Predict Software Reliability by Support Vector Machine

Md. Baharul Islam
Senior Lecturer
Department of Multimedia Technology and Creative Arts,
Daffodil International University, Dhaka, Bangladesh,

Md. Kamrul Hasan
Project Manager
Atomix System Limited
Dhaka, Bangladesh

A. H. M. Saiful Islam
Senior Lecturer
Department of Computer Science and Engineering,
Daffodil International University, Dhaka, Bangladesh,

Ziaur Rahman
Lecturer
Department of ICT, Mawlana Bhashani Science &
Technology University, Bangladesh

Abstract- The application of computer system has now crossed many different fields. Software has become an essential part of many industrial, military and even commercial systems. We may find programs containing millions of lines of code in both microcomputers and supercomputers. Software reliability is an important research area because of safety critical system. In this paper, a model that can be used for software reliability prediction is explored. The proposed model is implemented using Support Vector Machine (SVM) and has been applied on a custom set of test data. It focused on a particular dataset behavior in predicting reliability. Focusing on a particular dataset behavior is performed to develop an accurate model since the recent work focused on developing a model which can be more accurate. We also build a SVM model to find the relationship between reliability and error rate.

***Keywords*- Bugs, Error, Reliability, Software, SVM**

I. INTRODUCTION

The IEEE Standard Glossary of Software Engineering Terminology defines software reliability is: "The ability of the software to perform its required function under stated conditions for a stated period of time". The ability to predict the reliability of a software system would enable project management for better performing product assurance and readiness. Three bases are used for estimating reliability that are failure record, behavior for a random sample of input points, or quantity of actual and "seeded" faults detected during testing.

As the complexity and constraints under the software is increasing, it is difficult to produce software without faults. One of the ways to deal with this problem is to predict software quality attributes such as fault proneness, effort, testability, maintainability and reliability during phases of software development.

The software metrics [1-4] may be used in predicting these quality attributes. The behavior of several quantitative models ranging from simple linear discriminate analysis to more complex logistic regression, decision tree, and SVM have been proposed. The regression and machine learning approaches are inherently different because of raising the question to analyze the performance of these methods. Several empirical studies [5-12] have been carried out to predict the fault proneness models.

It is need to empirically validate the machine learning methods in predicting software quality attributes. Therefore data-based empirical studies can be used to verify the capability of machine learning methods. The evidence gathered through these empirical studies is considered to be the strongest support for testing a given hypothesis. We have to find empirically validating the results of machine leaning methods such as SVM and the relation between reliability metrics and error rate.

In order to analysis the performance of the SVM method, we use data set of RUP model for software development life cycle. We consider six phases and collect the defects of each phase. The considerable six phases with type of faults are given in Table 1.

The contributions of this paper are summarized as follows: We performed the system description of SVM with our result and based on our result we proposed a software reliability model. The results showed that SVM method predict reliability with high accuracy.

However our analysis is based on only one data set. This study should be replicated on different data sets to generalize our findings. This paper is organized followed by literature review, experimental methodology, result with discussion and conclusion.

Table 1: Count the defects or error of each phases

S/N	Phase	Fault type or root cause
1	Requirements	• Missing requirements. • Misinterpreted requirements. • Requirements not clear. • Changed requirements. • Conflicting requirements.
2	Design	• Design not to requirements. • Missing design. • Top level design logic. • Low level design logic. • Design not robust.
3	Code	• Code not implemented to design. • Code not implemented to requirements. • Missing code. • Initialization error. • Storing error. • Mismatched parameters. • Math operations not robust. • I/O operations not robust. • Memory errors. • Domain errors.
4	Unit Testing	• New fault generated in unit Testing
5	Functional Testing	• New fault generated in Functional Testing
6	Integration testing	• New fault generated in Integration Testing

II. LITERATURE REVIEW

To predict software reliability, there are several works that have been done such as ANN, GA, Asian network, SVM. SVM is to find the relationship between object-oriented metrics and fault proneness [8]. The proposed model is empirically evaluated using public domain KC1 NASA data set. The performance of the SVM method was evaluated by Receiver Operating Characteristic (ROC) analysis. Based on results, it is reasonable to claim that such models could help for planning and performing testing by focusing resources on fault-prone parts of the design and code.

SVM method may also be used in constructing software quality models. Reference [13] proposes a novel GEP-based non-parametric software reliability modelling approach. There are a lot of models for predicting software reliability discovered by researchers such as Particle Swarm Optimization-Support Vector Machine (PSO-SVM) [14], altering paradigms such as Component-Based Software Engineering (CBSE), Autonomic Computing, Service-Oriented Computing (SOC), Fault-Tolerant Computing (FTC) [15], weighted criteria value [16], Genetic Algorithms [17], Support vector Machine (SVM) [18], Machine Learning Methods [19], Support Vector Regression [20], Neural Networks [21], Soft computing techniques [22], Sequential Bayesian Technique [23].

III. EXPERIMENTAL METHODOLOGY

Consider N training samples $x1$, $y1$, $x2$, $y2$, … $\{xN, yN\}$, where $xi \in Rm$ is a m-dimensional feature vector representing the t[th] training sample, and $yi \in \{-1, 1\}$ is the class label of xi . A hyper-plane in the feature space can be described as the equation $wT\ x + b = 0$, where $w \in Rm$ and b is a scalar. When the training samples are linearly separable, SVM yields the optimal hyper-plane that separates two classes with no training error, and maximizes the minimum distance from the training samples to the hyper-plane. It is easy to find that the parameter pair (w, b) corresponding to the optimal hyper-plane is the solution to the following optimization problem:

$$\text{Minimize:} \quad L(w)= \frac{1}{2}\|w\|^2 \tag{1}$$

$$\text{Subject to:} \quad y_i\,(w^T x_i + b) \geq 1, \quad i=1,2,\ldots,N \tag{2}$$

For linearly non-separable cases, there is no such a hyper-plane that is able to classify every training sample correctly. However the optimization idea can be generalized by introducing the concept of soft margin. The new optimization problem thus becomes:

$$\text{Minimize:} \quad L(w, \zeta_i)= \frac{1}{2}\|w\|^2 + C\sum \zeta_i \tag{3}$$

$$\text{Subject to:} \quad y_i\,(w^T x_i + b) \geq 1- \zeta_i \quad i=1,2,\ldots,N \tag{4}$$

Where ξ_i are called slack variables that are related to the soft margin and C is the tuning parameter used to balance the margin and the training error. Both optimization problems (3) and (4) can be solved by introducing the Lagrange multipliers α_i that transform them to quadratic programming problems. For the applications where linear SVM does not produce satisfactory performance, nonlinear SVM is suggested [10]. The basic idea is to map x by nonlinearly mapping $\phi(x)$ to a much higher dimensional space in which the optimal hyper-plane is found. The nonlinear mapping can be implicitly defined by introducing the so-called kernel function K (xi, xj) which computes the inner product of vectors $\phi(x_i)$ and $\phi(x_j)$. The typical kernel functions include the radial basis function and the polynomial function.

$$\text{Radial Basis:} \quad K(x_i, x_j) = \exp\left(\frac{\| xi - xj \|^2}{\sigma 2}\right) \tag{5}$$

$$\text{Polynomial:} \quad K(x_i, x_j) = (x_i^T x_j + 1)^d \tag{6}$$

We considered seven modules in RUP software life cycle; therefore we have a data set of 21 * 6 matrices. First we collected the raw dataset. That has six phases and each phase have three cycles.

- Sample dataset
  - Take 7 modules
  - RUP Process
  - Dataset now stand (21*6)
    - C1 ---------- 42 (7*6)
    - C2 ---------- 42 (7*6)
    - C3 ---------- 42 (7*6)

Table 2: Raw dataset

	1			2			3			4			5			6			7		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**
R	8	5	3	7	6	4	6	4	5	11	8	4	9	6	4	8	7	3	5	5	4
D	8	5	3	7	6	4	6	4	5	11	8	4	9	6	4	8	7	3	5	5	4
C	33	19	13	37	26	16	29	25	17	31	27	17	27	22	11	34	23	16	30	19	13
U	117	45	25	90	55	30	85	55	35	99	56	34	90	46	23	88	48	26	76	38	24
F	12	8	6	15	8	6	13	9	6	9	9	5	12	9	6	11	8	4	9	7	5
I	20	10	4	28	12	8	16	10	8	18	13	9	14	8	6	10	6	6	17	11	9

Table 2 represents a data matrix of defects. It has six phases that we took as features. Six features are R, U, C, U, F and I. We collected raw data where feature values are scattered. So we have to normalize those values in a range. We normalize those values in 0-1 range that is shown in Table 3.

Table 3: Normalized data

	1			2			3			4			5			6			7		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**
R	.7	.4	.2	.6	.5	.3	.5	.3	.4	1	.7	.3	.8	.5	.3	.7	.6	.2	.4	.4	.3
D	.7	.4	.2	.6	.5	.3	.5	.3	.4	1	.7	.3	.8	.5	.3	.7	.6	.2	.4	.4	.3
C	.8	.5	.3	1	.7	.4	.7	.6	.4	.8	.7	.4	.7	.5	.2	.9	.6	.4	.8	.5	.3
U	1	.3	.2	.7	.4	.2	.7	.4	.2	.8	.4	.2	.7	.3	.1	.7	.4	.2	.6	.3	.2
F	.8	.5	.4	2	.5	.4	.8	.6	.4	.6	.6	.3	.8	.6	.4	.7	.5	.2	.6	.4	.3
I	.7	.3	.1	1	.4	.2	.5	.3	.2	.6	.4	.3	.5	.2	.2	.3	.2	.2	.6	.3	.3

Now we preprocess the data for SVM training. In this case each column represents as feature and each row represents a software life cycle of RUP of a module. Therefore we got a data set which has six columns and twenty rows that are shown in Table 4. But this data is not ready for SVM because of this data is not classified yet. For the classification purpose, we took a threshold t=0.46. We considered that below of threshold of all features is -1 class and rest of portion are +1 class. Table 5 represents the classification of data.

Table 4: Dataset without class labeled

R	**D**	**C**	**U**	**F**	**I**
0.7273	0.7273	0.8919	1.0000	0.8000	0.7143
0.4545	0.4545	0.5135	0.3846	0.5333	0.3571
0.2727	0.2727	0.3514	0.2137	0.4000	0.1429
0.6364	0.6364	1.0000	0.7692	1.0000	1.0000
0.5455	0.5455	0.7027	0.4701	0.5333	0.4286
0.3636	0.3636	0.4324	0.2564	0.4000	0.2857
0.5455	0.5455	0.7838	0.7265	0.8667	0.5714
0.3636	0.3636	0.6757	0.4701	0.6000	0.3571
0.4545	0.4545	0.4595	0.2991	0.4000	0.2857
1.0000	1.0000	0.8378	0.8462	0.6000	0.6429
0.7273	0.7273	0.7297	0.4786	0.6000	0.4643
0.3636	0.3636	0.4595	0.2906	0.3333	0.3214
0.8182	0.8182	0.7297	0.7692	0.8000	0.5000
0.5455	0.5455	0.5946	0.3932	0.6000	0.2857
0.3636	0.3636	0.2973	0.1966	0.4000	0.2143
0.7273	0.7273	0.9189	0.7521	0.7333	0.3571
0.6364	0.6364	0.6216	0.4103	0.5333	0.2143
0.2727	0.2727	0.4324	0.2222	0.2667	0.2143
0.4545	0.4545	0.8108	0.6496	0.6000	0.6071
0.4545	0.4545	0.5135	0.3248	0.4667	0.3929
0.3636	0.3636	0.3514	0.2051	0.3333	0.3214

Table 5: Dataset with class labeled

R	D	C	U	F	I	Y
0.7273	0.7273	0.8919	1.0000	0.8000	0.7143	1
0.4545	0.4545	0.5135	0.3846	0.5333	0.3571	1
0.2727	0.2727	0.3514	0.2137	0.4000	0.1429	-1
0.6364	0.6364	1.0000	0.7692	1.0000	1.0000	1
0.5455	0.5455	0.7027	0.4701	0.5333	0.4286	1
0.3636	0.3636	0.4324	0.2564	0.4000	0.2857	-1
0.5455	0.5455	0.7838	0.7265	0.8667	0.5714	1
0.3636	0.3636	0.6757	0.4701	0.6000	0.3571	1
0.4545	0.4545	0.4595	0.2991	0.4000	0.2857	-1
1.0000	1.0000	0.8378	0.8462	0.6000	0.6429	1
0.7273	0.7273	0.7297	0.4786	0.6000	0.4643	1
0.3636	0.3636	0.4595	0.2906	0.3333	0.3214	-1
0.8182	0.8182	0.7297	0.7692	0.8000	0.5000	1
0.5455	0.5455	0.5946	0.3932	0.6000	0.2857	1
0.3636	0.3636	0.2973	0.1966	0.4000	0.2143	-1
0.7273	0.7273	0.9189	0.7521	0.7333	0.3571	1
0.6364	0.6364	0.6216	0.4103	0.5333	0.2143	1
0.2727	0.2727	0.4324	0.2222	0.2667	0.2143	-1
0.4545	0.4545	0.8108	0.6496	0.6000	0.6071	1
0.4545	0.4545	0.5135	0.3248	0.4667	0.3929	1
0.3636	0.3636	0.3514	0.2051	0.3333	0.3214	-1

Table 6: Training data with lebel

R	D	C	U	F	I	Y
0.7273	0.7273	0.8919	1.0000	0.8000	0.7143	1
0.4545	0.4545	0.5135	0.3846	0.5333	0.3571	1
0.5455	0.5455	0.7027	0.4701	0.5333	0.4286	1
0.4545	0.4545	0.4595	0.2991	0.4000	0.2857	-1
0.7273	0.7273	0.7297	0.4786	0.6000	0.4643	1
0.3636	0.3636	0.4595	0.2906	0.3333	0.3214	-1
0.5455	0.5455	0.5946	0.3932	0.6000	0.2857	1
0.3636	0.3636	0.2973	0.1966	0.4000	0.2143	-1
0.2727	0.2727	0.4324	0.2222	0.2667	0.2143	-1
0.4545	0.4545	0.8108	0.6496	0.6000	0.6071	1
0.4545	0.4545	0.5135	0.3248	0.4667	0.3929	1

Table 7: Testing data without label

R	D	C	U	F	I
0.2727	0.2727	0.3514	0.2137	0.4000	0.1429
0.6364	0.6364	1.0000	0.7692	1.0000	1.0000
0.3636	0.3636	0.4324	0.2564	0.4000	0.2857
0.5455	0.5455	0.7838	0.7265	0.8667	0.5714
0.3636	0.3636	0.6757	0.4701	0.6000	0.3571
1.0000	1.0000	0.8378	0.8462	0.6000	0.6429
0.8182	0.8182	0.7297	0.7692	0.8000	0.5000
0.7273	0.7273	0.9189	0.7521	0.7333	0.3571
0.6364	0.6364	0.6216	0.4103	0.5333	0.2143
0.3636	0.3636	0.3514	0.2051	0.3333	0.3214

We are randomly selecting training and testing data. Table 6 and 7 represents the training data with label and testing data without label respectively.

IV. EXPERIMENTAL RESULT AND DISCUSSION

We train the training data with SVM and check the performance. We plot this in a feature space with kernel RBF that is shown in figure 1. We used test data for classifying and measure the performance by calculating the sensitivity, specificity, completeness and error rate in SVM. Table 8 represents the feature matrix of test data.

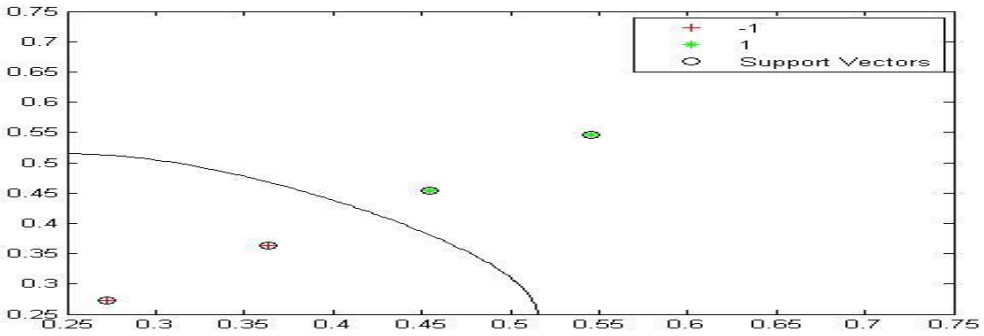

Figure 1: support vectors with RBF kernel

Table 8: Feature matrix of test data

Sensitivity	Specificity	Completeness	Error Rate
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
0.6667	1.0000	0.9000	0.1
1.0000	0.7143	0.8000	0.2
1.0000	1.0000	1.0000	0
1.0000	0.8571	0.9000	0.1
1.0000	0.8571	0.9000	0.1
0.6667	1.0000	0.9000	0.1
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
0.6667	1.0000	0.9000	0.1
0.6667	1.0000	0.9000	0.1
0.6667	1.0000	0.9000	0.1
0.6667	1.0000	0.9000	0.1
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
1.0000	1.0000	1.0000	0
Sum of Square Mean Error Rate (SSMER)			0.0476

We calculated the Sum of Square Mean Error Rate (SSMER) by the following equation and we also plot the sensitivity and completeness that is shown in figure 2. From the figure 2 we noticed that completeness never below the sensitivity.

$$\text{ssmer}=(x)^n = \left(\sum_{k=0}^{n}\binom{n}{k}x^k\right)^2/n$$

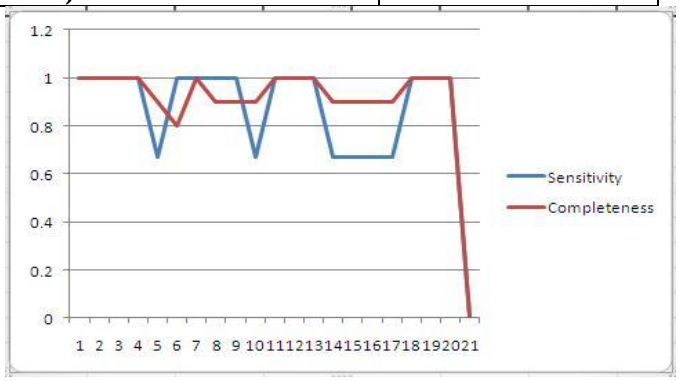

Figure 2: Sensitivity vs. Completeness

From our experiment, we got a co-relation between reliability and SSMER. The relation between them is given below:

Reliability: $R \propto SSMER$ (7)

Reliability is proportional to the SSMER that is found from SVM Trainings. We established this equation from

Reliability, $R= (\log (d) +t)*SSMER$ (8)

Re-written, $R= Abs (\log(d)+t)*SSMER$ (9)

Where

Abs =Absolute value
Log = Logarithm
d = Number of defects or error
t = Threshold
SSMER= Sum of Square Mean Error Rate

I took threshold t=0.46 and from Support Vector Machine SSMER=0.0476. The result is given in table 9 that calculated from equation (9).

Table 8: Co-relation matrix

R	D	C	U	F	I
0.0371	0.0371	0.0273	0.0219	0.0325	0.0379
0.0594	0.0594	0.0536	0.0674	0.0518	0.0709
0.0837	0.0837	0.0717	0.0954	0.0655	0.1145
0.0434	0.0434	0.0219	0.0344	0.0219	0.0219
0.0507	0.0507	0.0387	0.0578	0.0518	0.0622
0.0700	0.0700	0.0618	0.0867	0.0655	0.0815
0.0507	0.0507	0.0335	0.0371	0.0287	0.0485
0.0700	0.0700	0.0406	0.0578	0.0462	0.0709
0.0594	0.0594	0.0589	0.0793	0.0655	0.0815
0.0219	0.0219	0.0303	0.0298	0.0462	0.0429
0.0371	0.0371	0.0369	0.0570	0.0462	0.0584
0.0700	0.0700	0.0589	0.0807	0.0742	0.0759
0.0314	0.0314	0.0369	0.0344	0.0325	0.0549
0.0507	0.0507	0.0466	0.0663	0.0462	0.0815
0.0700	0.0700	0.0796	0.0993	0.0655	0.0952
0.0371	0.0371	0.0259	0.0355	0.0367	0.0709
0.0434	0.0434	0.0445	0.0643	0.0518	0.0952
0.0837	0.0837	0.0618	0.0935	0.0848	0.0952
0.0594	0.0594	0.0319	0.0424	0.0462	0.0456
0.0594	0.0594	0.0536	0.0754	0.0582	0.0664
0.0700	0.0700	0.0717	0.0973	0.0742	0.0759

From above this Reliability matrix I can tell that for higher defects have lower reliability and for lower defects have higher reliability.

V. CONCLUSION

A new SVM system was developed to predict the software reliability. This model focused on a particular dataset behavior for predicting reliability. Focusing on a particular dataset is performed to develop an accurate model. The experimental results have shown that the developed model can predict accurate results in most points of the target dataset. We hope to use the neural network technique to allow the training for some parameters used in the proposed model to predict more accurate results as our future work. And we also try to find out any other co-relation between and feature matrix.

REFERENCES

[1] K. K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, Software reuses metrics for object-oriented systems, Proceedings of the 3rd ACIS International Conference on Software Engineering Research, Management and Applications (SERA), 2005, 48-55.

[2] L. Briand, W. Daly, and J. Wust, Unified framework for cohesion measurement in object-oriented systems. Empirical Software Engineering, 3(1), 1998, 65-117.

[3] L. Briand, W. Daly, and J. Wust, A unified framework for coupling measurement in object-oriented systems. IEEE Transactions on Software Engineering, 25(1), 1999, 91-121.

[4] M. Cartwright, and M. Shepperd, An empirical investigation of an object-oriented software system. IEEE Transactions of Software Engineering, 26(8), 1999, 786-796.

[5] K. G. Popstojanova, M. Hamill, and R. Perugupalli, Large Empirical Case Study of Architecture–based Software Reliability, Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering, 2005.

[6] A. Singhal, A. Singhal, A Systematic Review of Software Reliability Studies, Software Engineering: an International Journal, 1(1), 2011, 96-114.

[7] A. Quyoum, M. D. Dar, and S. M. K. Quadri, Improving Software Reliability using Software Engineering Approach- A Review, International Journal of Computer Applications, 10(5), 2010, 41-47.

[8] Y. Singh, A. Kaur and R. Malhotra, Software Fault Proneness Prediction Using Support Vector Machines, Proceedings of the World Congress on Engineering (WCE 2009), July 1 - 3, 2009, London, U.K.

[9] K. K. Mohan, A. K. Verma, and A. Srividya, Early Software Reliability Prediction Using ANN for Process Oriented Development at Prototype Level, IEEE 20th International Symposium on Software Reliability Engineering (ISSRE), 16-19 November, 2009, India.

[10] K. Khatatneh, and T. Mustafa, Software Reliability Modeling Using Soft Computing Technique, European Journal of Scientific Research, 26(1), 2009, 154-160.

[11] L. Prasad, A. Gupta, and S. Badoria, Measurement of Software Reliability Using Sequential Bayesian Technique, Proc. of the World Congress on Engineering and Computer Science, 20-22 October, 2009, San Francisco, USA

[12] V. N. Vapnik, An Overview of Statistical Learning Theory, IEEE Trans. on Neural Networks, 10(5), 1999, 988-999.

[13] H. F. Li, M. Y. Lu, M. Zeng and B. Q. Huang, A Non-Parametric Software Reliability Modeling Approach by Using Gene Expression Programming, Journal of Information Science and Engineering 28, 2012, 1145-1160.

[14] Z. Xiaonan Y. Junfeng, D. Siliang and H. Shudong, A New Method on Software Reliability Prediction, Mathematical Problems in

Engineering, Hindawi Publishing Corporation, 2013 (Article ID 385372), 1-8.

[15] R. Wason, P.Ahmed and M. Q. Rafiq, Novel Software Reliability Estimation Model for Altering Paradigms of Software Engineering, IJCSI International Journal of Computer Science Issues, 9(3), 2012, 92-99.

[16] M. Anjum, M. A. Haque and N. Ahmad, Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value, International Journal of Information Technology and Computer Science, 2(1), 2013, 1-14.

[17] S. H. Aljahdali and M. E. E. Telbany, Genetic Algorithms for Optimizing Ensemble of Models in Software Reliability Prediction, ICGST-AIML Journal, 8(1), 2008, 5-13.

[18] F. Xing, P. Guo, and M. R. Lyu, A Novel Method for Early Software Quality Prediction Based on Support Vector Machine, Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering, 2005.

[19] R. Malhotra and A. Jain, Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality, Journal of Information Processing Systems, 8(2), 2012, 241-262.

[20] Z. Qiuhong, Research of Software Failure Prediction Based on Support Vector Regression, The 2nd International Conference on Computer Application and System Modeling 2012, Paris, France.

[21] V. Ramakrishna, N. Rao, T. M. Padmaja, Software Reliability Prediction using Neural Networks, International Journal of Computer Applications, 60(7) , 2012, 44-48.

[22] N. R. Kiran, and V. Ravi, Software reliability prediction by soft computing techniques, The Journal of Systems and Software, 2007, www.elsevier.com/locate/jss

[23] L. Prasad, A. Gupta, and S. Badoria, Measurement of Software Reliability Using Sequential Bayesian Technique, Proceedings of the World Congress on Engineering and Computer Science 2009 (WCECS 2009), October 20-22, San Francisco, USA.