AMSE JOURNALS-AMSE IIETA publication-2017-Series: Advances B; Vol. 60; N°3; pp 539-550
Submitted Sept. 26, 2017; Revised Dec. 26, 2017; Accepted Jan. 03, 2018

A Fuzzy Neural Network Approach for Assessment and Enhancing Software Reliability

*Theyyagura M.K. Reddy, **Madhwaraj Kango Gopal

*Research Scholar, Department of IT, VFSTR Deemed to be University, Vadlamudi, Guntur, India (manikantareddy.t@gmail.com)
**Professor, Department of IT, VFSTR Deemed to be University, Vadlamudi, Guntur, India (drkgm_it@vignanuniversity.org)

Abstract

Software Reliability is the probability of non-failure software procedure for a predefined duration in a predetermined domain. Software Reliability is similarly an imperative factor manipulating structure with reliability [2]. It contrasts from hardware reliability in the way that it reflects the outline faultlessness, and to provide reliable software. The elevated intricacy of software is the major contributing element of Software Reliability issues. Software reliability engineering (SRE) surveys how well software based items and administrations meet client's operational needs. SRE utilizes quantitative techniques in view of reliability measures to do this evaluation. The essential objective of SRE is to boost consumer loyalty. SRE uses such quantitative strategies as factual estimation and expectation, estimation, and displaying. As the reliability of an item or administration is profoundly subject to working conditions and the reliability of software is identified with how the product is utilized, the quantitative portrayal of the utilization of software is an indispensable part in SRE. Software Cost Estimation with resonating unwavering quality, profitability and improvement exertion is a testing and burdensome undertaking. This has prompted the product group to give much required push and dig into broad research in Software exertion estimation for developing refined strategies. Estimation by similarity is one of the practical strategies in Software exertion estimation field. Be that as it may, the technique used for the estimation of Software exertion by similarity can't deal with the all-out information in an express and exact way. Another approach has been created in this paper to assess Software exertion for ventures spoke to by all out or numerical information utilizing thinking by similarity and fluffy approach. The current chronicled datasets, investigated with fluffy rationale, deliver precise brings about correlation with the dataset examined with the before approaches. Software designing is a more extensive training of which SRE is a sub train. Software building is worried about all parts of outlining, executing, and dealing with the advancement of software [8]. Different parts of software building incorporate the financial aspects of creating software and the interfaces between software, frameworks, and people and with the practices and procedures for guaranteeing the nature of conveyed software. In this paper we ponder the product reliability of frameworks with the assistance of past failure related informational collections by utilizing Fuzzy Neural Networks (FNN) methods, Numerical cases are appeared with both real and mimicked datasets. Better execution of software reliability evaluation is watched, contrasted and unique FNN demonstrate with no such verifiable failure related information joined.

Keywords: Software reliability, Quality evaluation, Software failure, Fuzzy-neural approach, Software reliability engineering.

1. INTRODUCTION

Numerous product reliability methods have been created from recent decades. They are produced through either a scientific or information driven approach. Expository software reliability development methods spoke to by Non-Homogeneous Poisson Process (NHPP), are stochastic methods concentrating on software failure process. Information driven methods are produced from verifiable software issue related information, following the approach of relapse or time arrangement examination.

FNN (Fuzzy Neural Network) software reliability methods have as of late excited more research intrigue [4-10]. Customarily, the two sorts of methods just consider single failure recognition process (FRP) and information for investigation are just from FRP. In any case, while information from both FRP&FCP (failure correction process) are accessible, NHPP and FNN methods can be stretched out into matched NHPP methods and consolidated FNN methods, giving more precise forecasts [10, 11]. As a rule, information driven approach is considerably less prohibitive in presumptions contrasted with logical approach.

By and large, exact forecasts cannot be gotten in the early period of testing through both methodologies, as there is insufficient information for parameter estimation or learning at this stage. Be that as it may, early software reliability expectation is valuable for auspicious software improvement process control. S.Gokhale et al. [13] attempted to grow early expectation with Bayes system with subjective or/and target information from more seasoned undertakings. S. Krishnamurthy et al. [14] proposed a more useful way to deal with grow early reliability forecast in view of NHPP methods. In that paper, NHPP methods were acclimated to fuse failure history data from a comparative task by accepting a similar failure rate. This approach has likewise been reached out to combined NHPP methods, taking both the testing and investigating situations as the same [15]. What's more, verifiable failure related information reuse is a pragmatic approach for current develops software producers, as they have a lot of reusable data from past discharges or comparative undertakings put away in their database.

Software Reliability Includes the Following Parameters

Software Measurement

Software estimation is fundamental for accomplishing the essential administration goals of expectation, advance, and process change.

Software Metrics

Measurements which are utilized to assess the product procedures, items and administrations is alluded to as the product measurements.

Software Defect

Deformity (or Fault or bug) is an aftereffect of a passage of incorrect data into Software. "Imperfections are raised when expected and genuine test outcomes vary".

In this paper, we propose to consolidate authentic failure related information with FNN methods (both conventional and expanded ones) to enhance early reliability expectation for current software testing venture. To represent this approach, two reasonable datasets from FRP are connected with conventional FNN display. We have taken reenacted dataset one from each: FRP and FCP and has connected it with expanded FNN demonstrate.

In the early time of the item life cycle, an insightful method is required in light of the way that no breakdown data are available. This kind of strategy figures the amount of slip-ups in the program sooner than testing. In testing stage, the reliability of the item upgrades through investigating. A reliability improvement exhibit is relied upon to assess the present steadfastness level and the time and resources required to fulfill the objective trustworthiness level. In the midst of this stage, reliability estimation relies upon the examination of disappointment data. After the entry of an item program, the extension of new modules, ejection of old modules, clearing of distinguished goofs, mix of new code with officially created code, change of customer condition, change of equipment, and organization commitment must be considered in the appraisal of Software reliability. The proposed work uses a fuzzy approach for enhancing the software reliability and the results also show that the proposed method enhances the reliability level of the software.

2. RELATED DATA

C. T. Lin consolidated an item reliability database [1]. His objective was to gather failure break data to help Software bosses in watching test status, envisioning designs and to help Software examiners in supporting Software trustworthiness methods. These methods are associated in the preparation of Software reliability building. The dataset involves Software deficiency data on 16 wanders. Careful controls were used in the midst of data collection to ensure that the data would be of high gauge. The data was assembled all through the mid 1970s. It addresses wanders from an arrangement of uses including consistent request and control, word taking care of, business, and military applications.

For our circumstance, we used data from three exceptional undertakings. They are Real Time Control, Military and Operating System.

A MATLAB device stash for showing of fluffy structures [6] was used to realize the going with results. The timetables of the instrument compartment contain the Gustanfson-Kessel (GK) gathering estimation, whose utilization is given in [3].

3. SOFTWARE FAILURE MECHANISMS

Software failures might be because of mistakes, ambiguities, oversights or distortion of the determination that the product should fulfill, recklessness or ineptitude in composing code, lacking testing, off base or sudden use of the product or other unanticipated issues. Hardware shortcomings are for the most part physical deficiencies, while software flaws are configuration issues, which are harder to imagine, order, recognize, and remedy [3]. Configuration flaws are firmly identified with fuzzy human components and the plan procedure, which we don't have a strong comprehension. In hardware, outline shortcomings may likewise exist, yet physical failures generally command. An incomplete rundown of the particular qualities of software contrasted with hardware is recorded underneath:

a) Failure cause: Software deserts are primarily configuration surrenders.

b) Wear-out: Software has no imperativeness connected wreck organize. Errors can happen out of the blue.

c) Repairable system thought: interrupted start overs can help settle Software issues.

d) Instance dependence & life cycle: Software reliability is not a segment of equipped time.

e) Ecological parts: Do not impact Software reliability, beside it might impact program inputs.

f) Reliability conjecture: Software reliability can't be expected from any substantial set up, since it depends absolutely on individual factors in design.

g) Replication: can't upgrade Software constancy if undefined Software portions are used.

h) Borders: Software borders are essentially sensible other than illustration.

I) Break Down rate aides: Typically not obvious from examinations of discrete announcements.

j)Strong foundation: In any case, in Software industry, we have not watched this pattern. Code reuse has been around for quite a while, however to an extremely constrained degree. Entirely talking there are no standard parts for software, with the exception of some institutionalized rationale structures.

4. Classification of Software Reliability Methods

The number of methods as there are and numerous all the more developing, none of the methods can catch a fantastic measure of the unpredictability of software; limitations and presumptions must be made for the evaluating procedure [10]. Along these lines, there is no single method that can be utilized as a part of all circumstances. No method is finished or even illustrative. One method may function admirably for an arrangement of certain product, however might be totally off trail for different sorts of issues. Many Software methods hold the going with parts: assumptions, features, and a numerical limit that relates the constancy with the factors. Software reliability methods can be comprehensively characterized into two sorts. These are:

A) Deterministic Method and

B) Probabilistic Method

Each method is particular and can be connected to just a segment of the circumstances and a specific class of the issues. We need to deliberately pick the correct method that suits our particular case. Besides, the displaying comes about cannot be aimlessly accepted and connected. We might additionally clarify the over two methods and their sorts in following areas.

a) deterministic method

The deterministic sort is used to contemplate the segments of a curriculum by counting the amount of executives, operands, and rules, the direct stream of a program by checking the twigs and following the implementation ways and the data stream of a agenda by focus the data distribution and data transient [18]. Execution measures of the deterministic sort are gotten by exploring the program surface and do exclude any sporadic event.

b) probabilistic methods

The probabilistic sort addresses the disappointment occasions and the disappointment departures as the probabilistic events. It can in like manner be furthermore apportioned into different social affairs of methods: (a) screw up seeding, (b) disappointment rate, (c) twist correcting, (d) reliability advancement, (e) program organization, (f) key in zone, (g) execution way, (h) Identical Poisson process.

4.1 Software Reliability Development Representation

In the past decades, a few methods were familiar with assess the reliability of Software systems [10]–[12]. The subject of building advancement methods is from research work [13], [14] which helps in assessing the reliability of an item system previous to its release to the market. Bona fide application, for instance, weapon systems and NASA space convey submissions were researched [15]– [17]. Defects may be involvement in feature released Software. This is a test for Software associations. It might impact their reputation and reserve. Software reliability improvement methods were through and through used to help in handling the amount of deficiencies which is still abides in the item [18]. In this way, deciding the effort required to settle disappointments, the time essential earlier than Software can be released and the estimation of repair. Software trustworthiness improvement methods use structure exploratory data for testing to envision the amount of blemishes remaining in the item.

Software reliability methods can be masterminded to two sorts of methods as demonstrated by gauge style either from:

• the framework restrictions in this way called "deformation thickness" methods

• the test data in this way "Software reliability advancement" methods.

Some known SRGM are Logarithmic, Exponential, Power, S-Shaped and Inverse Polynomial methods [17][19]. They are regular informative methods. They commonly depict the disappointment technique as a part of implementation time (or timetable time) and a plan of darken limitations. The method limitations frequently evaluated using smallest square estimation or most extraordinary likelihood frameworks [13].

4.2 Proposed Fuzzy-Neural Network (FNN) Approach

Setting up a FNN is basically a six-stage system.

1. Right off the bat, the information to be utilized should be characterized and displayed to the FNN as an example of information with the coveted result or target.

2. Besides, the information is ordered to be either in the preparation set or approval (likewise got test and out-of-test) set. The FNN just uses the preparation set in its learning procedure in building up the method. The approval set is utilized to test the method for its prescient capacity and when to stop the preparation of the FNN.

3. Thirdly, the FNN structure is characterized by choosing the quantity of shrouded layers to be built and the quantity of neurons for each concealed layer.

4. Fourthly, all the FNN parameters are set before beginning the preparation procedure.

5. Next, the preparation procedure is begun. The preparation procedure includes the calculation of the yield from the information and the weights. The back-proliferation calculation is utilized to "prepare" the FNN by modifying its weights to limit the contrast between the current FNN yield and the coveted yield.

6. At long last, an assessment procedure must be directed to decide whether the FNN has "learned" to fathom the main job.

4.3 Reliability Analysis

The reliability of a mechanical framework is not the same at each snapshot of the framework's lifetime. Truth be told, three particular "ages" can be seen in many frameworks: the earliest stages time frame, typical operation, and destroy. The earliest stages age is the era quickly following the framework's fabricate and establishment [12]. For most innovative frameworks, this is a basic period; amid which outline and assembling defects will become visible and cause a failure. This situation is alluded to as a "baby mortality" failure. Amid routine operation, failures are typically the aftereffect of chance occasions in the framework's condition, and are not time-subordinate. Startling outside occasions are the prime wellspring of failures amid this day and age [14]. There is a sharp increment in failure toward the finish of the helpful existence of a framework. This is a direct result of parts age and destroys factor of software.

4.4 Proposed Fuzzy-Neural Network (FNN) Method Organization

The goal is to estimate the elements of failure estimations amid the testing procedure and of speaking to it in a solitary nonlinear method we can broaden it by an arrangement of nearby direct methods. Every neighborhood method ought to have the capacity to speaks to the connection between the chronicled issues also, the present failure p(q) in a specific scope of working conditions.

$p(q-1), p(q-2), p(q-3), p(q-4)$

The proposed structure f a fuzzy method can effectively considered for testing by methods for fuzzy If-Then standards. The projected method demonstrate condition is given as takes after:

$$p(q) = SR(p(q-1), p(q-2), p(q-3), p(q-4))$$

Utilizing enrollment capacities and the predecessor of the govern we can characterize the fuzzy locale in the item space. The forerunner factors give the state of the procedure status now. The ensuing of the lead is ordinarily a neighborhood straight relapse demonstrate which relates y(k) with

p(q − 1), p(q − 2), p(q − 3), p(q − 4).

The anticipated strategies do not consider any earlier learning about the working administrations. On the off chance that an adequately number of estimations are gathered which mirrors the working scopes of intrigue, the created fuzzy method will be a productive one.

5. Experiment Results

We run the MATLAB Neural Network Toolbox [36] alongside three participation capacities. The informational collection has the following sections: 1) 65% of the gathered information for preparing and 2) 35% for testing (i.e. approval).

In Figure 1, we demonstrate the bug arrival frequency of 6 projects we considered and for the continuous direct application. We utilized three groups to construct the fuzzy method. Figure 2 demonstrate the time vs reliability over the preparation and testing information for the ongoing direct requests. The fuzzy participation capacities for the armed force purpose and working frameworks are appeared in Fig.3 demonstrate the real and anticipated execution time of delay with reliability failures over the preparation and testing information for the armed forces and working frameworks requests.

The Table 1 illustrates the parameters of the considered projects. The Rsquare value The time vs Reliability measure is illustrated in figure.2

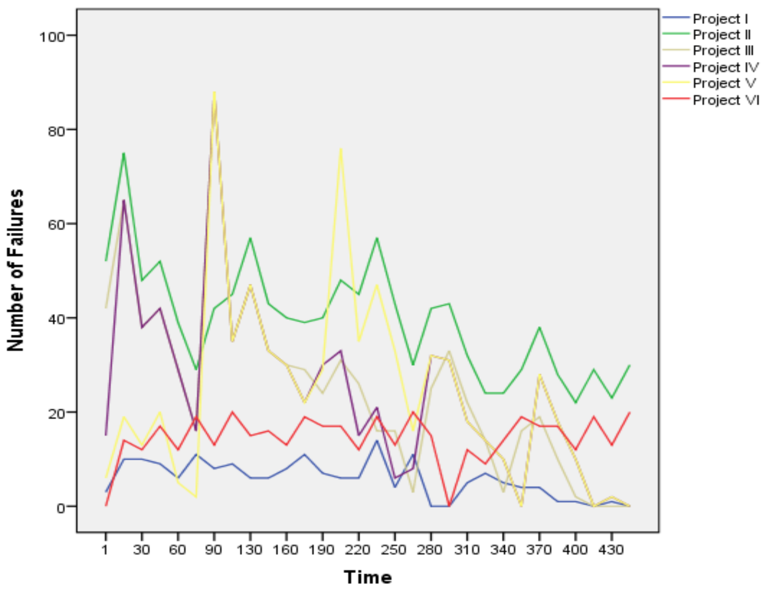

Fig.1. Bug Arrival Frequency for Six Projects

Project	Lambda	Beta	R-square
Project I	0.0008	0.585	0.785
Project II	0.001	1.550	0.228
Project III	0.002	0.695	0.287
Project IV	0.035	1.311	0.348
Project V	0.029	2.292	0.363
Project VI	0.027	1.699	0.279

Table 1. Parameters of considered projects

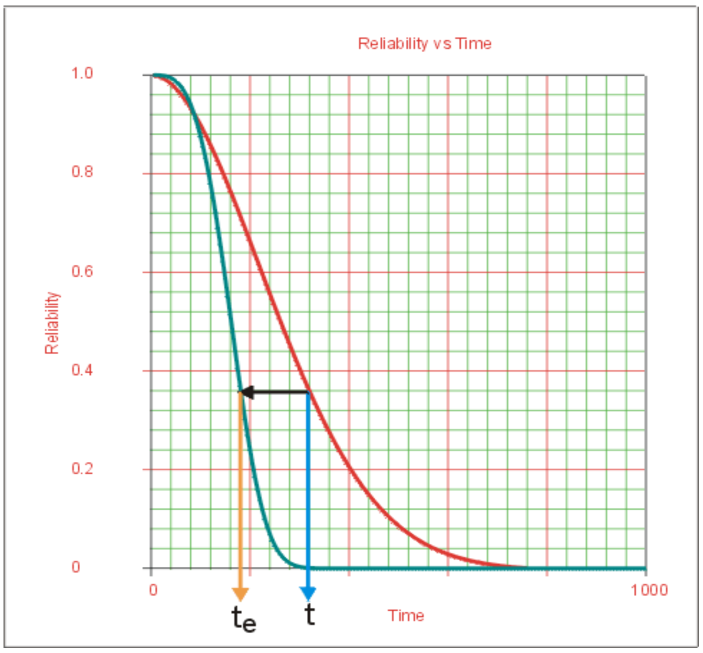

Fig.2. Time Vs Reliablity

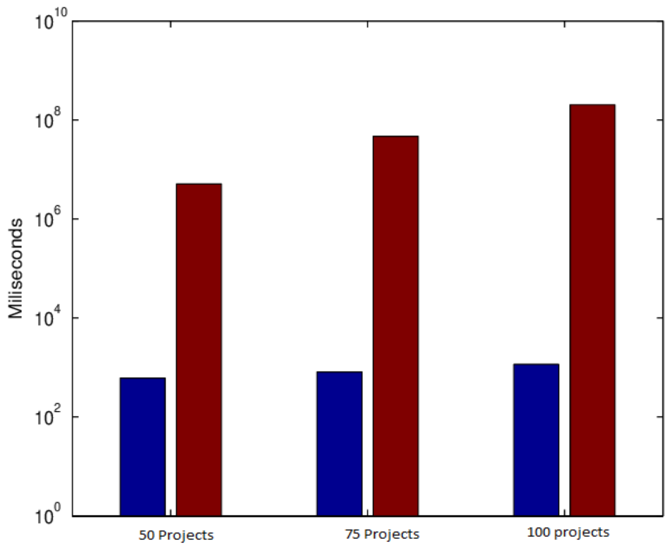

Fig.3. Execution Time of Delay Vs Reliability experiment

6. Conclusion

In this paper we built up an arrangement of fuzzy methods for foreseeing the software reliability in different methodologies. A fuzzy inorder relapse methods were produced for foreseeing the gathered flaws of software building requests or methods. The created fuzzy methods actualized based on the Takagi-Sugeno procedure. The created fuzzy methods were tried utilizing three sorts of uses. They are continuous control, armed forces and working frameworks submissions. The proposed measurements foresee the aggregate number of issues in class. Nonetheless, a similar report can be reached out to foresee the deficiencies as per their seriousness level. Proposed measurements can additionally assess under various modern and scholastic conditions to reclassify the outline measurements as per particular mechanical/scholarly needs to improve reliability. The class-level shortcomings expectation is additionally used/reached out to assess the aggregate deficiencies initiated at framework level. The improvement in the level of reliability through classes update having particular number of shortcomings anticipated through flaws expectation method could be additionally measured utilizing the information from ongoing applications for concentrated and early criticism to achieve reliability. Further, it can draw in the scientists at the colleges, explore labs with subsidized undertaking.

References

1. C. T. Lin, C. Y. Huang, C. C. Sue, Measuring and Assessing Software Reliability Growth Through Simulation Based Approaches, 2007, Proceedings of the 31st IEEE Annual

International Computer Software and Applications Conference (COMPSAC 2007), pp. 439-446, Beijing, China.

2. J. Lo, S. Kuo, M.R. Lyu, C. Huang, Optimal Resource Allocation and Reliability Analysis for Component-Based Software Applications, 2002, Proc. 26th Ann. Int'l Computer Software and Applications Conf. (COMPSAC), pp. 7-12.
3. John D. Musa, Operational Profiles in Software-Reliability Engineering, March 1993, IEEE Software, v.10 n.2, p.14-32.
4. Kishor S. Trivedi, Probability and statistics with reliability, queuing and computer science applications, 2001, John Wiley and Sons Ltd., Chichester, UK
5. K. Kanoun M. Kaaniche C. Beounes J.C. Laprie and J. Arlat, Reliability Growth of Fault-Tolerant Software, June 1993, IEEE Trans. Reliability, vol. 42, no. 2, pp. 205-219.
6. Kumar, M., Ahmad, N., Quadri, S.M.K. (2005), Software reliability growth methods and data analysis with a Pareto test-effort, RAU Journal of Research, vol.15, no. 1-2, pp 124-8
7. Norman F. Schneidewind, Fault Correction Profiles, 2003, Proceedings of the 14th International Symposium on Software Reliability Engineering, p. 257.
8. Quadri, S.M.K., Ahmad, N., Peer, M.A. Software optimal release policy and reliability growth methoding, 2008, Proceedings of 2nd National Conference on Computing for Nation Development, INDIACom-2008, New Delhi, India, pp. 423-431.
9. R.C.Tausworthe, A General Software Relibility Process Simulation Technique, April 1991, NASA JPL Publication, pp. 91-97.
10. R.C.Tausworthe and M.R, Lyu, A generalized technique for simulating software reliability, March 1996, IEEE Software, vol.13, no.2, pp.77-88.
11. Robert C. Tausworthe, Michael R. Lyu, A Generalized Technique for Simulating Software Reliability, March 1996, IEEE Software, vol.13, no.2, p.77-88
12. S.Gokhale, M. R.Lyu, K. S. Trividi, Reliability Simulation of Component-Based Software Systems, November 1998, Proceedings of the 19th International Symposium on Software Reliability Engineering, pp. 192-201, Paderborn, Germany.
13. S.Gokhale, Michael R. Lyu, K.S. Trivedi." Reliability Simulation of Fault-Tolerant Software and Systems, December 1997, In Proc. of Pacific Rim International Symposium on Fault-Tolerant Systems (PRFTS '97), pp. 197-173, Taipei, Taiwan.
14. S. Krishnamurthy, A. P. Mathur, On the Estimation of Reliability of a Software System Using Reliabilities of Its Components, November 02-05, 1997, Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE '97), p.146.

15. Swapna S. Gokhale, Kishor S. Trivedi, A time/structure based software reliability method, 1999, Annals of Software Engineering, vol. 8 no. 1-4, pp. 85-121.
16. Swapna S. Gokhale, Kishor S. Trivedi, Michael R. Lyu, Reliability Simulation of Fault-Tolerant Software and Systems, 1997, Proceedings of the 1997 Pacific Rim International Symposium on Fault-Tolerant Systems, pp.167.
17. S. Y. Kuo, C. Y. Huang, M. R. Lyu, Framework for Methoding Software Reliability, Using Various Testing Efforts and Fault-Detection Rates, Sept. 2001, IEEE Transactions on Reliability, vol. 50, no. 3, pp. 310-320.
18. Tausworthe, Robert C., A General Software Reliability Process Simulation Technique, March 1991, Technical Report 91-7, Jet Propulsion Laboratory, Psaadena, CA.
19. Wood, A., Predicting software reliability, 1996, IEEE Computers, vol.11, pp. 69-77.
20. A.Von Mayrhauser, Y.K. Malaiya, J. Keables, P. K. Srimani, On the Need for Simulation for better Characterization of Software Reliability, 1993, International Symposium on Software Reliability Engineering, Denver, CO.