International Journal of Engineering Research and Advanced Technology (IJERAT)

DOI: 10.31695/IJERAT.2020.3665

E-ISSN : 2454-6135

Volume.6, Issue 11
November -2020

Optimization of Interval Type-2 Fuzzy Logic System for Software Reliability Prediction

Ini Umoeka[1], Imo Eyoh[2], Edward Udo[3], Veronica Akwukwuma[4]

[1-3]Department of Computer Science, University of Uyo, Uyo

Akwa Ibom State, Nigeria

[4]Department of Computer Science, University of Benin, Benin City

Edo State, Nigeria

ABSTRACT

Since real world application is fraught with high amount of uncertainty, such as applicable to software reliability, there should be a method of handling the uncertainty. This paper presents a model to properly handle uncertainty in software data for effective prediction of the reliability of software at the early phase of software development process. In this paper we employed a Takagi-Sugeno-Kang (TSK)-based interval type 2 fuzzy logic systems with artificial neural network learning for the prediction of software reliability. The degree of membership grades of the interval type 2 fuzzy sets (IT2FSs) are obtained using interval type Gaussian membership function with fixed mean and uncertain standard deviation. The parameters of the IT2FLS membership functions are optimized using gradient descent (GD) back-propagation algorithm. As inputs to the system, reliability relevant software requirement metrics and the software size metrics are used. The proposed new approach makes use of qualitative data of requirement metrics of twenty three real software projects to examine its predictive ability. The performance of the model is evaluated using five performance metrics and found to provide better results when compared with existing approaches.

Key Words: Interval type-2 fuzzy logic; Software reliability; Software metrics; Gradient descent back propagation algorithm.

1. INTRODUCTION

Software reliability is an area of software engineering which deals with the failure free execution of a software. Failures caused by software defects are very common and can have adverse effect on both reliability and safety of the system. The reliability of software is very important at the early stage of software development process [1]. Thus, the necessity of guaranteeing the reliability of these software systems by fixing the faults as early as possible during the process of software development cannot be overemphasized. Since reliability has become a cardinal factor in software systems, its prediction is of utmost importance. Many sensitive operations within the human endeavours are now relying solely on software and people may be confident that software will respond correctly to commands and situations. This assumption is not always true, so we need to assess the reliability level of software through proper analyses and predictions. Since most software are complex and difficult in nature, reliability analyses and predictions also tend to be complex and difficult. As such, developers may find it very difficult to state the level of software reliability.There are many software reliability prediction (SRP) models in literature, but it is quite difficult for practitioners to implement these techniques in software production environment because they must decide on the amount of data to collect and appropriate software reliability model and methods to adopt. Since the nature of software engineering requires measurements to be made, reliability prediction approaches continue to be identified so as to aid software developers; and in order to express software product reliability, reliability relevant metrics are needed.The choice of which metrics to use depends on the type of software system under scrutiny and the requirements of the application domain. Software reliability and quality has become a primary concern for both designers and developers throughout the entire software development process. According to [2], software reliability is the probability to perform failure free operation and produce correct output for a specified time under specified conditions while Software quality is defined as the degree to which a system component or process meets customer requirement [3].Among the many software quality attributes like functionality, usability, capability, maintainability, performance, serviceability and documentation; reliability is a major factor to assure quality of the software [4], [5]. Generally, software

reliability is seen as the major factor that affects the quality of any software because it quantifies software failures that renders the system inoperative or risky [6]. Software reliability can be predicted either in the later or early stage of software development process [1, 7-11].

Predicting the accuracy of software reliability via software reliability models is mostly possible at the later stage of SDLC [1]. However, predicting reliability in the early phase of SDLC is cost effective [1], [12], [13]. Most of SRP models have their basis on failure or fault data. But, the availability of failure or fault data in the early phases of SDLC is always a major challenge. However, qualitative values of software metrics are available in the early phases of SDLC and this could be used to predict residual defects in software. The metrics which influence software reliability in SDLC have been identified in [14], [15], [16],[17]. In fact, most of these software metrics are associated with uncertainty which could be as a result of unrealistic assumptions and some measures that cannot be described precisely during software development process. Since real world application is loaded with high amount of uncertainty, there should be a method of handling the uncertainty. Hence the concept of fuzzy logic system, which plays a vital role in uncertainty modeling, is of utmost importance.

2. RELATED WORK

Several studies have been presented in the literature for predicting software faults and reliability based on type-1 fuzzy logic approaches ([8-22]). Type-1 fuzzy set (T1FS) proposed by Zadeh in 1965 [23] is an extension of the binary logic. Unlike binary logic with 0 or 1 membership function, T1FS membership functions are multi-valued in the interval [0, 1] and can cope with uncertainty in data to some degree. Type-1 fuzzy logic system (T1FLS), when compared with traditional approaches, prove to be more robust as they are capable of dealing with the incompleteness, vagueness and imprecision in the data. Cai et al., [18] discussed the development of fuzzy software reliability models in place of probabilistic software reliability models (PSRMs). Their claim was based on the evidence that software reliability is fuzzy in nature. The authors showed a demonstration of how to develop a fuzzy software reliability model (FSRM) to characterize software reliability.Khatatneh and Mustafa [19] presented a T1FL model technique on a custom set of test data. The model is characterized as a growth reliability model whose focus was on a particular dataset behavior in predicting reliability. Their model predicted failures during the software testing process on a dataset from command and control applications.Aljahdali [20] presented a model for SRP using T1FL and Normalized Root of Mean of the Square of Error (NRMSE). The author's model design was based on the Takagi-Sugeno (TS) fuzzy model.Kumar *et al*., [21] proposed a T1FL model for SRP using three parameters (availability, Failure Probability and Recoverability) as an integrated measure of software reliability.Jaikumar and Ramani [22] developed T1FLS to detect defects in software using metrics from different SDLC including requirements, design, coding and testing phases respectively.Various works have been proposed for predicting software faults and reliability in the early stage of SDLC ([7], [8], [24], [9], [1], [25], [10], [11]). Among these methods, some researchers ([8], [24], [9], [1], [25], [10],[11]) have used T1FLS. For example, Pandey and Goyal [8] presented a T1FL model for early software fault prediction. The model predicts number of faults at the end of each software development phase before testing, using reliability relevant software metrics and the level of developer's Capability Maturity Model (CMM) level.Pandey and Goyal [24] also presented another model for fault prediction based on type-1 fuzzy sets. The model considered software metrics of the requirement, design, coding and testing phase. This approach constructed fuzzy profiles of each input metric using triangular membership functions. Yadav et al., [9] developed a model to predict the number of residual faults before testing phase. The model made use of T1FLS with the software metrics to predict the remaining defects in the software expected during testing or when the software would be actually used.Yadav and Yadav [1] presented another model to predict the software defect density indicator at the requirement, design and coding phase of SDLC based on T1FLS using the reliability relevant software metrics of early artifacts. The proposed model made use of twenty real software projects.Again, Yadav and Yadav [25] proposed another T1FLS-based model that calculates the number of software defects at the end of testing phase. The proposed model considered requirement analysis, design, coding and testing metrics. These metrics are assessed in linguistic terms. Rizvi*et* al., [10] presented a reliability prediction model using fuzzy inference system of type-1 to predict the reliability of the developing software. The study focused on the reliability prediction prior to coding phase and the model was statistically validated through the dataset gotten from twenty real projects. Rizvi*et al*., [11] again, proposed a reliability prediction model that utilizes early stage based measures from requirements and object-oriented design stage. The model made use of FLS of type-1 to predict reliability at the requirements as well as design level through its output variable. However, T1FLS, despite its widespread use has membership functions (MFs) that are precise and makes it inadequate to handle uncertainty associated with the inputs and outputs of FLS [26], [27], [28]. The uncertainty in T1FSs (see Figure 2.) disappears the moment its MFs are specified [29], leaving crisp numerical values.

DOI : 10.31695/IJERAT.2020.3665

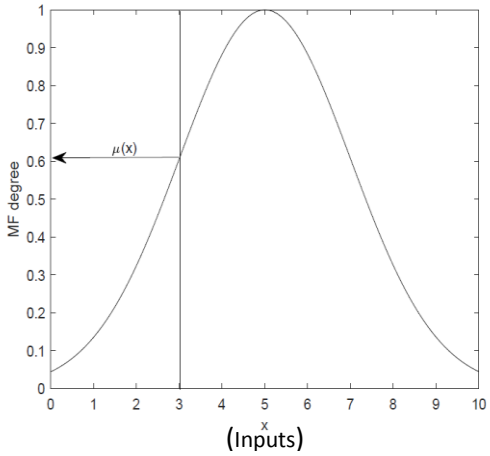


Figure 2: A Gaussian type-1 Fuzzy Set [30]

Based on this premise, Zadeh [31] presenteda type-2 fuzzy set (T2FS) as an extension of T1FS whose MFs are T1FS with a third dimension. T2FSs may be general T2FS (GT2FS) or interval T2FS (IT2FS) (see Fig. 3). GT2FS are three dimensional FS with varying weights on the third dimension.

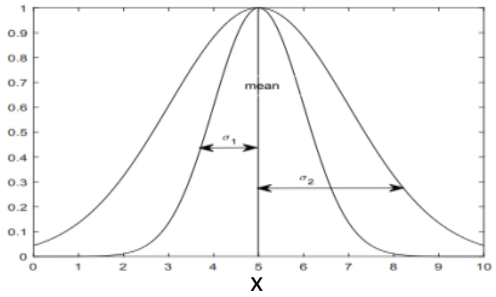


Fig.3 Interval type-2 fuzzy set with uncertain standard deviation and fixed mean

The representation of GT2FS makes it computationally expensive to work with. Thus, many researchers resort to the use of IT2FS whose third dimension takes the value 1 and therefore carries no information [32] and can simply be represented on a 2-dimensional plane. These make IT2FS simple to use with less computational burden. An IT2FLS makes use of type-2 fuzzy sets (T2FSs), which are higher order fuzzy sets that can adequately handle the uncertainty in the linguistic information [26], [27], [29]. Hence, this study presents a TSK-based IT2FLS-SRP with ANN learning at the early stage (requirement) of SDLC.Shown in Figure 4, is the structure of IT2FLS.

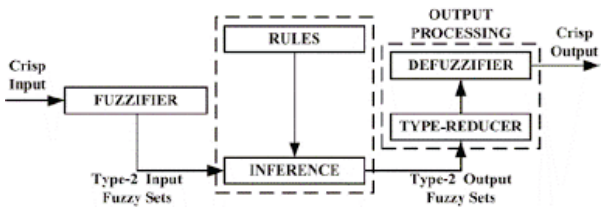


Figure 4: Structure of IT2FLS

Closely related to this study is the work of Chatterjee et al., [33]. The authors developed a fuzzy-rule based generation algorithm using IT2FLS to predict fault in early phase of software development. The model in [33] made use of Gaussian MFs with uncertain mean which according to the literature is not differentiable at all points [34]. The authors in [33] also adopted Mamdani fuzzy inference which involves high computational cost due to the time wastage in defuzzification process. Moreover, the model proposed in [33] has no form of parameter optimization. Based on these premise, we are motivated to address these shortcomings by proposing an optimized IT2FLS-SRP for the prediction of software reliability in the early stage (requirement). The term 'early' in SDLC refers to requirements, design and coding phases,but in this study, only requirements phase is considered. Different from [33], the parameters of the IT2FLS-SRP employed in this paper are optimized for the first time using GDbackpropagation method. The proposed model adopts a TSK fuzzy inference. Unlike Mamdani fuzzy inference in [33], the TSK fuzzy inference is computationally efficient, accurate and more flexible than Mamdani [35] and specifically works well with optimization problems

DOI : 10.31695/IJERAT.2020.3665

[36]. The proposed IT2FLS-SRP makes use of Gaussian MF with uncertain standard deviations. According to [34], the Gaussian MF that has uncertain standard deviations is the only known MF that is continuous at all points and well suited for optimization problem undertaken in this work. The contributions of this paper therefore are as follows: 1) optimization of the parameters of IT2FLS-SRP for the first time using GDbackpropagation algorithm. 2) managing the varying degrees of uncertainties in the rule base using a user-defined parameter, β.

The rest of this paper is organized as follows: Proposed model for reliability prediction is presented in section 3. Section 4 describes the case studies for twenty three real projects. Reliability prediction based on model accuracy is presented in section 5. Section 6 describes the model prediction based on evaluation performance metrics and conclusion is given in section 7.

3. PROPOSED MODEL FOR RELIABILITY PREDICTION

The proposed model utilizes the most significant reliability relevant metrics in [15] and [16] from early stages of SDLC. The inputs to the IT2FLS-SRP model are the requirement metrics (RMs) and Kloc. The three requirement metrics in [15] and [16] also used in this work include requirement complexity (RC), requirement stability (RS) and Experience of requirement team (ERT) with requirement phase reliability (RPR) as the model output. The architecture of the proposed model is shown in Figure 5.

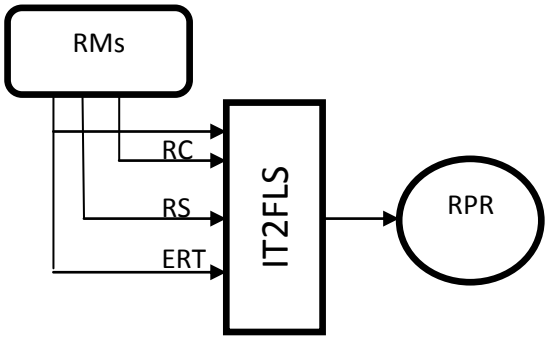


Figure 5: Proposed Model Architecture

3.1 Requirements Software Metrics

Therequirements metrics employed as inputs in the proposed model are as follows [37]:
(a). Requirement complexity (RC) High RC means increase in the number of faults which leads to low reliability of software.
(b). Requirement stability (RS) This metric quantifies the stability of requirement specification in the requirement phase of SDLC. More RS increases the number of faults which causes the reliability to be low.
(c). Experience of requirement team (ERT). This metric determines the relevant and skill set of team staff involved in project execution at the requirement stage of SDLC. These metrics are chosen because they are top significant reliability relevant metrics [16] from early stages of SDLC.
The TSK-IT2FLS is implemented using the architectural design model as depicted in Figure 6.

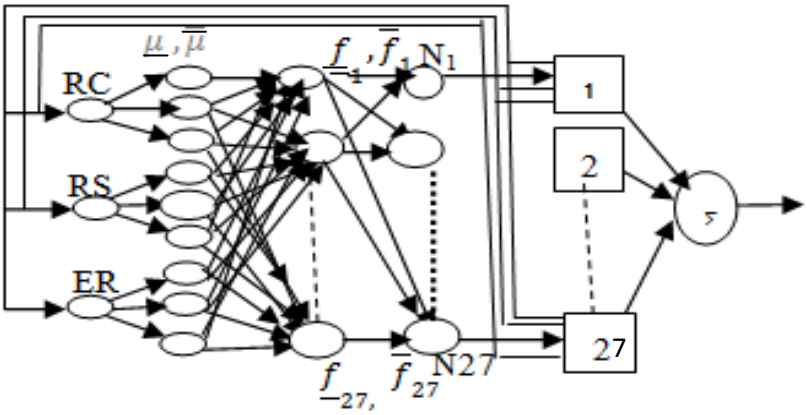


Figure 6: Architecture of TSK-IT2IFLS-SRP

DOI : 10.31695/IJERAT.2020.3665

The architectural design comprises six layers. Layer 1 (Input Layer): The external input signals are distributed in this layer. Layer 2 (Membership function layer) translates the external inputs into IT2FSs (RMs). IT2FS are obtained using IT2 Gaussian MF with a fixed mean and uncertain standard deviation which is calculated as in (1).

$$\mu_{\tilde{A}_{ik}}(x_i) = \exp\left[-\frac{(x_i - m_{ik})^2}{2\sigma_{,ik}^2}\right] \quad (1)$$

where μ is MF, $\tilde{A}_{ik}$ is IT2FS, x'_is are the inputs, m and σ are antecedent parameters. The IT2 MF with uncertain standard deviation is as shown in Figure 3.

In this paper, the range (universe of discourse) of the membership grades of all input metrics is presented in a normalized form, i. e. between 0 and 1. The input metrics, with their ranges and linguistic terms, low (L), Medium (M) and high (H) are as shown in Table 1.

Table 1: Requirements phase metrics with their linguistics states and fuzzy range

Requirements Metrics	Linguistics State	Fuzzy Range
RC	{L, M, H}	{0, 1}
RS	{L, M, H}	{0, 1}
ERT	{L, M, H}	{0, 1}

This paper adopts domain experts MFs definitions for software metric inputs using Gaussian IT2 membership functions. The partitioning of the requirement metrics input space into IT2FS is as shown in Fig. 4

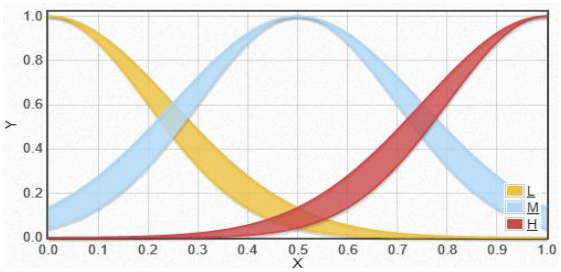


Figure 7: IT2 Membership functions for requirement metrics input variables

In layer 3 (Rule layer), the fuzzy rules for each input variables are obtained using IF-THEN TKS fuzzy inference. The TSK-IT2FLS used here consists of IT2FS in the antecedent part and crisp values in the consequent part otherwise known as A2-CO. The IT2-TSK fuzzy rules used in this paper is represented in (2)

$$R_k: IF x_1 is \tilde{A}_{1k} and x_2 is \tilde{A}_{2k} and \ldots and x_n is \tilde{A}_{nk} THEN y_k = \sum_{i=1}^{n} w_{ik}\, x_i + b_k \quad (2)$$

Here $x_1, x_2, \ldots, x_n$ are the input variables, $\tilde{A}_{1k}$, $\tilde{A}_{2k}, \ldots, \tilde{A}_{nk}$ are antecedent IT2FSs of the k^{th} rule of the i^{th} inputs which are represented as Gaussian MF and $y_k (k = 1, \ldots, M)$ is the output of the k[th] rule which is a linear combination of the input vector. w_{ik} (consequent coefficient)$(i = 1, \ldots, n)$ and b_k $(k = 1, \ldots, M)$ are the function parameters in the consequent part of the rules. A set of firing strength for each range of input variables are computed using algebraic product (t-norm operator) operation. The firing strength indicates an interval type-1 fuzzy set $\left[\underline{f_k}, \overline{f}_k\right]$ and can be computed as [38]:

$$F_k = \left[\underline{f_k}, \overline{f}_k\right], \quad 1, \ldots, M \quad (3)$$

$$\underline{f_k} = \prod_{i=1}^{n} \underline{\mu}_{\tilde{A}_{ik}} = \underline{f_k}(x) = \underline{\mu}_{\tilde{A}_{1k}}(x_1) * \underline{\mu}_{\tilde{A}_{2k}}(x_2) * \ldots * \underline{\mu}_{\tilde{A}_{nk}}(x_n) \quad (4)$$

$$\overline{f}_k = \prod_{i=1}^{n} \overline{\mu}_{\tilde{A}_{ik}} = \overline{f}_k(x) = \overline{\mu}_{\tilde{A}_{1k}}(x_1) * \overline{\mu}_{\tilde{A}_{2k}}(x_2) * \ldots * \overline{\mu}_{\tilde{A}_{nk}}(x_n) \quad (5)$$

DOI : 10.31695/IJERAT.2020.3665

This model made use of three input metrics each with three linguistic terms which are low, (L) Medium (M) and high (H) with 27 rules as reflected in Table 2

Table 2 Fuzzy rules for Requirement input metrics

Rule No	RC	RS	ERT
1	L	L	L
2	L	L	M
…..	…..	…..	….
27	H	H	H

In layer four (Normalization layer), the firing strengths for each of the rules are normalized to obtain normalized output (N_k). Each normalized output $[\overline{N}_k, \underline{N}_k]$ is computed as the ratio of k[th] firing strength and the sum of the firing strengths of all rules as in (6) and (7).

$$\overline{N}_k = \frac{\sum_{k=1}^{M} \overline{f}_k y_k}{\sum_{k=1}^{M} \overline{f}_k}. \quad (6)$$

$$\underline{N}_k = \frac{\sum_{k=1}^{M} \underline{f}_k y_k}{\sum_{k=1}^{M} \underline{f}_k}. \quad (7)$$

In layer 5 (consequent layer), TSK inference operation is performed on the TSK consequent output which is a linear combination of software metrics input and is represented as:

$$y_k TSK = \sum_{i=1}^{n} w_{ik} x_i + b_k. \quad (8)$$

Layer 6 is the summation layer that gives the final crisp output of the model and is computed as in (9) [39].

$$y = (1-\beta)\frac{\sum_{k=1}^{M} \underline{f}_k y_k}{\sum_{k=1}^{M} \underline{f}_k} + \beta \frac{\sum_{k=1}^{M} \overline{f}_k y_k}{\sum_{k=1}^{M} \overline{f}_k} \quad (9)$$

where$M$ is number of active rules,$\underline{f}_k$and $\overline{f}_k$ are determined using (3) and (4) respectively and y is output signal of TSK IT2FLS which is determined using (8).β is the user defined parameter that weighs the sharing of the lower and the upper firing levels of each fired rule. It should be noted that Begian-Melek-Mendel (BMM) method entails that$\underline{y}_k = \overline{y}_k \equiv y_k$.

Next is the parameter update of the model where the antecedent and consequent parameters of the fuzzy rules are updated with the gradient descent (GD) backpropagationalgorithm. Equation (10) shows the cost function for a single output [36]:

$$E = \frac{1}{2}(y^a - y)^2 \quad (10)$$

Here, y^a is the actual output and y is the output of the proposed approach. This paper makes use of IT2 Gaussian MFs with uncertain standard deviation and fixed mean as seen in (11) and (12).

$$\overline{\mu}_{ik}(x_i) = exp\left[-\frac{(x_i - m_{ik})^2}{2\sigma_{2,ik}^2}\right] \quad (11)$$

$$\underline{\mu}_{ik}(x_i) = exp\left[-\frac{(x_i - m_{ik})^2}{2\sigma_{1,ik}^2}\right] \quad (12)$$

The generic GD backpropagation update rule for any parameter is as shown in (13):

$$\theta_{i+1} = \theta_i - \varphi \frac{\partial E}{\partial \theta_i} \quad (13)$$

where θ denotes generic parameter such as w, b, m, σ, β and$\varphi$is the learning rate (step size) that must be carefully chosen as a large value may lead to instability, and small value on the other hand may lead to a slow learning process.

4. PROJECT DATA SETS FOR CASE STUDIES

The proposed model is implemented using the project datasets in [7]. Twenty three real software data points are used for case studies and reproduced in Table 3 below:

DOI : 10.31695/IJERAT.2020.3665

Table 3 Assessment of Software Project

S/N	Software Project [7]	Size of project (KLOC)	RC	RS	ERT	ACTUAL DEFECT
1	1	6.0	M	L	H	148
2	2	31.0	L	H	H	31
3	3	53.9	H	H	H	209
4	5	14.0	L	M	H	373
5	7	21.0	M	M	L	204
6	8	5.8	L	H	M	53
7	9	2.5	M	H	H	17
8	10	4.8	H	H	H	29
9	11	4.4	H	H	H	71
10	12	19.0	H	L	H	90
11	13	49.1	H	L	H	129
12	14	58.0	H	H	H	672
13	15	154.0	H	L	H	1768
14	16	27.7	L	M	H	109
15	17	33.0	L	M	H	688
16	19	87.0	H	M	H	476
17	20	50.0	H	L	L	928
18	21	22.0	L	M	H	196
19	22	44.0	M	L	L	184
20	23	61.0	H	M	M	680
21	24	99.0	M	L	M	1597
22	27	52.0	H	M	H	412
23	29	11.0	M	H	H	91

5.RELIABILITY PREDICTION BASED ON MODEL ACCURACY

The inputs into the system consist of the three requirement metrics together with the KLOC. For efficient learning, the KLOC is normalized to lie in closed interval of 0 and 1. After prediction, the final results are de-normalized to obtain the predicted values in their original form. The predicted results using IT2FLS-TSK is compared with existing works in the literature as shown in Table 4.

Table 4 Predicted number of faults at the requirement phase

SN	Project No [7]	Actual Values	Predicted values			
			[7]	[37]	[33]	Proposed Model
1	1	148	75	55.94	85	146.6
2	2	31	52	5.48	37	32.8
3	3	209	254	210.61	-	228.6
4	5	373	349	-	-	365.1
5	7	204	262	113.43	139	206.5

DOI : 10.31695/IJERAT.2020.3665

6	8	53	48	53.81	-	53.0
7	9	17	57	52	41	23.4
8	10	29	209	26.17	64	155.1
9	11	71	51	40.61	-	69.2
10	12	90	347	176.25	219	91.4
11	13	129	516	336.59	-	129.7
12	14	672	674	697.48	-	680.2
13	15	1768	1526	1650.89	1946	1768.1
14	16	109	145	127.94	-	100.0
15	17	688	444	135.74	371	573.6
16	19	476	581	573.44	-	470.0
17	20	928	986	869.23	-	927.2
18	21	196	259	105.51	-	195.4
19	22	184	501	291.02	-	191.1
20	23	680	722	690.17	-	682.2
21	24	1597	1514	-	-	1597.8
22	27	412	430	400.17	-	392.2
23	29	91	116	110.06	82	91.4

DOI : 10.31695/IJERAT.2020.3665

Table 5: Absolute errors predicted for nine projects

Project No.	Actual Value	Predicted Values				Absolute Error			
		[7]	[37]	[33]	Proposed model	[7]	[37]	[33]	Proposed model
1	148	75	55.94	85	146.6	73	92.06	63	1.4
2	31	52	5.48	37	32.8	21	26.48	6	1.8
7	204	262	113.43	139	206.5	58	90.57	65	2.5
9	17	57	52	41	23.4	40	35	24	6.4
10	29	209	26.17	64	155.1	180	2.83	35	126
12	90	347	176.25	219	91.4	257	85.25	129	1.4
15	1768	1526	1650.89	1946	1768.1	242	117.2	178	0
17	688	444	135.74	371	573.6	244	552.26	317	114.4
23	91	116	110.06	82	91.4	25	19.06	9	0.4
TOTAL						**1140**	**1018.71**	**828**	**254.3**

Table 5 shows the actual and predicted outputs together with the absolute error for the nine projects undertaken by all other researchers. This is done to ascertain the accuracy level of each model. As shown in Table 5, IT2FLS-SRP provides the least absolute error in all the projects except for project number 10. The predicted value for project 9 in [37] was obtained from [33]. From the table, total absolute error for [7] is **1140**, [37] is **1018.71**, [33] is **828** while the proposed model has the least total absolute error of **254.3** which shows a better and accurate predictions over other models.

6. MODEL EVALUATION BASED ON PERFORMANCE METRICS

The proposed model is evaluated using 5 performance metrics as presented in (14) to (18). The proposed model has predicted software reliability at the requirement phase for twenty-three software projects.

6.1 Performance Metrics

1. Root mean square error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i^a - y_i)^2} \qquad (14)$$

2. Normalized root mean square error (NRMSE)

$$NRMSE = \frac{1}{n}\sum_{i=1}^{n}(y_i^a - y_i)^2 \Big/ \frac{1}{n}\sum_{i=1}^{n} y_i * \frac{1}{n}\sum_{i=1}^{n} y_i^a \qquad (15)$$

3. Mean Magnitude of Relative Error (MMRE)

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i^a - y_i|}{y_i} \qquad (16)$$

4. Balanced Mean Magnitude of Relative Error (BMMRE)

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i^a - y_i|}{Min(y_i^a, y_i)} \qquad (17)$$

Smaller values of RMSE, NRMSE, MMRE and BMMRE entail better prediction accuracy.

DOI : 10.31695/IJERAT.2020.3665

5. Coefficient of Determination (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i^a - y_i)^2}{\sum_{i=1}^{n}(y_i^a - \overline{y})^2} \qquad (18)$$

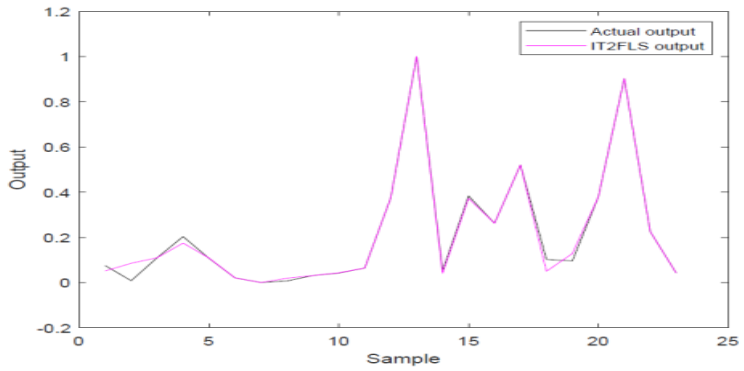
where $\overline{y}$ is the mean of the actual values, y_i^a.

Better prediction accuracy is obtained if the value of R^2 is closer to 1 [7]. In (13) to (18), y_i^a is the actual defect, y_i is the predicted defect and n is the number of testing data points. The result is as depicted in Table 6.

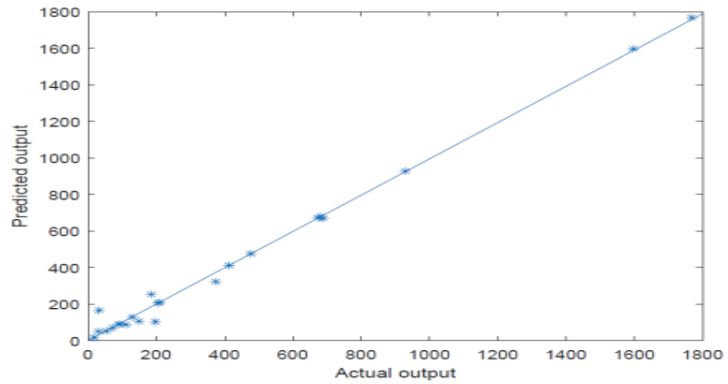
Table 6 Performance comparison of the proposed model

Models	*RMSE*	*NRMSE*	*MMRE*	*BMMRE*	R^2
Chatterjee et al., [33]	132.8230	0.0758	0.6280	0.7247	0.9398
Pandey and Goyal [37]	195.7192	0.1118	0.6744	1.5648	0.8692
Fenton et al., [7]	158.3891	0.0904	1.4922	1.5696	0.9144
Proposed	**102.65**	**0.0543**	**0.1204**	**0.2484**	**0.9402**

From Table 6, the performance metrics for the proposed model indicate better prediction than those in the literature. R^2 value for the proposed model is **0.9402** (94%) which is closer to 1 showing more accurate prediction than others in the literature. In designing the model, 17 data sets are used for training and 6 data sets are used for testing. All input data were normalized in interval [0, 1]. During learning the initial values of consequent parameter w and b were randomly selected in the interval $[0, 1]$. The IT2FLS model is executed for 500 epochs having learning rate set at 0.1. Figure 8 depicts the learning of IT2FLS and the relationship between the actual and predicted values for SRP at the requirement phase.



(a)



(b)

Figure 8: (a). The learning of IT2FLS and (b). Relationship between the actual and the predicted outputs

DOI : 10.31695/IJERAT.2020.3665

7. CONCLUSION

In this study, an IT2FLS based model is presented for predicting the reliability of software at the requirement phase of the software development life cycle. The proposed approach employed reliability relevant software metrics of the requirement phase of SDLC and software size metrics as inputs to the system. The uncertainties associated with the reliability software requirement metrics, namely RC, RS and ERT and software size metrics are properly captured and modeled using IT2 TSK-IT2FLS. The gradient descent back-propagation method is used to optimize the parameters of the IT2FLS for effective performance. The performance evaluation of the model is carried out based on five performance measures and the results compared with existing models of [33], [37],[7] proves that the proposed model is better and more accurate than the existing models. The predicted outputs for 23 software projects are very close to the actual outputs.

This model may serve as a guide to software managers, engineers, practitioners and researchers for decision making about software reliability during early stage of SDLC. It may also assist them in making informed and useful decisions in the face of uncertainty. This work can be applied in software measurement quality domains and software metrics modeling. The application domain covers education, defense, transportation, medical, industries and other government agencies making use of software.

In the future, we intend to learn the parameters of the proposed model using other optimization tools such as particle swarm optimization (PSO), extended Kalman filter (EKF), simulated annealing, and bee colony algorithm. We also intend to model SRP at the latter stage of software development life cycle.

REFERENCES

1. Yadav, H. B. and Yadav,D. K. (2014). Early Software Reliability Analysis using Reliability Relevant Software Metrics. International Journal of System Assurance Engineering and Management, 8(S4) pp 2097-2108
2. Ravinder, K; Kiran, K. and Arvind K. Measuring Software Reliability : A Fuzzy Model, ACM SIGSOFT Software Engineering Notes, 36(6), (2011), pp. 1-6
3. IEEE standard glossary of software engineering terminology STD – 729-1991.
4. Sangeetha.M , Arumugam.C, Sapna P.G and Senthil Kumar .K.M A cooperative approach to ensure software product quality" IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1. (2011)
5. Palviainen, M., Evesti, A. and Ovaska, E., (2011): The reliability estimation prediction and measuring of component-based software. Journal of Systems and Software 84 (6), 1054–1070.
6. Agrawal M., Chari K. Software Effort, Quality and Cycle Time: A Study of CMM Level 5 Projects, IEEE Transaction on Software Engineering, vol. 33, no. 2, (2007), pp 145-156
7. Fenton, N., Neil, M., Marsh, W., Hearty, P., Radliński, L., and Krause, P. (2008). On the effectiveness of early life cycle defect prediction with Bayesian Nets. *Empirical Software Engineering*, *13*(5) pp. 499.
8. Pandey, A. K. and Goyal, N. K. (2009). A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics Reliability Engineering Centre, IIT Kharagpur, INDIA, International Journal of Electronics Engineering, 1(2), pp. 239-245S.
9. Yadav, D. K., Charurvedi, S. K. and Mishra, R. B. (2012). Early software defects prediction using fuzzy logic, Int. J. Performability Eng. 8 (4) pp399–408S
10. Rizvi, S. W. A., Khan, R. A. and Singh, V. K. (2016).Software Reliability Prediction using Fuzzy Inference System: Early Stage Perspective, *International Journal of Computer Applications (0975 – 8887) Volume 145(10),pp 16-23*
11. Rizvi, S. W. A.; Khan, R. A. and Singh, V. K. (2017). Early Stage Software Reliability Modeling using Requirements and Object-Oriented Design Metrics: Fuzzy Logic Perspective, *International Journal of Computer Applications (0975 – 8887) Volume 162, No 2*, pp. 44-59
12. Rizvi, S. W. A. and Khan, R. A. (2009). A Critical Review on Software Maintainability Models. Proceedings of the Conference on Cutting Edge Computer and Electronics Technologies, pp144-148.
13. Rizvi, S. W. A. and Khan, R. A. (2010). Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD). Journal of Computing, 2(4), (2010), pp 26-32.
14. Zhang, X. and Pham, H. (2000). An analysis of factors affecting software reliability. The Journal of Systems and Software 50(1) pp.43–56
15. Li, M., Smidts, C. and Brill, R. W. (2000). Ranking software engineering measures related to reliability using expert opinion. In: Proceedings of the 11th international symposium on software reliability engineering (ISSRE pp 246–258

DOI : 10.31695/IJERAT.2020.3665

16. Li, M. and Smidts, C. (2003). A ranking of software engineering measures based on expert opinion. IEEE Transactions on Software Engineering, 29(9) (2003) pp811–824
17. Kumar, T. R., Rao, T. S. and Ch. V. M. K. Hari, A predictive approach to estimate software defects density using Probabilistic Neural Networks for the given Software Metrics, T. Ravi Kumar Journal of Engineering Research and Application ISSN : 2248-9622, Vol. 8, Issue 7 (Part -II) July 2018, (2018) pp 8-15
18. Cai, K. Y., Wen, C. Y. and Zhang, M. L. (1991). A critical review on software reliability modeling", Reliability Engineering and System Safety 32 (3) pp. 357–371.
19. Khatatneh, K. and Mustafa, T. (2009). Software Reliability Modeling Using Soft Computing Technique, in European Journal of Scientific Research ISSN 1450-216X Vol.26 No.1, pp.147-152
20. Aljahdali, S. (2011). Development of Software Reliability Growth Models for Industrial Applications Using Fuzzy Logic. Journal of Computer Science, 7(10), pp. 1574-1580.
21. Kumar, R., Khatter, K. and Kalia, A. (2011). Measuring Software Reliability- A Fuzzy Model, ACM SIGSOFT Software Engineering Notes, 36 (6) pp 1-6
22. Jaikumar, M. and Ramani, A. V. (2017). Software defect prediction using fuzzy logic system. International Journal of Innovations & Advancement in Computer Science, IJIACS, Volume 6, Issue 3, pp. 118-124
23. Zadeh, L. A. (1965). Fuzzy sets," Information and control, vol. 8, no. 3, pp. 338-353.
24. Pandey, A. K. and Goyal, N. K. (2010). Fault prediction model by fuzzy profile development of reliability relevant software metrics. International Journal of Computer Applications, 11(6), 34–41.
25. Yadav, H. B. and Yadav, D. K. (2015). A Fuzzy Logic based Approach for Phase-wise Software Defects Prediction using Software Metrics, *Information and Software Technology*, 63, pp. 44–57.
26. Hagras, H. A., (2004): A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," IEEE Transactions on Fuzzy Systems, vol. 12, no. 4, pp. 524-539.
27. Hagras, H. (2007) "Type-2 FLCs: A new generation of fuzzy controllers," *IEEE Comput. Intell. Mag.*, vol. 2, no. 1, pp. 30–43
28. Mendel, J. M. (2001, July). On the importance of interval sets in type-2 fuzzy logic systems. In *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)* (Vol. 3, pp. 1647-1652). IEEE.
29. Mendel, J. M. and John, R. B. (2002): Type-2 fuzzy sets made simple," Fuzzy Systems, IEEE Transactions on, vol. 10, no. 2, pp. 117-127
30. Wu, D., and Mendel, J. M. (2014). Designing practical interval type-2 fuzzy logic systems made simple. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 800-807.
31. . Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning-i," Information Sciences, vol. 8, , pp. 199-249.
32. Mendel, J. M., John, R. I. and Liu, F. (2006). Interval type-2 fuzzy logic systems made simple," IEEE Transactions on Fuzzy Systems, vol. 14, no. 6, pp. 808-821,.
33. Chatterjee, S., Maji, B., & Pham, H. (2019). A fuzzy rule-based generation algorithm in interval type-2 fuzzy logic system for fault prediction in the early phase of software development. *Journal of Experimental & Theoretical Artificial Intelligence*, *31*(3), 369-391.
34. Kayacan, E. and Khanesar, M. (2016). Fuzzy Neural Networks for Real Time Control Applications: Concepts, Modeling and Algorithms for Fast Learning, 720
35. Hamam, A., &Georganas, N. D. (2008, October). A comparison of Mamdani and Sugeno fuzzy inference systems for evaluating the quality of experience of Hapto-Audio-Visual applications. In *2008 IEEE International Workshop on Haptic Audio visual Environments and Games* (pp. 87-92). IEEE.
36. Eyoh, I. J. (2018). Interval Type-2 Atanassov-Intuitionistic Fuzzy Logic for uncertainty Modelling, Doctoral Dissertation, University of Nottingham p.22
37. Pandey, A. K. and Goyal, N. K. Multistage model for residual fault prediction, in Early Software Reliability Prediction, Springer, India pp. 59–80.
38. Lin, Y.-Y., Liao, S.-H., Chang, J.-Y. and Lin, C.-T. (2014). Simplified interval type-2 fuzzy neural networks," IEEE Transactions on Neural Networks and Learning Systems,25(5) pp. 959-969.
39. Begian, M. B., Melek, W. W., & Mendel, J. M. (2008, May). Parametric design of stable type-2 TSK fuzzy systems. In *NAFIPS 2008-2008 Annual Meeting of the North American Fuzzy Information Processing Society* (pp. 1-6). IEEE.

DOI : 10.31695/IJERAT.2020.3665