Application of soft computing techniques for software reliability prediction

A Dissertation submitted in the partial fulfilment for the award of

Degree of Master of Technology

In

Software Engineering

By

DEEPAK KUMAR VINODIA

2K15/SWE/07

Under the Esteemed Guidance of:

Dr. Ruchika Malhotra

Associate Head & Assistant Prof.

Department of CSE, DTU

DEPARTMENT OF COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI

CERTIFICATE

This is to certify that a thesis entitled **"Application of soft computing techniques for software reliability prediction"** by **Mr. Deepak Kumar Vinodia** bearing Roll No. **2k15/SWE/07,** is an authentic work which has been carried out under my supervision. The thesis is completed in partial fulfillment of award for Degree in **Master of Technology** in **Software Engineering** at **Delhi Technological University**. The content embodied in this thesis has not been submitted elsewhere for the award of any degree to the best of my knowledge and belief.

(Dr. Ruchika Malhotra)

Associate Head and Assistant Professor

Department of Computer Science and Engineering

Delhi Technological University

Delhi -110042

DECLARATION

I hereby declare that a thesis entitled **"Application of soft computing techniques for software reliability prediction"** is a bonafide work carried out by me. The thesis is completed in partial fulfillment of award for Degree in **Master of Technology** in **Software Engineering** at **Delhi Technological University**. The content embodied in this thesis has not been submitted to elsewhere for the award of any degree.

Deepak Kumar Vinodia

M. Tech. in Software Engineering

Department of Computer Science and Engineering

Delhi Technological University

Delhi -110042

ACKNOWLEDGEMENT

I take this opportunity to express my deep sense of gratitude and respect towards my guide Dr. Ruchika Malhotra, Department of computer science and engineering, DTU.

I am very much indebted for her generosity, expertise and guidance which I received while working on this thesis. Without her support and guidance this work would not have been completed. She has not only guided me throughout the work but also provided valuable suggestions about documentation and presentation. I feel blessed to have such a guide.

I would like to express my gratitude towards the University for providing me Infrastructure, Laboratories, testing facilities and environment which allowed me to work without any obstructions.

I would also like to appreciate the support provided by lab assistants, research scholars and peer group who aided me with all the knowledge they had regarding various topics.

I would also like to thank Almighty God with her blessings I had this opportunity and strength to work on this wonderful thesis and studies. Last but not least, I would like to thank my parents who supported and guided me unconditionally throughout the journey of my life.

Deepak Kumar Vinodia

M. Tech. in Software Engineering

Department of Computer Science and Engineering

Delhi Technological University

Delhi -110042

TABLE OF CONTENTS

List of Tables

List of Figures

ABSTRACT

***Background**:* Software reliability prediction has become a key activity in the field of software engineering. It is the process of constructing models that can be used by software practitioners and researchers for assessing and predicting the reliability of the software product. This activity provides significant information about the software product such as "when to stop testing" or "when to release the software product" and other important information. Thus, effective reliability prediction models provide critical information to software stakeholders.

Method: In this paper, we have conducted a systematic literature review of studies from the year 2005 to 2016, which use soft computing techniques for software reliability prediction. The studies are examined with specific emphasis on the various soft computing techniques used, their strengths and weaknesses, the investigated datasets, the validation methods and the evaluated performance metrics. The review also analyses the various threats reported by software reliability prediction studies and statistical tests used in literature for evaluating the effectiveness of soft computing techniques for software reliability prediction.

Results: After performing strict quality analysis, we found 31 primary studies. The conclusions made based on the data taken from the primary studies indicate wide use of public datasets for developing software reliability prediction models. Moreover, we identified five most commonly used soft computing techniques for software reliability prediction namely, Neural Networks, Fuzzy Logic, Genetic Algorithm, Particle Swarm Optimization and Support Vector Machine.

Conclusion: This review summarizes the most commonly used soft computing techniques for software reliability prediction, their strengths and weaknesses and predictive capabilities. The suitability of a specific soft computing technique is an open issue as it depends heavily on nature of the problem and its characteristics. Every software project has its own growth behavior and complexity pattern. Hence, more number of studies should be conducted for the generalization of the results. The review also provides future guidelines to researchers in the domain of software reliability prediction.

Keywords: Software reliability, soft computing technique, software quality, reliability prediction

CHAPTER 1

INTRODUCTION

1.1 Introduction

Reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time [1]. Software may face failures during its operational phase if bugs are present. Underestimation of the complexity of the projects may cause schedule and cost overruns. These overruns are primary concerns for project stakeholders and can adversely affect the quality of software. Thus, assessing and predicting software reliability is very important to ensure the quality of the final software product. It provides crucial information about the software product. It can be used to decide the amount of testing and release date of the software product by management team. Many studies have been conducted in literature for prediction of software reliability. As a result, hundreds of reliability prediction models are produced to estimate and predict the reliability pattern. However, there is not a single model that can be classified as a generic model which can be easily adapted to different project attributes in all conditions. This is because every project has their own growth behaviour and complexity patterns.

Researchers have proposed the application of various Soft Computing Techniques (SCT) to software reliability prediction [2, 3]. SCT are typically useful for solving real life problems. These techniques are well suited in situations where problems are imprecise, uncertain and partially incorrect [4]. The implementation of SCT for assessing and predicting software reliability is beneficial because they can solve nonlinear real world problems only by using software failure history data. They do not need to consider any assumptions about organization, development process and nature of faults or their complexity. Previous studies have already established that the use of various SCT for assessing and predicting the software reliability has shown significant improvement over other traditional techniques [2, 3, 5]. SCT like Artificial Neural Networks (ANN), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Support Vector Regression (SVR) and Fuzzy Logic (FL) have been used for software reliability prediction because they produce more precise estimation results as compared to statistical or other traditional prediction techniques.

Though, a number of studies have successfully used SCT for software reliability prediction, in order to identify research gaps and to provide future guidance to researchers, it is necessary to conduct a systematic literature review (SLR) of the existing literature in this domain. Thus, this thesis conducts a SLR of studies from 2005 to 2016 that use SCT for software reliability prediction. We summarize the current trends in the domain by extracting data from 31 primary studies. These studies were selected after application of strict quality assessment. We summarized current trends related to (1) most commonly used SCT, (2) strengths and weaknesses of the techniques (3) datasets investigated (4) validation methods used (5) performance metrics evaluated (6) predictive capabilities of SCT (7) statistical tests used and (8) the

identified threats for software reliability prediction. We have further provided future guidelines for the researchers in this domain.

1.1.1 SOFTWARE RELIABILITY

Today, the world around us is computerized. Along with this, many important activities are computer controlled, such as economical transactions, medical monitoring etc. Our dependencies on computer applications are increasing day by day. Thus the correct response of the software is very much essential. Hence, software reliability has become the most important quality characteristics of the software. In order to assess a particular reliability level, first we need to find current reliability of the software. The characteristics of the software varies depending on many factors like type of software, operating environment etc. there is a need to establish a standardization of the software which represents the main purpose of the software Thus achieving desired software reliability with meeting user requirements are of great interest. “When should testing stop” or “when to release the software” are very relevant questions for different stakeholders of the software. Extensive testing is required before the software is ready to be released in the market. Extensive testing requires huge cost and time. Cost calculations and testing effort calculation has become mandatory activities in context of software engineering. It is necessary to find the milestone when the software reliability is of satisfactory and the costs are realistic. This involves the activities of software reliability estimation and prediction. Prediction is required to manage the software development and testing in respect of costs. First, the estimation of current reliability level is required and then prediction can be made for future reliability. The following activities are important with respect to software reliability;

- Mapping of application for specifying the usage
- Testing based on the specified operational profile
- Modelling the results of testing in order to estimate the current level of reliability.
- Predicting the future software reliability to restrict the software testing and development efforts.

1.1.2 SOFTWARE RELIABILITY GROWTH MODEL

Software reliability is defined as the probability of failure free software operation for a specified period of time in a specified environment. The assessment of software reliability is one of the most important activity during the development of the software. A failure is a departure of system behaviour in execution from user requirements and it is the result of a fault. A fault is a defect that causes or can potentially cause the failure when executed. Software reliability growth models (SRGM), are those models that attempt to predict software reliability from testing data of the software. These models try to find a correlation between testing data and functions like logarithmic or exponential functions. The effectiveness of these models are depending on the degree of correlation between mathematical function and testing data. SRGM are employed for estimating the software growth i.e. how software reliability improves as testing proceeds.

These models can be used for predicting the software reliability and also helps in deciding when to stop testing. SRGMs help in determination of many software development metrics such as failure rate, number of faults in the beginning, number of residual faults, reliability value within a specified interval of time period, release time and cost analysis etc. The utility of a SRGM is dependent to its predictive ability and stability. The prediction of number of faults remaining should be as close as actual faults discovered during the field test (predictive ability). Model parameters should not change majorly as the new data is included (Stability).

Many SRGMs have a parameter that refers to the total number of defects present in a software program. If this parameter is known and the existing number of defects discovered is also known then we can know that how many more defects are remaining in the program. (Figure 1.1). If we know the remaining defects in the program, it helps in decision making in deciding the release date of the software. Also, it helps in knowing that how much testing is required in future.. It provides an estimation about the number of faults that the customers will experience while operating the software. This estimation helps in deciding the resources required during the maintenance phase of the software.

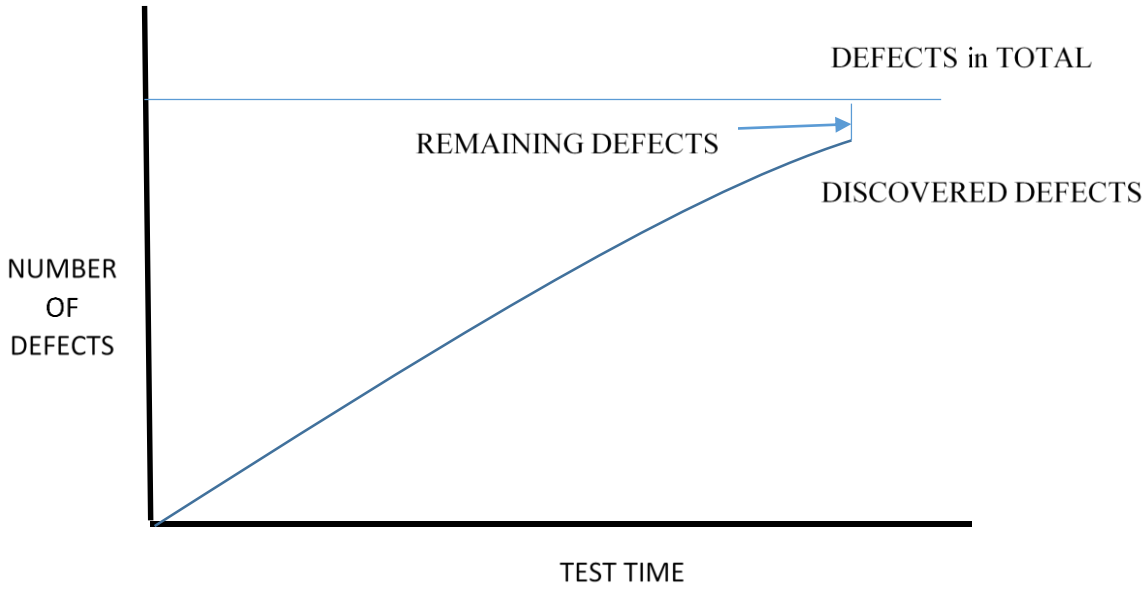

Figure 1.1: REMAINING DEFECTS

1.2 Motivation

Software reliability prediction is the process of developing models that can be used by software practitioners for assessing and predicting the reliability of the software product. Software reliability prediction is very essential activity for the development of quality software. Many models have been developed for assessing and predicting the software reliability. SCTs have been used for solving real life problems efficiently and effectively. These techniques are well suited where the problems are uncertain, imprecise and partially

incorrect. The application of SCTs for software reliability prediction have shown enormous improvement over other traditional techniques.

Several Soft Computing techniques such as ANNs, FLs, GAs, PSOs, and Hybrid approach have been proposed in the literature to solve classification and optimization problems. However, there is a lack of systematic literature review that provides comprehensive analysis about current trends related to the mostly used SCT, datasets, validation techniques and their strengths and weaknesses for software reliability prediction. Also, there is a lack of review which addressed the concern over threats to validity and statistical tests. There are five reasons to conduct this SLR.

1. To make general conclusions about the SCT used for software reliability prediction.
2. To analyse the methods, dataset trends, validation methods and statistical tests used in the studies based on usage of SCT for software reliability prediction.
3. To summarize strengths and weaknesses of SCT for software reliability prediction.
4. To identify the various threats to validity addressed in studies which use SCT for software reliability prediction.
5. To identify different statistical tests used in the studies.

1.3 Thesis Organization

This thesis is organised as: In Chapter 2, the related work and background is given. Chapter 2 includes related work about the existing literature on reviews and usage of SCT for software reliability prediction and background of software reliability.

In Chapter 3, the investigated research questions (RQs) and research methodology are given. This chapter includes various RQs formed and process of SLR followed throughout the conduction of the thesis.

In chapter 4, the results and answers of research questions (RQ's) are described. This chapter consists of discussion and findings while answering the RQs.

In chapter 5, threats to validity is presented. This chapter addresses the limitations of the study conducted as part of thesis.

In chapter 6, the conclusion and future directions are given in chapter 6. This chapter describes the summary of results and findings. Also, the chapter includes future directions to the researchers who are interested to work in the similar domain.

In Chapter 7, the appendix A.1 is provided. This appendix is a summary about the primary studies which includes 5 attributes (year, primary studies, Journal/conference, datasets and methods used) and 31 primary studies (from year 2005 to 2016).

In Chapter 8, References are provided

2.1.1 Systematic Literature Review Studies

Apoorva and Ankur (2011) [7] mainly focused on analysing Journals and Conferences publications on software reliability. They emphasised on the research methods, identification of journals and research topics selected for reliability studies. Afzal and Torkar (2011) [8] conducted a systematic review that analysed the effectiveness of genetic programming for developing predictive models in software engineering. However, the focused on only a specific SCT i.e. GP, while ignoring others. Jatain and Mehta (2014) [9] conducted a study which focused on the various parameters that affects the performance of the software reliability models. Sharma (2015) [10] performed a review of various software faults, detection of fault tolerance, performance testing, and evaluation of reliability of software systems. Ong L (2016) [11] has reviewed meta-heuristic techniques for software reliability prediction. He has analysed trends of meta-heuristics techniques, validation methods used and comparison criteria used for software reliability prediction. Malhotra R (2017) [12] has reviewed various search based techniques used for prediction purposes. This review focused on prediction of effort, maintainability, defect and change proneness. Yadav N (2016) [13] presented a review of different evolutionary algorithms used for software reliability prediction. He emphasised on role of well-known evolutionary algorithms for optimization of software reliability. Sangwan T (2017) [14] reviewed various intelligence approaches used for software reliability prediction. He focused mainly on metrics based approaches. The existing reviews are focused on some specific objectives like review by Yadav N mainly focused on evolutionary algorithms and review by Ong L mainly concerned with meta-heuristic techniques. Also, these review lack in some important aspects such as threats to validity and statistical tests concerns. So this review presents a comprehensive analysis of various SCTs used for software reliability prediction. This review involves trends in various SCT, dataset analysis, capability measures analysis, threats to validity concern, statistical tests concerns, strengths and weaknesses of mostly used SCT. Also, this review provides guidelines to future researchers who are interested to work in the similar domain.

2.1.2 Soft Computing Techniques Studies

Several SCTs such as ANNs, FLs, GAs, PSOs, and Hybrid approach have been proposed in theory to solve classification and optimization problems (Yogesh Singh. 2010, 2011; Sultan Aljahdali 2011; Mohammed E. El-Telbany 2008; TaehyounKim 2015; Malhotra et al. 2013; Manjubala Bisi 2015; Eduardo et al. 2010; Raj and Ravi 2007).

There are several studies which applied ANNs for software reliability prediction successfully (Yogesh Singh. 2010, 2011). However, effectiveness of NN based prediction models depend on the type of dataset

that is of changing nature. Therefore ANNs have the problem of overfitting the results. This happens when dealing with unknown data sets. Overfitting occurs mostly due to the reason that model gets tuned well with training data but for new data the performance of the model degrades. So overfitting is the main issue with NNs. The usage of fuzzy logic systems in software reliability prediction is found to be efficient and decisive. Because of huge computation and small learning rate of the model, the soft computing techniques based of FIS is more effective than other soft computing techniques (Sultan Aljahdali 2011). However, the challenge is to make it make it more efficient by employing new technique that require less resources and provide improved predictive accuracy.

The GAs soft computing technique have the capability of optimal parameters estimation in process of learning through historical data (Mohammed E. El-Telbany 2008). The proposed models are constructed using linear ensemble, However the non-linear models are also required to be built for performance comparison. The real-valued GA (RGA) provides improvement over simple GA technique (TaehyounKim 2015). Simple GA requires extra processes for encoding and decoding the chromosomes and binary genetic operator produces useless chromosomes that produces unstable parameter estimation. RGA eliminate the need of binary genetic operator and hence improve the performance over GA technique.

The Particle Swarm Optimization (PSO) has been applied successfully in many optimization problems. PSO is used for SRGM parameter estimation and comparative study is carried out (Malhotra et al. 2013). The results shows lesser mean errors for PSO than GA and hence better performance of PSO.

The hybrid approach (ANN-PSO) has shown better predictive capability (Manjubala Bisi 2015). A novel ANN based approach is proposed with an additional layer in between hidden and input layer for increasing the input values using Log function. PSO is used for training the ANN.

.

2.2 CLASSIFICATION OF SRGM

SRGMs have been classified into two categories of models: S-shaped and concave. Fig. 1.2 shows both types of models. The most important aspect of both models is that they demonstrate the same behaviour, that is, As the test time progresses (no. of defects detection and repairing increases), the defect detection rate decreases.

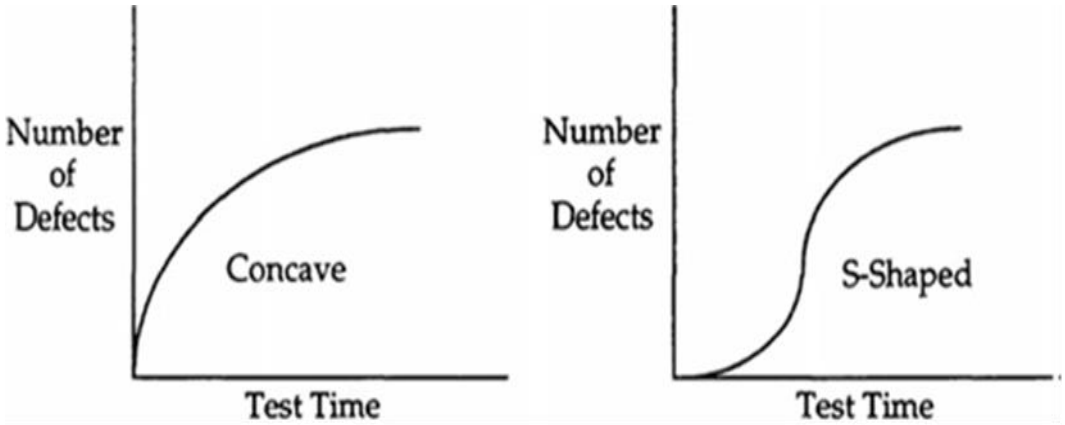

Figure 2.1: S – SHAPED AND CONCAVE MODEL

The measurement of the reliability of software can be done in many ways. Failure intensity is mostly used metric to describe software reliability. It is defined as the number of failures experienced per unit "time" period. It is also known as failure rate. Another measure for software reliability is the mean time to failure (MTTF). MTTF is calculated mostly by making the inverse of the failure rate. The computation of failure rate can be done for different cases (total discovered failures, total sole failures, or for some particular category of failures). Failure rate represents the user perception about the quality of software. When the software testing is under progress, the failure intensity decreases over the period of time**.** The software reliability can be described for systems with no repair as follows:

$$\mathbf{R(t) \sim e^{-Lt}} \text{ ---------- (1)}$$

Where,

R(t) denotes reliability of the system, and **"t"** is the mission time.

For example, suppose that the system is operational under specified conditions, and repairing of faults is not done.

Suppose the number of failures experienced over the operation of 10,000 hours is 8. Then, failure intensity is 0.0008 (8/10000 = 0.0008) failures per hour, and the related MTTF is about 1250 hours (1/0.0008 =1250). If the system operates as expected at time t=0 hours. We can find the reliability of the system functioning without failure for 10 hours using equation (1)

$$\mathbf{R(10) = e^{-0.0008*10} = 0.991.}$$

When the software is under test, failure intensity, decreases with time **t** during which software is under test and usage as per specified conditions. Most of the software reliability models address this aspect of software reliability, but before any model is selected, it is a necessary to ensure the presence of the growth using testing data. Each model has certain strengths as well as weaknesses. Thus, It is very important to choose a suitable model for a particular environment.

If failure intensity data is known, estimation of model parameters can be done. There are several ways of estimation. Two traditional methods for estimation includes maximum likelihood and least squares. There are two different ways of using a model. One way is to describe the historical data. The other is the prediction of future reliability measures and activities when the software is under testing or operation. Prediction can be done about some important activities like "when software can be released" or “when reliability will reach a specified value". Predictions are more useful for different stakeholders of the software product, but also it involves high risk. No single model can be recommended for all situations, and model performance can change significantly depending on different conditions.

2.3 Terms related to software reliability

Term	Explanation
M(t)	The total number of failures experienced by time t.
μ(t)	Mean value function for an SRGM. This represents the expectation of the number of failures expected by time t as estimated by the model.
L(t)	Failure intensity, representing the derivative of the mean value function.
Z(ΔT/ T(i-1))	Hazard rate of the software, which represents the probability density of experiencing the ith failure at ti-1+!t given that that (i-1)st failure occurred at ti-1.
z(t)	Per-fault hazard rate, which represents the probability that a fault, that had not been activated so far, will cause a failure instantaneously when activated. This term is usually assumed to be a constant (Ø) by many of the models.
N	Initial number of faults present in the software prior to testing.

2.4 Popular Software reliability Models

Following are the popular software reliability models

1) Jelinski-Moranda Model
2) Goel-Okumoto Model
3) Generalized Goel NHPP Model
4) Inflected S-Shaped Model
5) Logistic Growth Curve Model
6) Musa-Okumoto Model

These models are described briefly in subsequent sections.

2.4.1 Jelinski-Moranda Model

This model was introduced in 1972. It is one of the most popular SRGM. This is a continuous time-independently, identical error behaviour and distributed inter failure times model. The rate at which software failure occurs at any time is propotional to the current fault density of the program. The order statistics distribution is the Exponential distribution.

Following are the primary assumptions of the model:

1. When testing begins, there are constant no. of faults available in the program.
2. All faults belongs to the same category.
3. Faults are repaired immediately and perfectly.

4. The detection of faults is independent with each other.
5. The number of residual faults are exponentially distributed with time between failures.
6. The severity of the faults which causes failures are equal..
7. The failures are detected randomly, i.e. given N_0 and φ the times between failures differs (Δt_1 ,Δt_2 ,....,$\Delta t n_0$).
8. The faults are removed instantly, whenever failure occurs and repair does not introduce any new fault in the software.

$$Z(\Delta t \mid t_{i-1}) = \phi[n_0 - M(t_{i-1})] = \phi[n_0 - (i-1)] \ldots\ldots\ldots (1)$$

The failure intensity function can be described as the product of the residual faults and the probability density function of the time till activation of a single fault,

$$\frac{dm(t)}{d(t)} = n_0[1 - \exp^{(\phi t)}] \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Mean value function m(t) can be described as

$$m(t) = n_0\,[1 - \exp^{-\phi t}] \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The failure intensity function can be given as [from equations (2) and (3)],

$$\frac{dm(t)}{d(t)} = \phi[n_0 - m(t)] \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

As per eq. (4), the failure intensity of the software at time **t** is proportional to the probable no. of residual faults in the software. Several SRGMs can be stated in a form corresponding to equation (4). One of the most commonly discussed assumptions of the J-M model is the equation (2) due to the reason that this eq. shows that hazard rate decreases with repair of faults. This phenomena is shown in Figure 2.2.

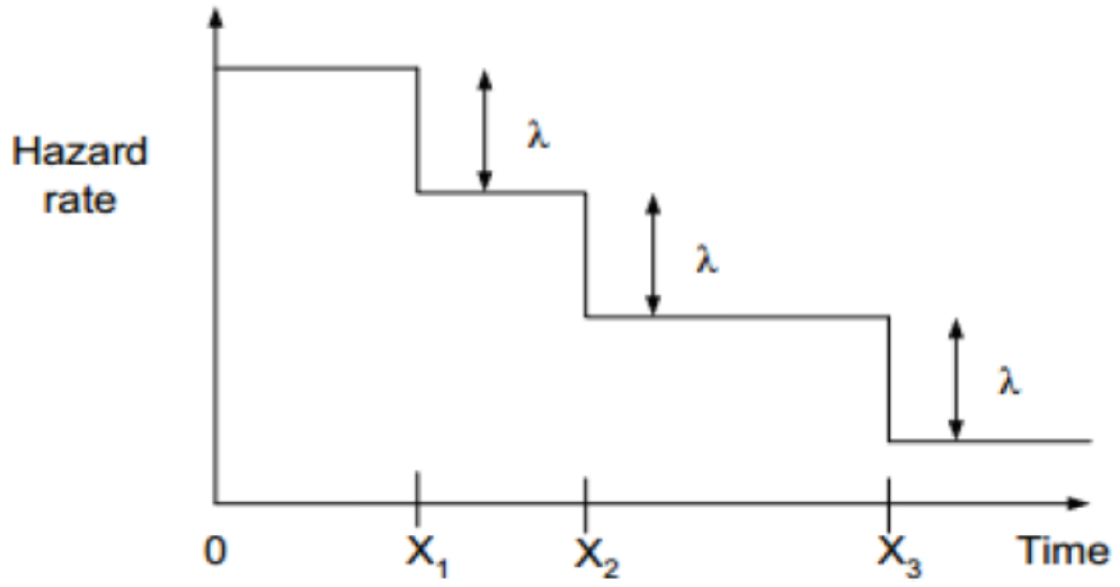

Figure 2.2: Hazard rate of Jelinski-Moranda Model

2.4.2 Goel-Okumoto Model

This model was suggested in year 1972. It is popularly known as Non homogeneous Poisson process (NHPP) model in the area of software reliability modeling. It is also called the exponential NHPP model. Above mentioned equations (2), (3) and (4) of the J-M model are also applicable for Goel-Okumoto model. Considering failure detection as a NHPP with an exponentially decreasing rate function, the mean value function can be expressed as:

$$m(t) = a(1-\exp^{(-bt)}),\ a>0, b>0$$

and the intensity function of this model is given as

$$\lambda(t) = ab*(\exp^{(-bt)}),\ a>0, b>0$$

Where parameter **a** is the probable total number of faults that would be detected and parameter **b** represents the rate of fault detection.

A typical plot of (t) for the Goel-Okumoto model is shown in Figure 2.3 where (t) is shown when $\boldsymbol{a}$ **= 11** and $\boldsymbol{b}$ **= 0.14**. The parameter **a** is related to the scale and **b** is related to the shape of the mean-value function.

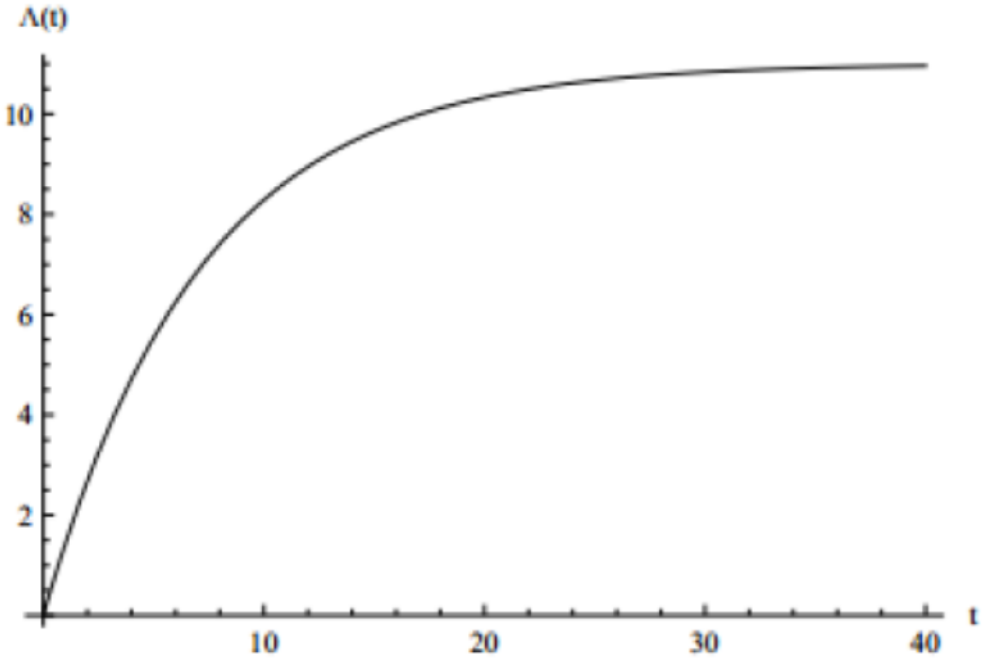

Figure 2.3: Mean Value function of G-O Model

2.4.3 Generalized Goel NHPP Model

Goel has proposed a model that is generalized form of G-O model. He has described the situation that software failure intensity increases marginally at the start and then begins to decrease as the testing proceeds. There is an additional parameter **c** in the Goel model. The mean value function and intensity function are given as follows:

$$m(t) = a\,(1-\exp[-bt]^c)\ ,\ a>0, b>0, c>0$$

$$\lambda(t) = [abct]^{(c-1)}\exp -[-bt]^c\ ,\ a>0, b>0, c>0$$

Where, **a** refers to the probable total number of faults that would be detected and **b** and **c** are parameters that refers to the quality of testing.

2.4.4 Inflected S-Shaped Model

This model provides solution to the technical problem of G-O model. It was proposed by Ohba and its main concept is that the detected software reliability growth follows S-shape if faults in a program are dependent mutually, i.e., some faults are not visible before some others are corrected. The mean value function of S-shaped model is as under:

$$m(t) = a * \frac{1-\exp[-bt]}{1+\psi(r)*\exp[-bt]} \qquad \psi(r) = \frac{1-r}{r}\ ,\ a>0,>0,>0$$

Where,

Parameter **r** is the inflection rate that denotes the ratio of the number of visible faults to the total number of faults in the software, **a** is the probable total number of faults that would be discovered, **b** is the rate of fault detection, and is the inflection factor.

If we take ψ r =β then the inflection S-shaped model mean value function and intensity function are given as follows:

$$(t)=a*\left[\ \frac{1-\exp[-bt]}{1+\beta*\exp[-bt]}\quad a>0, b>0, \beta>0\right.$$

$$\lambda(t) = \frac{abexp[-bt](1+\beta t)}{(1+\beta*\exp[-bt])2},\ a>0, b>0, \beta>0$$

2.4.5 Logistic Growth Curve Model

The general behaviour of the software reliability is that it improves as the testing proceeds, hence, this phenomenon can be considered as growth process. The growth happens due to detection and repair of faults during testing phase. Therefore, under specified conditions, the models that are developed to predict economic population growth could also be utilised to predict growth of software reliability. These models attempt to correlate the collective number of detected faults at a given time with a mathematical function

such as exponential or logarithmic. Logistic growth curve model is one of them and it follows an S-shaped. Its mean value function and intensity function are given as under:

$$m(t) = \frac{a}{1+k*\exp[-bt]},\ a>0,>0,k>0$$

$$\lambda(t) = \frac{abexp[-bt]}{(1+k*\exp[-bt])2},\ a>0,>0,k>0$$

Where, a is the probable total number of faults that would be discovered and **k** and **b** are estimated parameters using test data.

2.4.6 Musa-Okumoto Model

Musa-Okumoto have observed that the reduction in rate of failure resulting from detection and repair of faults during early failures are often bigger. This is because there is a tendency of most frequently occurring once. They incorporated this characteristics of faults in their model. The mean value function and intensity function of the model given as under:

$$m(t) = a * \ln(1+bt),\ a>0,>0$$

$$\lambda(t) = \frac{ab}{(1+bt)},\ a>0,>0$$

Where parameter **a** is the probable total number of faults that would be discovered and parameter **b** is the rate of fault detection.

CHAPTER 3

Research Methodology

3.1 METHODOLOGY

This section discusses the steps for conducting the review and states the research questions.

3.1.1 *Review Method*

A systematic approach for reviewing the literature on the software reliability prediction using SCT is chosen. Systematic literature reviews (SLR) have become a well-established method for analysing current trends and identifying research gaps in software engineering. An SLR is defined as a process of identifying, assessing, and interpreting all available research evidence with the purpose to provide answers for specific research questions [6]. This literature review has been undertaken as a SLR based on the original recommendations given by Charters and Kitchenham (2007).

As shown in Fig. 3.1, SLR is accomplished using three steps namely: planning phase, conducting phase and reporting phase. In the planning phase, the requirements are identified (Step 1). The goals for performing the review are presented during introduction of this paper. Then, the existing systematic reviews on software reliability prediction are identified and reviewed. The review protocol was designed in such a manner so as to reduce the chances of any preferences by the researcher (Step 2). Firstly, the research questions have been stated, the search strategy was developed, the process of study collection along with exclusion and inclusion criteria was defined. The quality evaluation of all the collected studies is performed, and finally data extraction is performed with respect to the research questions. The final phase includes reporting the results of the review according to the investigated research questions.

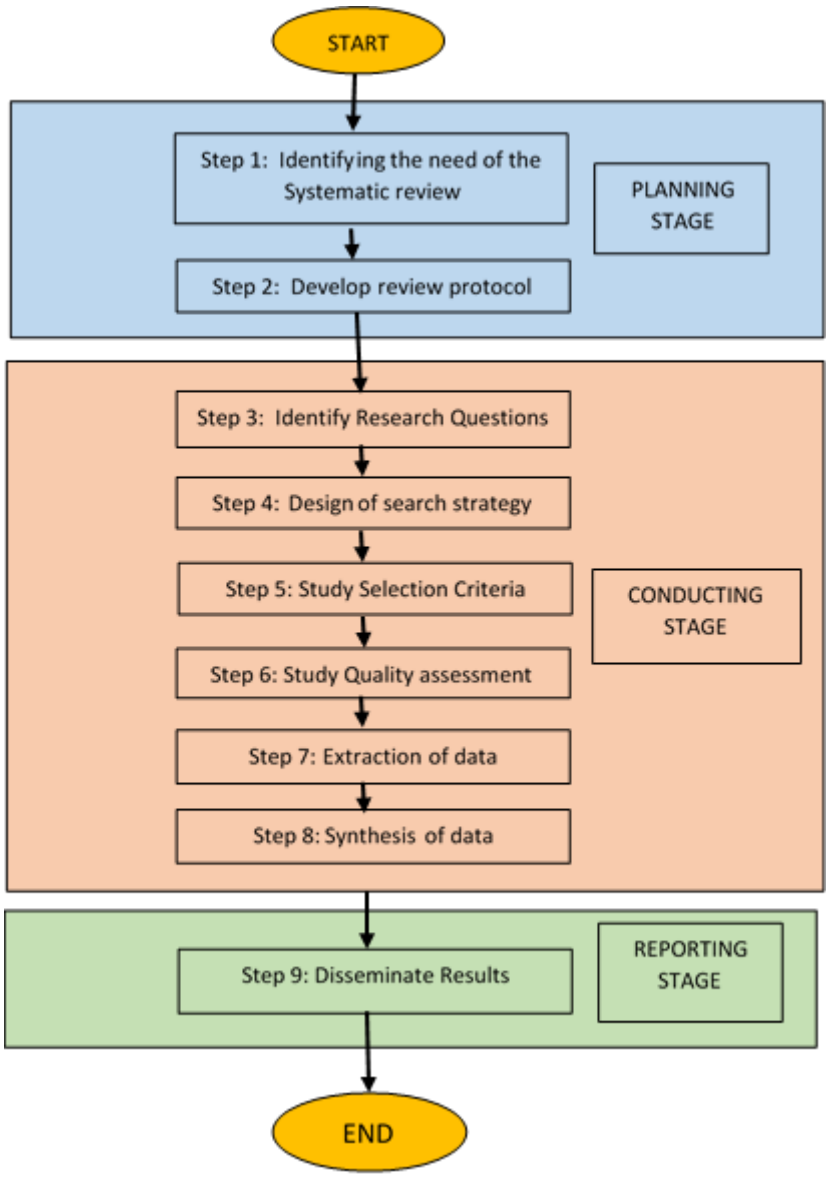

Figure 3.1: Steps involved in systematic review

3.1.2 *Research Questions*

Formulation of research questions is a very important step of review. It provides significant information to the readers about the review and what it investigates. These questions are primarily of interest to researchers. The research questions and motivation for selecting the research questions addressed by this literature review are listed in Table 3.1.

Table 3.1: RQ's on Literature review

ID	Research Questions	Motivation
RQ1	Which are the various SCT used in literature for software reliability prediction?	To identify the most commonly used SCT in the literature which are used in developing models for software reliability prediction
RQ2	Which data sets are used for developing software reliability prediction models using SCT?	To explore different datasets mostly used in software reliability prediction studies using SCT
RQ3	Which performance measures are used to evaluate software reliability prediction models developed using SCT?	To identify the most common performance measures investigated for software reliability prediction using SCT
RQ4	What is the predictive capability of mostly used SCT for software reliability prediction?	To estimate predictive capabilities of mostly used SCT for software reliability prediction.
RQ5	What are the strengths and weaknesses of various SCT for developing software reliability prediction models?	Identify the strengths and weaknesses of each SCT used for software reliability prediction
RQ6	Which validation methods are used for developing software reliability prediction models using SCT?	To identify various validation techniques mostly used in the literature for software reliability prediction using SCT.
RQ7	Which statistical tests have been used for comparing the predictive abilities of software reliability prediction models using SCT?	To identify the various statistical techniques used to compare the performance of developed software reliability prediction models using SCT
RQ8	What are the possible source of threats to validity for developing software reliability prediction using SCT?	Aggregate different threats to validity or limitations for software reliability prediction using SCT

The objective of this thesis is to analyse and provide empirical guidance based on the studies which use SCT for software reliability prediction. Table 3.1 shows the eight RQs addressed in this SLR. The data corresponding to all the RQs (RQ1 to RQ8) are extracted from the primary studies.

3.1.3 *Search Strategy*

The search method i.e. Step 4 involves activities such as choosing the digital libraries sources, describing the search terms, performing a model search, filtering the search terms and producing a preliminary list of major studies from various sources corresponding to the search terms.

The following five electronic databases were searched for collecting the primary studies:

1. IEEE xplore
2. Google scholar
3. ACM digital library
4. ScienceDirect
5. Web of Sciences

The search terms have been established based on the following steps:

1. Selecting search terms based on RQ's.
2. Selection of search terms based on related titles, keywords and abstracts.
3. Selection of substitutes, alternative antonyms of search terms.
4. Creation of refined search string by selecting suitable terms in conjunction with Boolean ANDs and ORs.

The search string is described as:

(Application OR Software * OR systems) AND (reliability * OR quality OR error-prone) AND (predict* OR prone* OR probability OR assess* OR detect* OR estimation* OR classification*) AND (soft computing*OR *Neural*OR Particle Swarm*OR*Genetic*OR*Fuzzy*OR*Hybrid*OR*Support Vector*) AND (Machine Learning*OR*ant*OR*perceptron) AND (meta-hueristics*ORdifferential*OR*evolutionary*)

3.1.4 *Inclusion and Exclusion criteria*

Deciding the proper inclusion and exclusion criteria is essential to assess the suitability of each chosen study. The selection or rejection of a study is derived from the criteria of exclusion and inclusion respectively. This criteria is shown in the Table 3.2.

Table 3.2: Criteria of Inclusion and Exclusion

Inclusion Criteria	Studies using SCT for software reliability Prediction
	Studies which performed comparative analysis of different methods for software reliability prediction
	Studies published in conference and journals
	Studies published during year 2005 to 2016
	Studies involving both small and large datasets in academia and industry
	Only journal publications are included for similar conference studies which have been extended to a journal
Exclusion Criteria	Studies which develop models other than software reliability prediction
	Studies pertaining to hardware reliability
	Studies without strong validation for software reliability prediction
	Studies other than software reliability prediction using software reliability growth models (SRGM)
	Studies which are based on software quality modelling
	Studies not presented in English

3.1.5 *Selection of relevant studies*

This study is focused on the literature based on the reliability prediction of software using SCT in their prediction processes. Other than that, studies that published online in the time frame of these 11 years, which are from year 2005 to 2016, are included to make sure only latest and updated studies are reviewed. The studies published in conference or journal are included because professional reviews have been done to these studies and the results or findings of the studies are reliable. We have included journal publication of study in case if the study is published in the conference and extended to journal.

The total number of papers collected from the various conference/Journal on the usage of SCT for software reliability prediction were 49. After applying criteria of exclusion and inclusion 42 studies were selected. Then, the full texts of 42 studies were analysed for further quality analysis. After analysing studies based on quality question assessment 11 studies were dropped. Finally, 31 studies were considered in this SLR.

3.1.6 *Quality assessment criteria*

We analysed the following quality questionnaire (Table 3.3) for studies assessing the quality of the studies. The table states 10 questions where each study was given a score of either NO (0), PARTLY YES (0.5) or YES (1) according to each quality question. The studies can have score of maximum 10 or minimum 0. A score of 10 shows that the study is having information to answer all the RQs and score 0 means the study cannot answer any of the investigated RQs.

TABLE 3.3: List of Quality Questions

Q #	Quality Questions	NO	PARTLY	YES
Q1	Are the objectives of the study clearly given?			
Q2	Is the data set size appropriate?			
Q3	Is the use of specific soft computing technique justified?			
Q4	Are the used SCT's clearly defined?			
Q5	Are the performance measures used to assess the Software reliability prediction model clearly defined?			
Q6	Are the findings and results described clearly?			
Q7	Does the study report its findings clearly?			
Q8	Are the validation methods defined clearly?			
Q9	Is there any comparative analysis performed between SCT's or between SCT's and statistical methods for software reliability prediction?			
Q10	Is the research methodology used in the study repeatable?			

3.1.7 *Data Extraction and data synthesis*

The data extraction was done by filling forms for each study. The form was utilised to extract the information about which RQs are answered by which studies. All the studies were tabulated as author name, dataset used, year of publication, method used and publishing details. These details were mentioned in the data extraction cards. These templates were used to collect the desired data. Further the data was stored in the excel file for data synthesis purposes.

The data synthesis involves gathering of information from all the studies with respect to a particular RQ and summarising the facts obtained from the analysis. Both kind of analysis quantitative as well as qualitative were performed for answering the RQs. Quantitative analysis involves values of performance measures and qualitative analysis involves strengths and weaknesses of SCT, classification of various SCT and data set used.

CHAPTER 4

Research Results and Discussion

4.1 Research Result and discussion

In this section, we have analysed various research questions formulated and discussed each RQ in depth. Answers to each RQ is provided with interpretation and discussion on the findings. This section also provides the overview of primary studies and their rankings based on quality assessment questions. This discussion provides significant information regarding the present state of research in the field of software reliability and propose future guidelines to the readers with respect to RQ's.

4.1.1 *Description of primary studies*

We considered 49 studies that applied SCT for software modelling and prediction. After evaluating the studies based on the criteria of exclusion and inclusion, 07 studies [43-49] were removed. We further analysed the studies with quality assessment questions and 11 studies [50-60] were removed after quality assessment. Finally, we included 31 primary studies [2, 3, 5, 15-42] for conducting the SLR.

4.1.2 *Publication Source*

Table 4.1 presents the journals of the selected primary studies. The number of studies in a journal along with their relative percentages are given. Maximum number of primary studies were found in the Journal of systems and software. The other Journal includes Int J of Syst Assur Eng Manage, J of comp Science etc. and conferences includes IEEE conference on reliability, Int conf on recent trends in computing etc. The 68 percentage of studies were taken from journals and 32 percentage of studies were taken from conferences. Hence majority of the studies were selected from journals.

Table 4.1: The details of publications

Publication Name	Type	Number	Percentage
The Journal of systems and software	Journal	5	16.66
International Journal of Systems Assur Eng Management	Journal	3	10.00
Journal of computer sciences	Journal	1	3.33
Baltic Journal of Modern Computing	Journal	1	3.33
ICGST-AIML Journal	Journal	1	3.33
International Journal of computer applications	Journal	1	3.33
International Journal of Advanced Resources in computer science and software engineering	Journal	1	3.33
International Journal of computer science issues	Journal	1	3.33
International Journal of Advanced computer science and applications	Journal	1	3.33
Journal of Industrial and Intelligent Information	Journal	1	3.33
Information and Software Technology	Journal	1	3.33
Journal of Neurocomputing	Journal	1	3.33
Applied Soft Computing	Journal	1	3.33
Journals of IET software	Journal	1	3.33
Expert systems with applications	Journal	1	3.33

4.1.3 *Quality Assessment Questions*

We have assigned scores to the questions of quality assessment. We have formed the categories according to the scores. The categories are: Extreme high(10<score<8), high(8< score <6), average(6< score <4), low(4< score <2), extreme low(2< score <0). The largest score that a study could get was 10 and the smallest was 0. The studies which obtained extreme low score were dropped from the SLR. Total number of 11 studies were dropped due to their low score factor. Six studies obtained the score in extreme high category (SR5, SR9, SR14, SR16, SR24, SR26). The readers who wish to study further on the usage of SCT for software reliability prediction should consider these studies.

Table 4.2 displays the unique ID given to each study with their references. The details of the studies consisting publishing year, method used, dataset used and year of publication is given in appendix A.1.

Table 4.2: Paper ID and the reference of primary studies

PAPER ID	PAPER	Reference	PAPER ID	PAPER	Reference
SR1	Pie(2006)	[15]	SR17	Nasar(2013)	[28]
SR2	Su(2006)	[16]	SR18	Rita(2013)	[29]
SR3	Zhang(2006)	[17]	SR19	Jin(2014)	[30]
SR4	Afzal(2008)	[18]	SR20	Mohanthy(2014)	[31]
SR5	Raj(2008)	[3]	SR21	Tyagi(2014)	[32]
SR6	Sultan(2008)	[19]	SR22	Bisi(2015)	[33]
SR7	Sebakhy(2009)	[20]	SR23	Indhurani(2015)	[34]
SR8	Yuan(2009)	[21]	SR24	Kim(2015)	[35]
SR9	Costa(2010)	[22]	SR25	Park(2015)	[36]
SR10	Singh(2010)	[23]	SR26	Roy(2015)	[37]
SR11	Jin(2011)	[24]	SR27	Yadav(2015)	[38]
SR12	Sultan(2011)	[25]	SR28	Arunima(2016)	[39]
SR13	Bisi(2012)	[26]	SR29	Bhuyan(2016)	[40]
SR14	Singh(2012)	[2]	SR30	Lou(2016)	[41]
SR15	Latha(2013)	[27]	SR31	Sheta(2016)	[42]
SR16	Malhotra(2013)	[5]			

4.1.4 *Publication Year*

The Fig 4.1 depicts the details about the publication year of studies. The period of studies selected if from year 2005 to 2016. The figure shows that the average number of studies conducted every year is 2. The highest number of studies were published in the year 2014 and 2015. This indicates increased interest of researchers in exploring soft computing techniques for software reliability prediction. The least number of studies were published in the year 2005.The figure indicates that the trends of conducting studies on using SCT for software reliability prediction has increased over the years.

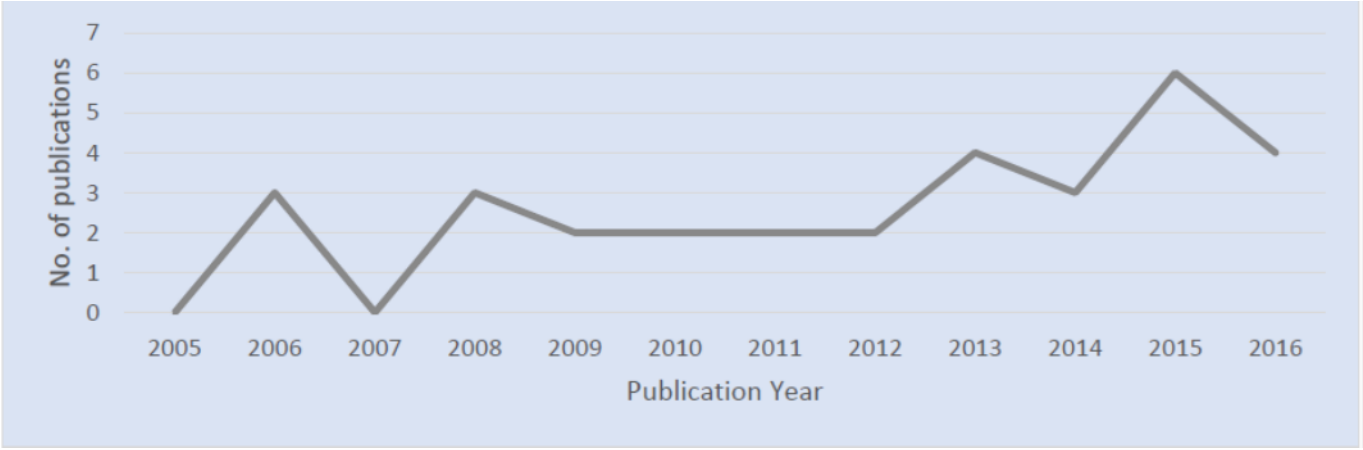

Figure 4.1: Year-wise distribution of studies

4.2 RQ1: Which are the various SCT used in literature for software reliability prediction?

During the period of 2005 to 2016, many soft computing methods have been applied and proposed as the method for prediction of software reliability. The identified SCT are as follows:-

- Neural Network (NN)
- Particle Swarm Optimization (PSO)
- Genetic Algorithm (GA)
- Fuzzy Logic (FL)
- Genetic Programming (GP)
- Ant Colony Optimization (ACO)
- Support Vector Machine (SVM)
- Differential Evolution (DE)
- Simulated Annealing (SA)
- Grey wolf optimization
- Decision TREE
- Bagging
- Hybrid Approach (NN+PSO, (GA-SA)+Support Vector Regression (SVR), Adaptive Neuro Fuzzy Inference System (AFNIS))

Fig 3 shows the distribution of various SCT used and Table 6 presents the number of studies and their relative percentages. The most frequent SCT used was NN. There were many variants found of NN technique including backpropagation neural network (BPNN), Pi–Sigma network (PSN), threshold-accepting-based neural network (TANN), generalized regression neural network (GRNN), Multi-layer perceptron (MLP) and multivariate adaptive regression splines (MARS). Nine primary studies used NN technique and its variants for software reliability prediction purpose (SR2, SR5, SR10, SR13, SR14, SR18, SR23,SR26, SR28). The top five SCT used were NN (25%), FL (11%), SVM (11%), GA (8%) and PSO (8%). There were studies which used relatively new optimization techniques such as Grey wolf

optimization (SR31), Decision Tree (SR25) and bagging (SR28). There were studies which used techniques that can be classified under evolutionary techniques (PSO, GA, GP, ACO, DE). Evolutionary techniques are inspired by nature and are population based. They are also known as meta-heuristics search techniques. The 52% of primary studies used evolutionary techniques for software reliability prediction. Evolutionary techniques requires multiple runs (minimum 10) for obtaining better results as these techniques are stochastic in nature. Very few studies (about 10%) which are under evolutionary category used multiple runs for obtaining the results. There were studies which used hybrid approaches for software reliability prediction. Hybrid approaches are used to enhance the performance of the developed models. The idea behind hybrid approach is to combine the best features of the constituent techniques. The hybrid approaches found in the studies are: ANN+PSO, (GA-SA)+SVM and AFNIS. The 10% of the studies used hybrid approaches for software reliability prediction purpose.

Table 4.3: SCT applied for reliability prediction of software

Method	Number of studies	Percentage
Neural Network (NN)	9	29.03
Support Vector Machine (SVM)	4	12.90
Fuzzy Logic (FL)	4	12.90
Genetic Algorithm (GA)	3	9.67
Particle Swarm Optimization (PSO)	3	9.67
Hybrid Techniques (NN+PSO, GA-SA+SVR, ANFIS)	3	9.67
Ant Colony Optimization (ACO)	2	6.45
Genetic Programming (GP)	2	6.45
Decision TREE (DT)	2	6.45
Support Vector Regression (SVR)	2	6.45
Grey wolf optimization (GWO)	1	3.22
Differential Evolution (DE)	1	3.22
Simulated Annealing (SA)	1	3.22
Bagging	1	3.22

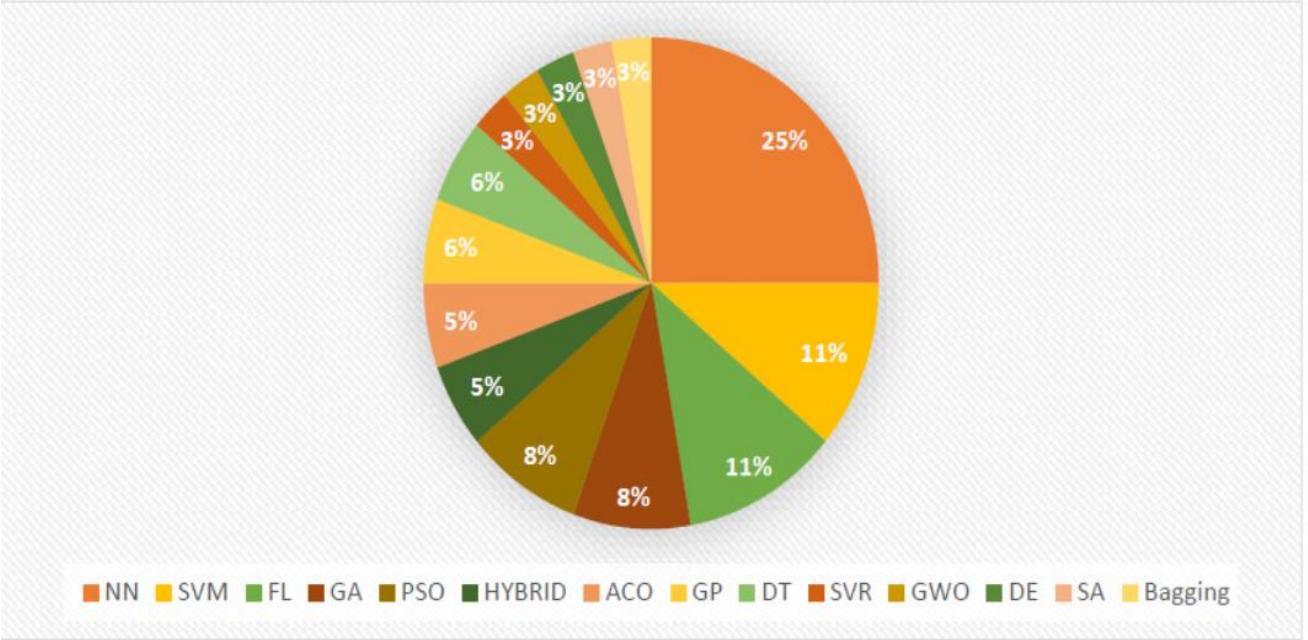

Figure 4.2: Distribution of SCTs Used

4.3 RQ2: Which datasets are used for developing software reliability prediction models using SCT?

In order to create a software reliability prediction model using SCT, a data set is required for training as well as validating the model. Thus, a dataset consists of two portions, first portion is a training set and second is a test set. A training set consists of data points that are provided as inputs to a learning system that analyses the data and learns from it for future predictions. A test set or evaluation set consists of data points that are used for assessing the model i.e. how well the model has learnt from the training data so that it can predict unseen data points correctly.

Table 4.4 presents the different datasets used along with their brief description and the studies which investigated them. Fig 4.3 shows the distribution of dataset types used in the studies. The dataset used are: DACS/Musa, Private/Industrial, Lyu and Wood. The percentage categorization of these datasets are: 58.06% of the research studies used DACS/Musa dataset, 32.25% of the research studies used private/industrial datasets, 6.45% of the research studies used Lyu dataset and 3.22% of the research studies used Wood dataset.

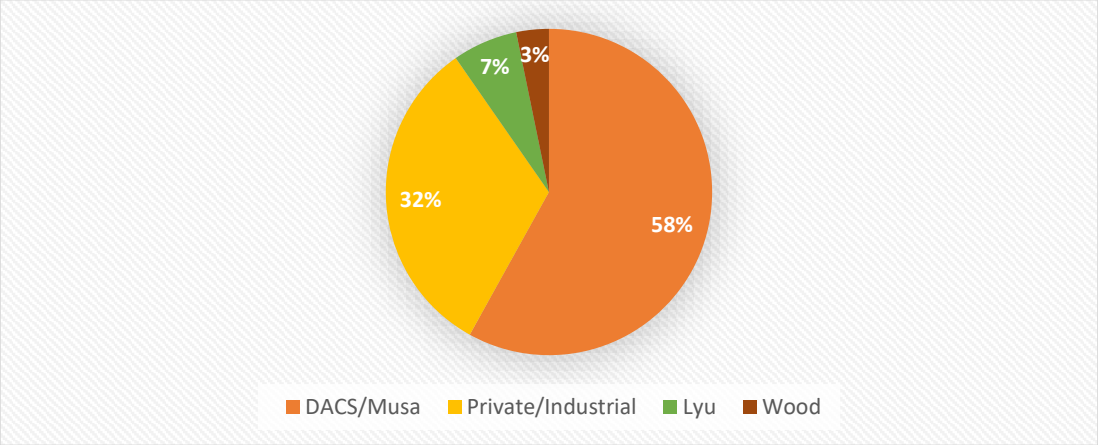

Figure 4.3: Distribution of dataset

Table 4.4: Commonly used Data sets

Data Set	Description	Studies used
DACS/Musa	This dataset is published by John Musa [5, 6]. It consists of software reliability data of 16 projects. These projects includes different fields like operating systems, real-time systems, word processing systems, etc.	SR1, SR2, SR5, SR6, SR7, SR9, SR10, SR12, SR14, SR15, SR16, SR 18, SR19, SR23, SR24, SR29, SR30
Private/Industrial	These datasets is not available in public. This dataset contains failure data of some specific applications. The example are: Armoured Force Engineering test data, classroom based project test data, Electronic Switching system test data etc.	SR3, SR4, SR8, SR11, SR20, SR21, SR22, SR28, SR31
Lyu	This dataset is a public dataset and contains three dataset DS1, DS2 & DS3. DS1 is a failure data of propulsion Laboratory project. DS2 is a failure data of Brazilian switching system and DS3 is failure data of real time command and control application [Lyu 1996].	SR25, SR26
Wood	This dataset is a public dataset and contains failure data of commercial applications [wood 1996].	SR25

4.4 RQ3: Which performance measures are used to evaluate software reliability prediction models developed using SCT?

Many performance measures are employed to gauge the correctness of the models proposed in the studies. These performance measures are used for comparison and evaluation of the models which are developed using SCT for software reliability prediction. Table 4.5 shows the used performance measures, their brief description and the studies in which they are evaluated. Fig 4.4 shows the distribution of performance measure used in the studies. It was found that the most commonly used performance measure was Root Mean Squared Error (RMSE) followed by Mean Squared Error (MSE). Both of these measures provides the differences between actual and predicted values. So, the lower the values of RMSE or MSE, the better is the performance of the developed model. The percentage distribution shows that 32% of studies used RMSE and 20% of studies used MSE as their performance measures.

Table 4.5: Performance measures used

Performance Measures	Description	Studies
RMSE	Root Mean Square is a measure that shows the difference between actual and predicted values. The formula for calculating RMSE is given as follows: $RMSE=\sqrt{\sum_{i=1}^{n}\frac{(predicted-actual)^2}{n}}$	SR3, SR7, SR9, SR13, SR14, SR16, SR18, SR22, SR23, SR28
AE / RE	Absolute Error and Relative Error are the measure used to compare the accumulated actual failures with accumulated predicted failure data. The formulas are given as follows: $RE=\left\|\frac{\hat{y}-y}{y}\right\|\times 100$ $AE=\frac{1}{n}\sum_{i=1}^{n}RE_i$ Where $\hat{y}$ is estimated accumulated failure and y is actual accumulated failure	SR2, SR26
NRMSE	It is the normalised root mean squared error and the formula is as under: $NRMSE=\sqrt{\frac{\sum_{i-1}^{n}(yi-yi\hat{})^2}{\sum_{i=1}^{n}(yi)^2}}$ Where n refers to the number of observations, $\hat{y}$ is predicted value and y is actual value	SR5, SR6, SR11, SR20
MSE	Mean Squared Error is a measure to show the mean of the square of the differences between actual and predicted values. The formula for calculating MSE is as follows: $MSE=\frac{1}{n}\sum_{i=1}^{n}(mi-m(ti))^2$ Where m is real failure and m(t) is estimated failure.	SR4, SR18, , SR19, SR24, SR28, SR31
VAF	Variance Accounted For is a measure used to show that how close is the estimated value to the actual value. The formula for calculating VAF is as follows: $VAF=[1-\frac{var(y-\hat{y})}{var\ (y)}]\times 100\%$ Where y is the actual and $\hat{y}$ is the estimated value.	SR12, SR31
MRE	Magnitude Relative error is a measure to show the relative differences between actual and estimated error values. $MRE=\frac{\sum_{i=1}^{n}\frac{Actual-estimated}{Actual}}{n}$	SR8, SR16,
MAE	Mean Absolute Error is a measure to show the mean value of absolute error values. $MAE=\sum_{i=1}^{n}\frac{(predicted-actual)}{n}$	SR10, SR14

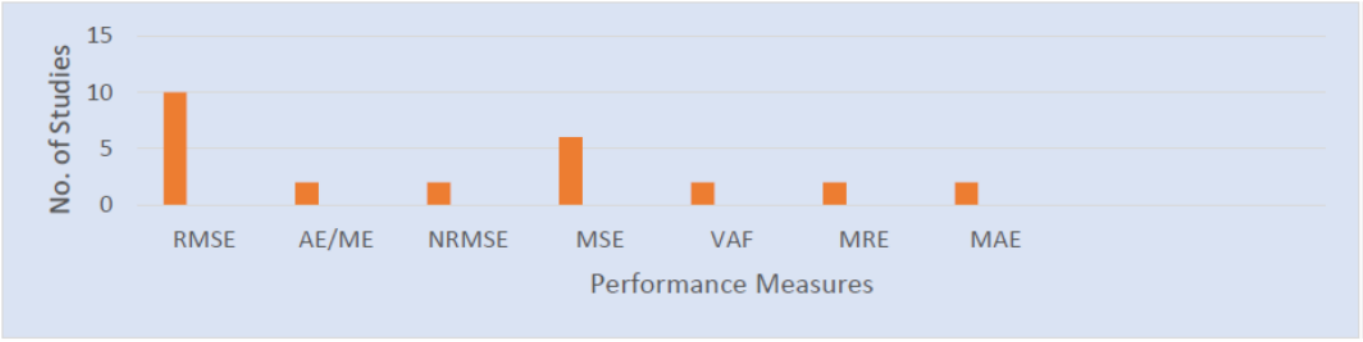

Figure 4.4: Distribution of performance measures used

4.5 RQ4: What is the predictive capabilities of mostly used SCT for software reliability prediction?

During the analysis of RQ4, it was found that the most commonly used performance measures are RMSE and MSE for software reliability prediction using SCT. We have shown the distribution of studies in which they are used while answering RQ4. Table 4.6 summarises the average values (Min, Max and correlation coefficient) of the RMSE performance measures along with the techniques. Table 4.7 shows the average values (Min, Max and correlation coefficient) of the MSE performance measure along with the techniques. The correlation coefficient is the measure to show degree of relationship between actual and predicted values. The range of correlation coefficient is from -1 to +1. The relationship becomes stronger as the value reaches near +1. The correlation coefficient is more than 0.8 for all the techniques, hence the usage of SCT are effective for software reliability prediction purposes. We have selected these tuples of the tables after analysing more than two studies, which have used same SCT, same predictive capability measure in order to avoid any biases in the observations.

Table 4.6: Predictive capability of SCT in terms of RMSE

SCT	RMSE		
	Min	Max	Correlation coefficient
NN	0.9234	4.7867	0.9901
SVM	0.8777	3.9734	0.9987
PSO	11.4541	15.5135	0.9781

Table 4.7: Predictive capability of SCT in terms of MSE

SCT	MSE		
	Min	Max	Correlation coefficient
GA	25.0881	32.3467	0.8453
SVM	0.2012	0.8456	0.9324

4.6 RQ5: What are the strengths and weaknesses of various SCT for developing software reliability prediction models?

Each soft computing technique has some advantages as well as disadvantages with respect to software reliability prediction capabilities. Table 4.8 shows the strengths and weaknesses of SCT for software reliability prediction along with the studies in which they are discussed. These strengths and weaknesses are the opinion of the different authors. The applicability of any SCT depends on the many factors like, type and characteristics of the problem. We have drawn the conclusion about strengths and weaknesses based on more than two studies. This is to avoid any biases in the conclusion.

Table 4.8: Strengths and Weaknesses of SCT

SCT	Strength	Weaknesses	Supporting Studies
NN	Better prediction performance when limited test data is available	Overfitting of the results with real life data set is the main drawback of NNs	SR2, SR5, SR13, SR14, SR23, SR26
FL	Efficient and decisive. The decisive capability is better than a conventional intelligent system.	Poor validation capability	SR10, SR12, SR21
GA	Performs better for complex search spaces	Large number of runs required	SR6, SR24, SR26
PSO	Faster convergence. Reduces time and space complexity	Suffers from local optimum problem	SR8, SR16, SR18
SVM	Better performance in handling high dimensionality data	Suitable for specific kind of applications	SR7, SR14

4.7 RQ6: Which validation methods are used for developing software prediction models using SCT?

Validation methods are employed for comparing and evaluating the performance of prediction models. Two methods for validation are identified as dataset and case study. The most commonly validation method used is dataset, as the dataset method provides the real environment data that is documented for reliability prediction purposes. Table 4.9 shows the validation methods used along with the studies that employed them. The table shows that majority of the studies have used datasets as their validation method. The dataset usually contains information like cumulative number of fault, time duration between successive failure, etc. The validation is done through comparing the estimated values with the actual values.

Some studies have used case study as their validation method (SR25, SR26). Case study method is employed to validate the model at early phases of software development life cycle. Case study method is used during design and implementation phases when the test data is not available. Various metrics (object

oriented metrics, traditional software metrics and process metrics) are used for case study method of validation.

Table 4.9: Validation Method Used

Validation Method	Studies
Dataset	SR1, SR2, SR3, SR4, SR5, SR6, SR7, SR8, SR9, SR10, SR11, SR12, SR13, SR14, SR15, SR16, SR18, SR19, SR20, SR21, SR22, SR23, SR24, SR25, SR26, SR28, SR29, SR30, SR31
Case Study	SR17, SR27

4.8 RQ7: Which statistical tests have been used for comparing the predictive abilities of software reliability prediction models using SCT?

Statistical tests are used to compare the results obtained using different techniques. They provide strong support to the results and conclusions made in the study. Only 10% of studies included statistical tests for validation of the results. Table 4.10 shows the different statistical tests used along with the unique ID of the studies.

Table 4.10: Statistical tests used

Statistical test	Studies
ANOVA	SR24
t-test	SR19
Wilcoxon signed rank test	SR1

4.9 RQ8: What are the possible source of threats to validity for developing software reliability prediction models using SCT?

Identification of possible sources of threats is important before setting up experiment. This enables readers to know the possible limitations of the studies. Cook and Campbell has given four categories of possible threats to a study namely: Conclusion validity threats, construct validity threats, internal and external validity threats [61]. We have identified the limitations reported in our primary studies and classified them according to cook and Campbell guidelines. Table 4.11 shows the threats identified their classification and supporting studies. The identified threats are categorized either internal or external validity threats. No threats are found which can be categorised under conclusion or construct validity category. About 10% of studies included threat to validity concern in their studies.

Table 4.11: Classification of threats

Threats category	Threats identified	Supporting Studies
Internal validity threat	The environment in which the software is operational is critical for the predictive performance of the models developed using SCT. So change in the environments is a possible source of threat.	SR14, SR25, SR19
External validity threat	The results obtained during the studies cannot be generalised due to various reasons such as the datasets used are limited, setting of parameters is stringent and study configurations are not flexible. So the threat of generalization is a most prominent threat identified.	SR14, SR25, SR19

CHAPTER 5

Threat to Validity

5.1 Threats to Validity

The objective of this review is to analyse the studies on reliability prediction of software based on techniques of soft computing. This review is not aware about the existence of any biases in the chosen studies. There may be more studies on the usage of SCT for software reliability prediction but not included in the review. This study does not include any unpublished results.

Selection of the primary studies based on quality assessment questions was done independently so that any biases are avoided. In order to answer various RQs, the data were extracted from several studies. The performance measures data were taken from multiple studies. However, the studies were performed under varied experimental set up (selection of optimal parameters, validation method used, performance measures used etc.). So this could be a threat to the study. The strengths and weaknesses presented are based on the opinion of the respective authors. We have only reported strengths and weaknesses of SCT which have been mentioned in more than two studies.

CHAPTER 6

Conclusion and Future Directions

6.1 Conclusion and Future Work

This section discusses the results obtained during the study and provides future guidelines to the researchers who intend to work in this domain.

The main objective of this thesis is to select and analyze the SCT, datasets trends, validation methods, strengths and weaknesses of the SCT used, performance measures, statistical tests and threats to validity for software reliability prediction studies. The review is performed based on the studies published between year 2005 and 2016. Depending on the designed quality assessment questions, finally 31 software reliability prediction studies were chosen and investigated.

6.1.1 Discussion of results

- (RQ1) The study revealed that fourteen SCT are used for software reliability prediction (NN, SVM, FL, GA, PSO, Hybrid, ACO, GP, DT, SVR, GWO, DE, SA, Bagging). Top five mostly used techniques found are: NN, FL, GA, PSO and SVM. The distribution of percentages are: NN (25%), FL(11%), SVM (11%), GA (8%) and PSO (8%). Only 10% of studies used hybrid approaches of SCT (ANN+PSO, AFNIS and GA-SA+SVR) for software reliability prediction. These studies showed improved performance over statistical techniques and other SCTs.
- (RQ2) The dataset analysis revealed that mostly four different datasets are used in the studies. The identified datasets are: DACS/Musa dataset (58.06% of studies), Private/Industrial dataset (32.25% of studies), Lyu dataset (6.45% of studies) and wood dataset (3.22% of studies).
- (RQ3) There are various performance measures used in the studies to assess the predictive capabilities of the models developed using SCT. The performance measures identified are: RMSE, AE/RE, VAF, MSE, MAE, MRE and NRMSE. Two most commonly used performance measures are identified as RMSE and MSE. The RMSE was used in 32% of the studies and MSE was used in 20% of the studies.
- (RQ4) Two mostly used performance measures were identified as RMSE and MSE. Both these measures provides the difference between actual and estimated values. Correlation coefficient is used to measure the degree of relationship between actual and predicted values and it was found to be more than 0.8 for mostly used SCT with RMSE as well as with MSE performance measures. Hence the predictive capability of SCTs were found quite satisfactorily.

- (RQ6) The strengths and weaknesses of each SCT are identified. NN provides better predictive performance when limited test data is available but they suffer from overfitting. FL is efficient and decisive but have poor validation capabilities. GA is effective technique when search space is complex but requires large number of runs. PSO provides faster convergence. PSO also reduces the time and space complexity but not suitable for small datasets.
- (RQ7) The validation techniques identified are either by dataset or by case study. Mostly dataset method is used. The dataset technique utilises the data obtained through real environment for predicting purposes. Subsequently, the validation is done by comparing the actual and predicted values. The case study method is used in early phases of development lifecycle (Design and implementation). The case study method is based on software metrics.
- (RQ7) The statistical tests used to validate the results were identified. Only 10% of the studies used statistical tests. The statistical tests found are: ANOVA, t-test and Wilcoxon signed rank test.
- (RQ8) The possible threats to validity are identified and found to be under the category of internal and external validity threats. No threats were found in the studies which can be classified under either conclusion or construct category.

6.1.2 Future Directions

The following are the guidelines for the researchers interested in conducting the research work using SCT for software reliability prediction

1. It was found that only 10% studies included the limitations or threats to validity concerns. A researcher should analyse the probable sources of threats during the design of experiment. This would help in effective implementation and accurate results of the experiment. Thus, in future the researchers must analyse the possible sources of threats to validity in the study and list them in their publications.
2. Only 4% studies used ensembles in their prediction processes and the results were found encouraging. Hence, more number of studies should be conducted using ensembles.
3. Most of the studies (about 90%) used public datasets. The studies which use public datasets can be easily replicated. Hence the researchers should use public dataset in their study
4. The strengths and weaknesses of the SCT should be considered before selection of any SCT for software reliability prediction.
5. Only three studies used a hybrid SCT (ANN+PSO, AFNIS and GA-SA+SVR) for software reliability prediction which showed enhanced performance. Hence, more number of studies should be conducted based on hybrid SCT.
6. The researchers should make datasets public so that more numbers of studies can be conducted using that dataset and further, the results can be generalized.

7. The researchers should explore and use industry datasets to achieve practical usefulness of SCT for software reliability prediction activity.
8. Very few studies (about 10%.) used statistical tests for the validation of the results obtained. The validation tests provide strong support to the results obtained. The researchers should conduct the statistical tests in their studies.
9. Out of 31 studies, 16 studies used evolutionary techniques. Out of 16 studies, only 5 studies included the execution of multiple runs of technique used for reporting the results. Since the SCT are stochastic in nature so multiple runs (minimum 10) are required to established the results effectively. The researcher should use multiple runs of the SCT used and the results should be reported accordingly.
10. Very few studies (25%) compared the performance of SCT used with either statistical techniques or other SCTs. Due to insufficient number of such studies, the performance analysis among various SCTs could not be established. The researchers should include comparative analysis of SCT used with either statistical techniques or other SCTs so that the performance of the SCT used can be analysed with respect to other SCTs.

Finally, the list of primary studies is given in chapter7 (appendix A.1). This list has of 5 attributes (year, primary studies, Journal/conference, datasets and methods used) and 31 primary studies (from year 2005 to 2016).

CHAPTER 7

Appendix A.1: List of primary Studies used in the review

S. No.	Author Name	Journal/Conference	YEAR	METHODS USED	DATASET
1	Wei-Chiang Hong and Ping-Feng Pai	The J of Systems and Software	2006	SVM and Simulated annealing (SA)	Telemetry systems AT&T Bell Laboratories and Musa 1979
2	Yu-Shen Su, Chin-Yu Huang	The J of Systems and Software	2006	Neural Network (NN)	Musa 1987 (PUBLIC)
3	ZHANG Yongqiang, CHEN Huashan	IEEE	2006	GP	Armoured Force Engineering test data (PRIVATE)
4	Wasif Afzal, Richard Torkar	IEEE	2008	GP	Industrial dataset (PRIVATE)
5	N. Raj Kiran, V. Ravi	The J of Systems and Software	2008	BPNN, TANN, PSN, MARS, GRNN, MLR	Musa 1979 (PUBLIC)
6	Mohammed E. El-Telbany and Sultan H. Aljahdali	ICGST-AIML Journal	2008	GA	John Musa (PUBLIC)
7	Emad A. and El-Sebakhy	Expert Systems with applications	2009	support vector machine (SVM)	AT & T Bell Telemetry system & Real time control system (PRIVATE)
8	Yuan Ju-mei	IEEE, International Conference	2009	PSO	Teaching data package, Univ. Denmark (PRIVATE)
9	Aurora Trinidad Ramirez Pozo, Eduardo Oliveira Costa, and Silvia Regina Vergilio	IEEE Transaction on Reliability	2010	GP	John Musa 1989 (PUBLIC)
10	Yogesh Singh, Pradeep Kumar	IEEE	2010	ANN	DACS, Misra & Ohba (PUBLIC)
11	C. Jin,	Journal of IET, software	2011	Hybrid Approach (GA-SA+SVR)	Failure data of turbochargers and Pai and Hong dataset
12	Sultan Aljahdali, Alaa F. Sheta	International Conference on Information Technology: New Generations	2011	Fuzzy Logic	John Musa (PUBLIC)
13	Neeraj Kumar Goyal and Manjubala Bisi	International Journal of Computer Applications	2012	Neural Network (NN) with Encoded Input	18 different datasets (PRIVATE)
14	Pradeep Kumar Yogesh Singh	Int J Syst Assur Eng Manag	2012	NN, SVM, Decision Tree & FIS	DACS (PUBLIC)
15	Lilly Florence and Latha Shanmugam	J of Comp Sc	2013	ACO	John Musa (PUBLIC)
16	Udayan Chanda, Md. Nasar and Prashant Johri	J of Industrial and Int Info	2013	Differential Evolution	Software Metrics
17	Ruchika Malhotra, Arun Negi	Int J Syst Assur Eng Manag	2013	PSO	John Musa 1989 (PUBLIC)
18	Nada N. Saleem and Rita G. Al gargoor	Int J of Comp Sc Issues	2013	ANN, PSO	DS1, DS2 & DS3 (PRIVTE) - Command & Control app - Operating system

S. No.	Author Name	Journal/Conference	YEAR	METHODS USED	DATASET
19	Cong Jin, Shu-Wei Jin	Applied Soft Computing	2014	Support Vector Machine (SVM)	Telemetry systems AT&T Bell Laboratories and Musa 1979
20	Azmath Mubeen, Ramakanta Mohanthy and Venkatshwarlu Naik,	Fourth Int Conf on Comm Systems and Network Tech	2014	ACO	Literature dataset
21	Kirti Tyagi and Arun Sharma	Applied computing and informatics	2014	ANFIS and FIS	Classroom based projects
22	Neeraj Kumar Goyal and Manjubala Bisi	Int conf on Computational Intelligence and NW	2015	Hybrid (ANN-PSO)	DS1, DS2 & DS3 (PRIVATE) - Electronic switching system - Wireless Network product - Bug tracking system
23	Indhurani Lakshmanana, Subburaj Ramasamy	Int Conf on Recent Trends in Computing	2015	ANN	John Musa (PUBLIC)
24	TaehyounKim, KwangkyuLee, JongmoonBaik,	The J of Systems and Software	2015	GA	DACS dataset (Musa, 1980) Literature dataset (PUBLIC)
25	JinheePark, JongmoonBaik	The J of Systems and Software	2015	Decision TREE	Musa 1987, Lyu 1996, Wood 1996 & Tohma Jacoboy 1989 (PUBLIC)
26	Pratik Roy, G.S. Mahapatra , K.N. Dey	Expert Systems with applications	2015	ANN, GA	Lyu 1996 (PUBLIC)
27	Harikesh Bahadur Yadav, Dilip Kumar Yadav	Information and Software Technology	2015	Fuzzy Logic	Software Metrics
28	Arunima Jaiswal, Ruchika Malhotra	Int J Syst Assur Eng Manag	2016	GRNN, BPNN, SVM, Bagging	Industrial dataset
29	Durga Prasad MOHAPATRA, Manmath Kumar BHUYAN, Srinivas SETHI	Baltic J. Modern Computing	2016	Neural Network (NN)	John Musa 1980 & Iyer 1996 (PUBLIC)
30	Jungang Lou, Yunliang Jiang	Journal of Neurocomputing	2016	Support vector Machine (SVM)	Real time control application and flight dynamic applications
31	Alaa F. Sheta and Amal Abdel-Raouf	Int J of Advanced Comp Sc and Apps	2016	Grey Wolf Optimization	Real time control application (PRIVATE)

CHAPTER 8

References

[1] Yadav, A., & Khan, R. A. (2012). Development of encapsulated class complexity metric. *Procedia Technology*, *4*, 754-760.

[2] Kumar, P., & Singh, Y. (2012). An empirical study of software reliability prediction using machine learning techniques. International Journal of Systems Assurance Engineering and Management, 1-15.

[3] Kiran, N. R., & Ravi, V. (2008). Software reliability prediction by soft computing techniques. *Journal of Systems and Software*, *81*(4), 576-583.

[4] Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, *27*, 504-518.

[5] Malhotra, R., & Negi, A. (2013). Reliability modeling using particle swarm optimization. International Journal of System Assurance Engineering and Management, 3(4), 275-283.

[6] Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. sn.

[7] Singhal A., Singhal A. (2011). A systematic review of software reliability studies. *Software Engineering : An International Journal (SEIJ)*, Vol. 1, No. 1.

[8] Afzal W., Torkar R. (2011). On the application of genetic programming for software engineering predictive modeling: A systematic review, *Expert Systems with Applications* 38 (2011) 11984–11997.

[9] Jatain, A., & Mehta, Y. (2014, February). Metrics and models for software reliability: A systematic review. In *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on* (pp. 210-214). IEEE.

[10] Sharma, L. K., Saket, R. K., & Sagar, B. B. (2015, March). Software Reliability Growth Models and tools-a review. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on* (pp. 2057-2061). IEEE.

[11] Ong L., Isa M., Razak A (2016). Meta-Heuristics Methods for Software Reliability Prediction: A Systematic Literature Review. *Journal of Emerging Trends in Computing and Information Sciences*, Vol. 7, No. 3.

[12] Malhotra, R., Khanna, M., & Raje, R. R. (2017). On the application of search-based techniques for software engineering predictive modeling: A systematic review and future directions. *Swarm and Evolutionary Computation*, *32*, 85-109.

[13] Yadav N., Yadav V., Verma P. (2016). Role of Evolutionary Algorithms in Software Reliability Optimization, *5th International Conference on System Modeling & Advancement in Research Trends*. ISBN: 978-1-5090-3543-4.

[14] T., Sangwan O., (2017). Computational Intelligence based Approaches to software reliability, *7th International Conference on Cloud Computing, Data Science & Engineering – Confluence*, 978-1-5090-3519-9/17.

[15] Pai, P. F., & Hong, W. C. (2006). Software reliability forecasting by support vector machines with simulated annealing algorithms. *Journal of Systems and Software*, *79*(6), 747-755.

[16] Su, Y. S., & Huang, C. Y. (2006). Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. *Journal of Systems and Software*, *80*(4), 606-615.

[17] Zhang, Y., & Chen, H. (2006, October). Predicting for MTBF failure data series of software reliability by genetic programming algorithm. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on* (Vol. 1, pp. 666-670). IEEE.

[18] Afzal, W., & Torkar, R. (2008, October). Suitability of genetic programming for software reliability growth modeling. In *Computer Science and its Applications, 2008. CSA'08. International Symposium on* (pp. 114-117). IEEE.

[19] Aljahdali, S. H., & El-Telbany, M. E. (2008). Genetic algorithms for optimizing ensemble of models in software reliability prediction. *ICGST-AIML J*, *8*(1), 5-13.

[20] El-Sebakhy, E. A. (2009). Software reliability identification using functional networks: A comparative study. Expert systems with applications, 36(2), 4013-4020.

[21] Yuan, J. M. (2009, August). Adaptive multi-model synthesis dynamic prediction of software reliability based on particle swarm optimization. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on* (pp. 2357-2362). IEEE.

[22] Costa, E. O., Pozo, A. T. R., & Vergilio, S. R. (2010). A genetic programming approach for software reliability modeling. *IEEE Transactions on Reliability*, *59*(1), 222-230.

[23] Singh, Y., & Kumar, P. (2010, December). Prediction of software reliability using feed forward neural networks. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on* (pp. 1-5). IEEE.

[24] Jin, C. (2011). Software reliability prediction based on support vector regression using a hybrid genetic algorithm and simulated annealing algorithm. *IET software*, *5*(4), 398-405.

[25] Aljahdali, S., & Sheta, A. F. (2011, April). Predicting the reliability of sof*tware systems using fuzzy logic. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on* (pp. 36-40). IEEE

[26] Bisi, M., & Goyal, N. K. (2012). Software reliability prediction using neural network with encoded input. *International Journal of Computer Applications*, *47*(22), 46-52.

[27] Shanmugam, L., & Florence, L. (2013). Enhancement and comparison of ant colony optimization for software reliability models.

[28] Nasar, M., Johri, P., & Chanda, U. (2013). A Differential Evolution Approach for Software Testing Effort Allocation. *Journal of Industrial and Intelligent Information Vol*, *1*(2).

[29] Gargoor R., Saleem N. (2013). Software Reliability Prediction Using Artificial Techniques, *International Journal of Computer Science Issues*, Vol. 10, Issue 4, No 2.

[30] Jin, C., & Jin, S. W. (2014). Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms. Applied Soft Computing, 15, 113-120.

[31] Mohanthy R., Naik V., Mubeen A. (2014). Predicting Software Reliability Using Ant Colony Optimization Technique, *Fourth International Conference on Communication Systems and Network Technologies.*

[32] Tyagi, K., & Sharma, A. (2014). An adaptive neuro fuzzy model for estimating the reliability of component-based software systems. *applied Computing and informatics*, *10*(1), 38-51.

[33] Bisi, M., & Goyal, N. K. (2015, January). Predicting cumulative number of failures in software using an ANN-PSO based approach. In Computational Intelligence and Networks (CINE), 2015 International Conference on (pp. 9-14). IEEE

[34] Lakshmanana I., Ramasamy S. (2015). An artificial neural-network approach to software reliability growth Modelling, *International Conference on Recent Trends in Computing*, 2015.

[35] Kim, T., Lee, K., & Baik, J. (2015). An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm. *Journal of Systems and Software*, *102*, 134-144.

[36] Park, J., & Baik, J. (2015). Improving software reliability prediction through multi-criteria based dynamic model selection and combination. *Journal of Systems and Software*, *101*, 236-244.

[37] Roy, P., Mahapatra, G. S., & Dey, K. N. (2015). Neuro-genetic approach on logistic model based software reliability prediction. Expert systems with Applications, 42(10), 4709-4718.

[38] Yadav, H. B., & Yadav, D. K. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. *Information and Software Technology*, *63*, 44-57.

[39] Jaiswal, A., & Malhotra, R. (2016). Software reliability prediction using machine learning techniques. In *Proceedings of Fifth International Conference on Soft Computing for Problem Solving* (pp. 141-163). Springer, Singapore.

[40] Bhuyan M., Mohapatra D., Sethi S. (2016). Software Reliability Assessment using Neural Networks of Computational Intelligence Based on Software Failure Data. *Baltic J. Modern Computing*, Vol. 4 (2016), No. 4, pp. 1016–1037.

[41] Lou, J., Jiang, Y., Shen, Q., Shen, Z., Wang, Z., & Wang, R. (2016). Software reliability prediction via relevance vector regression. *Neurocomputing*, *186*, 66-73.

[42] Sheta, A. F., & Abdel-Raouf, A. (2016). Estimating the Parameters of Software Reliability Growth Models Using the Grey Wolf Optimization Algorithm. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, *7*(4), 499-505.

[43] Kaur, A. (2015). Comparative Analysis of Reliability Models–Based on Uncertainty Factors. *IJ of Advanced Research in Computer Science and Software Engineering*, *5*(2).

[44] Gueorguiev, S., Harman, M., & Antoniol, G. (2009, July). Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (pp. 1673-1680). ACM.

[45] Costa, E. O., & Pozo, A. (2006, November). A (\mu+\lambda)-GP Algorithm and its use for Regression Problems. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on* (pp. 10-17). IEEE.

[46] Zhang, Y., & Cheng, H. (2009). Improved genetic programming algorithm applied to symbolic regression and software reliability modeling. *Journal of Software Engineering and Applications*, *2*(05), 354.

[47] Azar, D., Harmanani, H., & Korkmaz, R. (2009). A hybrid heuristic approach to optimize rule-based software quality estimation models. *Information and Software Technology*, *51*(9), 1365-1376.

[48] Vivanco, R., & Jin, D. (2008, October). Enhancing predictive models using principal component analysis and search based metric selection: a comparative study. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (pp. 273-275). ACM.

[49] Wu, B. H. (2011, October). An evolutionary approach to evaluate the quality of software systems. In *Advanced Computational Intelligence (IWACI), 2011 Fourth International Workshop on* (pp. 381-386). IEEE.

[50] Tang, J. L., Cai, Q. R., & Liu, Y. J. (2010, April). Gear fault diagnosis with neural network based on niche genetic algorithm. In *Machine Vision and Human-Machine Interface (MVHI), 2010 International Conference on* (pp. 596-599). IEEE.

[51] Khoshgoftaar, T. M., & Liu, Y. (2007). A multi-objective software quality classification model using genetic programming. *IEEE Transactions on Reliability*, *56*(2), 237-245.

[52] XIiong, B., Zhang, L., Yang, N., & Li, J. (2009, March). A Genetic Algorithm Based Method of Fault Maintenance in Software-Intensive System. In *Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on* (Vol. 3, pp. 1056-1059). IEEE.

[53] Bouktif, S., Sahraoui, H., & Antoniol, G. (2006, July). Simulated annealing for improving software quality prediction. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 1893-1900). ACM.

[54] Rao, K. M., & Anuradha, K. (2016, May). A hybrid method for parameter estimation of software reliability growth model using Modified Genetic Swarm Optimization with the aid of logistic exponential testing effort function. In *Research Advances in Integrated Navigation Systems (RAINS), International Conference on* (pp. 1-8). IEEE.

[55] Yang, S., Lu, M., & Ge, L. (2013, June). Bayesian Network Based Software Reliability Prediction by Dynamic Simulation. In *Software Security and Reliability (SERE), 2013 IEEE 7th International Conference on* (pp. 13-20). IEEE.

[56] Altaf, I., Majeed, I., & Iqbal, K. A. (2016, November). Effective and optimized software reliability prediction using Harmony search algorithm. In *Green Engineering and Technologies (IC-GET), 2016 Online International Conference on* (pp. 1-6). IEEE.

[57] Yadav, N., Yadav, V., & Verma, P. (2016, November). Role of Evolutionary algorithms in Software Reliability Optimization. In *System Modeling & Advancement in Research Trends (SMART), International Conference* (pp. 45-48). IEEE.

[58] Sheptunov, S. A., Larionov, M. V., Suhanova, N. V., Salakhov, M. R., & Solomentsev, Y. M. (2016, October). Simulating reliability of the robotic system software on the basis of artificial intelligence. In *Quality Management, Transport and Information Security, Information Technologies (IT&MQ&IS), IEEE Conference on* (pp. 193-197). IEEE.

[59] Aljahdali, S. H., & El-Telbany, M. E. (2009, May). Software reliability prediction using multi-objective genetic algorithm. In *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*(pp. 293-300). IEEE.

[60] Oliveira, E., Pozo, A., & Vergilio, S. R. (2006, November). Using boosting techniques to improve software reliability models based on genetic programming. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on* (pp. 643-650). IEEE.

[61] Cook T. and Campbell D (1979), Quasi-experimentation – design and analysis issues for field settings, *Houghton Miffin Company*, Boston, 1979.