ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Fuzzy Analysis for the Cost Effective Software Evolution Exertion Appreciation

Mr. Lakshmana Rao Padala
Research Scholar JJTU,
Assistant.Manager CSI-ED,
Chennai, Tamil Nadu.

Dr. E.Mohan
Principal P.T.Lee.
Chengalvaraya Naicker College of Engineering and Technology, Kanchipuram, Tamil Nadu, India.

Abstract :
Software Cost Assessment is the most significant and demanding activity in Software evolution effort indication. Software effort assessment is unpredictable in nature as it hugely depends upon some variables that are not known at the initial state of evolution. Fuzzy analysis plays a main role to analyse and predict the Software cost assessment. Fuzzification is the one of the key attribute which includes the size of the project, incorporates expert's knowledge in a well-defined manner, allows total transparency in the indication system by indication of results through rules or other means, adaptability towards continually changing evolution technologies and environments. Properly addressing all these issues would position fuzzy logic computing based indication techniques as models of choice for effort indication, considering the promising features already present in them are made. Cost Effective Software Evolution Exertion Appreciation is one of the most challenging tasks in software sector. Our most intension in this paper is to present the Fuzzy Analysis for the Cost Effective Software Evolution Exertion Appreciation.

Key words:
Fuzzy Analysis, Fuzzification, Cost Effective, Software Evolution, Exertion.

A.INTRODUCTION:
Software reliability is a key part in software quality along with functionality, usability, performance, serviceability, maintainability and documentation. The standard definition of reliability for software is the probability that a system will continue to function without failure for a specified period in a specified environment. The study of software reliability can be categorized into three parts: modelling, measurement and improvement. Software reliability modelling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, but no single model can capture a necessary amount of the software characteristics.

Assumptions and abstractions must be made to simplify the problem. There is no single model that is universal to all the situations. Software reliability measurement is fresh [1]. Measurement is far from commonplace in software, as in other engineering field. "How good is the software, quantitatively?" As simple as the question is, there is still no good answer. Software reliability cannot be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Evolution process, faults and failures found are all factors related to software reliability. Software reliability improvement is hard. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software [2]. Until now there is no good way to conquer the complexity problem of software. Complete testing of a moderately complex software module is infeasible. Defect-free software product cannot be assured.

Realistic constraints of time and budget severely limit the effort put into software reliability improvement. If the time and budget is not considered carefully, software reliability can be the reliability bottleneck of the whole system. Securing software reliability is no easy task. As hard as the problem is, promising progresses are still being made toward more reliable software. More standard components and better process are introduced in software engineering field. In the last few years many research studies has been carried out in this area of software reliability modeling and forecasting. They included the application of fuzzy logic models neural networks, Fuzzy logics (FL) based neural networks, Recurrent neural networks, Bayesian neural networks, and support vector machine (SVM) based techniques, to name a few. Software Reliability issues are focused on Fuzzy logic computing techniques for developing and maintaining software systems whose reliability can be quantitatively evaluated. In order to estimate as well as to predict the reliability of software systems, failure data need to be properly measured by various means during software evolution and operational states.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Although software reliability has remained an active research subject over the past forty years, challenges and open questions still exist. The various modeling technique for software reliability is reaching its prosperity, but before using these techniques, we must carefully select the appropriate fuzzy logic computing tool that can best suit the case in question. Measurement in software is still in its infancy. No good quantitative methods seem to have been developed to represent software reliability without excessive limitations.

B.Software Reliability Indication:

Using Fuzzy Logic Fuzzy logic is derived from fuzzy set theory which deals with reasoning which is approximate rather than precisely deduced from classical predicate logic. It can be thought of as the application side of fuzzy set theory dealing with well thought out real world expert values for a complex problem.[3] Fuzzy logic was initiated in 1965 [4][5] by Lotfi A.Zadeh, professor of computer science. Fuzzy logic is proven to be capable of modeling highly nonlinear and multidimensional models. Fuzziness refers to non statistical imprecision and vagueness in information and data. The difference between fuzzy logic and probabilistic logic consists in the fact that the fuzzy logic uses truth degrees as a mathematical model for vague facts while the probabilistic one is a mathematical model for random facts [6]. The linguistic values are used for writing the If – Then rules. Researchers in this area have felt that fuzzy logic is vital for Software reliability indication. Yuan et. Al in [7] used fuzzy subtractive clustering integrated with module order modeling for software quality indication.

First Fuzzy Subtractive clustering is used to predict the number of faults then module order modeling is used to predict whether modules are fault prone or not. Xu et al [8] introduced the fuzzy nonlinear regression (FNR) modeling technique as a method for predicting fault ranges in software modules. A case study of full scale industrial software system was used to illustrate the usefulness of FNR Modeling. Jeff Tian in [9] assessed software reliability by grouping data into clusters. The series of data clusters associated with different time segments are used directly as a piecewise linear model for reliability assessment and problem identification. The model is evaluated in the testing of two large software systems from IBM. Adnan et al in [9] explored the potential of indication techniques which have been used for assessing software reliability.

The DACS Services at the Department of Defense (DoD) Software Information clearinghouse provides an authoritative source for the state of the art software information, supplying technical support for the software community. John Musa of Bell Telephone Laboratories compiled a software reliability database.. The dataset consists of software failure data on 16 projects. Careful controls were employed during data collection to ensure that the data would be of high quality. The data was collected throughout the mid 1970s. It represents projects from a variety of applications including real time command and control, word processing, commercial, and military applications. Cai et al. [9] advocated the evolution of fuzzy software reliability models in place of probabilistic software reliability models (PSRMs). Their argument was based on the proof that software reliability is fuzzy in nature. A demonstration of how to develop a fuzzy model to characterize software reliability was also presented. Karunanithi et al. [9] carried out a detailed study to explain the use of connectionist models in software reliability growth indication. It was shown through empirical results that the connectionist models adapt well across different datasets and exhibit better predictive accuracy than the well-known analytical software reliability growth models.

C.Software Reliability Indication Using Fuzzy logic:

In recent years, many papers have presented in various models for software reliability indication. In this section, some works related to Artificial Fuzzy logic modeling for software reliability modeling and indication is presented. Many factors like software evolution process, and software test or use characteristics, software complexity, and nature of software faults and the possibility of occurrence of failure affect the software reliability behavior. Fuzzy logic methods normally approximate any non linear continuous function. So more attention is given to Fuzzy logic based methods now-a-days. Fuzzy logicbased software reliability model was first presented by Karunanithi et al. [9][8] to predict cumulative number of failures. They consider execution time as the input of the neural network. In their approach they used different networks like Feed Forward neural networks, Recurrent neural networks like Jordan Fuzzy logicand Elman neural network. Two different training regimes like Indication and Generalization are also used in their study. They compared their results with some statistical models and found better indication than those models. Karunanithi et al. [8] also used connectionist models for software reliability indication.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

They applied the Falman's cascade Correlation algorithm to find out the architecture of the neural network. They considered the minimum number of training points as three and calculated the average error (AE) for both end point and next-step indication. Their results concluded that the connectionist approach is better for end point indication. Cai et al. [7] Proposed a Fuzzy logicbased method for software reliability indication. They used back PROPAFLTION algorithm for training. They evaluated the performance of the approach by varying the number of input nodes and number of hidden nodes. They concluded that the effectiveness of the approach generally depends upon the nature of the handled data sets. Tian and Noore [6] proposed an on-line adaptive software reliability indication model using evolutionary connectionist approach based on multiple-delayed-input single-output architecture. The proposed approach, as shown by their results, had a better performance with respect to next-step predictability compared to existing Fuzzy logic model for failure time indication. Tian and Noore [5] proposed an evolutionary Fuzzy logicmodeling approach for software cumulative failure time indication.

Their results were found to be better than the existing Fuzzy logic models. It was also shown that the Fuzzy logic architecture has a great impact on the performance of the network. Viswanath [5] proposed two models such as Fuzzy logic based exponential encoding and Fuzzy logic based Loflrithmic encoding for indication of cumulative number of failures in software. He encoded the input i.e. the execution time using the above two encoding scheme. He applied the approach on four datasets and compared the result of the approach with some statistical models and found better result than those models. Ho et al. [5] performed a comprehensive study of connectionist models and their applicability to software reliability indication and found them to be better and more flexible than the traditional models. A comparative study was performed between their proposed modified Elman recurrent neural network, with the more popular feed forward neural network, the Jordan recurrent model, and some traditional software reliability growth models. Numerical results show that the proposed network architecture performed better than the other models in terms of indications. Despite of the recent advancements in the software reliability growth models, it was observed that different models have different predictive capabilities and also no single model is suitable under all circumstances.

Software Reliability Indication Using Fuzzy logic Fuzzy logics (FLs) were developed by Prof. John Holland and his students at the University of MichiFLn during the 1960s and 1970s. Fuzzy logic can be used by represent a solution to your problem as a genome (or chromosome) [5]. The fuzzy logic then creates a population of solutions and applies fuzzy operators such as mutation and cross-over to evolve the solutions in order to find the best one. The three most important aspects of using fuzzy logics are: (1) definition of the objective function, (2) definition and implementation of the fuzzy representation, and (3) definition and implementation of the fuzzy operators. Once these three have been defined, the generic fuzzy logic should work fairly well. Beyond that you can try many different variations to improve performance, find multiple optima (species - if they exist), or parallelize the algorithms [9].Fuzzy logics are machine learning and optimization schemes, much like neural networks. However, fuzzy logics do not appear to suffer from local minima as badly as neural networks do. Fuzzy logics are based on the model of evolution, in which a population evolves towards overall fitness, even though individuals perish. Evolution dictates that superior individuals have a better chance of reproducing than inferior individuals, and thus are more likely to pass their superior traits on to the next fuzzification.

This "survival of the fittest" criterion was first converted to an optimization algorithm by Holland in 1975, and is today a major optimization technique for complex, non-linear problems .Oliveira et al. [18, 19] proposed the using of fuzzy programming (FP) to obtain software reliability model for forecasting the reliability and extended this work by boosting the GP algorithm using re-weighting. The re-weighting algorithm calls many times the learning algorithm with assigned weights to each example. Each time, the weights are computed according to the error (or loss) on each example in the learning algorithm. In this way, the learning algorithm is manipulated to look closer at examples with bad indication functions. Sheta [7] uses fuzzy logics to estimate the COCOMO model parameters for NASA Software Projects. The same idea is implemented for estimating the parameters of different SRGM models using PSO [6].

D. Issues and challenges :

In order to estimate as well as to predict the reliability of software systems, failure data need to be properly measured by various means during software evolution and operational states.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Although software reliability has remained an active research subject over the past 35 years, challenges and open questions still exist. The various modeling technique for Software Reliability is reaching its prosperity, but before using these technique, we must carefully select the appropriate model that can best suit our case. Measurement in software is still in its infancy. No good quantitative methods have been developed to represent Software Reliability without excessive limitations.

E.Future Directions:

Software Reliability Engineering relates to whole software life cycle. We discuss possible future directions with respect to four areas: software architecture, testing and metrics [1].

A.Reliability for software architectures:

Due to the ever-increasing complexity of software systems, modern software is seldom built from scratch. Revolutionary and evolutionary object-oriented design and programming paradigms have vigorously pushed software reuse. In the light of this shift, reliability engineering for software evolution is focusing on two major aspects: software architecture, and component-based software engineering. The software architecture of a system consists of software components, their external properties, and their relationships with one another. As software architecture is the foundation of the final software product, the design and management of software architecture is becoming the dominant factor in software reliability engineering research. In this popular software evolution technique, many research issues are identified, such as reliability, software reusability, clean interface design, fault tolerance etc.

B.Testing for reliability assessment:

Software testing and software reliability have traditionally belonged to two separate communities. Software testers test software without referring to how software will operate in the field, as often the environment cannot be fully represented in the laboratory. Software reliability measurers insist that software should be tested according to its operational profile in order to allow accurate reliability assessment and indication. One approach is to measure the test compression factor, which is defined as the ratio between the mean time between failures during operation and during testing. Another approach is to ascertain how other testing related factors can be incorporated into software reliability modeling, so that accurate measures can be must collect software metrics as an indication of a maturing software evolution process. Industrial software engineering data, particularly those related to system failures, are historically hard to obtain across a range of Fuzzification. Novel methods are used to improve reliability indication are actively being researched. For example, by extracting rich information from metrics data using a sound statistical and probability foundation, Moreover, traditional reliability models can be enhanced to incorporate some testing completeness or effectiveness metrics, such as code coverage, as well as their traditional testing-time based metrics. The key idea is that failure detection is not only related to the time that the software is under testing, but also what fraction of the code has been executed by the testing.

Conclusions:

This paper presented the usages of the fuzzy logic computing techniques for software reliability. Employing effective software reliability engineering techniques to improve product and process reliability would be the industry's best interests as well as major challenges. As the majority of faults are found in a few of its modules so there is a need of fuzzy logic to module the affected severely as compared to other modules and proper maintenance need to be done in time especially for the critical applications. Preliminary results are quite interesting and more insights will provide a special fuzzy logic computing assisted architecture for enabling the specialist in software reliability engineering.

References:

[1] G. Pour, "Component-Based Software Development Approach: New Opportunities and Challenges," Proceedings Technology of Object-Oriented Languages, 1998. TOOLS 26, pp. 375-383.
[2] Tyagi, K., Sharma, A., 2014, "An adaptive neuro fuzzy model for estimating the reliability of component-based software system", applied Computing and informatics 10, 38–51.
[3] Cheung, R.C., 1980. A user oriented software reliability model. IEEE Trans. Softw. Eng. 6 (2), 118– 125.
[4] J.D Musa (1987), Software Reliability measurement, prediction, application McGRAW-HILL International Edition. ISBN 0-07-100208- 1
[5] Michael R. Lyu (May 2005) Handbook of Software Reliability Engineering: Introduction. IEEE Computer Society Press and McGraw-Hill Book Company

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[6] Swapna S. Gokhale. Kishor S. Trivedi (2002), Reliability Prediction and Sensitivity Analysis Based on Software Architecture, IEEE, Software reliability, pp 64-75, ISBN: 0-7695-1763-3.
[7] Musa, John. Software Reliability Engineering, New York, NY, McGraw-Hill, 1998.
[8] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 1-13, 1975.
[9] 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.