.**

The relationship between MTTF, MTTR, MTBF is shown in Figure 4.3.

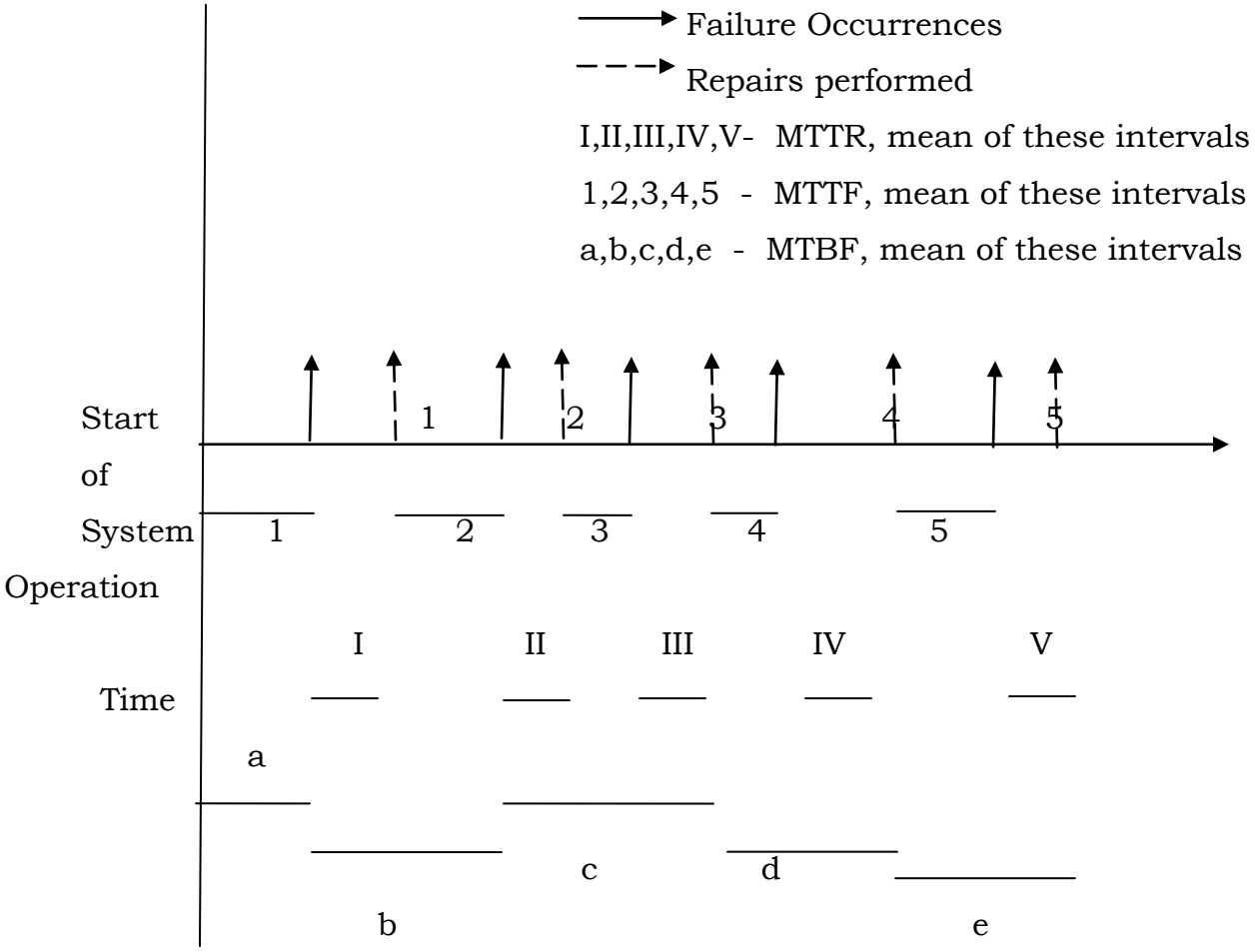

Figure 4.3: Relationship between MTTF, MTTR, MTBF

Methods to Assess Software Reliability

There are two methods to assess the software reliability. The methods are as follows:

- Counting failures at periodic intervals of time($\mu(\tau)$) : Observe the way of cumulative failure count = Total Number of failures observed until execution.

  τ - Time from start of system execution.

- Based on Failure Density($\lambda(\tau)$):

  Observe the trend of number of failures per unit time. i.e., no. of failures observed per unit time after 'τ' time units from the staring stage of the system execution, which can be called as failure intensity at time, 'τ'.

For the research, the researcher has used the "Counting failures at periodic intervals of time($\mu(\tau)$)" method. According to this method, the failures are observed at two intervals of time 'x1' and 'x2'. The value of different parameters is calculated as:

Software Reliability = MTBF / (1+MTBF).

Normalized MTBF = (sum of MTBF values observed)/ number of readings.

Availability = MTBF/(MTBF+MTTR) , is the likelihood that a software system will work at a given time.

In the study, the parameters are a combination of normalized MTBF and availability to arrive at the effective assessment of Software Reliability using the proposed approach.

4.4 Parameters Estimation and Experimental Results

The experiments are conducted to prove that the proposed approach gives better results than the conventional Fuzzy Inference System (FIS) and other traditional Software Reliability assessment models. The MATLAB software has been used as a base for conducting the experiments.

4.4.1 Paramers Estimation

The calculations of normalized Mean Time Between Failures (MTBF) and availability are done mathematically and the results are shown in Table 4.3 and table 4.4 respectively. Then, the assessed values are given as input to Neuro Fuzzy Systems based approach for the better assessment of Software Reliability. Based on the experiments conducted and the results observed, the researcher concludes that the proposed approach perform well. To conduct the experiments the dataset is recorded based on the factors like total production time, uptime, downtime, No. of breakdowns at two intervals of time 'x1' and 'x2' and the recorded values are shown in Table 4.2.

Table 4.2: Production Time Analysis for the Program Dataset

S.No	Program #	Total Production time(Hrs.)	Uptime at x1(Hrs.)	Uptime at x2(Hrs.)	Downtime at x1(Hrs.)	Downtime at x2(Hrs.)	No. of breakdowns at x1(Hrs.)	No. of breakdowns at x2(Hrs.)
1	GE01	256	216	202	40	54	3	11
2	GE02	324	260	203	64	121	9	16
3	GE03	236	168	154	68	82	2	19
4	GE04	600	450	435	150	165	16	23
5	GE05	371	300	265	71	106	13	35
6	GE06	447	430	410	17	37	15	21
7	GE07	865	560	525	305	340	10	25
8	GE08	843	615	575	228	268	4	31
9	GE09	943	720	706	223	237	17	28
10	GE10	135	85	78	50	57	4	6
11	GE11	242	130	132	112	110	36	22
12	GE12	369	240	206	129	163	24	30
13	GE13	122	68	64	54	58	23	9
14	GE14	107	72	74	35	33	6	15
15	GE15	371	265	253	106	118	18	34
16	GE16	453	370	398	83	55	21	37
17	GE17	325	285	256	40	69	27	29

Calculations

For all observations, the following parameters can be calculated as follows:

Total Production time= Uptime+ down time

$$\textbf{MTBF} = \frac{\text{Total uptime (total time- total downtime)}}{\text{Number of Breakdowns}} \quad \text{for all observations}$$

Where,

MTTF= Mean Time To Failure (in hours/minutes/seconds).

MTTR= Mean Time To Repair (in hours/minutes/seconds).

MTBF= Mean Time Between Failures (in hours/minutes/seconds).

$$\text{MTTR} = \frac{\text{Total downtime}}{\text{Number of breakdowns}} \quad \text{for all observations}$$

$$\text{MTTF} = \frac{(\text{Failure at obs.1} + \text{Failure at obs.2} + \ldots + \text{Failure at obs.N})}{\text{Number of software programs under test}}$$

$$\text{Availability (For Repairable software systems)} = \frac{\text{MTBF}}{(\text{MTBF} + \text{MTTR})}$$

(or)

$$\text{For non-repairable, hardware systems} = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})}$$

A new automated java program for Software Reliability calculation, Approximation based on Euler's mathematical theorem [159] is presented in **Appendix-B.**

Table 4.3: Calculation of MTBF & MTTR

S.No.	Program	MTTF	MTTR	MTBF
1	GE01	0	9.12	45.18
2	GE02	0	7.336	20.78
3	GE03	0	19.158	46.05
4	GE04	0	8.25	23.51
5	GE05	0	4.25	15.32
6	GE06	0	1.447	24.09
7	GE07	0	22.05	38.5
8	GE08	0	32.82	86.14
9	GE09	0	10.791	33.784
10	GE10	0	11	17.12
11	GE11	0	4.056	4.80
12	GE12	0	5.042	8.43
13	GE13	0	4.396	5.03
14	GE14	0	4.016	8.46
15	GE15	0	4.679	11.08
16	GE16	0	2.719	14.18
17	GE17	0	1.93	9.69

Table 4.4: Calculation of Availability

S.No.	Program	MTTR	MTBF	Availability
1	GE01	9.12	45.18	0.832
2	GE02	7.336	20.78	0.739
3	GE03	19.158	46.05	0.706
4	GE04	8.25	23.51	0.739
5	GE05	4.25	15.32	0.783
6	GE06	1.447	24.09	0.943
7	GE07	22.05	38.5	0.635
8	GE08	32.82	86.14	0.724
9	GE09	10.791	33.784	0.757
10	GE10	11	17.12	0.608
11	GE11	4.056	4.80	0.543
12	GE12	5.042	8.43	0.609
13	GE13	4.396	5.03	0.533
14	GE14	4.016	8.46	0.678
15	GE15	4.679	11.08	0.703
16	GE16	2.719	14.18	0.839
17	GE17	1.93	9.69	0.833

The ratio of Availability, MTTR and MTBF w.r.t. number of programs in the dataset is represented in Figure 4.4, Figure 4.5, Figure 4.6 respectively.

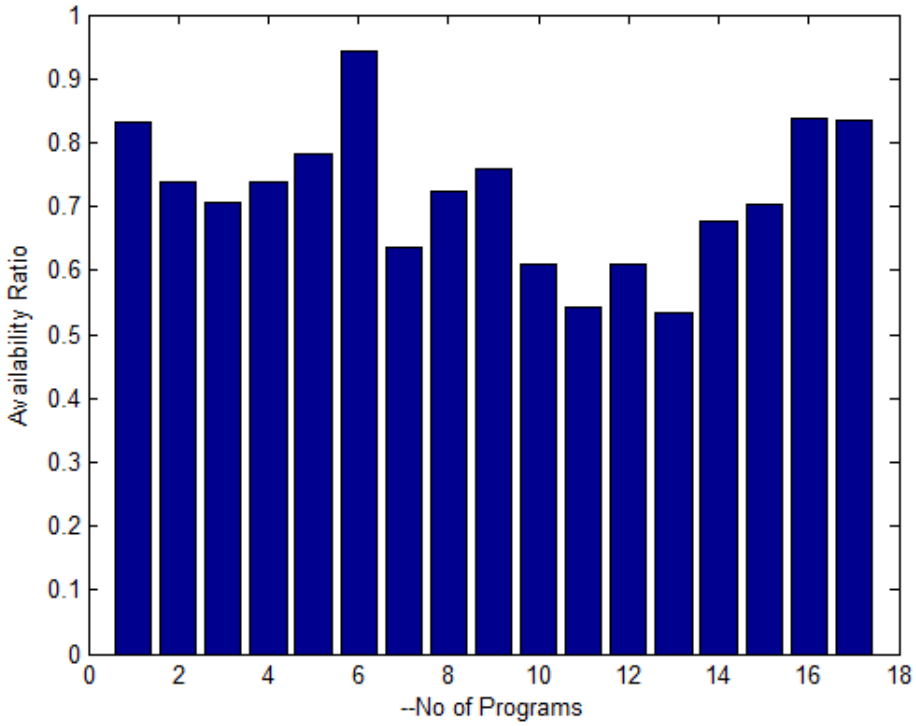

Figure 4.4: Analysis of Availability Ratio w.r.t. Number of Programs

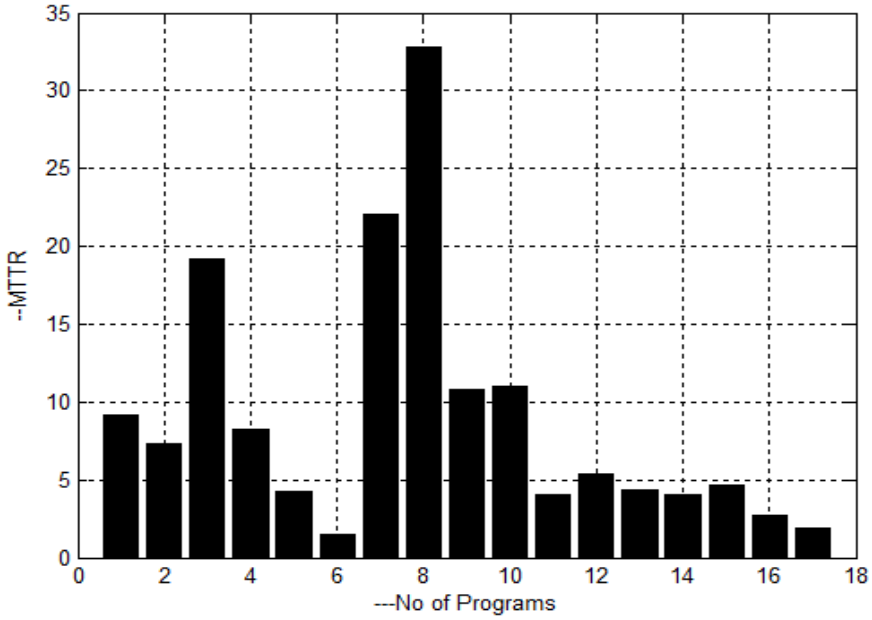

Figure 4.5: Analysis of MTTR Ratio w.r.t. Number of Programs

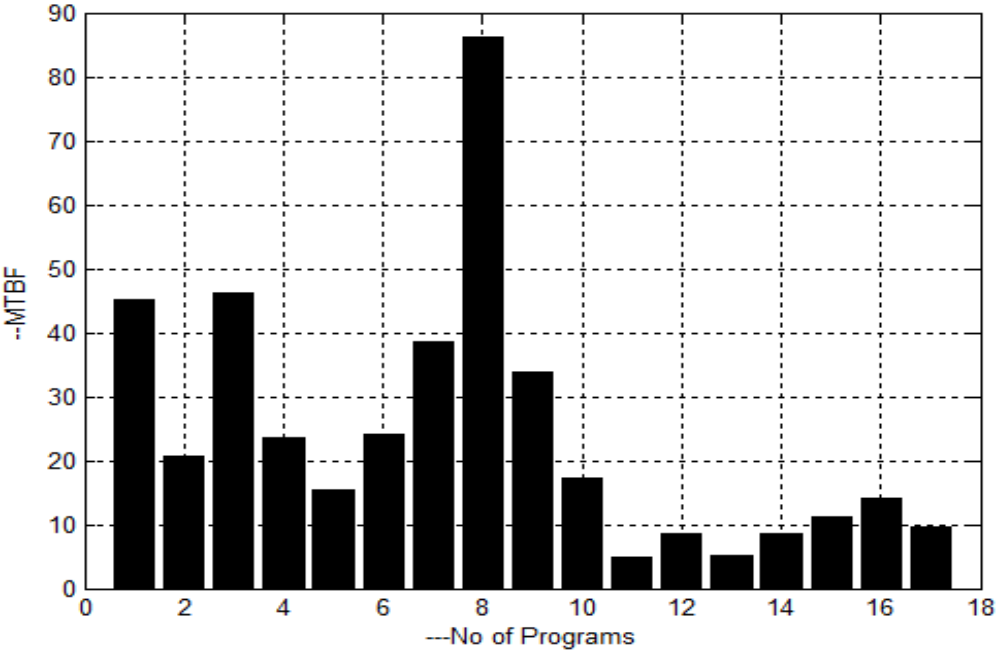

Figure 4.6: Analysis of MTBF Ratio w.r.t. Number of Programs

4.4.2 Implementation of the Proposed Approach through MATLAB

The experiment is conducted with the 17 programs of Glace EMR Medical Billing. The analysis is done using FIS (fuzzy interference system) and the proposed Neuro Fuzzy model in MATLAB environemt. The Fuzzy Inference Systems based model is presented with the parameters Normalized MTBF and availability in Figure 4.7. The generated membership function based on employed fuzzy IF-THEN rules ranging from “very low” to “very high” are shown in Figure 4.8. The dataset will be given as training data to ANFIS systems and the comparison between ANFIS output and FIS output is shown in Figure 4.9. The sequence of the steps from feeding of the input to generation of output at different layers of Neuro Fuzzy System like input, processing of input through membership function and Fuzzy IF-THEN rules, the generation of output is shown in Figure 4.10. The generated error after learning of inputs through training is based on number of epochs in Figure 4.11. Finally, the performance level of the proposed approach for conversion of input space to output space is depicted as a surface in Figure 4.12.

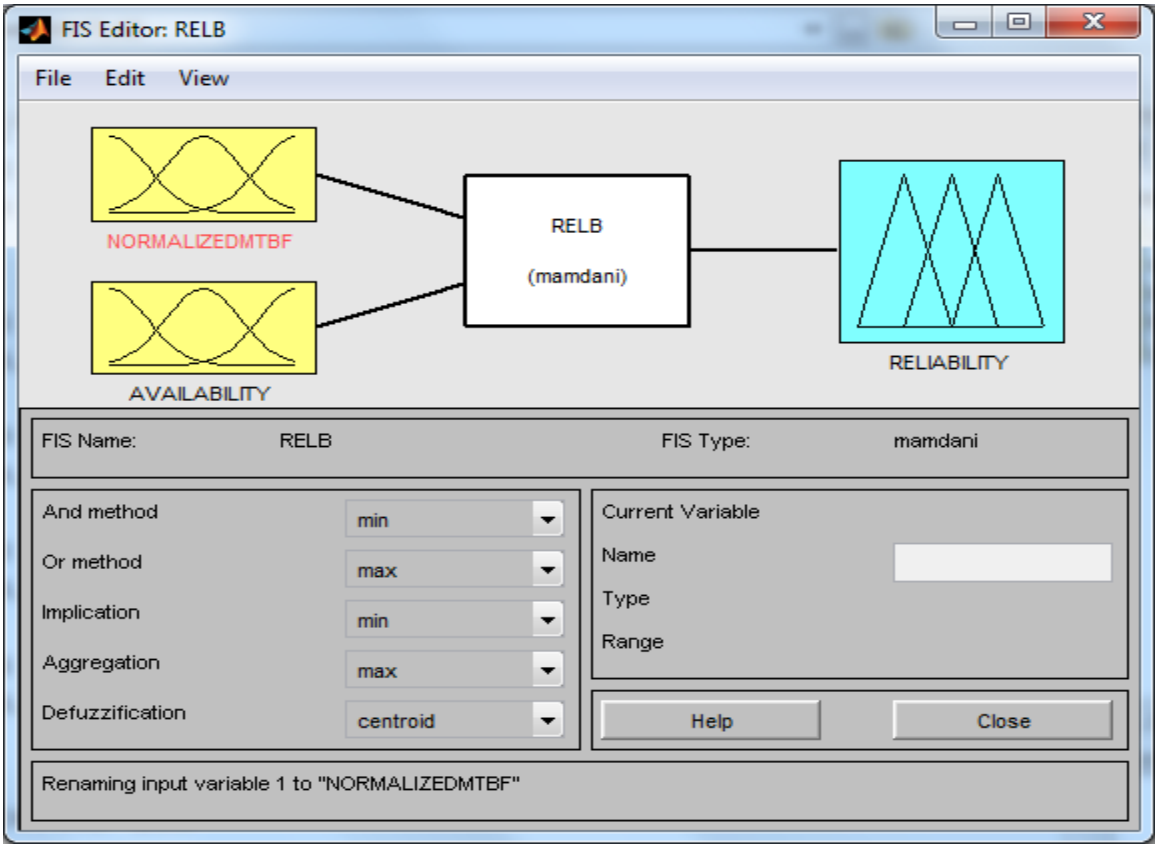

Figure 4.7: FIS System Model

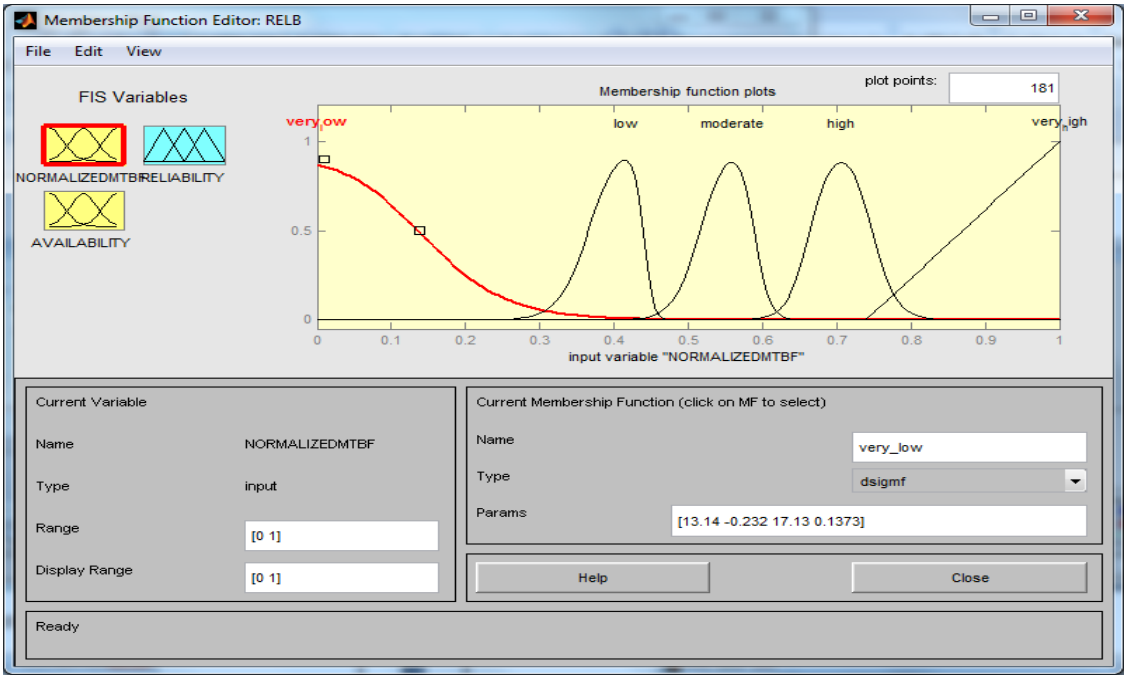

Figure 4.8: Membership Function for MTBF and Availability

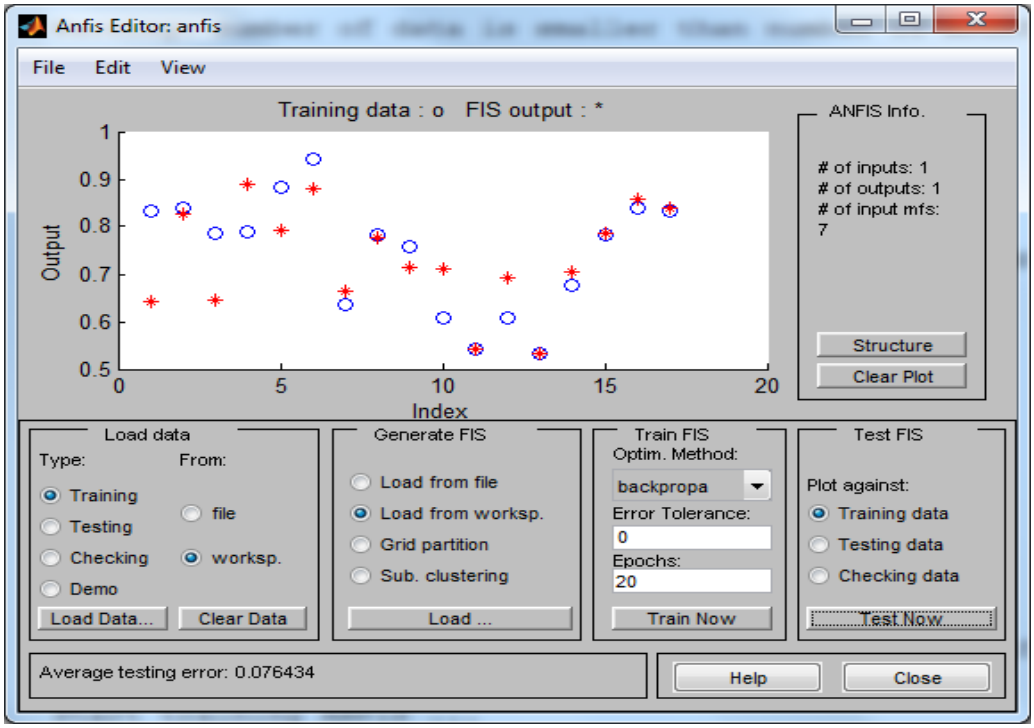

Figure 4.9: Neuro Fuzzy Inference Model

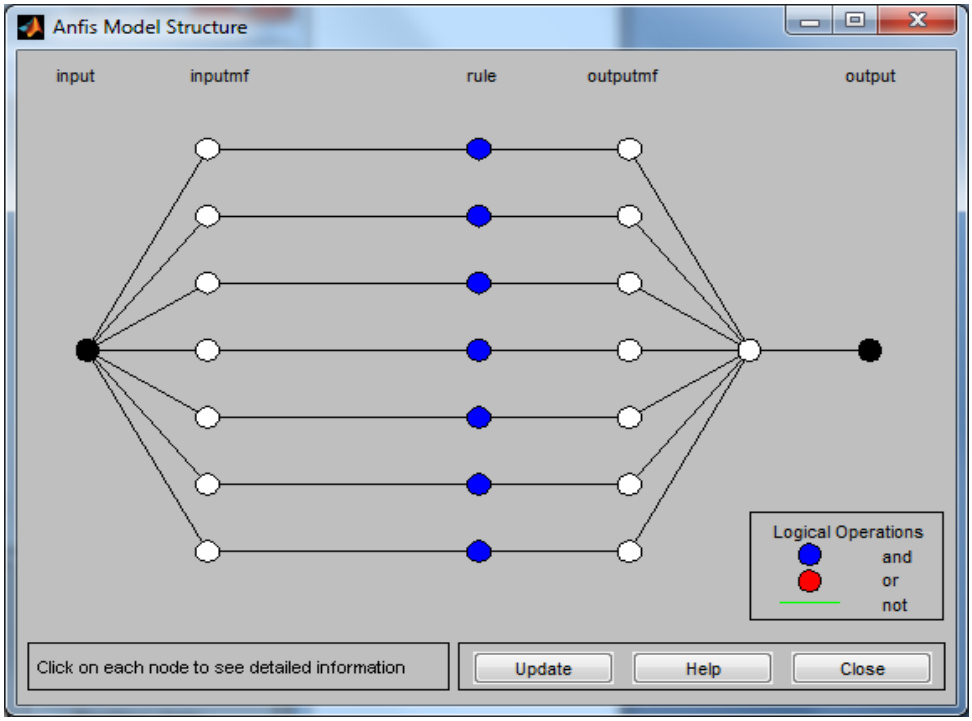

Figure 4.10: Neuro Fuzzy Structure

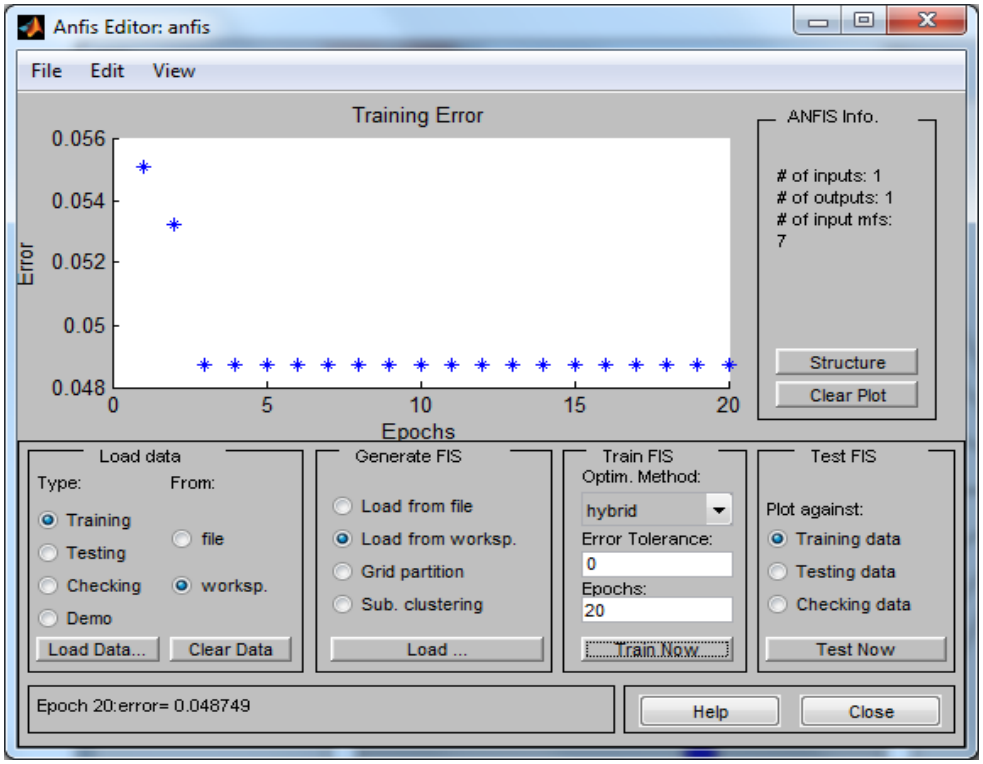

Figure 4.11: Error Tolerance

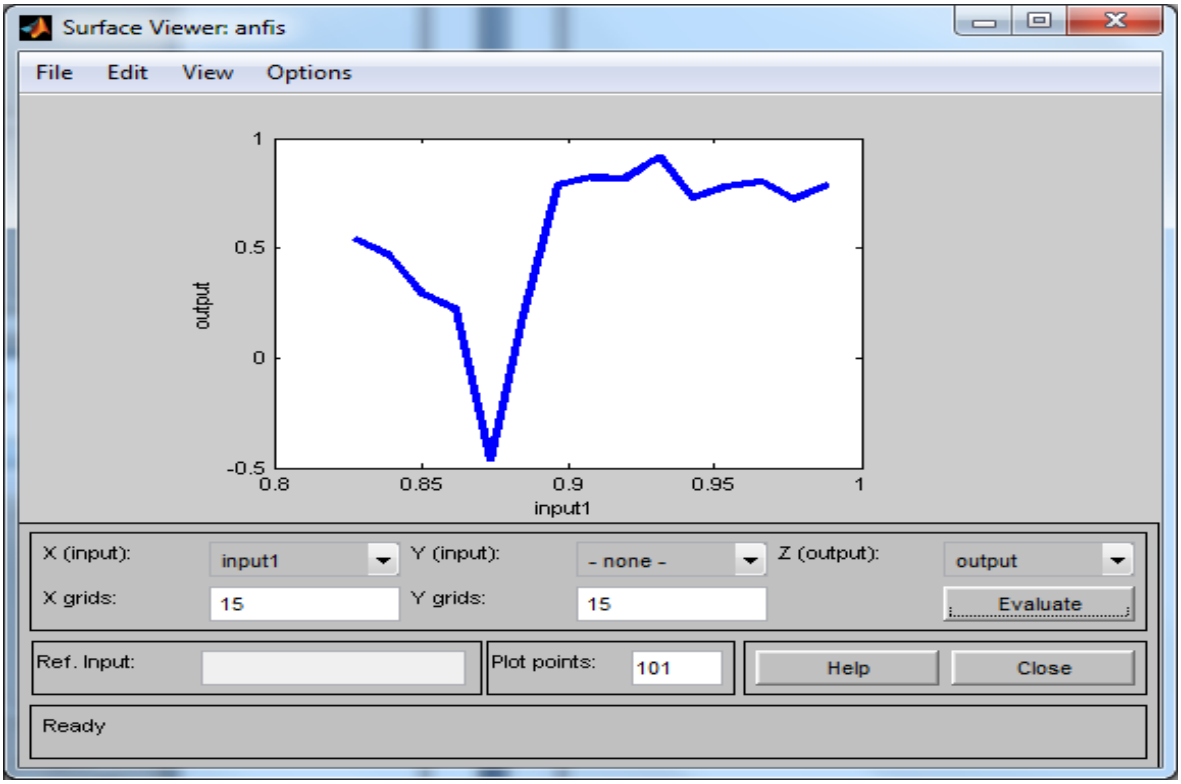

Figure 4.12: Performance Analysis

4.5 Theoretical Validation

The theoretical validation is performed using mathematical approximation of assessed software reliability using the proposed approach. This mathematical approximation is based on Euler's theorem discussed in details in section 3.4. A theoretical valuation is done with the formula mentioned in the context. Five iterations are performed for getting the accuracy level of the Approximated Software Reliability and the results are shown in Table 4.5, Table 4.6, Table 4.7, Table 4.8, Table 4.9 respectively as follows:

1st Iteration

Table 4.5: Calculation of Reliability & its Approximation at 1st Iteration

x	y(Measured	f(a)=MTBF/(1+MTBF)	Approximated value= y - h * f(a)
1	45.18	0.97835	45.123
2	20.78	0.95409	20.725
3	46.05	0.97875	45.993
4	23.51	0.9592	23.454
5	15.32	0.93873	15.266
6	24.09	0.96014	24.034
7	38.5	0.97468	38.443
8	86.14	0.98852	86.083
9	33.78	0.97125	33.728
10	17.12	0.94481	17.065
11	4.8	0.82759	4.752
12	8.43	0.89396	8.378
13	5.03	0.83416	4.982
14	8.46	0.89429	8.408
15	11.08	0.91722	11.027
16	14.18	0.93412	14.126
17	9.69	0.90645	9.637

2nd iteration

Table 4.6: Calculation of Reliability & its Approximation at 2nd Iteration

x	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y - h * f(a)
1	45.12	0.97832	45.07
2	20.73	0.95397	20.67
3	45.99	0.97872	45.94
4	23.45	0.95911	23.4
5	15.27	0.93852	15.21
6	24.03	0.96005	23.98
7	38.44	0.97465	38.39
8	86.08	0.98852	86.03
9	33.73	0.9712	33.67
10	17.07	0.94464	17.01
11	4.752	0.82615	4.704
12	8.378	0.89337	8.326
13	4.982	0.83283	4.934
14	8.408	0.89371	8.356
15	11.03	0.91685	10.97
16	14.13	0.93389	14.07
17	9.637	0.90599	9.584

3rd Iteration

Table 4.7: Calculation of Reliability & its Approximation at 3rd Iteration

x	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y - h * f(a)
1	45.07	0.97829	45.01
2	20.67	0.95385	20.62
3	45.94	0.97869	45.88
4	23.4	0.95901	23.34
5	15.21	0.93832	15.16
6	23.98	0.95996	23.92
7	38.39	0.97461	38.33
8	86.03	0.98851	85.97
9	33.67	0.97116	33.62
10	17.01	0.94448	16.96
11	4.704	0.82468	4.656
12	8.326	0.89277	8.274
13	4.934	0.83148	4.886
14	8.356	0.89312	8.304
15	10.97	0.91649	10.92
16	14.07	0.93365	14.02
17	9.584	0.90552	9.531

4th Iteration

Table 4.8: Calculation of Reliability & its Approximation at 4th Iteration

x	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y - h * f(a)
1	45.01	0.97827	44.952
2	20.62	0.95374	20.56
3	45.88	0.97867	45.822
4	23.34	0.95892	23.286
5	15.16	0.93811	15.104
6	23.92	0.95987	23.866
7	38.33	0.97457	38.272
8	85.97	0.9885	85.912
9	33.62	0.97111	33.56
10	16.96	0.94431	16.9
11	4.656	0.8232	4.608
12	8.274	0.89217	8.222
13	4.886	0.83011	4.838
14	8.304	0.89252	8.252
15	10.92	0.91611	10.868
16	14.02	0.93341	13.964
17	9.531	0.90504	9.479

5th Iteration

Table 4.9: Calculation of Reliability & its Approximation at 5th Iteration

x	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y - h * f(a)
1	44.95	0.97824	44.895
2	20.56	0.95362	20.505
3	45.82	0.97864	45.765
4	23.29	0.95882	23.23
5	15.1	0.9379	15.05
6	23.87	0.95978	23.81
7	38.27	0.97454	38.215
8	85.91	0.98849	85.855
9	33.56	0.97106	33.504
10	16.9	0.94413	16.845
11	4.608	0.82168	4.56
12	8.222	0.89156	8.17
13	4.838	0.82871	4.79
14	8.252	0.89192	8.2
15	10.87	0.91574	10.815
16	13.96	0.93317	13.91
17	9.479	0.90457	9.427

% Reliability= (Average of Approximated vales/ Average of observed Values) x 100

% Reliability after 5th iteration = (23.973/ 24.027)*100=0.9977=99.77%

In 5th iteration, as the study got 99.77% so iteration process will be stopped, since the occurrence of good approximated % of reliability.

4.6 Statistical Validation

A statistical hypothesis is an assertion or conjecture concerning one or more population. To prove that a hypothesis is true, or false, with absolute certainty, we would need absolute knowledge, hypothesis testing concerns on how to use a random sample to judge if it is evidence that supports or not the hypothesis. A statistical hypothesis is a scientific hypothesis that is testable on the basis of observing a process that is modeled through a set of random variables.

4.6.1 Hypothesis Testing

Hypothesis testing or significance testing is a method for testing a claim or hypothesis about a parameter in a population, using data measured in a sample. For obtaining the significance of the approach, the researcher has proposed a null hypothesis as well as an alternate hypothesis. The hypothesis is tested through chi-square test. The objective is to reject the null hypothesis and accept the alternate one. The hypotheses are as follows:

H0: Reliability estimates obtained through MTBF and availability are not significantly comparable/close to those obtained from mathematical approximated values theoretically.

Ha: Reliability estimates obtained through MTBF and availability are significantly comparable/close to those obtained from mathematical approximated values theoretically.

4.6.2 Chi-Square Test for Aprroximated Software Reliability

A **chi-squared test**, also referred to as χ^2 **test** (or **chi-square test**). Chi-square is a statistical test commonly used to compare observed data with data the researchers would expect to obtain according to a specific hypothesis. For example, if, according to Mendel's laws, you expected 10 of 20 offspring from a

cross to be male and the actual observed number is 8 males, then you might want to know about the "goodness to fit" between the observed and expected. Were the deviations (differences between observed and expected) the result of chance, or are they due to other factors. How much deviation can occur before you, the investigator, must conclude that something other than chance is at work, causing the observed to differ from the expected. The chi-square test is always testing what scientists call the **null hypothesis,** which states that there is no significant difference between the expected and observed result [154].

Calculation of Chi-Square variable for Approximated Software Reliability

The Chi- Square test is performed on MTBF values of all 17 programs of dataset. Table 4.3 shows values of MTBF and MTTR which reflects that for all 17 software systems, all of the metrics are highly correlated with each other, with measured MTBF values and Approximated Software Reliability being the most significantly correlated. In order to further assure, χ^2 test has been used for testing the proposed null hypothesis. The results of chi-sqaure test are presented in Table 4.10.

Table 4.10: Calculation of Chi-Square for Approximated Software Reliability (SR)

x	Observed frequency(f_i)	Expected Frequency (e_i)	f_i-e_i	(f_i-e_i) / e_i^2
1	45.18	1.426104	43.7539	21.51366
2	20.78	1.426104	19.3539	9.516256
3	46.05	1.426104	44.6239	21.94144
4	23.51	1.426104	22.0839	10.85859
5	15.32	1.426104	13.8939	6.83159
6	24.09	1.426104	22.6639	11.14377
7	38.5	1.426104	37.0739	18.22913
8	86.14	1.426104	84.7139	41.65357
9	33.784	1.426104	32.3579	15.91028
10	17.12	1.426104	15.6939	7.716644
11	4.80	1.426104	3.373896	1.658935
12	8.43	1.426104	7.003896	3.443795
13	5.03	1.426104	3.603896	1.772025
14	8.46	1.426104	7.033896	3.458546
15	11.08	1.426104	9.653896	4.746792
16	14.18	1.426104	12.7539	6.271055
17	9.69	1.426104	8.263896	4.063333

From the Table 4.10 by finding out the average of chi-square variable the value obtained is: χ^2 = 11.21938. The computed value of chi-square 11.21938 is greater than the critical value of chi-square for 1 degree of freedom at 0.05 level of significance, which is 3.84 (as per Chi-Square Distribution enclosed as Table 1 at **Appendix-C**). The test indicates that there is significant relationship between approximated Software Reliability and the measured values of MTBF

of the dataset of all 17 systems at 0.05 level of significance. Hence, Null Hypothesis is strongly rejected and alternate hypothesis is accepted.

4.7 Comparison of Proposed Approach with conventional Fuzzy System

The inputs to the Neuro Fuzzy system are Normalized MTBF and availability which is shown in the Figure 4.7. The researcher has noted down that for the reliability, the outcome from the conventional system is **84.5 %** and from the proposed approach is **95.5 %** using MATLAB software tool. The program has been run in MATLAB environment (See **Appendix-A**). The results obtained are shown in Figure 4.13. From the above the performance assessment, the improvement of **11%** is achieved with the proposed approach. The evaluation criteria like Mean Squared Error(MSE) and Average Error(AE) [155] are used to evaluate the proposed model performance against conventional Fuzzy Inference System(FIS) based approach.

MSE= ((Theoretical validation Value- practical Validation Value)/ Total no of readings)^2

```
MSE=  ((99.70-95.5)./17).^2=0.061038
```

AE = ((Theoretical validation Value- practical Validation Value)/ Total no of readings)=0.247;

```
AE=  ((99.70-95.5)./17)=0.0247
```

Figure 4.13 shows the results obtained through the experiment done using MATLAB for obtaining the Reliabilty Assessment differences between conventional FIS and the proposed approach. The performance comparison table is prepared based on MSE and AE and shown in the Table 4.11.

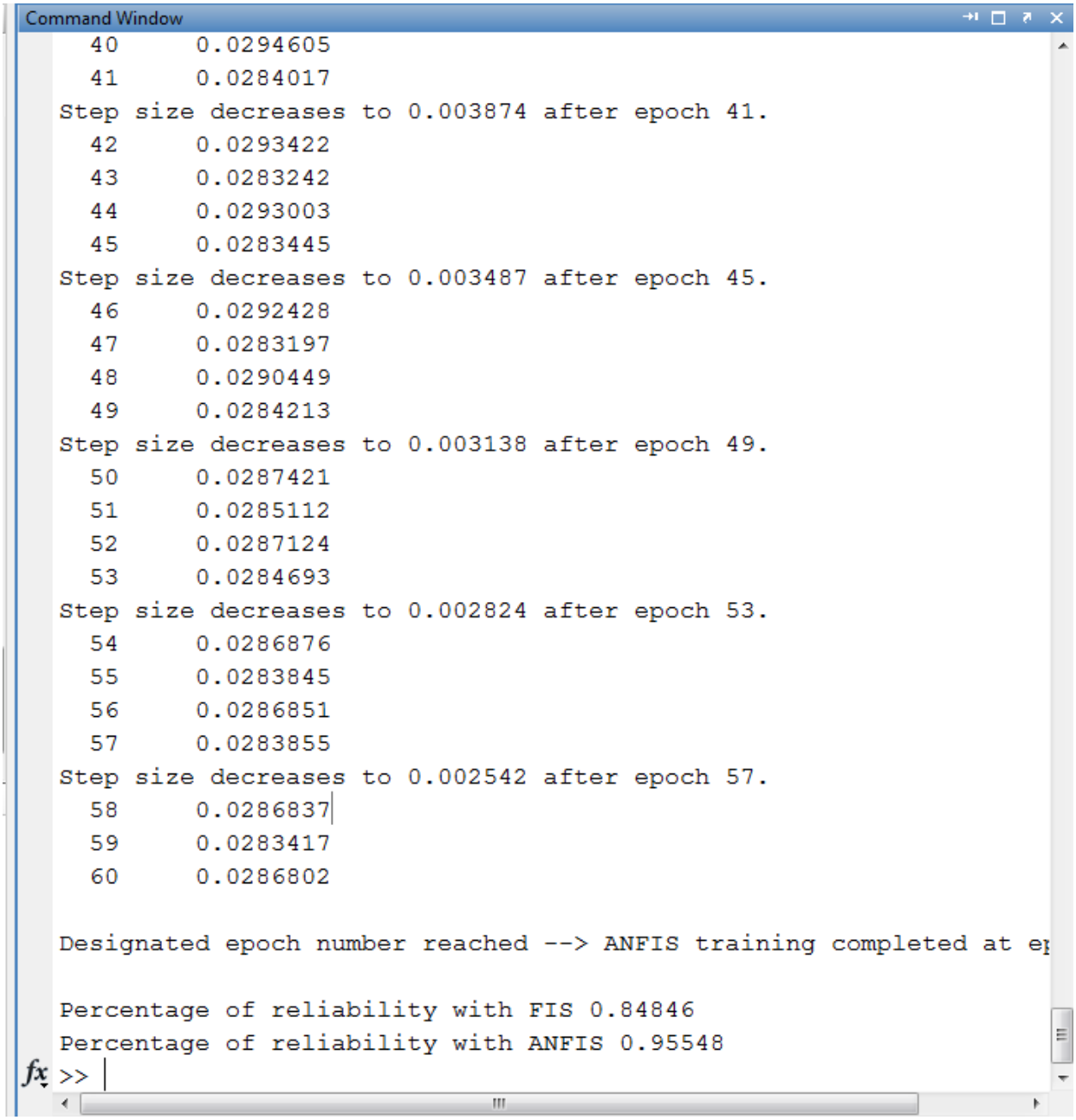

Figure 4.13: Practical Validation of the Reliability Percentage obtained using MATLAB

Table 4.11: Performance Comparison between FIS & ANFIS of SR Estimation

Method	MSE	AE
FIS	0.799	0.894
ANFIS	0.061	0.247

4.8. Conclusion

The iterations process was performed based on the Mathematician Euler's theorem for Approximation to arrive at the accuracy of the assessed Software Reliability. Two evaluation criteria like Mean Squared Error (MSE) and Average Error (AE) are used to evaluate the performace of the proposed approach against the conventional Fuzzy Inference Systems based approach. The comparison table has been chalked out.

Experimental results are obtained on the data of 17 software programmes, with iterative approaximations and the reliability is calculated using FIS and ANFIS models. It is found that the ANFIS is obtaining an improvement of about 11.5 % when campared with FIS and also a decrement of 0.73 units of error and average of about 0.55 units. From these statistical readings this can be concluded that the proposed ANFIS model outperforms and achievs a considerable performance.

5.1 Background

The researcher has discussed vividly about the literature review on Software Reliability, Software Reliability Engineering process, Neural Networks, Fuzzy Logic, Neuro Fuzzy models including the earlier approaches that are present on the same context for the assessment of Software Reliability. The research focused on the need of reliability, importance of reliability assessment. With the review of literature, the research gaps in the existing literature are identified. Their flaws and overcoming have also been discussed. In order to bridge the gap, a model for reliability assessment based on Neuro Fuzzy Systesm has been developed. The same is implemented, validated and compared.

Fuzzy modeling and Neuro Fuzzy modeling, relation between the findings and the approach used in the models is also studied. The implementation of the proposed approach under an experimental dataset has also been discussed. Also, analysis of the performance and evaluation of the outcomes of the proposed approach has been performed. The implementation and validation is done using the MATLAB software environment. A detailed explanation to the experimentaiton procedure followed is explained in chapter 4 and the experimental results are depicted. The proposed approach has also been compare with the conventional FIS based system of Software Reliability Assessment. A brief explanation of the problem findings, achievements and contribution are explained in this chapter.

5.2 Significant contribution

In this research work a Neuro Fuzzy interference model has been designed for the assessment of reliability of a software growth model.The algorithm mainly focuses on MTBF and availability which is analyzed and calculated theoretically and practically. The proposed approach is compared against conventional fuzzy model with the same sigmoid function. The experiments are

conducted on 17 program's dataset and found that the proposal approach is obtaining an improvement of 11% in assessment of reliability of SRGM.

From the research this is found that Neuro Fuzzy model performs better in terms of less error in prediction as compared to existing analytical models and hence it is a better alternative to do Software Reliability test. As the weights are randomly initialized, thus the model gives different results for the same datasets and thus the performance of the model varies. The usefulness of a Neuro Fuzzy model is dependent on the nature of dataset up to a greater extent.

Significant contributions are:

- A Neuro Fuzzy interference model is designed for the assessment of reliability of a software growth model, the model mainly focuses on MTBF and Availability which is analyzed and calculated theoretically and practically.
- The fuzzy rules for the assessment of Software Reliability are designed.
- Reliability of a software at design phase has been evaluated with Neuro Fuzzy system.
- The proposed model has been evaluated and the results are compared withconventional FIS system.

5.3 Research Findings

The findings of the study are as follows:

- A novel ANFIS model is proposed in this thesis, the below are the few findings of the research.
- Reliability is one of the key factors for the proper functioning and utilization of the software , the ANFIS model proposed in this research may able to estimate this reliability accurately.
- Mathematical relationship between the assessement analysis and Neuro Fuzzy mnodel has been performed.

- Design and implementation of the Neuro Fuzzy model with the given inputs are done.
- Analysis and evalution, validation of model is done with approximations.
- Performed hypothesis testing to check about the results using Chi-Square Test.
- Automation of calculation part is done in JAVA Programming language with MySQL as background and is presented at **Appendix-B**.
- Hybrid combination of Normalized MTBF and Availability are taken into consideration for the effective assessment of Software Reliability using Neuro Fuzzy based Systems.
- Fuzzy Rules are implemented properly to guide the Assessment process.
- Fuzzy Membership function like sigmoid function is taken into consideration for the assessment using ANFIS.
- Mathematical Approximation is done with iterations to find out the accuracy level of the assessed Software Reliability.
- A Neuro Fuzzy based Software Reliability assessment is implemented with feed forward and recurrent Neuro Fuzzy network model.
- The observations conclude that proposed network model performs better in terms of less error in prediction as compared to existing analytical models and hence it is a better alternative to do Software Reliability test using neural network. However it can be seen from the Software Reliability estimates that the ANFIS method proposed in this research provides a good fit than analytical models.
- On the other hand a minor limitation of this network model is that the connection weights are randomly initialized, thus the network model gives different results for the same datasets and thus the performance of the network varies. The usefulness of this network model is dependent on the nature of dataset up to a greater extent.

5.4 Future Work and Suggestions

Software Reliability can be predicted using hybrid intelligent system. In addition to neural network model genetic programming can be applied further. Novel recurrent architectures for Genetic Programming (GP) and Group Method of Data Handling (GMDH) to predict Software Reliability can be proposed. Further, research can be extended by developing GP and GMDH based ensemble models to predict Software Reliability. In the ensemble models, GP and GMDH are considered as constituent models.

Research is a continuing activity. As a future researcher plans the following tasks are to be completed:

- The researcher plans to predict Software Reliability using hybrid intelligent system. In addition to neural network model genetic programming can be applied further.
- The researcher plans to develop a novel recurrent architecture for Genetic Programming (GP) and Group Method of Data Handling (GMDH) and also GMR(Group Maturity Rating) in combination of Fuzzy logic for predicting Software Reliability.
- The researchers may plan to conduct more experiments on real time projects data to derive more clear conclusions about the metrics and models values.
- The researchers may plan to extend this work to the other machine learning techniques like Neuro Fuzzy systems approach, support vector machine approach, self-organizing maps approach, decision-region approach etc. for the better estimation of the Software Reliability at different stages of Software Development Life Cycle (SDLC) process. The researchers can also incorporate recent evolutionary computational mechanisms for the purpose of assessing the Software Reliability.
- Accuracy of the Fuzzy Rule Generation for the Fuzzy Inference System can further be improved with decision tree techniques.

5.5 Limitations and Delimitations

As every coin has two faces. The proposed work too suffers from the follwiing limitations:

- This approach is tested on a smaller dataset using one software only.
- It would be more accurate if it has been tested on multiple datasets on differet software outcomes.
- The Neuro Fuzzy system also suffers from stability issues, where the output may not retain the same all the time.
- No research study completes in all aspects, there is always a scope for further improvement.
- High values of MTBF and Availability increases the Software Reliability, means there exists a linear relation between the parameters considered.
- The approach can be used to demonstrate how to measure Software Reliability only but doesn't demonstrate how to maximize them for the betterment of Software Reliability.
- There seems no hybrid approach exists which takes more than one parameter for the assessment of Software Reliability using Neuro Fuzzy System.
- As per the current literature survey is concerned, there seems a gap of assessing Software Reliability Growth Model that suits to all the stages of Software Development Life Cycle.
- None of the factors exist that directly effects the reliability of the software.
- The model can be used to maximize and measure the availability, MTBF and maximize and assess Software Reliability based on Neuro Fuzzy systems approach.
- The model is validated with only a small dataset(17 programs of GlaceEMR software).
- The research concentrated on reliability factors like availability, MTBF and uptime, downtime, number of breakdowns only.

On the other hand the approach has the following delimitations:

- The Neuro Fuzzy approach is most accurate mode of measurement.
- The approach is very easy to implement.
- The research has taken hybrid parameters like normalized MTBF and Availability for assessment of Software Reliability.

5.6 Conclusion

From the research the researchers found that Neuro Fuzzy model performs better in terms of less error in prediction as compared to existing analytical models and hence it is a better alternative to do Software Reliability test. The usefulness of a Neuro Fuzzy model is dependent on the nature of dataset up to a greater extent.The preliminary computational results in the MATLAB environment seem quite promising and give insight into the generalization capability of these models.

The results of the Fuzzy Logic and Neural Networks models are very promising. The error difference between the actual and estimated response is small. This finding gives a good indication of prediction capabilities of the developed fuzzy model and Neural Networks for assessing the Software Reliability.After evaluation of the proposed model, the proposed improved Neuro Fuzzy systems based approach for Software Reliability assessment is compared against the existing conventional Fuzzy Logic based Software Reliability growth assessment and evaluation model based on the experimental results.

References

[1] Hoyer, R. W. and Hoyer, B. B. Y.(2001). "What is quality?". *Quality Progress*, no. 7, pp. 52-62.

[2] Crosby P.B. (1979). “Quality is Free the Art of Making Quality Certain”. *New York, McGraw–Hill*.

[3] Deming & Edwards W. (1986). ”Out of Crisis”. *MIT Press*.

[4] Feigenbaum A. V.(1983). "Total Quality Control”. *McGraw – Hill*.

[5] IEEE Std 610.12 (1990). *IEEE Standard Glossary of Software Engineering Terminology*, New York.

[6] ISO/IEC Std 9126-1 (2001)” Software Engineering – Product Quality “, *Part 1: Quality Models, International Organization for Standards 2001*.

[7] McCall J.A, Richards P.K & Walters G.F (1977)” Factors in Software Quality”, *Nat'l Tech Information Services* – Vol. 1-2.

[8] AIAA/ANSI (1992) “Recommended Practice for Software Reliability, the American Institute of Aeronautics and Astronautics”, *Washington DC, Aerospace Center*, R-013, ISBN 1-56347-024-1.

[9] Michael R. Lyu(1996), “Handbook of Software Reliability Engineering”, *IEEE*.

[10] Wasserman, Gary(2002), “Reliability verification, Testing and analysis in engineering design”, *Marcel Dekker incorporated, New York, USA*, pp2-10.

[11] Schneidewind, N. F., (2001), “Modeling the fault correction processes”, *Proceedings of the 12th International Symposium on Software Reliability Engineering*, pp. 185-190.

[12] Myrtveit, I., Stensrud, E. and Shepperd, M., (2005), “Reliability and validity in comparative studies of software prediction models”, *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380-391.

[13] Falcone, G. Hierachy(2010), “Aware Software Metrics in Component Composition Hierarchies”, *PhD thesis, University of Mannheim,* 2010.

[14] Bass, L., Clements, P., and Kazman, R.(2003), “Software Architecture in Practice”, *Second Edition. Addison-Wesley, Reading, MA, USA*.

[15] Jain, R.(1991), ”The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling”, *Wiley*.

[16] Smith, C. U. and Williams, L. G. (2000), Software performance anti patterns, *in Workshop on Software and Performance*, pages 127-136.

[17] Immonen, A. and Niemela, E. (2008), “Survey of reliability and availability prediction methods from the viewpoint of software architecture”, *Software and System Modeling*,7(1):49{65).

[18] AIAA/ANSI (1992)“Recommended Practice for Software Reliability, The American Institute of Aeronautics and Astronautics”, *Washington DC, Aerospace Center*, R-013, ISBN 1-56347-024-1.

[19] Rosenberg Linda, Hammer Ted & Shaw Jack (1998) “Software Matrices and Reliability”, *ISSRE*.

[20] N. E. Fenton and M. Neil(1999), “A critique of software defect prediction models”, *IEEE Transactions on Software Engineering*, 25(5):675–689.

[21] N. E. Fenton and N. Ohlsson(2000), “Quantitative analysis of faults and failures in a complex software system”, *IEEE Transactions on Software Engineering*, 26(8):797–814.

[22] Wood Alan(1996), “Software Reliability Growth Models”, *Tandem Technical Report- Vol. 19.1*-part number 130056.

[23] Lyu M.R(1996), “Handbook of Software Reliability Engineering”, *IEEE Computer Society Press*.

[24] Schneidewind N. F(1993), “Software Reliability Model with Optimal Selection of Failure Data” , *IEEE Transactions on Software Engineering - Vol 19(11)*, Pp No. 1095-1104.

[25] L. Tian and A. Noore (2004), “Software Reliability Prediction Using Recurrent Neural Network with Bayesian Regularization”, *International Journal of Neural Systems*, vol. 14, no. 3, pp. 165–174.

[26] P.Werbos. (1988) “Generalization of Back propagation with Application to Recurrent Gas Market Model,” *Neural Network, vol. 1*, pp. 339–356.

[27] R. Shadmehr and D. Z. DSArgenio. (1990) “A Comparison of a Neural Network Based Estimator and Two Statistical Estimators in a Sparse and Noisy Data Environment,” *in IJCNN, vol. 1, Washington D.C*, pp. 289–292.

[28] N. Karunanithi, Y. Malaiya, and D. Whitley (1991) “Prediction of Software Reliability Using Neural Networks,” *in Proceedings IEEE International Symposium on Software Reliability Engineering. Austin, TX: IEEE*, pp. 124–130.

[29] T. M. Khoshgoftaar, A. S. Pandya and H. More (1992) “A Neural Network Approach For Predicting Software Development Faults”, *Research Triangle Park, NC: Proceedings of Third International Symposium on Software Reliability Engineering*, pp. 83–89.

[30] Y. Singh and P. Kumar(2010) “Prediction of Software Reliability Using Feed Forward Neural Networks,” *in Computational Intelligence and Software Engineering (CiSE), I. Conference, Ed. IEEE*, pp. 1–5.

[31] M. M. T. Thwin and T. S. Quah, Eds.(2002), “Application of Neural Network for Predicting Software Development Faults using Object-Oriented Design Metrics”, *vol. 5. Proceedings of the 9th International Conference on Neural Information Processing (ICONIP’02)*.

[32] N. Karunanithi and D. Whitley(1992), “Prediction of Software Reliability Using Feed forward and Recurrent Neural Nets,” in *Neural Networks, 1992. IJCNN, vol.1. Baltimore, MD: IEEE*, pp. 800–805.

[33] R. Sitte (1999) “Comparison of software-reliability-growth predictions: Neural Networks vs. parametric-recalibration,” *Reliability, IEEE Transactions*, vol. 48, no. 3, pp. 285–291.

[34] N. RajKiran and V. Ravi (2007), “Software Reliability Prediction using Wavelet Neural Networks,” in *International Conference on Computational Intelligence and Multimedia Applications, vol. 1. Sivakasi, Tamil Nadu: IEEE*, pp. 195 – 199.

[35] J. H. Lo (2009), "The Implementation of Artificial Neural Networks Applying to Software Reliability Modeling," *Control and Decision Conference, 2009. CCDC '09, Chinese*, pp. 4349 – 4354.

[36] L. Zhao, J. pei Zhang, J. Yang, and Y. Chu (2010), "Software Reliability growth model based on fuzzy wavelet neural network" in *2nd International Conference on Future Computer and Communication (ICFCC), vol. 1. Wuhan: IEEE*, pp. 664–668.

[37] Quah, T. and Thwin, M.(2002), "Application of Neural Networks for Predicting Software Development Faults Using Object-Oriented Metrics", *Proceedings of Ninth International Conference on Neural Information Processing (ICONIP'02), vol. 5*, pp. 2312-2316.

[38] R. Sitte. (1999), "Comparison of Software Reliability growth prediction: Neural Networks VS Parametric recalibration," *IEEE Trans. on Reliability*, vol. 48, no.3, pp. 285-291.

[39] Aljahdali, S., Sheta, A., and Rine, D. (2002), "Predicting Accumulated Faults in Software Using Radial Basis Function Network", *Proceedings of the ISCA 17th International Conference on Computers and their Application*, pp. 26-29.

[40] Kaufmann A and Gupta M. M.(1988), *Fuzzy mathematical models in engineering and management science, North-Holland, Amstredam.*

[41] Cai K.Y.(1996), System failure and fuzzy methodology: an introductory overview, *Fuzzy Sets and Systems*, 113-133

[42] Specht, D.F., (1991). A general regression neural network, *IEEE Transactions on Neural Networks,* 2 (6), 568-576.

[43] Kasabov, N.K., (2002). DENFIS: dynamic evolving neural-Fuzzy Inference System and its application for time-series prediction, *IEEE Transactions on fuzzy systems* 10(2).

[44] Makridakis, S., Anderson, A., Carbone, R., Fildes, R., Hibdon, M., Lewandowski, R., Newton, J., Parzen, E., Winkler, R., (1982), "The

accuracy of extrapolation (time series) methods: results of a forecasting competition", *Journal of Forecasting 1*, 111-153.

[45] Perrone, M.P., Cooper, L.N., (1993), "When networks disagree: ensemble methods for hybrid neural netwoks", in *Mammone, R.J. (Ed.), Neural Networks for speech and Image processing", Chapman Hall*, pp. 126-142.

[46] Yu, L., Wang, S.Y., Lai, K.K., (2005), "A novel non-linear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates", *Computers and Operations Research* 32 (10), 25232541.

[47] Olmeda, I., Fernandez, E., (1997), "Hybrid classifiers for financial multi-criteria decision making: the case of bankruptcy prediction", *Computational Economics* 10, 317-335.

[48] Zhou, Z-H., Wu, J., Tang, W., (2002), "Ensembling Neural Networks: many could be better than all", *Artificial Intelligence* 137, 239-263.

[49] J.T. Duane. (1964), "Learning curve(approach to reliability monitoring)", *IEEE Trans. on Aerospace*, AS-2. 563-566.

[50] Z. Jelinski and P. Moranda (1972), "Software Reliability research in Statistical Computer Performance Evaluation", *W. Freiberger, Ed. New York: Academic*, pp. 465-484.

[51] M. L. Shooman (1972), "Probabilistic models for Software Reliability prediction," in *Statistical Computer Performance Evaluation, W. Freiberger, Ed. New York: Academic*, pp. 485-502.

[52] J. D. Musa(1971), "A theory of Software Reliability and its application", *IEEE Trans. Software Eng. , vol. SE-1*, pp. 312-327.

[53] J. D. Musa and K. Okumoto (1983), "A logarithmic Poisson execution time model for Software Reliability measurement," in *Proceedings of 7th International Conference on Software Eng., Orlando, FL*, pp. 230-237.

[54] A. L. Goel, K. Okumoto (1979), "Time-dependent error detection rate model for Software Reliability and other performance measures", *IEEE Trans. Rel.*, R-28(3), pp. 206-211.

[55] Yamada, M. Ohba, and S. Osaki (1983), "S-shaped reliability growth modeling for software error detection", *IEEE Trans. Rel.*, vol. R-32, pp. 475-478.

[56] J. D. Musa, K. Okumoto (1982), "Software Reliability models: concepts, classification, comparisons, and practice", *Proc. Electronic Systems Effectiveness and Life Cycle Costing Conference, Norwich, U. K.*, July 19-31, *NATO ASI Series, Vol. F3, (Ed: J. W. Skwirzynski) Springer-Verlag, Heidelberg*, pp. 395-424.

[57] Härtler, G. (1989), "The non-homogeneous Poisson process—a model for the reliability of complex repairable systems." *Microelectronics Reliability*, 29.3: 381-386.

[58] Yamada, Shigeru, Koichi Tokuno, and Shunji Osaki. (1992), "Imperfect debugging models with fault introduction rate for Software Reliability assessment", *International Journal of Systems Science*, 23.12: 2241-2252.

[59] Yager, Susan E., et al. (1997), "Microcomputer playfulness: stable or dynamic trait?", *ACM SIGMIS Database* 28.2: 43-52.

[60] Hossain, Syed, and Ram C. Dahiya. (1993), "Estimating the parameters of a non-homogeneous Poisson-process model for Software Reliability", *IEEE Transactions on Reliability,* 42.4: 604-612.

[61] Kuo, Lynn, and Tae Young Yang. (1996), "Bayesian computation for non-homogeneous Poisson processes in Software Reliability", *Journal of the American Statistical Association*, 91.434: 763-773.

[62] Singpurwalla, Nozer D., and Simon P. Wilson. (1994), " Software Reliability modeling", *International Statistical Review/Revue International de Statistique* : 289-317.

[63] Tian, Jeff. (1999), "Measurement and continuous improvement of Software Reliability throughout software life-cycle", *Journal of Systems and Software* 47.2: 189-195.

[64] Inoue, Shinji, and Shigeru Yamada. (2007), "Discrete program-size dependent Software Reliability assessment: Modeling, estimation, and goodness-of-fit comparisons", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 90.12: 2891-2902.

[65] R. Kruse, J. Gebhardt and F. Klawonn (1994a). Foundations of Fuzzy Systems. *Wiley, Chichester*.

[66] R. Kruse, J. Gebhardt and R. Palm, eds. (1994b), Fuzzy Systems in Computer Science. *Vieweg, Braunschweig*.

[67] W. Pedrycz and H. C. Card (1992). "Linguistic Interpretation of Self-Organizing Maps", *in Proc. IEEE Int. Conf. on Fuzzy Systems* 1992, pages 371–378, San Diego, CA.

[68] D. Nauck, U. Nauck and R. Kruse (1996), "Generating Classification Rules with the Neuro Fuzzy System NEFCLASS. *in Proc. Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96*, pages 466–470, Berkeley, CA.

[69] Glossary Working Party of the International Software Testing Qualification Board (ISTQB), van Veenendaal, E. (ed.) (2006), *Standard glossary of terms used in Software Testing, Version 1.2*.

[70] Standards Coordinating Committee of the IEEE Computer Society (1991), *IEEE standard computer dictionary 610*.

[71] Storey, N (1996), Safety-Critical Computer Systems, *Prentice Hall*.

[72] Ammann, P, Offutt, J. (2008), Introduction to Software Testing, *Cambridge University Press*.

[73] Bonthu Kotaiah, Raees Ahmed Khan, Muralidhar Vejendla (2013),"Model For Software Errors Prediction using Machine Learning to Improve the Software Reliability", *International Journal of Soft Computing and Software Engineering [JSCSE]*, Vol. 3, No. 3, pp. 314-319, eISSN: 2251-7545, Doi: 10.7321/jscse.v3.n3.47

[74] Bonthu Kotaiah and R.A. Khan (2012), "A Survey on Software Reliability Assessment by using Different Machine Learning Techniques",

International Journal of Scientific & Engineering Research(IJSER), Volume 3, Issue 6, P. No.1-7.

[75] Bonthu Kotaiah, R. A. Khan (2013), "Different Issues in Predicting the Software Reliability", *International Journal of Science and Engineering Investigations (IJSEI)*, Vol. 2, Issue 12, Pp.65-68, ISSN: 2251-8843.

[76] Bonthu Kotaiah, R. A. Khan (2013), "Software Reliability Growth Models-A Revisit", *IJAC(International Journal Of Advanced Computing (Print Journal)*, ISSN. 0975-7686, P.No.32-36, Online Link: http:// Journals.Griet.ac.in/Images/V5i1jan2013_7.Jpg

[77] Bonthu Kotaiah, R.A. Khan (2015),"An Analysis of Software Reliability Assessment with Neuro Fuzzy based Expert Systems ", *Elsevier B.V. ., Procedia Computer Science*, ISSN No.1877-0509, Proceedings of International conference on Soft Computing and Software Engineering(SCSE), University of California, USA (Paper ID-40819).

[78] Bonthu Kotaiah, R. A. Khan, D.S.R. Krishna (2013), "A Neural Network methodology for Software Reliability prediction of long-term MTTF", *Mathematical Sciences International Research Journal*, Volume 2, Issue No.1, ISSN 2278-8697 Pp. No.101-103.

[79] Bonthu Kotaiah, R.A. Khan (2015), "Effective Assessment of Software Reliability by Using Neuro-Fuzzy System", *International Journal of Research (IJR)* e-ISSN: 2348-6848, p-ISSN: 2348-795X Volume 2, Issue 08, Pp. No. 250-270.

[80] Bonthu Kotaiah, R.A. Khan (2013), "Assessing the level of Software Reliability using Neuro Fuzzy System", *International Conference on Information & Engineering Sciences (ICIES -2013)* (Print Journal).

[81] Bonthu Kotaiah, R.A. Khan(2013), "GMR(Group Maturity Rating) with Fuzzy Logic Based Software Defects Rating for the Assessment of Software Development Life Cycle(SDLC) Performance", *Advances in Engineering and Technology Convergence(AETC)*, Bangkok, Thailand, ISBN No: 978-93-82208-89-1, P.No.25-29.

[82] Bonthu Kotaiah, R.A. Khan (2013), "MLP Neural Networks Based Approach for the Assessment of Software performance to Improve the Software Reliability", *The International Conference on Mathematics, Statistics and Computer Engineering (ICMSCE)*, ISSN No.2568-4345, (Print Journal).

[83] Bonthu Kotaiah, R.A. Khan (2013), "The Prelimilinary Prediction of Software Reliability Prediction using Neural Network Based Systems", *International Conference on Mathematical Modelling And Numerical Simulation(ICMMANS), BBAU, Conference Proceedings, Pp. No.62.*

[84] Bonthu Kotaiah, R.A. Khan(2013), "Comparison of Software Reliability Assessment Methods with Neuro Fuzzy Based Systems", *International Conference on Nanoscience And NanoTechnology (ICNN), BBAU, Conference Proceedings. Pp. No.256.*

[85] Bonthu Kotaiah, R.A. Khan(2013), "Software Reliability Assessment by using Neural Networks with Fuzzy Logic based systems", *International Conference on Advances in Computer Science(AET-ACS), Elsevier, onlinelink:http:searchdl.org/public/book_serieselsevierst/6/540.pdf.*

[86] Bonthu Kotaiah, R.A. Khan, Muralidhar V.(2013), " Model for Software Errors Prediction Using Machine Learning to Improve The Software Reliability", *The Proceedings of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13], San Francisco State University, California, U.S.A., Doi: 10.7321/jscse.v3.n4.47.*

[87] Bonthu Kotaiah, B. Sunil Kumar, R.A. Khan(2012), "A Survey on Software Reliability Assessment by using different Machine Learning Techniques, *International Conference on Information Technology, Electronics and Communications(ICITEC-2012)*, *Hyderabad, India*, Pp No.11-17.

[88] Bonthu Kotaiah, R.A. Khan (2014), "A Comparative Study on Software Reliability assessment with Neural Networks and Neuro Fuzzy based

systems", "*International Conference on Modeling and Computing" (ICMC-2014)*, PP No.12, Souvenier, BBAU, Lucknow, Conference Proceedings.

[89] Bonthu Kotaiah , R.A. Khan(2013), **"**Software Reliability Assessment by using Neural Networks with Fuzzy Logic Based Systems**",** *Advances In Engineering And Technology Conference(AET)*, Delhi, NCR, Elsevier, Conference Proceedings.

[90] Bonthu Kotaiah , R.A. Khan(2014), "Issue in the Assessment of Software Reliability with Mathematical Modelling Based Safety-Critical System", *National Conference on Information Security Challenges(NCISC- 2014)*, Organized by Department of Information Technology, Babasaheb Bhimrao Ambedkar University(BBAU), Lucknow, Conference Proceedings.

[91] Bonthu Kotaiah, R.A. Khan(2012), " A Survey on Software Reliability Assessment for Machine Learning Techniques", *National Conference on Recent Trends in Information Technology*, PVP Siddartha Institute of Technology, Vijayawada, Conference Proceedings, Pp No.95-102.

[92] Bonthu Kotaiah , R.A. Khan(2015), "Software Reliability Measurement with Neural Network based Soft Computing Systems", *"Third International Conference on Emerging Research in Computing, Information, Communication And Applications" (ERCICA-2015)*, *Springer*, *Conference Proceedings*.

[93] "Khan, M Taimoor; Mustafa. K; Ahson, SI(2006), "Software Quality, Concepts and Practices", *Alpha Science Intl ltd, England*.

[94] "Mustafa, K; Khan RA.(2005), "Quality Metric Development Framework (QMDF),*Journal of Computer Science*,1,3,437-444.

[95] "Yadav A, Khan, RA (2009) ",Complexity: A Reliability Factor, *IEEE International Advance Computing Conference*, 2375-2378.

[96] "Yadav A, Khan, RA (2009) ",Critical review on Software Reliability models, *International Journal of recent trends in Engineering*, 2-3.

[97] "Nayak, Sandeep Kumar; Khan, Raees Ahmad; Beg, Md Rizwan(2012)",

Reliable Requirement Specification: Defect Analysis Perspective, *Global Trends in Information Systems and Software Applications*,740-751,Springer.

[98] "Yadav A, Khan RA (2011), "Class cohesion complexity metric(C3 M), *"2nd International Conference on Computer and Communication Technology (ICCCT), 2011",* 363-366,IEEE.

[99] "Nazir, Mohd, Khan, Raees A, Mustafa, K. (2010)", "Testability Estimation Framework, *Internaional Journal of Computer Applications*.

[100] Cai, K., Bai, C., Zhong, X., (2003), "Introduction to reliability models of component based software system", *J.Xi'an Jiaotong Univ.* 37 (6), 560–564.

[101] Cheung, R.C., (1980), "A user oriented Software Reliability model", *IEEE Trans. Softw. Eng.* 6 (2), 118–125.

[102] Dimov, Aleksandar, Sasikumar, Punnekkat, (2010), "Fuzzy reliability model for component-based software systems", *36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 39–46.

[103] Dong, W., Huang, N., Ming, Y., (2008), Reliability analysis of component-based software based on relationships of components, *IEEE Conference on Web Services*, pp. 814–815.

[104] Fiondella, Lance, Rajasekaran, Sanguthever, Gokhale, Swapana, (2013), Efficient Software Reliability analysis with correlated component failures, *IEEE Trans. Reliab.* 62 (1), 244–255.

[105] Gokhle, S.S., (2007), "Architecture based Software Reliability analysis: overview and limitations", *IEEE Trans. Dependable Secure Comput.* 4 (1), 32–40.

[106] Gokhle, S.S., Dong, W.E., Trivedi, K.S., Horgan, J.R., (1998), "An analytical approach to architecture based Software Reliability prediction", *Proc. of the Third International Computer Performance and Dependability Symposium*, Durham, NC, pp. 13–22.

[107] Goswami V., Acharya, Y.B., (2009), "Method for reliability estimation of COTS components based software systems", *International Symposium on Software Reliability Engineering*.

[108] Hsu, C., Huang, C., (2011), "An adaptive reliability analysis using path testing for complex component based software systems", *IEEE Trans. Reliab.* 60 (1), 158–170.

[109] Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W. Eric Wong, Aditya P. Mathur, (2013), "Enhancing Software Reliability estimates using modified adaptive testing", *Information and Software Technology Journal Elsevier*, pp. 288–300.

[110] Huang, N., Wang, D., Jia, X., (2008), FAST ABSTRACT: an algebra-based reliability prediction approach for composite web services, *19th International Symposium on Software Reliability Engineering*, pp. 285–286.

[111] Jang, J.R., (1992), ANFIS: adaptive-network-based Fuzzy Inference System, *IEEE Trans. Systems, Man, and Cybernetics*.

[112] Koziolek H., Becker, S., (2005), Transforming operational profiles of software components for quality of service predictions, *Proc. 10th WCOP'05*, pp. 1–8.

[113] Krishnamurthy, S., Mathur, A.P., (1997), on the estimation of reliability of a software system using reliabilities of its components, *Proceeding of the Eighth International Symposium on Software Reliability Engineering, Albuquerque, NM*, pp. 146–155.

[114] Littlewood, B., (1979), Software Reliability model for modular program structure. *IEEE Trans. Reliab.* 28 (3), 241–246.

[115] Lo, J., (2010), Early Software Reliability prediction based on support vector machines with genetic algorithms, *Fifth IEEE Conference on Industrial Electronics and Applications*, pp. 2221–2226.

[116] Palviainen, Markov, Evesti, Antti, Ovaska, Eila, (2011), The reliability estimation, prediction and measuring of component – based software. *J. Syst. Softw.,* 1054–1070.

[117] Kirti Tyagi, Arun Sharma, (2014), An adaptive Neuro Fuzzy model for estimating the reliability of component-based software systems, *Applied Computing and Informatics*, 10, 38–51.

[118] Lee, T., Jung, S. O., In, H. P., & Lee, H. J. (2007), "Cyber Threat Trend Analysis Model Using HMM", in *Information Assurance and Security, Third International Symposium* on (pp.177-182). *IEEE*.

[119] Greitzer, Frank L., Andrew P. Moore, Dawn M. Cappelli, Dee H. Andrews, Lynn A. Carroll and Thomas D. Hull (2008), "Combating the insider cyberthreat", *Security & Privacy, IEEE* 6, no. 1:61-64.

[120] Ingalsbe, Jeffrey A., Louis Kunimatsu, Tim Baeten and Nancy R. Mead, (2008) "Threat modeling: diving into the deep end", *Software, IEEE* 25, no. 1:28-34.

[121] Ongsakorn, P., Turney, K., Thornton, M., Nair, S., Szygenda, S. & Manikas, T. (2010). Cyber threat trees for large system threat cataloging and analysis. *In Systems Conference, 4th Annual IEEE* (pp. 610-615). IEEE.

[122] Gegick, Michael and Laurie Williams(2005), "Matching attack patterns to security vulnerabilities in software intensive system designs", in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1-7.ACM.

[123] Gandotra, V., Singhal, A., & Bedi, P. (2009), "Threat mitigation, monitoring and management plan-A new approach in risk management", in *Advances in Recent Technologies in Communication and Computing*.

[124] Tang, X. & Shen, B. (2009), "Extending Model Driven Architecture with Software Security Assessment", in *Third IEEE International Conference on Secure Software Integration and Reliability Improvement(SSIRI)* (pp. 436-441).

[125] Dhillon, Danny(2011), "Developer-Driven Threat Modeling: Lessons Learned in the Trenches", *Security & Privacy, IEEE* 9, no. 4: 41-47

[126] Okubo, T., Kaiya, H. & Yoshioka, N. (2011), "Effective security impact analysis with patterns for software enhancement", in *Sixth International Conference on Availability, Reliability and Security (ARES)* (pp. 527-534). IEEE.

[127] Stango, A., Prasad, N. R. & Kyriazanos, D. M. (2009), "A threat analysis methodology for security evaluation and enhancement planning", in *Third International Conference on Emerging Security Information, Systems and Technologies. SECURWARE'09. (pp. 262-267). IEEE.*

[128] Frantz, T. L., & Carley, K. M. (2009), "Information assurances and threat identification in networked organizations", in *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009).* (pp. 1-5). *IEEE.*

[129] Debar, H., Becker, M., & Siboni, D. (1992), "A neural network component for an intrusion detection system", in *IEEE Computer Society Symposium on Research in Security and Privacy, Proceedings* (pp. 240-250). *IEEE.*

[130] Mukkamala, S., Janoski, G., & Sung, A. (2002), "Intrusion detection using Neural Networks and support vector machines", in *Proceedings of the International Joint Conference on Neural Networks (IJCNN'02)*, (Vol. 2, pp. 1702-1707), IEEE.

[131] Gong, R. H., Zulkernine, M. & Abolmaesumi, P. (2005), "A software implementation of a genetic algorithm based approach to network intrusion detection" in *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005.* (pp. 246-253). *IEEE.*

[132] Haslum, K., Abraham, A. & Knapskog, S. (2008), "Hinfra: Hierarchical neuro-fuzzy learning for online risk assessment", in *Second Asia*

International Conference on Modeling & Simulation,2008", AICMS 08. (pp. 631-636). *IEEE.*

[133] Ngai, E. W. T., & Wat, F. K. T. (2005), "Fuzzy decision support system for risk analysis in E-commerce development", *Decision support systems*, 40(2), 235-255.

[134] Gandotra, V., Singhal, A. & Bedi, P. (2010), "A step towards secure software system using Fuzzy Logic", in *2nd International Conference on Computer Engineering and Technology (ICCET)*(Vol. 1, pp. V1-427), *IEEE.*

[135] BLi, S. (2000), "The development of a hybrid intelligent system for developing marketing strategy, *Decision Support Systems*, 27(4), 395-409.

[136] Singh, R.P., Rawat, V., Pawar, M. & Mishra, R.K.(2012), "Design of an optimal multi-layer neural network for eigenfaces based face recognition", *Recent Research in Science and Technology*, 4(1), 24–32.

[137] Wong, J., Ho, D., & Capretz, L. F. (2008), "Calibrating function point backfiring conversion ratios using neuro-fuzzy technique", *International Journal of Uncertainty, Fuzziness and Knowledge- Based Systems*, 16(06), 847-862.

[138] Haslum, K., Abraham, A. & Knapskog, S. (2008), "Hinfra: Hierarchical neuro-fuzzy learning for online risk assessment", in *Second Asia International Conference on Modeling & Simulation-AICMS 08*(pp. 631-636). IEEE

[139] Boehm, B. W. (1991), Software risk management: principles and practices, *Software, IEEE*, 8(1), 32-41.

[140] Hu, Y., Zhang, X., Sun, X., Zhang, J., Du, J. & Zhao,J. (2010), "A unified intelligent model for software project risk analysis and planning", in *International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII), (Vol. 4, pp. 110-113). IEEE.*

[141] Bragina, T. & Tabunshchyk, G. (2011), "Fuzzy model for the software projects design risk analysis", in *the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), in 11th International Conference* (pp. 335-341). *IEEE*.

[142] Pooja Rani & Dalwinder Singh Salaria (2013), "Neuro-Fuzzy based Software Risk Estimation Tool", in *Global Journal of Computer Science and Technology Software & Data Engineering, Vol. 13 Issue 6, Global Journals Inc. (USA)*, ISSN: 0975-4172,0975-4350(p.No.1-7)

[143] Aggarwal, K.K., Singh, Y., Kaur A, Malhotra R. (2009), "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study". *Software Process: Improvement and Practice" 2009*; (14): pp. 39-62, DOI= http://onlinelibrary.wiley.com/ doi/10.1002/spip.389.

[144] Han, J., Kamber, M. (2001), *Data Mining: Concepts and Method*. Harchort India Private Limited.

[145] Ho, S, Xie, M, Goh, T.N. (2003), "A study of the connectionist models for Software Reliability prediction", *Computers and Mathematics with Applications 2003*; (46): pp. 1037-1045.

[146] Hosmer, D., Lemeshow, S. (1989), *Applied Logistic regression,* John Wiley and Sons.

[147] Jun, Z. (2007), Predicting Software Reliability with neural network ensembles. *Expert Systems with Applications*; 36(2): pp. 216-222. DOI= http://10.1016/j.eswa.2007.12.029.

[148] Kai, Y.C., Lin, C., Wei, D.W., Zhou, Y.Y., and David, Z. (2001), on the neural network approach in Software Reliability modeling, *Journal of Systems and Software 2001*; (58): pp. 47-62.

[149] Karunanithi, N., Whitley, D., and Malaiya, Y.K. (1992), "Prediction of Software Reliability using connectionist models", *IEEE Transactions on Software Engineering*; 18(7): pp. 563-574.

[150] Lyu, M.R. (1999), "*Handbook of Software Reliability Engineering"*, McGraw Hill, India; 131-151.

[151] MATLAB TOOLBOX (2014), *http://www.mathworks.com* "MatLab Toolbox for ANN, FIS, ANFIS".

[152] Kirti Tyagi, Arun Sharma (2014), "An adaptive Neuro Fuzzy model for estimating the reliability of component-based software systems", *Applied Computing and Informatics 10, Elsevier B.V.* , P. No. 38–51,

[153] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a), "Learning internal representations by error propagation", in D. E. Rumelhart, & J. L. McClelland (Eds.), "Parallel distributed processing: Explorations in the microstructure of cognition (pp. 318–362)", *Cambridge, MA: MIT Press*, 318–362.

[154] Yates, F (1934). "Contingency table involving small numbers and the x^2 test". *Supplement to the Journal of the Royal Statistical Society* 1(2): 217–235.

[155] Vinu V Das, Janahanlal Stephen, Yogesh Chaba (2011), Computer Networks and Information Technologies: Second International Conference on Advances in Communication, Network, and Computing, CNC 2011, Bangalore, India, March 10-11, Proceedings, Springer, 15-Mar-2011.

[156] Bonthu Kotaiah, T.Arundathi, P.M. Jahan (2014), "Comparative analysis of machine learning techniques for Software Reliability prediction", Mathematical Sciences International Research Journal: Vol. 3 Spl Issue (2014) ISSN 2349-1353.

[157] Bonthu Kotaiah, D.S.R. Krishna, T.Arundathi (2014), "Assessment of Software Reliability via Mathematical Modelling", Proceedings of the Second International Conference on "Emerging Research in Computing, Information, Communication and Applications"(ERCICA-2014), ISBN: 9789351072638, Elsevier, Pp. No.864-867.

[158] Bonthu Kotaiah, T.Arundathi, M.D. Fasihuddin, P.M.Jahan(2014), "Machine Learning Techniques for the effective Assessment of Software Reliability", Proceedings of the Second International Conference on "Emerging Research in Computing, Information, Communication and Applications"(ERCICA-2014), ISBN: 9789351072638, Elsevier, Pp. No.868-872.

[159] Bonthu Kotaiah, R.A. Khan (2015), " Automation of Assessment and Approximation of Software Reliability", *International Journal of Research (IJR)* e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, Pp. No. 425-468.

Appendices

Appendix- A

MATLAB Program for Practical Validation

```
clear
% MTBF input
aa= VECTOR OF MTBF VALUES;
af=aa./mean(aa);
% Availability Input
b= VECTOR OF AVAILABILITY VALUES;
% Read the FIS structure named as RELB
F=readfis('RELB.fis');
% Evalate the input with the given fuzzy structre
ff=evalfis([aa./max(aa)+.7,b+.7]',F)
% this section is regarding ANFIS
% train the data for it give MTBF and Availability as inputs
trnData = [af , b];
numMFs = 7;
mfType = 'dsigmf';
epoch_n = 100;
% generate a new anfis with this training data
in_fis = genfis1(trnData,numMFs,mfType);
out_fis = anfis(trnData,in_fis,60);
ff'
mean(ff)
% evaluate the data with input anfis structure
oo=evalfis([b]',out_fis)'
mean(oo)
```

Appendix - B

Automation of Calculation and Approximation of Software Reliability

Source Code Files

1. main.java

```
package org.pra.ui;
import com.jtattoo.plaf.bernstein.BernsteinLookAndFeel;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

/**
 *
 * @authors Bonthu Kotaiah
 */
public class Main extends javax.swing.JFrame {

/**
* Creates new form Main
*/
public Main() {
        try {

UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassN
ame());
            UIManager.setLookAndFeel(new BernsteinLookAndFeel());
            initComponents();
            this.setLocationRelativeTo(null);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedLookAndFeelException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
```

```
     * WARNING: Do NOT modify this code. The content of this method is
    always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
    BEGIN:initComponents
    private void initComponents() {

        jLabel2 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();


    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel2.setFont(new java.awt.Font("Bookman Old Style", 0, 18)); //
    NOI18N
        jLabel2.setText("Welcome to the Software Reliability Calculator");

        jButton1.setText("Click Here");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
    javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
                .addContainerGap(47, Short.MAX_VALUE)
                .addComponent(jLabel2)
                .addGap(37, 37, 37))
            .addGroup(layout.createSequentialGroup()
                .addGap(200, 200, 200)
                .addComponent(jButton1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
        );
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

```
                .addGap(87, 87, 87)
                .addComponent(jLabel2)
                .addGap(69, 69, 69)
                .addComponent(jButton1)
                .addContainerGap(104, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        dispose();
        login mm = new login();
        mm.setVisible(true);
        setSize(350, 250);
    }//GEN-LAST:event_jButton1ActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
    java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.
    Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.
    Level.SEVERE, null, ex);
        }
        //</editor-fold>
/* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Main().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel2;
    // End of variables declaration//GEN-END:variables
}
```

2. login.java

```
package org.pra.ui;
import com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel;
import com.sun.java.swing.plaf.windows.WindowsLookAndFeel;
import java.text.ParseException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import javax.swing.plaf.basic.BasicLookAndFeel;
import javax.swing.plaf.metal.MetalLookAndFeel;
import javax.swing.plaf.nimbus.NimbusLookAndFeel;

/**
 *
 * @authors Bonthu Kotaiah
 */

public class login extends javax.swing.JFrame {
    /**
     * Creates new form login
     */
```

```
    public login() {
            initComponents();
            this.setLocationRelativeTo(null);
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        txtu = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        txtp = new javax.swing.JPasswordField();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jLabel1.setFont(new java.awt.Font("Bookman Old Style", 0, 18)); // NOI18N
        jLabel1.setText("Software Reliability Calculator");
        jLabel2.setText("User Name");
        jLabel3.setText("Password");
        jButton1.setText("Login");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(96, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
TRAILING, false)
```

```
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(txtp,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                    .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(txtu,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addComponent(jLabel1,
javax.swing.GroupLayout.Alignment.LEADING))
            .addGap(78, 78, 78))
        .addGroup(layout.createSequentialGroup()
            .addGap(181, 181, 181)
            .addComponent(jButton1)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(21, 21, 21)
            .addComponent(jLabel1)
            .addGap(52, 52, 52)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jLabel2)
                .addComponent(txtu,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(44, 44, 44)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jLabel3)
```

```
                    .addComponent(txtp,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
41, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(57, 57, 57))
        );
        pack();
    }// </editor-fold>//GEN-END:initComponents
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        String id = txtu.getText();
        char ch[] = txtp.getPassword();
        String p = String.valueOf(ch);
        if (id.equals("software") && p.equals("123456")) {
            dispose();
            reliabilitytable tr = new reliabilitytable();
            tr.setVisible(true);
            tr.getdata();
            setSize(350, 250);
        } else {
            JOptionPane.showMessageDialog(rootPane, "user name does't
match with password", "Error", JOptionPane.INFORMATION_MESSAGE);
            txtu.setText("");
            txtp.setText("");
            txtu.requestFocus();
        }
    }//GEN-LAST:event_jButton1ActionPerformed
    /**
     * @param args the command line arguments
     */
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPasswordField txtp;
    private javax.swing.JTextField txtu;
    // End of variables declaration//GEN-END:variables
}
```

3. reliabilitytable.java

```
package org.pra.ui;
import com.mysql.jdbc.Connection;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.event.ListSelectionEvent;
/**
 *
 * @authors Bonthu Kotaiah
 */
public class reliabilitytable extends javax.swing.JFrame {

    /**
     * Creates new form reliabilitytable
     */
    public reliabilitytable() {
        initComponents();
        this.setLocationRelativeTo(null);
    }
   /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();
        search = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
```

```
        jPanel1 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        jButton2 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jButton5 = new javax.swing.JButton();
        jButton6 = new javax.swing.JButton();
        jButton7 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        jScrollPane1.setViewportView(jTable1);

        jLabel1.setFont(new java.awt.Font("Bookman Old Style", 0, 18)); //
NOI18N
        jLabel1.setText("DATA SET OF SOFTWARE SYSTEM");

        jButton1.setText("Search Program");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
```

```
        jButton3.setText("Calculate New");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jPanel1.setBackground(new java.awt.Color(51, 255, 255));

        jLabel3.setForeground(new java.awt.Color(255, 51, 51));
        jLabel3.setText("PROG#=>PROGRAM#");

        jLabel4.setForeground(new java.awt.Color(255, 51, 51));
        jLabel4.setText("UAX1=>UPTIME AT X1(HRS.)");

        jLabel5.setForeground(new java.awt.Color(255, 51, 51));
        jLabel5.setText("DAX1=>DOWNTIME AT X1(HRS.)");

        jLabel6.setForeground(new java.awt.Color(255, 51, 51));
        jLabel6.setText("UAX2=>UPTIME AT X2(HRS.)");

        jLabel7.setForeground(new java.awt.Color(255, 51, 51));
        jLabel7.setText("DAX2=>DOWNTIME AT X2(HRS.)");

        jLabel8.setForeground(new java.awt.Color(255, 51, 51));
        jLabel8.setText("TPT=>TOTAL PRODUCTION TIME");

        jLabel2.setForeground(new java.awt.Color(255, 51, 51));
        jLabel2.setText("NOBX1=>NO. OF BREAKDOWNS AT X1(HRS.)");

        jLabel9.setForeground(new java.awt.Color(255, 51, 51));
        jLabel9.setText("NOBX2=>NO. OF BREAKDOWNS AT X2(HRS.)");

        jLabel10.setForeground(new java.awt.Color(255, 51, 51));
        jLabel10.setText("MTTR=>MEAN TIME TO REPAIR");

        jLabel11.setForeground(new java.awt.Color(255, 51, 51));
        jLabel11.setText("MTTF=>MEAN TIME TO FAILURE");

        jLabel12.setForeground(new java.awt.Color(255, 51, 51));
        jLabel12.setText("MTBF=>MEAN TIME BETWEEN FAILURES");

        jLabel13.setForeground(new java.awt.Color(255, 51, 51));
        jLabel13.setText("AFRSS=>AVAILABILITY FOR REPAIRABLE
SOFTWARE SYSTEM");
```

```
        jLabel14.setForeground(new java.awt.Color(255, 51, 51));
        jLabel14.setText("AFNRHS=>AVAILABILITY FOR NON- REPAIRABLE
HARDWARE SYSTEM");

        jLabel15.setFont(new java.awt.Font("Bookman Old Style", 1, 18)); //
NOI18N
        jLabel15.setText("ABBREVATIONS");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                    .addComponent(jLabel5)
                    .addComponent(jLabel7)
                    .addComponent(jLabel4)
                    .addComponent(jLabel6))
                .addGap(28, 28, 28)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                    .addComponent(jLabel2)
                    .addComponent(jLabel3)
                    .addComponent(jLabel9)
                    .addComponent(jLabel8))
                .addGap(62, 62, 62)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                    .addComponent(jLabel10)
                    .addComponent(jLabel13)
                    .addComponent(jLabel11)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel12)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel14)))
                .addContainerGap())
```

```
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(65, 65, 65)
                .addComponent(jLabel15)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(jLabel15)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
17, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE)
                    .addComponent(jLabel10)
                    .addComponent(jLabel4)
                    .addComponent(jLabel3))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE)
                            .addComponent(jLabel12)
                            .addComponent(jLabel2)
                            .addComponent(jLabel6))
                        .addGap(18, 18, 18))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel14)
                        .addGap(7, 7, 7)))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE)
                        .addComponent(jLabel9)
                        .addComponent(jLabel11))
                    .addComponent(jLabel5))
```

```
                .addGap(15, 15, 15)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE)
                    .addComponent(jLabel7)
                    .addComponent(jLabel8)
                    .addComponent(jLabel13))
                .addContainerGap())
        );

        jButton2.setText("Exit");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jButton4.setText("Refresh");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton5.setText("Calculate Approximated Value");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jButton6.setText("Delete program");
        jButton6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });

        jButton7.setText("Delete All Records");
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });
```

```
        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addContainerGap())
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jScrollPane1)
                        .addContainerGap())
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
TRAILING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jButton3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                                .addComponent(jButton5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                                .addComponent(jButton4,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                                .addComponent(jButton7)
                                .addGap(14, 14, 14)
                                .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(120, 120, 120))
                            .addComponent(jLabel1))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                    .addComponent(jButton6)
                    .addGap(35, 35, 35)
                    .addComponent(search,
javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(55, 55, 55))))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(25, 25, 25)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(search,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1)
                    .addComponent(jButton6))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jButton3)
                    .addComponent(jButton4)
                    .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton5)
                    .addComponent(jButton7)))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 206,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(7, 7, 7))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents


    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        try {
            // TODO add your handling code here
            Class.forName("com.mysql.jdbc.Driver");
            java.sql.Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
            String name = search.getText();
            if (name.equals("")) {
                JOptionPane.showMessageDialog(rootPane, "Please enter program
name for searching", "Error", JOptionPane.ERROR_MESSAGE);
            } else {

                String sql = "select * from reliabiltity_availbility where
program_name='" + name + "'";

                PreparedStatement ps = (PreparedStatement)
con.prepareStatement(sql);
                ResultSet rs = ps.executeQuery();
                int j = 0;
                while (rs.next()) {
                    j++;
                }
                String[][] sh = new String[j][15];
                rs.beforeFirst();
                j = 0;
```

```
            while (rs.next()) {
                sh[j][0] = rs.getString("id");
                sh[j][1] = rs.getString("program_name");
                sh[j][2] = rs.getString("uptime_at_x1");
                sh[j][3] = rs.getString("uptime_at_x2");
                sh[j][4] = rs.getString("downtime_at_x1");
                sh[j][5] = rs.getString("downtime_at_x2");
                sh[j][6] = rs.getString("total_production_time");
                sh[j][7] = rs.getString("no_of_breakdowns_at_x1");
                sh[j][8] = rs.getString("no_of_breakdowns_at_x2");
                sh[j][9] = rs.getString("mean_time_to_failure");
                sh[j][10] = rs.getString("mean_time_to_repair");
                sh[j][11] = rs.getString("mean_time_between_failures");
                sh[j][12] = rs.getString("availability1");
                sh[j][13] = rs.getString("availability2");
                sh[j][14] = rs.getString("reliability");
                j++;
            }
            jTable1.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    if (e.getClickCount() == 1) {
                        JTable target = (JTable) e.getSource();
                        int row = target.getSelectedRow();
                        String programId = sh[row][1];
                        dispose();
                        tableentry tr = new tableentry(programId);
                        tr.setVisible(true);
                        setSize(350, 250);
                    }
                }
            });
            String[] cols = {"ID", "PROG#", "UAX1", "UAX2", "DAX1", "DAX2",
"TPT", "NOBX1", "NOBX2", "MTTF", "MTTR", "MTBF", "AFRSS", "AFNRHS",
"RELIABILITY"};
            if (sh.length == 0) {
                jTable1.setModel(new javax.swing.table.DefaultTableModel(sh,
cols));
                JOptionPane.showMessageDialog(rootPane, "No record found",
"Info", JOptionPane.ERROR_MESSAGE);
            } else {
                jTable1.setModel(new javax.swing.table.DefaultTableModel(sh,
cols));
            }
        }
    } catch (ClassNotFoundException ex) {
```

```
            Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }//GEN-LAST:event_jButton1ActionPerformed

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton3ActionPerformed
        // TODO add your handling code here:
        dispose();
        tableentry tr = new tableentry();
        tr.setVisible(true);
        setSize(350, 250);
    }//GEN-LAST:event_jButton3ActionPerformed

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton2ActionPerformed
        // TODO add your handling code here:
        shutdownSoftware();
    }//GEN-LAST:event_jButton2ActionPerformed

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton4ActionPerformed
        // TODO add your handling code here:
        this.getdata();
    }//GEN-LAST:event_jButton4ActionPerformed

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton5ActionPerformed
        try {
            // TODO add your handling code here:.
            Class.forName("com.mysql.jdbc.Driver");
            com.mysql.jdbc.Connection con = (com.mysql.jdbc.Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
            String iterationNumber = JOptionPane.showInputDialog("Enter
number of iterations you want to perform.");
            String perLabelText="Software Reliability after"+"
"+iterationNumber+" "+"Iterations";
            if (iterationNumber.equals("")) {
                JOptionPane.showMessageDialog(this, "Please enter the number
of iteration you want to perform.", "Info", JOptionPane.ERROR_MESSAGE);
            } else {
                String sql = "select * from reliabiltity_availbility";
```

```
            com.mysql.jdbc.PreparedStatement ps =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            int j = 0;
            while (rs.next()) {
               j++;
            }
            double values = j;
            String[][] sh = new String[j][15];
            rs.beforeFirst();
            int k = 0;
            while (rs.next()) {
               sh[k][11] = rs.getString("mean_time_between_failures");
               double meanTimeBetweenFailure =
Double.parseDouble(sh[k][11]);
               double h = 1 / values;
               double reliabilityFunction = meanTimeBetweenFailure / (1 +
meanTimeBetweenFailure);

               double approximatedValue = meanTimeBetweenFailure - (h *
reliabilityFunction);
               String mean_time_between_failure =
Double.toString(meanTimeBetweenFailure);
               String reliability_function =
Double.toString(reliabilityFunction);
               String approximated_value =
Double.toString(Math.round(approximatedValue * 1000.0) / 1000.0);
               String sql1 = "insert into
approximation_reliability(mean_time_between_failure, reliability_function,
approximated_value) values (?,?,?)";
               com.mysql.jdbc.PreparedStatement ps1 =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql1);
               ps1.setString(1, mean_time_between_failure);
               ps1.setString(2, reliability_function);
               ps1.setString(3, approximated_value);
               ps1.executeUpdate();
               k++;
            }

            int b = Integer.parseInt(iterationNumber) - 1;
            for (int i = 0; i < Integer.parseInt(iterationNumber); i++) {
               String sqli = "select * from approximation_reliability";
               com.mysql.jdbc.PreparedStatement psi =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sqli);
               ResultSet rsi = psi.executeQuery();
```

```
                String sqlTrun = "TRUNCATE TABLE
approximation_reliability";
                com.mysql.jdbc.PreparedStatement dsTrun =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sqlTrun);
                dsTrun.executeUpdate();
                int ji = 0;
                while (rsi.next()) {
                    ji++;
                }
                double values1 = ji;
                String[][] shi = new String[ji][4];
                rsi.beforeFirst();
                int ki = 0;
                while (rsi.next()) {
                    shi[ki][3] = rsi.getString("approximated_value");
                    double meanTimeBetweenFailure1 =
Double.parseDouble(shi[ki][3]);
                    double h1 = 1 / values1;
                    double reliabilityFunction1 = meanTimeBetweenFailure1 /
(1 + meanTimeBetweenFailure1);
                    double approximatedValue1 = meanTimeBetweenFailure1 -
(h1 * reliabilityFunction1);
                    if (i == b) {
                        sumMeanTime = sumMeanTime +
meanTimeBetweenFailure1;
                        sumApproximated = sumApproximated +
approximatedValue1;
                    }
                    String mean_time_between_failure1 =
Double.toString(meanTimeBetweenFailure1);
                    String reliability_function1 =
Double.toString(reliabilityFunction1);
                    String approximated_value1 =
Double.toString(approximatedValue1);
                    String sqli1 = "insert into
approximation_reliability(mean_time_between_failure, reliability_function,
approximated_value) values (?,?,?)";
                    com.mysql.jdbc.PreparedStatement psi1 =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sqli1);
                    psi1.setString(1, mean_time_between_failure1);
                    psi1.setString(2, reliability_function1);
                    psi1.setString(3, approximated_value1);
                    psi1.executeUpdate();
                    ki++;
                }
            }
```

```
            double AverageMean = sumMeanTime /
Double.parseDouble(Integer.toString(k));
            double AverageApprox = sumApproximated /
Double.parseDouble(Integer.toString(k));
            double percentageReliability = (AverageApprox / AverageMean) *
100;
            dispose();
            percentagereliability pr = new percentagereliability(AverageMean,
AverageApprox, percentageReliability,perLabelText);
            pr.setVisible(true);
            pr.getApproximationData();
          }
       } catch (ClassNotFoundException ex) {
          Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE,
null, ex);
       } catch (SQLException ex) {
          Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE,
null, ex);
       }
   }//GEN-LAST:event_jButton5ActionPerformed

   private void jButton6ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton6ActionPerformed
       // TODO add your handling code here:
       String name = search.getText();
       if (name.equals("")) {
          JOptionPane.showMessageDialog(rootPane, "Please enter program
name for deletion", "Error", JOptionPane.ERROR_MESSAGE);
       } else {
          if (JOptionPane.showConfirmDialog(this, "Do you realy want to
delete?", "Confirmation!", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION) {
            try {
               // TODO add your handling code here
               Class.forName("com.mysql.jdbc.Driver");
               java.sql.Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
               String sql = "delete from  reliabiltity_availbility where
program_name='" + name + "'";
               com.mysql.jdbc.PreparedStatement ds =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql);
               ds.executeUpdate();
               JOptionPane.showMessageDialog(null, "deleted
successfully.....", "DELETE", JOptionPane.INFORMATION_MESSAGE);
               this.getdata();
```

```
                search.setText(null);
            } catch (ClassNotFoundException ex) {

Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE, null,
ex);
            } catch (SQLException ex) {

Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }
   }//GEN-LAST:event_jButton6ActionPerformed

   private void jButton7ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton7ActionPerformed
       // TODO add your handling code here:
       if (JOptionPane.showConfirmDialog(this, "Do you realy want to delete
all the records?", "Confirmation!", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION) {
           try {
               // TODO add your handling code here
               Class.forName("com.mysql.jdbc.Driver");
               java.sql.Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
               String name = search.getText();
               String sql = "TRUNCATE TABLE reliabiltity_availbility";
               com.mysql.jdbc.PreparedStatement ds =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql);
               ds.executeUpdate();
               this.getdata();
               JOptionPane.showMessageDialog(null, "deleted successfully.....",
"DELETE", JOptionPane.INFORMATION_MESSAGE);
           } catch (ClassNotFoundException ex) {

Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE, null,
ex);
           } catch (SQLException ex) {

Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE, null,
ex);
           }
       }
   }//GEN-LAST:event_jButton7ActionPerformed
```

```
    private void shutdownSoftware() {
        if (JOptionPane.showConfirmDialog(this, "Do you realy want to quit?",
"Confirmation!", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION) {
            setVisible(false);
            dispose();
            System.exit(0);
        }
    }

    public void getdata() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
            String sql = "select * from reliabiltity_availbility";
            PreparedStatement ps = (PreparedStatement)
con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            int j = 0;
            while (rs.next()) {
                j++;
            }
            String[][] sh = new String[j][15];
            rs.beforeFirst();
            j = 0;
            int i = j;
            while (rs.next()) {
                sh[j][0] = Integer.toString(++i);
                sh[j][1] = rs.getString("program_name");
                sh[j][2] = rs.getString("uptime_at_x1");
                sh[j][3] = rs.getString("uptime_at_x2");
                sh[j][4] = rs.getString("downtime_at_x1");
                sh[j][5] = rs.getString("downtime_at_x2");
                sh[j][6] = rs.getString("total_production_time");
                sh[j][7] = rs.getString("no_of_breakdowns_at_x1");
                sh[j][8] = rs.getString("no_of_breakdowns_at_x2");
                sh[j][9] = rs.getString("mean_time_to_failure");
                sh[j][10] = rs.getString("mean_time_to_repair");
                sh[j][11] = rs.getString("mean_time_between_failures");
                sh[j][12] = rs.getString("availability1");
                sh[j][13] = rs.getString("availability2");
                sh[j][14] = rs.getString("reliability");
                j++;
            }
```

```
            String[] cols = {"ID", "PROG#", "UAX1", "UAX2", "DAX1", "DAX2",
"TPT", "NOBX1", "NOBX2", "MTTF", "MTTR", "MTBF", "AFRSS", "AFNRHS",
"RELIABILITY"};

            jTable1.setModel(
                    new javax.swing.table.DefaultTableModel(sh, cols));
            /**
             *
             * @param args the command line arguments
             */
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(reliabilitytable.class
                    .getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(reliabilitytable.class
                    .getName()).log(Level.SEVERE, null, ex);
        }
    }

    double sumMeanTime = 0;
    double sumApproximated = 0;
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
```

```
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField search;
    // End of variables declaration//GEN-END:variables
}
```

4. tableentry.java

```
package org.pra.ui;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @authors Bonthu Kotaiah
 */
public class tableentry extends javax.swing.JFrame {

    /**
     * Creates new form tableentry
     */
    public tableentry() {
        initComponents();
        this.setLocationRelativeTo(null);
    }

    public tableentry(String programId) {
        this.programId = programId;
        try {
            initComponents();
            Class.forName("com.mysql.jdbc.Driver");
            java.sql.Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
            String value = programId;
            String sql = "select * from reliabiltity_availbility where
program_name=?";
            com.mysql.jdbc.PreparedStatement ps =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql);
```

```
            ps.setString(1, value);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                String add1 = rs.getString("program_name");
                prog.setText(add1);
                String add2 = rs.getString("uptime_at_x1");
                uptimeAtx1.setText(add2);
                String add3 = rs.getString("uptime_at_x2");
                uptimeAtx2.setText(add3);
                String add4 = rs.getString("downtime_at_x1");
                downtimeAtx1.setText(add4);
                String add5 = rs.getString("downtime_at_x2");
                downtimeAtx2.setText(add5);
                String add6 = rs.getString("no_of_breakdowns_at_x1");
                noOfBreakdownsAtx1.setText(add6);
                String add7 = rs.getString("no_of_breakdowns_at_x2");
                noOfBreakdownsAtx2.setText(add7);
            }
            this.setLocationRelativeTo(null);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(tableentry.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (SQLException ex) {
            Logger.getLogger(tableentry.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
```

```
        jLabel9 = new javax.swing.JLabel();
        prog = new javax.swing.JTextField();
        uptimeAtx1 = new javax.swing.JTextField();
        uptimeAtx2 = new javax.swing.JTextField();
        downtimeAtx1 = new javax.swing.JTextField();
        downtimeAtx2 = new javax.swing.JTextField();
        noOfBreakdownsAtx1 = new javax.swing.JTextField();
        noOfBreakdownsAtx2 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Bookman Old Style", 0, 18)); //
NOI18N
        jLabel1.setText("Enter Values for Reliability Calculation");

        jLabel2.setText("Program #");

        jLabel4.setText("Uptime at x1 (Hrs.)");

        jLabel5.setText("Uptime at x2 (Hrs.)");

        jLabel6.setText("Downtime at x1 (Hrs.)");

        jLabel7.setText("Downtime at x2 (Hrs.)");

        jLabel8.setText("No. of Breakdowns at x1 (Hrs.)");

        jLabel9.setText("No. of Breakdowns at x2 (Hrs.)");

        jButton1.setText("Sumit");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Back");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
```

```
        jButton3.setText("Reset");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap(33, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                            .addComponent(jLabel2)
                            .addComponent(jLabel4)
                            .addComponent(jLabel5)
                            .addComponent(jLabel6)
                            .addComponent(jLabel7)
                            .addComponent(jLabel8)
                            .addComponent(jLabel9))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                            .addComponent(noOfBreakdownsAtx2,
javax.swing.GroupLayout.DEFAULT_SIZE, 105, Short.MAX_VALUE)
                            .addComponent(prog)
                            .addComponent(uptimeAtx1)
                            .addComponent(uptimeAtx2)
                            .addComponent(downtimeAtx1)
                            .addComponent(downtimeAtx2)
                            .addComponent(noOfBreakdownsAtx1))))
```

```
                .addGap(26, 26, 26))
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jButton2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(104, 104, 104)
                        .addComponent(jButton1)
                        .addGap(60, 60, 60)
                        .addComponent(jButton3)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jButton2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
12, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                    .addComponent(jLabel2,
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(prog,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(29, 29, 29)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(uptimeAtx1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addGap(23, 23, 23)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jLabel5)
                .addComponent(uptimeAtx2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(23, 23, 23)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jLabel6)
                .addComponent(downtimeAtx1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(downtimeAtx2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel7))
            .addGap(23, 23, 23)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jLabel8)
                .addComponent(noOfBreakdownsAtx1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(noOfBreakdownsAtx2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel9))
            .addGap(28, 28, 28)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jButton1)
                .addComponent(jButton3))
            .addGap(28, 28, 28))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed



        try {
            // TODO add your handling code here:
            String program_name = prog.getText();
            String uptime_at_x1 = uptimeAtx1.getText();
            String uptime_at_x2 = uptimeAtx2.getText();
            String downtime_at_x1 = downtimeAtx1.getText();
            String downtime_at_x2 = downtimeAtx2.getText();
            String no_of_breakdowns_at_x1 = noOfBreakdownsAtx1.getText();
            String no_of_breakdowns_at_x2 = noOfBreakdownsAtx2.getText();
            if (program_name.isEmpty() && uptime_at_x1.isEmpty() &&
uptime_at_x2.isEmpty() && downtime_at_x1.isEmpty() &&
downtime_at_x2.isEmpty() && no_of_breakdowns_at_x1.isEmpty() &&
no_of_breakdowns_at_x2.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill all the fields",
"Info", JOptionPane.ERROR_MESSAGE);
            } else {
                if (failureMeanTime.equals("")) {
                    failureMeanTime = JOptionPane.showInputDialog("Enter
number of software programs under test for MTTF calculation");
                }
                double numberOfSoftwareProgramsUnderTest =
Double.parseDouble(failureMeanTime);
                Class.forName("com.mysql.jdbc.Driver");
                Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
                String sqlRow = "select * from reliabiltity_availbility";

                PreparedStatement psRow = (PreparedStatement)
con.prepareStatement(sqlRow);
```

```
            ResultSet rsRow = psRow.executeQuery();
            int jRow = 0;
            while (rsRow.next()) {
               jRow++;
            }
            double uptime1 = Double.parseDouble(uptime_at_x1);
            double uptime2 = Double.parseDouble(uptime_at_x2);
            double downtime1 = Double.parseDouble(downtime_at_x1);
            double downtime2 = Double.parseDouble(downtime_at_x2);
            double totalProductionTime1 = uptime1 + downtime1;
            double totalProductionTime2 = uptime2 + downtime2;
            if (totalProductionTime1 != totalProductionTime2) {
               JOptionPane.showMessageDialog(rootPane, "Uptime at x1(Hrs.)
and Downtime at x1(Hrs.) must be equal to Uptime at x2(Hrs.) and
Downtime at x2(Hrs.)", "Info", JOptionPane.ERROR_MESSAGE);
            }else{
            String total_production_time =
Double.toString(totalProductionTime1);
            double meanTimeToFailure = (downtime1 + downtime2) /
numberOfSoftwareProgramsUnderTest;
            String mean_time_to_failure =
Double.toString(meanTimeToFailure);
            double breakDownAtx1 =
Double.parseDouble(no_of_breakdowns_at_x1);
            double breakDownAtx2 =
Double.parseDouble(no_of_breakdowns_at_x2);
            double meanTimeToRepair = ((downtime1/breakDownAtx1) +
(downtime2/breakDownAtx2))/2;
            double meanTimeBetweenFailure = ((uptime1/breakDownAtx1) +
(uptime2/breakDownAtx2))/2;
            String mean_time_to_repair =
Double.toString(meanTimeToRepair);
            String mean_time_between_failures =
Double.toString(meanTimeBetweenFailure);
            double availb1 = meanTimeBetweenFailure /
(meanTimeBetweenFailure + meanTimeToRepair);
            double availb2 = meanTimeToFailure / (meanTimeToFailure +
meanTimeToRepair);
            double relia = meanTimeBetweenFailure / (1 +
meanTimeBetweenFailure);
            String availability1 = Double.toString(Math.round(availb1 * 100.0)
/ 100.0);
            String availability2 = Double.toString(Math.round(availb2 * 100.0)
/ 100.0);
            String reliability = Double.toString(Math.round(relia * 100.0) /
100.0);
```

```
            if (programId.equals("")) {
                if(numberOfSoftwareProgramsUnderTest==jRow){
                    JOptionPane.showMessageDialog(rootPane, "You can't insert
more than"+" "+failureMeanTime+" "+"rows", "Error",
JOptionPane.ERROR_MESSAGE);
                }else{
                String sql = "insert into reliabiltity_availbility (program_name,
uptime_at_x1, uptime_at_x2,
downtime_at_x1,downtime_at_x2,no_of_breakdowns_at_x1,no_of_breakdown
s_at_x2,mean_time_to_failure,mean_time_to_repair,mean_time_between_fail
ures,total_production_time,availability1,availability2,reliability) values
(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
                PreparedStatement ps = (PreparedStatement)
con.prepareStatement(sql);
                ps.setString(1, program_name);
                ps.setString(2, uptime_at_x1);
                ps.setString(3, uptime_at_x2);
                ps.setString(4, downtime_at_x1);
                ps.setString(5, downtime_at_x2);
                ps.setString(6, no_of_breakdowns_at_x1);
                ps.setString(7, no_of_breakdowns_at_x2);
                ps.setString(8, mean_time_to_failure);
                ps.setString(9, mean_time_to_repair);
                ps.setString(10, mean_time_between_failures);
                ps.setString(11, total_production_time);
                ps.setString(12, availability1);
                ps.setString(13, availability2);
                ps.setString(14, reliability);
                int r = ps.executeUpdate();
                if (r > 0) {
                    JOptionPane.showMessageDialog(rootPane, "submitted
successfully", "INSERT", JOptionPane.INFORMATION_MESSAGE);
                    prog.setText(null);
                    uptimeAtx1.setText(null);
                    uptimeAtx2.setText(null);
                    downtimeAtx1.setText(null);
                    downtimeAtx2.setText(null);
                    noOfBreakdownsAtx1.setText(null);
                    noOfBreakdownsAtx2.setText(null);
                    prog.requestFocus();
                } else {
                    JOptionPane.showMessageDialog(rootPane, "error occurred",
"INSERT", JOptionPane.ERROR_MESSAGE);
                }
            }
            } else {
```

```
                if(numberOfSoftwareProgramsUnderTest==jRow){
                    JOptionPane.showMessageDialog(rootPane, "You can't insert
more than"+" "+jRow+" "+"rows", "Error", JOptionPane.ERROR_MESSAGE);
                }else{
                String sql1 = "update reliabiltity_availbility set
program_name='" + program_name + "', uptime_at_x1='" + uptime_at_x1 + "',
uptime_at_x2='" + uptime_at_x2 + "', downtime_at_x1='" + downtime_at_x1 +
"',downtime_at_x2='" + downtime_at_x2 + "',no_of_breakdowns_at_x1='" +
no_of_breakdowns_at_x1 + "',no_of_breakdowns_at_x2='" +
no_of_breakdowns_at_x2 + "',mean_time_to_failure='" +
mean_time_to_failure + "',mean_time_to_repair='" + mean_time_to_repair +
"',mean_time_between_failures='" + mean_time_between_failures +
"',total_production_time='" + total_production_time + "',availability1='" +
availability1 + "',availability2='" + availability2 + "',reliability='" + reliability +
"' where program_name='" + programId + "'";
                com.mysql.jdbc.PreparedStatement ds =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql1);
                int r = ds.executeUpdate();
                if (r > 0) {
                    JOptionPane.showMessageDialog(rootPane, "Updated
successfully", "UPDATE", JOptionPane.ERROR_MESSAGE);
                    prog.setText(null);
                    uptimeAtx1.setText(null);
                    uptimeAtx2.setText(null);
                    downtimeAtx1.setText(null);
                    downtimeAtx2.setText(null);
                    noOfBreakdownsAtx1.setText(null);
                    noOfBreakdownsAtx2.setText(null);
                    prog.requestFocus();
                } else {
                    JOptionPane.showMessageDialog(rootPane, "error occurred",
"UPDATE", JOptionPane.INFORMATION_MESSAGE);
                }
            }
            }
            }
        }
      } catch (ClassNotFoundException | SQLException ex) {
          Logger.getLogger(tableentry.class.getName()).log(Level.SEVERE, null,
ex);
      }
   }//GEN-LAST:event_jButton1ActionPerformed

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton2ActionPerformed
      // TODO add your handling code here:
```

```
        dispose();
        reliabilitytable rt = new reliabilitytable();
        rt.setVisible(true);
        rt.getdata();
        setSize(1000, 250);
    }//GEN-LAST:event_jButton2ActionPerformed

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton3ActionPerformed
        prog.setText(null);
            uptimeAtx1.setText(null);
            uptimeAtx2.setText(null);
            downtimeAtx1.setText(null);
            downtimeAtx2.setText(null);
            noOfBreakdownsAtx1.setText(null);
            noOfBreakdownsAtx2.setText(null);
            prog.requestFocus();

    }//GEN-LAST:event_jButton3ActionPerformed

    public String failureMeanTime = "";
    public String programId = "";
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JTextField downtimeAtx1;
    private javax.swing.JTextField downtimeAtx2;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JTextField noOfBreakdownsAtx1;
    private javax.swing.JTextField noOfBreakdownsAtx2;
    private javax.swing.JTextField prog;
    private javax.swing.JTextField uptimeAtx1;
    private javax.swing.JTextField uptimeAtx2;
    // End of variables declaration//GEN-END:variables
}
```

5. percentagereliability.java

```
package org.pra.ui;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTable;

/**
 *
 * @authors Bonthu Kotaiah
 */
public class percentagereliability extends javax.swing.JFrame {

    /**
     * Creates new form percentagereliability
     */
    public percentagereliability() {
        initComponents();
    }

    public percentagereliability(double averageMean, double averageApprox,
double percentageReliability, String perLabelText) {
        initComponents();
        this.setLocationRelativeTo(null);
        jLabel1.setText(perLabelText);
        jLabel5.setText(Double.toString(Math.round(averageMean * 100.0) /
100.0));
        jLabel6.setText(Double.toString(Math.round(averageApprox * 100.0) /
100.0));
        jLabel7.setText(Double.toString(Math.round(percentageReliability *
100.0) / 100.0));
    }

    public void getApproximationData() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
```

```
            com.mysql.jdbc.Connection con = (com.mysql.jdbc.Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
            String sql = "select * from approximation_reliability";
            com.mysql.jdbc.PreparedStatement ps =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            int j = 0;
            while (rs.next()) {
                j++;
            }
            String[][] sh = new String[j][15];
            rs.beforeFirst();
            j = 0;
            while (rs.next()) {
                sh[j][0] = rs.getString("approx_id");
                sh[j][1] = rs.getString("mean_time_between_failure");
                sh[j][2] = rs.getString("reliability_function");
                sh[j][3] = rs.getString("approximated_value");
                j++;

            }
            String[] cols = {"ID", "MTBF(y)", "f(a)=y/(1+y)", "APPROXIMATED
VALUE"};
            jTable1.setModel(new javax.swing.table.DefaultTableModel(sh, cols));
            /**
             *
             * @param args the command line arguments
             */
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(reliabilitytable.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
```

```
    private void initComponents() {
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_
CLOSE);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        jScrollPane1.setViewportView(jTable1);

        jLabel1.setFont(new java.awt.Font("Bookman Old Style", 0, 18)); //
NOI18N

        jLabel2.setText("Average of Measured Values:");

        jLabel3.setText("Average of Approximated Values:");

        jLabel4.setText("Percentage Reliability:");

        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            jButton1ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addGap(26, 26, 26)
            .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 64,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
417, Short.MAX_VALUE)
.addComponent(jLabel1)
            .addGap(187, 187, 187))
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1))
        .addGroup(layout.createSequentialGroup()
            .addGap(120, 120, 120)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                .addComponent(jLabel2)
                .addComponent(jLabel3)
                .addComponent(jLabel4))
            .addGap(95, 95, 95)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                .addComponent(jLabel5,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel6,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel7,
javax.swing.GroupLayout.DEFAULT_SIZE, 63, Short.MAX_VALUE))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );
```

```
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jButton1))
                .addGap(26, 26, 26)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 216,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(51, 51, 51)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel7)
                    .addComponent(jLabel4))
                .addContainerGap(25, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        try {
```

```
            // TODO add your handling code here:

            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
            String sql = "TRUNCATE TABLE approximation_reliability";
            com.mysql.jdbc.PreparedStatement ds =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sql);
            ds.executeUpdate();
            dispose();
            reliabilitytable pertr = new reliabilitytable();
            pertr.setVisible(true);
            pertr.getdata();
            setSize(350, 250);
        } catch (ClassNotFoundException ex) {

Logger.getLogger(percentagereliability.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SQLException ex) {

Logger.getLogger(percentagereliability.class.getName()).log(Level.SEVERE,
null, ex);
        }

    }//GEN-LAST:event_jButton1ActionPerformed

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosing
        try {
            // TODO add your handling code here:
            shutdownSoftware();
        } catch (ClassNotFoundException ex) {

Logger.getLogger(percentagereliability.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SQLException ex) {

Logger.getLogger(percentagereliability.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }//GEN-LAST:event_formWindowClosing

    private void shutdownSoftware() throws ClassNotFoundException,
SQLException {
```

```
        if (JOptionPane.showConfirmDialog(this, "Do you realy want to quit?",
"Confirmation!", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION) {
            Class.forName("com.mysql.jdbc.Driver");
            com.mysql.jdbc.Connection con = (com.mysql.jdbc.Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/reliability",
"root", "root");
             String sqlTrun = "TRUNCATE TABLE approximation_reliability";
               com.mysql.jdbc.PreparedStatement dsTrun =
(com.mysql.jdbc.PreparedStatement) con.prepareStatement(sqlTrun);
               dsTrun.executeUpdate();
            setVisible(false);
            dispose();
            System.exit(0);
        }
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    // End of variables declaration//GEN-END:variables
}
```

Database Structure

Database: `reliability`

TABLES

1. Table structure for table `reliabiltity_availbility`

```
CREATE TABLE IF NOT EXISTS `reliabiltity_availbility`( `id` int(11) unsigned
NOT NULL AUTO_INCREMENT,`program_name` varchar(255) NOT NULL,
`uptime_at_x1` varchar(255) NOT NULL, `uptime_at_x2` varchar(255) NOT
NULL, `downtime_at_x1` varchar(255) NOT NULL, `downtime_at_x2`
varchar(255) NOT NULL, `no_of_breakdowns_at_x1` varchar(255) NOT NULL,
`no_of_breakdowns_at_x2` varchar(255) NOT NULL, `mean_time_to_failure`
varchar(255) NOT NULL, `mean_time_to_repair` varchar(255) NOT NULL,
```

```
`mean_time_between_failures` varchar(255) NOT NULL, `total_ production_ time` varchar(255) NOT NULL, `availability1` varchar(255) NOT NULL, `availability2` varchar(255) NOT NULL, `reliability` varchar(255) NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=20 ;

-- Dumping data for table `reliabiltity_availbility`

INSERT INTO `reliabiltity_availbility` (`id`, `program_name`, `uptime_at_x1`, `uptime_at_x2`, `downtime_ at_x1`, `downtime_at_x2`, `no_of_breakdowns_at_x1`, `no_of_breakdowns_at_x2`, `mean_time_ to_ failure`, `mean_time_to_repair`, `mean_time_between_failures`, `total_production_time`,`availability1`, `availability2`, `reliability`) VALUES
(1, 'GE01', '216', '202', '40', '54', '3', '11', '5.53', '9.12', '45.18', '256.0', '0.83', '0.38', '0.98'),
(2, 'GE02', '260', '203', '64', '121', '9', '16', '10.88', '7.34', '20.79', '324.0', '0.74', '0.6', '0.95'),
(3, 'GE03', '168', '154', '68', '82', '2', '19', '8.82', '19.16', '46.05', '236.0', '0.71', '0.32', '0.98'),
(4, 'GE04', '450', '435', '150', '165', '16', '23', '18.53', '8.27', '23.52', '600.0', '0.74', '0.69', '0.96'),
(5, 'GE05', '300', '265', '71', '106', '13', '35', '10.41', '4.25', '15.32', '371.0', '0.78', '0.71', '0.94'),
(6, 'GE06', '430', '410', '17', '37', '15', '21', '3.18', '1.45', '24.1', '447.0', '0.94', '0.69', '0.96'),
(7, 'GE07', '560', '525', '305', '340', '10', '25', '37.94', '22.05', '38.5', '865.0', '0.64', '0.63', '0.97'),
(8, 'GE08', '615', '575', '228', '268', '4', '31', '29.18', '32.82', '86.15', '843.0', '0.72', '0.47', '0.99'),
(9, 'GE09', '720', '706', '223', '237', '17', '28', '27.06', '10.79', '33.78', '943.0', '0.76', '0.71', '0.97'),
(10, 'GE10', '85', '78', '50', '57', '4', '6', '6.29', '11.0', '17.13', '135.0', '0.61', '0.36', '0.94'),
(11, 'GE11', '130', '132', '112', '110', '36', '22', '13.06', '4.06', '4.81', '242.0', '0.54', '0.76', '0.83'),
(12, 'GE12', '240', '206', '129', '163', '24', '30', '17.18', '5.4', '8.43', '369.0', '0.61', '0.76', '0.89'),
(13, 'GE13', '68', '64', '54', '58', '23', '9', '6.59', '4.4', '5.03', '122.0', '0.53', '0.6', '0.83'),
(14, 'GE14', '72', '74', '35', '33', '6', '15', '4.0', '4.02', '8.47', '107.0', '0.68', '0.5', '0.89'),
(15, 'GE15', '265', '253', '106', '118', '18', '34', '13.18', '4.68', '11.08', '371.0', '0.7', '0.74', '0.92'),
(16, 'GE16', '370', '398', '83', '55', '21', '37', '8.12', '2.72', '14.19', '453.0', '0.84', '0.75', '0.93'),
```

(19, 'GE17', '285', '256', '40', '69', '27', '29', '6.41', '1.93', '9.69', '325.0', '0.83', '0.77', '0.91');

2. Table structure for table `approximation_reliability`

CREATE TABLE IF NOT EXISTS `approximation_reliability`(`approx_id` int(11) unsigned NOT NULL AUTO_INCREMENT, `mean_time_between_failure` decimal(10,3) NOT NULL,`reliability_function` decimal(10,5) NOT NULL, `approximated_value` decimal(10,3) NOT NULL, PRIMARY KEY (`approx_id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1;

Hardware/Software Requirements:

1. Netbeans 8.
2. Java JDK 1.8.
3. MySQL 5.5 / MySQL Workbench 5.2 CE(Community Edition)
4. RAM : 2 GB
5. Processor : Intel Dual Core or above
6. Size on Disk : 248 KB(44 Files, 14 folders)

Execution Flow of Automated Program

Database Import

Goto start menu -> Choose MySQL Workbench 5.2 CE -> Click on local instance MySQL -> click on Manage Import/Export -> select server as Local MySQL -> click ok -> give the password as 'root' -> Workbench will be loaded -> click on data import/restore -> click on 'import from self contained file' and choose path of .sql file of our program ->click on start import -> the database will be imported into MySQL workbench with name as 'reliability'.

Project Configuration in NETBEANS

Go to start menu -> from the programs choose NetBeansIDE 8.0 -> IDE will be loaded -> click on File -> click on open project -> browse for the project folder -> open project -> make sure that the libraries like 'mysql-connector-java-5.1.6-bin.jar' and 'JTattoo-1.6.11.jar' are properly exists or not. ->Right click on project folder -> choose either build or clan and build option for creating project '.jar' file in project 'dist' folder -> To run the project, right click

on project and choose run or press 'F5' from the keyboard -> The program will be opened.

Project Execution Sequence

When the program runs, first screen will be the main screen as shown in the Figure 1.

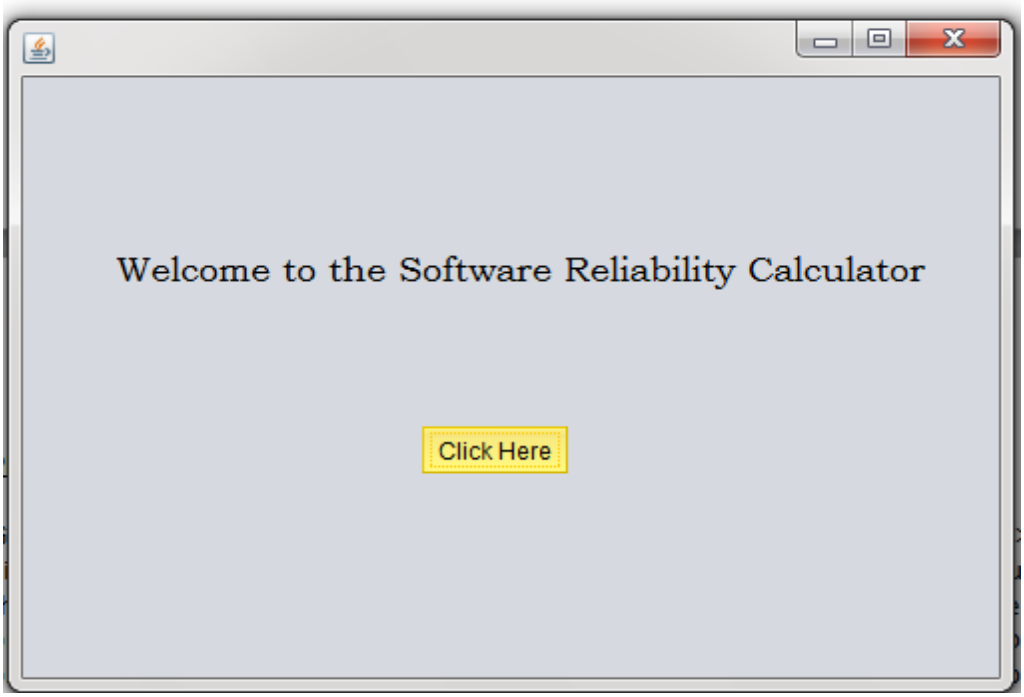

Figure 1: Welcome Screen

Click on 'click here' option, then the login screen will be appeared as shown in the Figure 2.

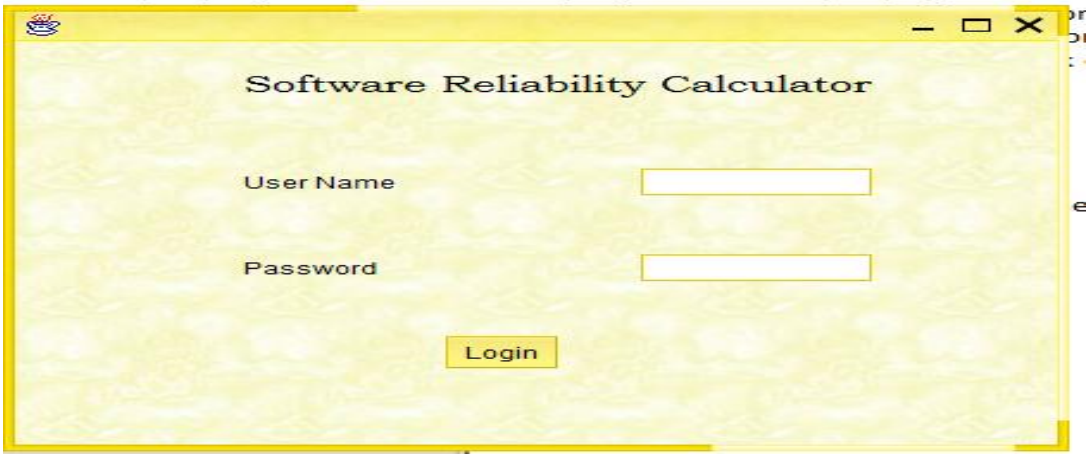

Figure 2: Login Screen

Supply the user name as 'software' and password as '123456'. Then the dataset for the software system will be displayed as follows and it is shown in the Figure 3.

DATA SET OF SOFTWARE SYSTEM

Calculate New | Calculate Approximated Value | Refresh | Delete All Records | Exit | Search Program | Delete program

ABBREVATIONS

UAX1=>UPTIME AT X1(HRS.) PROG#=>PROGRAM# MTTR=>MEAN TIME TO REPAIR

UAX2=>UPTIME AT X2(HRS.) NOBX1=>NO. OF BREAKDOWNS AT X1(HRS.) MTBF=>MEAN TIME BETWEEN FAILURES AFNRHS=>AVAILABILITY FOR NON- REPAIRABLE HARDWARE SYSTEM

DAX1=>DOWNTIME AT X1(HRS.) NOBX2=>NO. OF BREAKDOWNS AT X2(HRS.) MTTF=>MEAN TIME TO FAILURE

DAX2=>DOWNTIME AT X2(HRS.) TPT=>TOTAL PRODUCTION TIME AFRSS=>AVAILABILITY FOR REPAIRABLE SOFTWARE SYSTEM

ID	PROG#	UAX1	UAX2	DAX1	DAX2	TPT	NOBX1	NOBX2	MTTF	MTTR	MTBF	AFRSS	AFNRHS	RELIABILITY
1	GE01	216	202	40	54	256.0	3	11	5.53	9.12	45.18	0.83	0.38	0.98
2	GE02	260	203	64	121	324.0	9	16	10.88	7.34	20.79	0.74	0.6	0.95
3	GE03	168	154	68	82	236.0	2	19	8.82	19.16	46.05	0.71	0.32	0.98
4	GE04	450	435	150	165	600.0	16	23	18.53	8.27	23.52	0.74	0.69	0.96
5	GE05	300	265	71	106	371.0	13	35	10.41	4.25	15.32	0.78	0.71	0.94
6	GE06	430	410	17	37	447.0	15	21	3.18	1.45	24.1	0.94	0.69	0.96
7	GE07	560	525	305	340	865.0	10	25	37.94	22.05	38.5	0.64	0.63	0.97
8	GE08	615	575	228	268	843.0	4	31	29.18	32.82	86.15	0.72	0.47	0.99
9	GE09	720	706	223	237	943.0	17	28	27.06	10.79	33.78	0.76	0.71	0.97

Figure 3: Dataset for software system, showing Program no., uptime, down time, No. of breakdowns at time x1 and x2, total production time and calculated MTTF, MTTR, MTBF, Availability for both software and hardware systems, reliability.

The researcher can search a specific program by using search option.

Next choose calculate new option to add new dataset, a screen for adding new dataset as follows and it is shown in the Figure 4.

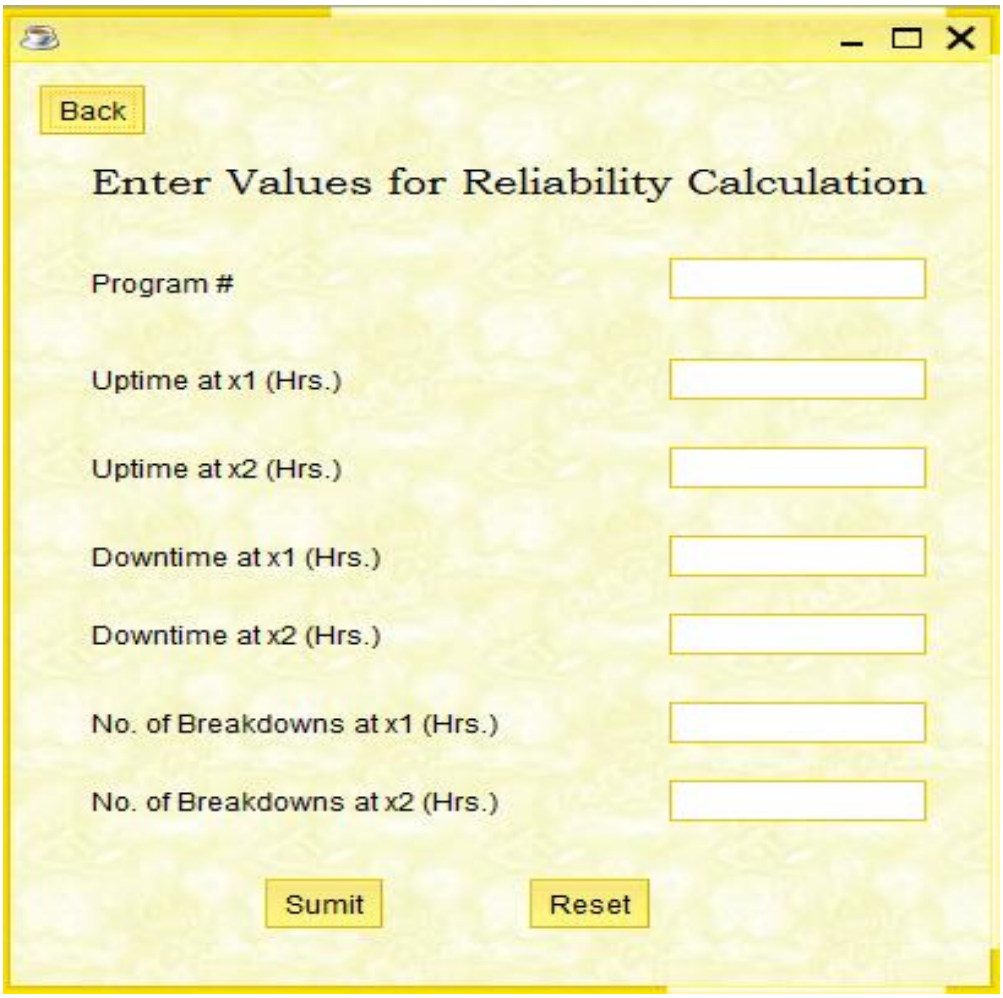

Figure 4: Dataset Add, Navigation, Reset screen

After adding the data, click 'submit' to load the dataset into database.

click 'reset' to clear all the data from dataset entry form.

click 'back' button anytime to go and see the list of datasets and assessed values.

click on 'calculate approximated value' to get the approximated value of Software Reliability as per mathematical Euler's theorem based formula.

click 'exit' to come out of the program

click ‘refresh’ to get back to datasets when no records matched with search criteria.

click ‘delete’ to delete the record after searching. option ‘delete all records’ can be used to delete all the records of dataset from the database.

Before the approximation, it will ask for how many iterations you need to perform. After supply input, the Approximated results, after the required number of steps are shown in the following Figure 5.

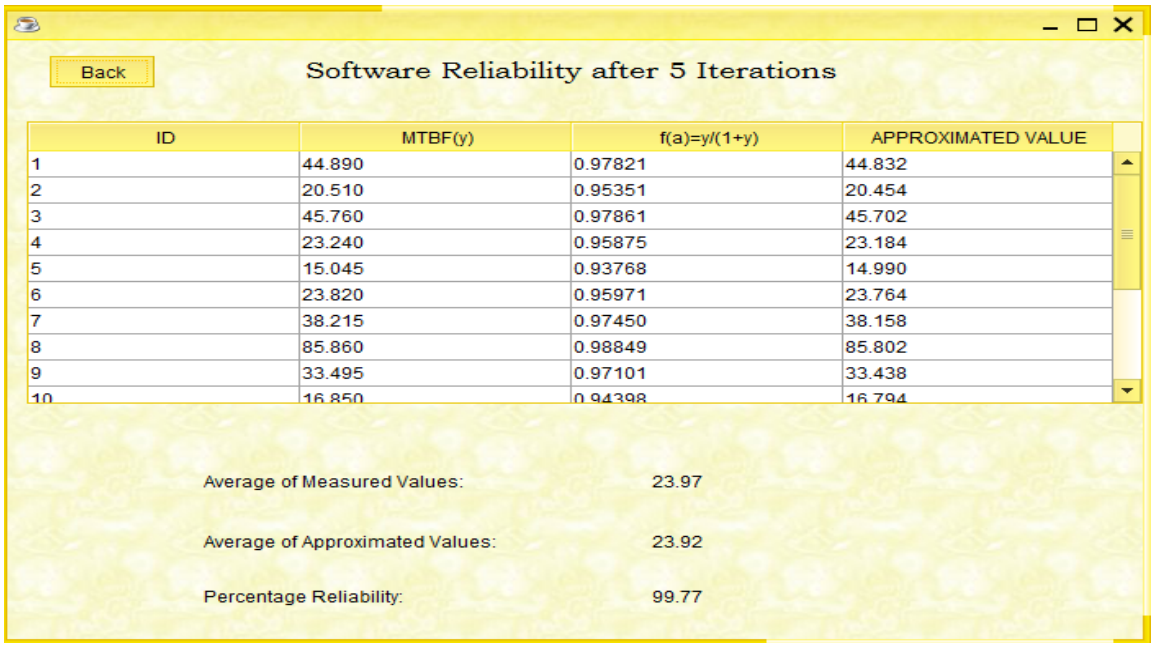

Figure 5: Approximated Value of Software Reliability

Appendix - C

Table 1: Chi-Square Distribution Table

Degrees of Freedom	99%	95%	90%	70%	50%	30%	10%	5%	1%
1	0.00016	0.0039	0.016	0.15	0.46	1.07	2.71	3.84	6.64
2	0.020	0.10	0.21	0.71	1.39	2.41	4.60	5.99	9.21
3	0.12	0.35	0.58	1.42	2.37	3.67	6.25	7.82	11.34
4	0.30	0.71	1.06	2.20	3.36	4.88	7.78	9.49	13.28
5	0.55	1.14	1.61	3.00	4.35	6.06	9.24	11.07	15.09
6	0.87	1.64	2.20	3.83	5.35	7.23	10.65	12.59	16.81
7	1.24	2.17	2.83	4.67	6.35	8.38	12.02	14.07	18.48
8	1.65	2.73	3.49	5.53	7.34	9.52	13.36	15.51	20.09
9	2.09	3.33	4.17	6.39	8.34	10.66	14.68	16.92	21.67
10	2.56	3.94	4.86	7.27	9.34	11.78	15.99	18.31	23.21
11	3.05	4.58	5.58	8.15	10.34	12.90	17.28	19.68	24.73
12	3.57	5.23	6.30	9.03	11.34	14.01	18.55	21.03	26.22
13	4.11	5.89	7.04	9.93	12.34	15.12	19.81	22.36	27.69
14	4.66	6.57	7.79	10.82	12.34	16.22	21.06	23.69	29.14
15	5.23	7.26	8.55	11.72	14.34	17.32	22.31	25.00	30.58
16	5.81	7.96	9.31	12.62	15.34	18.42	23.54	26.30	32.00
17	6.41	8.67	10.09	13.53	16.34	19.51	24.77	27.59	33.41
18	7.00	9.39	10.87	14.44	17.34	20.60	25.99	28.87	34.81
19	7.63	10.12	11.65	15.35	18.34	21.69	27.20	30.14	36.19
20	8.26	10.85	12.44	16.27	19.34	22.78	28.41	31.41	37.57

Source: Adapted from p.112 of Sir R.A. Fisher, *Statistical Methods fro Research Workers* (Edinburgh: Oliver and Boyd, 1958).