A Hybrid Quartile Deviation-based Support Vector Regression Model for Software Reliability Datasets

Y. Geetha Reddy[1], Dr. Y Prasanth[2]
Research Scholar[1], Professor[2]
Department of Computer Science and Engineering
KLEF, Guntur District, A.P., India

***Abstract*—Software reliability estimation using machine learning play a major role on the different software quality reliability databases. Most of the conventional software reliability estimation model fails to predict the test samples due to high true positive rate of the traditional support vector regression models. Most of the traditional machine learning based fault prediction models are integrated with standard software reliability growth measures for reliability severity classification. However, these models are used to predict the reliability level of binary class with less standard error. In this paper, a hybrid support vector regression-based quartile deviation growth measure is implemented on the training fault datasets. Experimental results are simulated on various reliability datasets with different configuration parameters for fault prediction.**

***Keywords*—Software fault detection; reliability prediction; support vector machine; exponential distribution; quartile deviation**

I. Introduction

Reliability, in its simplest form, means that a failure cannot occur within a certain period of time. The reliability concept thus stresses the probability, expected function (s), time, and operating conditions of four components. Reliability also depends on the conditions of the system that may or may not change over time. Software systems have increased significantly in size and complexity in recent decades, and the trend is expected to continue in the future [1]. Computer reliability and accessibility, usability, performance, serviceability, capabilities, and documentation are important attributes of software quality. Software reliability is difficult to achieve since software complexity seems to be high. While it is difficult to achieve a certain degree of reliability for any highly complex system, including software, system developers tend to upgrade the software layer with complexity and rapidly developing system sizes. The Software Reliability Growth Model (SRGMs) is a software reliability model (SRMs) design recognition class which is converted into a mathematical model. The reliability assessment of recent system updates is an important challenge in IT software management [2].

The probabilistic models are based on dynamic models and are represented as time-based statistical distributions. All these models are used to predict current trends and predict future trends in reliability. Probabilistic software reliability prediction models use statistical methods to estimate variables such as system error numbers, failure rates, software complexity, programme failures, etc. There are a number of software models in the literature, but none of them is ideal. The selection of an appropriate estimate model based on a specific application is a major research problem [3]. A data set that includes instances of defined classes and a test data set for which the class must be decided must therefore be entered. The quality of the data provided for learning, and also the type of algorithm used in machine learning, depends greatly on the ability to classify successfully. Categorical labels (discrete, unorderly) estimate classification results of continuously valued function models. It implies that numerical data values are expected instead of class marks to be incomplete or inaccessible. Regression analysis is the most widely used statistical method for numerical forecasting. Although other methods are available, the prediction also consists of identifying distribution trends based on available data. Genetic algorithms are also implemented to maximise the number of delayed input neurons and the number of neurons in the neural network's hidden architectural layer. We have demonstrated, using the software model for online adaptation that good-fitness and next-step predictability are better than traditional methods when cumulative software failure times are forecast because those variables' meanings are certainly not known. Many potential values can be equated to the likelihood of occurrence. Therefore, we really don't know when the next loss will happen. We know only a few possible failure times and their likelihood. “T” Two types of fault data, namely, time-domain data and interval-domain data, are widely used in software reliability modeling. The time-domain form is determined by the time the failure occurred. Learning supervised is a methodology for machine learning to build a data structure for preparation. Maximum Likelihood Assessment (MLE) is a common statistical method for the determination of the probability distribution parameters underlying a given dataset. Throughout the literature, there are many predictive models of the reliability of software-based neural networks, which are better known than certain statistical models [4-6]. Computer reliability is one of the key factors taken into account in maintaining the accuracy of the computer. Simply put, software reliability is about system failure or failure [7]. "Success and success" are two distinct variables commonly included in our software development. A fault could be identified as a fault or error during the development phase. As software constraints and modular complexity increase, the manufacture of a quality finished product is too difficult. Software defects may lose cash and time, so bugs for good performance products and decision-makers need to be predicted in advance. As a consequence, these bug accounts contain comprehensive data on bugs along with the seriousness level [8] within different bug tracking frameworks. Generally, software bugs are defective limitations that trigger inaccurate outputs. These limitations can be described as a collection of characteristics to discover the bugs. These features affect the bug prediction model's effectiveness. Applications for software defect detection include decision trees, multifunctional regressions, neural networks, SVM, nave Bayes, and various classification types and selection models. But the relevant flaws for suitable classification could not be selected in these designs. Nave Bayes is a highly efficient method of classification for predicting flaws based on samples of practice. A naive Bayes system sees bug predictions as binary classifiers, i.e., by evaluating historical measurement information, it trains and predicts the predictor. If the attribute kinds in the metric information are blended, errors owing to lacking values or uncertain data are hard to estimate. Three separate layers of dynamic analysis can be classified. A systemic testing layer is the first layer. This layer is designed into the policy to run target programs. These strategies are designed to efficiently achieve error states. The second layer is a layer of data retrieval. In order to check programme correctness, data on the inner behaviour of the target programmes is collected. In the third level, monitors create from the information collected an abstract model of the destination programme and then check for possible errors in the programme on the abstract model. The test limits are fundamental to all dynamic analysis techniques. Dynamic analysis cannot support full target-program analysis because it uses monitored partial programme compliance. The other restriction is that it is hard to implement dynamic analysis methods except for the completion of the target programs. Executable environments and sample instances involve dynamic analysis methods. [9] the significance of various software prediction-model metrics. In this model, correlations and metric events were introduced using distinct algorithms in the bug forecast model, and bug counts were calculated in each metric. Object-oriented measurement metrics for object-oriented quality software A model for bug-projection was proposed and its levels with high, medium, and low severity defects were found to be lower than traditional models with various severe values. The technique of regression is intended to predict the quantity and density of software defects. The technique of classification aims at determining whether or not a software module (e.g. a package, code, or file) has a higher risk of defect than another. This approach [10] uses fuzzy logic with neural networks in software reliability prediction. The recurrent neural network is trained using the back-propagation algorithm. The number of failures and cumulative execution time in the failure dataset are used as inputs to the network to predict the next step failure.

The rest of the paper is organized as follows. Section II describes the related works of the SRM and its limitations. Section III describes the proposed solution to the SRM based machine learning framework on different dataset. Section IV describes the experimental results and analysis. Finally, we conclude the paper in Section V.

II. Related Work

Lazarova et al. have developed various SRGMs concerning the growth rate software reliability index for error detection [11]. Li et.al, proposed a measuring method as an indicator collection, gathering data for the testing of all those metrics [12]. Mirchandaniet al. suggested the non-homogeneous Poisson method-based software reliability growth pattern because the detection of these errors might also lead to detection of other errors without failure [13]. Nagaraju proposed an evolutionary model of the neural network to estimate and predict the software reliability based on a multimedia architecture input and output. In this study, the development of neural network models for software-reliability predictions was proposed using an Exponential and Logarithmic Encoding Scheme. Neural network models with the two encoding schemes above have shown a better prediction of cumulative failures than some statistical models. However, [14] the value of the encoding parameter is calculated by repeated hit / test experiments. This paper presents recommendations for encoding parameter selection, which provide consistent results for various data sets. The proposed solution is implemented using 18 separate data sets and a clear result for all datasets is observed. The method was compared to known statistical models using three sets of change points.

Rani [15] proposed a neural network approach focused on predictions of software reliability. He compared the approach to parametric model recalibration with some meaningful predictive measures with the same data sets. We concluded that better predictors are neural network methods.

Rizvi et al. [16] proposed a system in which software reliability based on the neural network would be expected. They used the reverse propagation algorithm for instructions. They used several failure times in the last 50 to estimate the next failure as output. The performance of approaches was calculated by changing the number of input nodes and hidden nodes. We concluded that the success of the strategy usually depends on the quality of the data sets.

Sagar [17] submitted a neural network approach focused on the evolutionary prediction of device reliability. They used single output architecture with multiple delayed inputs. Vojdani [18] suggested two models for cumulative system failure estimation, such as exponential neural network encoding (NNEE) and logarithmic encoding (NNLE). He encoded the data with the above two encoding scheme, i.e. the time of execution. He used the four dataset method and compared the results with some statistical models and found better results than those models.

Wang et al. [19] have proposed to reuse data from previous projects / releases for failure to boost early reliability for current projects / releases. Wang et al.[20] proposed the combinational dynamic weighted model (DWCM) based on a neural network for the prediction of device reliability. Based on the software-reliability growth model (SRGM), they used various activation functions within the secret layer. The method was used on two sets of data and the effect was compared with certain statistical models. The experimental results indicate that the DWCM approach is more successful than traditional models. The neural network is a methodology for performance computation. The machine performance can previously be predicted on the basis of our neural network architecture. The system is also trained unless its desired output or destination can be achieved. For training purposes, we use different learning techniques that are freely described as supervised and unattended learning [21]. Software reliability is a quantitative study of every software designed since it affects directly software quality [22]. An efficient software reliability model is required in order to achieve good results. The previously developed reliability model is based on the analysis of faults linked to the code and context in which it was implemented [23]. All software reliability models are designed based on the execution time and calendar time. The time required or spent by the processor in the execution of instructions from the program is the execution time of any program [24].

Research Gaps: From the literature works, the main gaps identified for the software reliability estimation process are:

1) Difficult to predict the new reliability test data using machine learning approach.

2) Traditional SVM require different hyper parameters in order to improve the classification optimization.

III. Proposed Geometric Perturbation based Privacy Preserving Classifier

In this section, a statistical quartile deviation-based improved SVR prediction model is proposed on the software reliability datasets. In this work, a novel approach to predict the software reliability on the training and test software fault data. This model is integrated with the quartile deviation growth function in order to fit the S shaped curve. In the proposed model, reliability estimation is performed in two phases. In the initial phase, quartile deviation based error estimation is calculated on the training data for software reliability prediction. In the second phase, a hybrid support vector regression model is designed and implemented on the computed S-shaped training data as shown in Fig. 1.

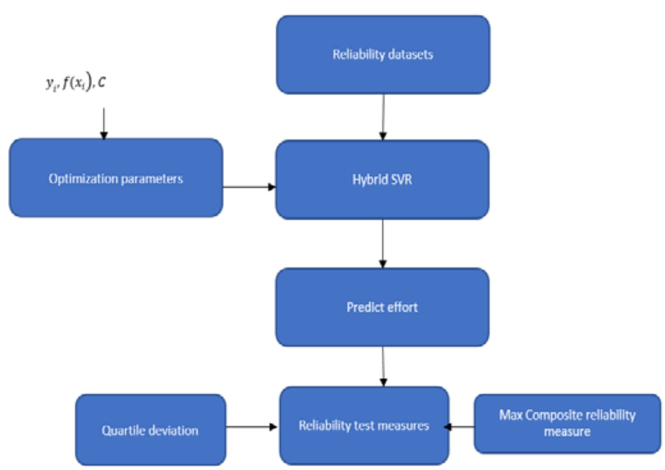

Figure 1: Max composite based SVR Framework

Fig. 1. Proposed Ensemble Deep Learning Framework for Privacy Preserving.

In the proposed model, an enhanced support vector regression is designed and implemented on the software fault dataset to improve the prediction rate and to minimize the error rate. The following proposed SVR model is implemented on the fault data. Initially, input data is given to hybrid SVR model to predict the effort rate. The prediction values of the SVR are tested using the Quartile deviation model and maximized composite reliability measures. These measures are used to find the deviation, skewness and shape of the dataset. The impact of failures on decision making is calculated using traditional software metrics. Extensive research was done using one or two software stage metrics to discover the error models. However, redundant and meaningless characteristics affect the effects of traditional designs. Also, the relationship between the new metrics and the traditional metrics is becoming too complex to make decisions as the number of software metrics increases. Generally, software metrics are used to gain quantitative insight into the software or its characteristics. The value of metrics is an ordinal, an interval, or a nominal scale. Software quality is assessed by the various features such as performance, documentation, easy maintenance and system soundness. Software reliability is considered as it is difficult to achieve with the complex nature of software. The software is therefore layered by the system developers throughout the design process to achieve a certain level of reliability, to support the later update of the software system and also to incorporate elasticity for increased system size. The reliability of software is inversely linked to the level of software complexity, since complexity is directly associated with enhanced capacity and strong software system features with enhanced functions. The main objective of this paper is to improve the software reliability prediction using the hybrid SVR model.

Let m(x) be the input data, m be the estimation function. M value is estimated by using multiple linear regression method. Then the objective function of the proposed SVR model is given as

$$C(x) \coloneqq \frac{1}{2} \| w \|^2 + \xi.\psi(x).\phi(x)$$

Where

$$\psi(x) = | m(x) - m(x) | = | m(x) - MLR(x) |$$

MLR(x) = Multiple linear regression

$$\phi(x, x') = e^{-\|x - x_i\|^2 / 2.\sigma^2}$$

$$\min_{\xi_k, \xi_k^*} C(x) = \frac{1}{2} \| w \|^2 + \lambda.\psi(x).\phi(x) + b$$

$$\min_{\xi_k, \xi_k^*} C(x) = \frac{1}{2} \| w \|^2 + \lambda.| \xi_k - \xi_k^* |.\phi(x) + b$$

IV. Experimental Results

In this work, experimental results are simulated with java environment for different software reliability datasets. The first, second, third and fourth datasets DS1, DS2, DS3, and DS4 are taken from Rome air development centre (RADC) projects. Each dataset and its type are given in Table I, Table II, Table III and Table IV.

TABLE I. DS1 FOR FAULT PREDICTION BASED ON SEVERITY LEVEL

W	CF	Label
1	16	L
2	24	L
3	27	L
4	55	M
5	41	L
6	49	L
7	54	M
8	58	M
9	69	M
10	75	H
11	81	H
12	86	H
13	90	H
14	93	H
15	96	H
16	98	H
17	99	H
18	100	H
19	100	H
20	100	H

TABLE II. DS2 FOR FAULT PREDICTION BASED ON SEVERITY LEVEL

W	CF	Label
1	28	L
2	29	L
3	29	L
4	29	L
5	29	L
6	37	M
7	63	M
8	92	H
9	116	H
10	125	H
11	139	H
12	152	H
13	164	H
14	164	H
15	165	H
16	168	H
17	170	H
18	176	H

TABLE III. DS3 FOR FAULT PREDICTION BASED ON SEVERITY LEVEL

W	F	label
40	71	M
41	72	M
42	74	M
43	74	M
44	80	M
45	84	M
46	84	M
47	84	M
48	84	M
49	85	H
50	86	H
51	89	H
52	90	H
53	90	H
54	92	H
55	108	H
56	120	H
57	128	H
58	129	H
59	139	H
60	146	H

TABLE IV. DS4 FOR FAULT PREDICTION BASED ON SEVERITY LEVEL

W	F	Label
33	79	L
34	80	L
35	82	L
36	83	L
37	83	L
38	84	L
39	84	L
40	85	M
41	85	M
42	87	M
43	87	M
44	87	M
45	89	M
46	89	M
47	91	H
48	91	H
49	94	H

Fig. 2 represents the mean time to failure rate and its runtime for the proposed exponential distribution function. Here, the proposed exponential function is used to test the reliability of the given input parameters.

Fig. 3 represents the F-measure rate for the proposed exponential distribution function to the existing models. Here, the proposed exponential function is used to test the reliability of the given input parameters. From the figure, it is observed that the proposed approach has better improvement over the conventional model on all the datasets.

Fig. 4 represents the recall rate for the proposed exponential distribution function to the existing models. Here, the proposed exponential function is used to test the reliability of the given input parameters. From the figure, it is observed that the proposed approach has better improvement over the conventional model on all the datasets.

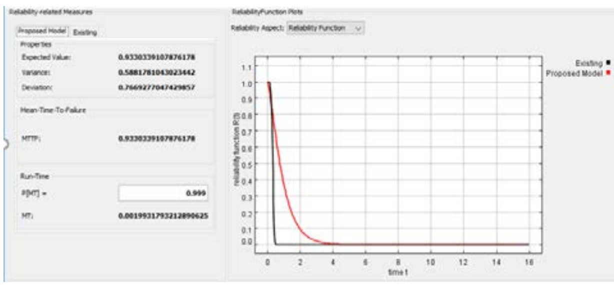

Fig. 2. Mean Time to Failure Rate and Runtime of the Proposed Model to the Exponential Model.

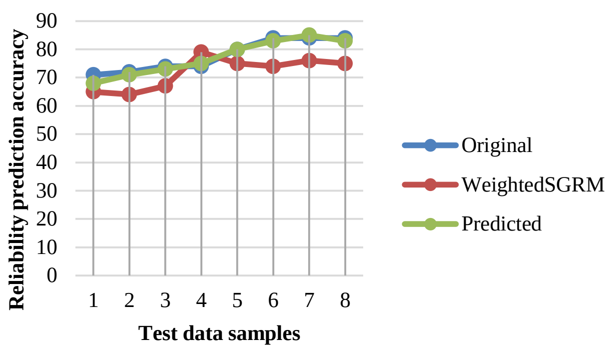

Fig. 3. Comparison of Proposed Fault Prediction Model to Existing Weighted SGRM Model on All Datasets.

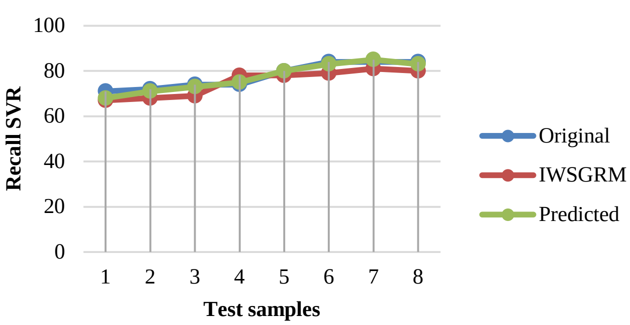

Fig. 4. Comparison of Proposed Fault Prediction Model to Existing Improved Weighted SGRM Model on All Datasets.

V. CONCLUSION

Software reliability fault prediction plays a vital role in small- and large-scale software applications. In this paper, a hybrid support vector regression-based quartile deviation model is implemented on the different software reliability datasets. Most of the traditional machine learning based fault prediction models are integrated with standard software reliability growth measures for reliability severity classification. However, these models are used to predict the reliability level of binary class with less standard error. Experimental results proved that the proposed reliability fault prediction model has better performance in terms of prediction and time is concerned.

VI. FUTURE WORK

In future work, a supervised learning model is integrated to the SVR model in order to predict the reliability for the new unclass labelled data.

REFERENCES

[1] J. Cho, S. M. Shin, S. J. Lee, and W. Jung, “Exhaustive test cases for the software reliability of safety-critical digital systems in nuclear power plants,” Nuclear Engineering and Design, vol. 352, p. 110151, Oct. 2019, doi: 10.1016/j.nucengdes.2019.110151.

[2] L. V. Utkin and F. P. A. Coolen, “A robust weighted SVR-based software reliability growth model,” Reliability Engineering & System Safety, vol. 176, pp. 93–101, Aug. 2018, doi: 10.1016/j.ress.2018.04.007.

[3] E. Abuta and J. Tian, “Reliability over consecutive releases of a semiconductor Optical Endpoint Detection software system developed in a small company,” Journal of Systems and Software, vol. 137, pp. 355–365, Mar. 2018, doi: 10.1016/j.jss.2017.12.006.

[4] C. Jin and S.-W. Jin, “Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization,” Applied Soft Computing, vol. 40, pp. 283–291, Mar. 2016, doi: 10.1016/j.asoc.2015.11.041.

[5] M. S. Alhammadi, B. S. Almaqrami, and B. Cao, “Reliability of Beta-angle in different anteroposterior and vertical combinations of malocclusions,” Orthodontic Waves, vol. 78, no. 3, pp. 111–117, Sep. 2019, doi: 10.1016/j.odw.2019.02.002.

[6] D. Amara and L. B. Arfa Rabai, “Towards a New Framework of Software Reliability Measurement Based on Software Metrics,” Procedia Computer Science, vol. 109, pp. 725–730, Jan. 2017, doi: 10.1016/j.procs.2017.05.428.

[7] J.-E. Byun, H.-M. Noh, and J. Song, “Reliability growth analysis of k-out-of-N systems using matrix-based system reliability method,” Reliability Engineering & System Safety, vol. 165, pp. 410–421, Sep. 2017, doi: 10.1016/j.ress.2017.05.001.

[8] F. Febrero, C. Calero, and M. Ángeles Moraga, “Software reliability modeling based on ISO/IEC SQuaRE,” Information and Software Technology, vol. 70, pp. 18–29, Feb. 2016, doi: 10.1016/j.infsof.2015.09.006.

[9] A. Lanna, T. Castro, V. Alves, G. Rodrigues, P.-Y. Schobbens, and S. Apel, “Feature-family-based reliability analysis of software product lines,” Information and Software Technology, vol. 94, pp. 59–81, Feb. 2018, doi: 10.1016/j.infsof.2017.10.001.

[10] V. Ivanov, A. Reznik, and G. Succi, “Comparing the reliability of software systems: A case study on mobile operating systems,” Information Sciences, vol. 423, pp. 398–411, Jan. 2018, doi: 10.1016/j.ins.2017.08.079.

[11] S. Lazarova-Molnar and N. Mohamed, “Reliability Assessment in the Context of Industry 4.0: Data as a Game Changer,” Procedia Computer Science, vol. 151, pp. 691–698, Jan. 2019, doi: 10.1016/j.procs.2019.04.092.

[12] Q. Li and H. Pham, “NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and

testing coverage," Applied Mathematical Modelling, vol. 51, pp. 68–85, Nov. 2017, doi: 10.1016/j.apm.2017.06.034.

[13] C. Mirchandani, "Adaptive Software Reliability Growth," Procedia Computer Science, vol. 140, pp. 122–132, Jan. 2018, doi: 10.1016/j.procs.2018.10.309.

[14] V. Nagaraju, V. Shekar, J. Steakelum, M. Luperon, Y. Shi, and L. Fiondella, "Practical software reliability engineering with the Software Failure and Reliability Assessment Tool (SFRAT)," SoftwareX, vol. 10, p. 100357, Jul. 2019, doi: 10.1016/j.softx.2019.100357.

[15] P. Rani and G. S. Mahapatra, "A novel approach of NPSO on dynamic weighted NHPP model for software reliability analysis with additional fault introduction parameter," Heliyon, vol. 5, no. 7, p. e02082, Jul. 2019, doi: 10.1016/j.heliyon.2019.e02082.

[16] S. W. A. Rizvi, V. K. Singh, and R. A. Khan, "Fuzzy Logic Based Software Reliability Quantification Framework: Early Stage Perspective (FLSRQF)," Procedia Computer Science, vol. 89, pp. 359–368, Jan. 2016, doi: 10.1016/j.procs.2016.06.083.

[17] B. B. Sagar, R. K. Saket, and Col. Gurmit Singh, "Exponentiated Weibull distribution approach based inflection S-shaped software reliability growth model," Ain Shams Engineering Journal, vol. 7, no. 3, pp. 973–991, Sep. 2016, doi: 10.1016/j.asej.2015.05.009.

[18] A. Vojdani and G. H. Farrahi, "Reliability assessment of cracked pipes subjected to creep-fatigue loading," Theoretical and Applied Fracture Mechanics, vol. 104, p. 102333, Dec. 2019, doi: 10.1016/j.tafmec.2019.102333.

[19] J. Wang, Z. Wu, Y. Shu, and Z. Zhang, "An optimized method for software reliability model based on nonhomogeneous Poisson process," Applied Mathematical Modelling, vol. 40, no. 13, pp. 6324–6339, Jul. 2016, doi: 10.1016/j.apm.2016.01.016.

[20] J. Wang and C. Zhang, "Software reliability prediction using a deep learning model based on the RNN encoder–decoder," Reliability Engineering & System Safety, vol. 170, pp. 73–82, Feb. 2018, doi: 10.1016/j.ress.2017.10.019.

[21] J. Yang, Y. Liu, M. Xie, and M. Zhao, "Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes," Journal of Systems and Software, vol. 115, pp. 102–110, May 2016, doi: 10.1016/j.jss.2016.01.025.

[22] O. Yazdanbakhsh, S. Dick, I. Reay, and E. Mace, "On deterministic chaos in software reliability growth models," Applied Soft Computing, vol. 49, pp. 1256–1269, Dec. 2016, doi: 10.1016/j.asoc.2016.08.006.

[23] M. Zhu and H. Pham, "A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal," Computer Languages, Systems & Structures, vol. 53, pp. 27–42, Sep. 2018, doi: 10.1016/j.cl.2017.12.002.

[24] B. Zou, M. Yang, E.-R. Benjamin, and H. Yoshikawa, "Reliability analysis of Digital Instrumentation and Control software system," Progress in Nuclear Energy, vol. 98, pp. 85–93, Jul. 2017, doi: 10.1016/j.pnucene.2017.03.006.