I J C T A, 9(41) 2016, pp. 863-872 ISSN: 0974-5572

DEFECT ESTIMATION AND SEVERITY ANALYSIS OF SOFTWARE ARTIFACTS

Chandan Kumar*,and Dilip Kumar Yadav**

Abstract:Defect estimation and severity analysis of software artifacts are very helpful for developing reliable software product. In order to achieve the reliable software within time and costs every software organization want to know how many software defect can be exit in developing software and which phase of software development life cycle is more severe. In this paper phase wise software defect severity analysis method is proposed using Bayesian networks. In the proposed method, severity of software defect in each phase of SDLC is predicted using top ranked reliability relevant software metrics. Bayesian belief network (BBN) and the linguistic values of software metrics related to requirement analysis, design, coding and testing phase have been considered to develop the proposed model. To validate the proposed model, 20 real software project data sets have been used. The predictive accuracy of the proposed model is validated and compared with existing work.

Key Words: *Defect Severity, Defect Estimation, Software Metrics, Bayesian Belief Network, Software Reliability*

1. INTRODUCTION

The causes of software failure are software defects, which are generated in the process of software development. The defects in software are generated either within a phase of software development process or at the interface between two different software development phases. However, software defects vary considerably with respect to their severity in different phases of software development. The IEEE standard states, ‘‘Identifying the severity of the defects is a mandatory category as is identifying the project schedule, and project cost impacts of any possible solution for the defects’’[1]. Software defect severity prediction plays an important role in producing reliable software product. In order to estimate the total defect of the software, it is also required to predict the severity of defect at the end of each phase of the SDLC. The importance of defect severity is identified as early as 1979 by Crosby [2]. The importance of measuring severity levels rather than simply identifying the number of defects is reemphasized by Jones [3]. Therefore, there is a need to develop phase wise software defect severity analysis prediction model that can be used to identify that which phase of the SDLC is more defective. The software practitioners can use that model with respect to high severity of defects and focus on those phases of the SDLC. The top agenda item for software developing companies is quality and reliability of the software. The reliability and quality of a software product is totally dependent on the process through which the software product is developed i.e. SDLC. The software quality does not depend only on technical

* Department of Computer Applications
National Institute of Technology Jamshedpur, Jamshedpur, Jharkhand, India
Email- chandan.jha150286@gmail.com

** Department of Computer Applications
National Institute of Technology Jamshedpur, Jamshedpur, Jharkhand, India
dkyadav1@gmail.com

competence but people, tools, techniques and management are also responsible. Therefore, the software development management system should address all the situations in which the software is being developed i.e. from requirement specification, to design, coding and testing.

In literature [4-6], it is found that BBN play a vital role to predict the quality and reliability of software product using defect estimation and detection approach. Amasaki et al. [4] proposed a BBN based software quality prediction model, they have tried to apply BBN to find the risky project using the software metrics that are highly correlated to SDLC phase. Fenton et al. [5] found that only size and complexity metrics are incapable for estimating the residual software defects accurately. In another study of Fenton et al. [6] suggested to use BBN in software defect and software reliability prediction. Their approach allows software analyst to incorporate the causal relationships of software metrics as well as combining the qualitative and quantitative measures of software metrics to solve the limitation of traditional software metrics methods. Kumar and Mishra [7] proposed a model for software reliability prediction, that can identified the phases of SDLC in which phase corrective actions are needed to be performed to achieve the required target reliability. Pandey et al. [8] proposed a residual fault prediction model using fuzzy logic approach. They have predicted defect in multistage of SDLC. A similar study performed by Yadav et al. [9], where defect density is predicted at the end of each phase of SDLC.

Khoshgoftaar et al. [10] have shown that software metrics plays vital role in software quality and reliability modelling. In literature [11-12] it is also found that software metrics have great impact on software reliability. Regarding this Zhang et al. [11] suggested thirty two software metrics in all stage of software development process. Software reliability has also influenced by the ranking of software reliability relevant metrics. Li et al. [12] ranked the reliability relevant software metrics according to their skill in predicting the software reliability by an expert opinion elicitation process. A systematic review of software metrics in software defect prediction approach is conducted by Catel et al. [13]. The ability of design phase metrics in software defect prediction is analyzed by Maa et al. [14]. Recently, a requirement phase based software fault prediction model is proposed by Chatterjee et al. [15], where only the software metrics belongs to requirement phase are considered for model development.

The observations based on the above literature survey and reviews are as follows:

- Software defect prediction can play a vital role in software reliability modeling.
- It is better to predict the severity of software defects in each phase of SDLC along with the total numbers of residual software defects for software quality improvement.
- Top ranked reliability relevant software metrics should be considered in software defect prediction modeling.
- Uncertainty modeling is a major challenge in software defect prediction modeling. Fuzzy logic and BBN are very useful approach for uncertainty modeling. BBN has some additional feature then fuzzy logic like: BBN is a probabilistic approach and it can handle the situations where some data entries are missing or unavailable.

Therefore, based on the above observations, in this paper, a BBN model is proposed for defect estimation and severity analysis of defect in different phases of SDLC.

The rest of the paper is organized as follows. Proposed methodology is presented in section II. Experimental results are presented by case study in section III. Model validations are done in section IV. Concluding remarks are given in section V.

2. PROPOSED METHODOLOGY

The proposed methodology heavily depends on BBN and top ranked reliability relevant software metrics. The detailed description of BBN is described in [16-17].Top ranked reliability relevant metrics for different phase (requirement, design, coding and testing) of SDLC are taken from Li et al. [12].

Requirements stability, requirement fault density, and review, inspection, and walk-through software metrics have been considered as input in the requirements analysis phase to predict the probabilistic values of defect severity at the end of the requirement analysis phase. Probabilistic values of defect severity in requirement phase (PDRP), software complexity, and design review effectiveness software metrics have been considered as input in the design phase to predict the probabilistic values of defect severity at the end of the design phase. The probabilistic values of defect severity in design phase (PDDP), programmer capability, and process maturity software metrics have been considered as input in coding phase to predict the probabilistic values of defect severity at the end of coding phase. Similarly, the probabilistic values of defect severity in coding phase (PDCP), staff experience, and quality of documented test cases have been considered as input in testing phase to predict the probabilistic values of defect severity at the end of the testing phase (PDTP). The following steps are involved in this proposed model.

Step 1- Selection of software metrics

Step 2- Model development (BBN Model)

Step 3- Designingof node probability table (NPT)

Step 4- Compile the BBN model

Step 5-Apply the evidence of software metrics on BBN model

Step 6- Analysis the severity of software defect in each of phase SDLC

Step 7- Estimate the residual software defects

In this section steps 1, 2, 3 and 4 are explained other steps like steps 5, 6 and 7 will be explained with the help of case studies which is shown in section IV.

3.1 Selection of software metrics –

There are numbers of software metrics are available in literature [6, 11-12]. To consider all the software metrics in any software defect prediction model is very critical task. Therefore, considering the most important software metrics in defect prediction model is another challenging task. Regarding this Li et al [12] were ranked the software metrics according to their ability in predicting the software reliability through expert elicitation process. Author considered the top three software metrics applicable in four different phases (requirements analysis, design, coding and testing) of SDLC from Li et al. [12] study.The considered top most reliability relevant software metrics are shown in Table 1.

RFD, Rs, and RIW metrics are taken as input metrics for predicting the severity of defect in requirement analysis phase. Similarly, PDRP, DRC and SC metrics are taken as input metrics for predicting the severity of defect in design phase. PDDP, PC and PM metrics are taken as input metrics for predicting the severity of defect in coding phase. PDCP, SE and QDT are taken as input metrics for predicting the severity of defect in testing phase.

Table-1 Considered top three software metrics present in first four phases of the SDLC

Sl. No	Requirements Phase	Design Phase	Coding Phase	Testing Phase
1	Requirement Fault Density (RFD)	Design Review Effectiveness (DRE)	Programmer Capability (PC)	Staff Experience (SE)
2	Requirements Stability (RS)	Software Complexity (SC)	Process Maturity (PM)	Quality of Document Test Cases (QDT)
3	Reviews,	Requirement	Software	Code defect

	Inspections and Walkthroughs(RIW)	Fault Density	Complexity	density

3.2 BBN model development

In literature [16-17] it is found that there are three different approaches have been used to construct the BBN model 1) Data-based approach 2) Knowledge-based approach and 3) Causal mapping approach. The data-based approaches use conditional independence semantics from the data to induce the BBN models. The knowledge-based approach use causal knowledge of domain experts in constructing BBN models. The causal mapping approach use cause-effect relations and expert knowledge in constructing BBN models [17]. We have considered the causal mapping approach for the construction of BBN model, because in literature [17] it is found that this is the perfect method to build the BBN modes. The proposed model is shown in Figure1. Netica tool [18] is used for model development.

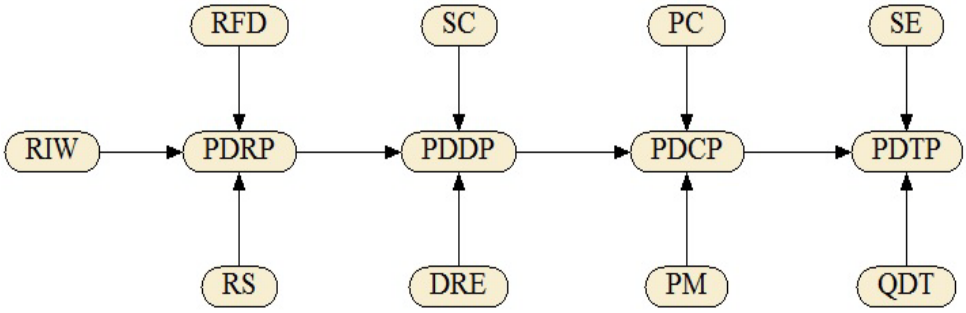

Fig 1. Proposed model

3.3 Designing of node probability table

There are different approaches are exist in literature [19-21] for designing the NPT of BBN. For example: Noisy-OR method [19], Noisy-MAX method [20] and Weighted function method [21]. However, all are problem dependent. Therefore, domain expert based NPT design can be universally apply for all types of problems but designing the large size of NPT from domain expert is not an easy task. So, fuzzy logic approach is applied to reduce the effort of domain expert in NPT development [22-23]. Here, fuzzy logic and domain expert opinion method is applied for NPT development.

3.4 Compile the BBN model

After NPT development for all the nodes of BBN model, the compilation process is carried out. The compile mode of BBN is shown in Figure 2.

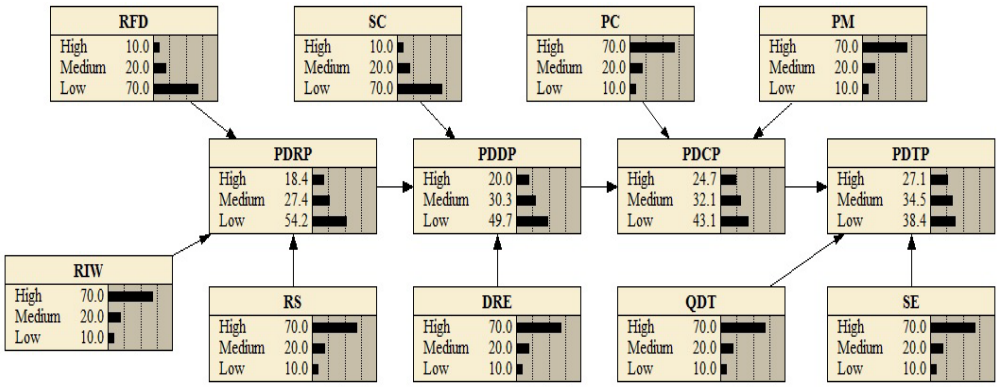

Fig 2. Compile mode of proposed BBN model

III. CASE STUDY

Twenty case studies are illustrated to explain the proposed methodology. Qualitative values of software metrics for 20 different software projects are taken from [6] and reproduced in Table 2 where the qualitative value of software metrics are represented in terms of Low (L), Medium (M) and High (H).

Table-2 Qualitative value of software metrics

Case Study No.	**Project No. [6]**	**Size (KLoC)**	**RS**	**RFD**	**RIW**	**SC**	**DRE**	**PC**	**PM**	**SE**	**QDT**
1	1	6	L	H	H	M	H	H	H	H	H
2	2	0.9	H	H	H	L	H	H	H	H	H
3	3	53.9	H	H	H	H	H	H	H	H	H
4	7	21	M	L	H	L	H	H	H	M	H
5	8	5.8	H	L	H	M	M	H	H	M	M
6	9	2.5	H	M	H	L	H	H	H	H	H
7	10	4.8	H	M	H	M	H	H	H	M	M
8	11	4.4	H	H	H	H	H	H	M	H	M
9	12	19	L	M	H	H	M	M	H	H	M
10	13	49.1	L	H	M	H	H	H	M	M	M
11	15	154	L	H	H	H	H	H	H	H	M
12	16	26.7	M	H	H	L	H	H	H	H	M
13	17	33	M	H	M	L	H	M	M	L	H
14	19	87	M	H	H	H	H	H	H	M	H
15	20	50	L	M	M	H	L	L	H	L	H
16	21	22	M	M	H	L	H	H	H	H	H
17	22	44	L	M	M	M	L	M	H	M	H
18	24	99	L	H	M	M	H	H	H	M	M
19	29	11	H	M	H	M	H	H	H	H	H
20	30	1	H	M	H	L	H	H	H	H	H

4.1 Model illustration: Case Study 1

Software project #1 [6] has been considered to explain the proposed approach. Following are the steps for finding the severity of software defect in each phase of SDLC for case study 1.

Step 1: Selection of software metrics: The selected reliability relevant software metrics and their qualitative values are shown in Table 2.

Step 2: BBN Model Development: Based on the selected metrics and their causal relationships, the BBN model is constructed which is shown in Figure 1.

Step 3: NPT Development:The node probability tables (NPT) are constructed for all the nodes. NPT development process is described in Section 3.3.

Step 4: Compilation of BBN Model: After NPT development for all the nodes of BBN model, the compilation process is carried out. The compile mode of proposed BBN model is shown in Figure2.

Step 5: Apply the evidence of software metrics on BBN model: To obtain the phase wise severity of software defect (probabilistic value) from the proposed model evidence of nodes (qualitative value of software metrics) are applied on compiled mode of proposed BBN model. The resultant values are as follows:

Case study no.: 1

Probabilistic values of defect severity in requirement phase (PDRP): High- 50.2, Medium- 30.6, and Low- 19.2

Probabilistic values of defect severity in design phase (PDDP): High- 25.5, Medium- 40.6, and Low- 33.9

Probabilistic values of defect severity in coding phase (PDCP): High- 20.5, Medium- 33.6, and Low- 45.9

Probabilistic values of defect severity in testing phase (PDTP): High- 19.5, Medium- 33.7, and Low- 46.8

Step 6: Analysis the severity of software defect in different phases of SDLC: From the result of probabilistic values of defect severity in case study 1, it can be observed that the requirement phase is more severe then design, coding and testing phase. High probability value of software defect in requirement phase is 50.2 whereas it is in design phase 25.5, in coding phase 20.5 and in testing phase 19.5. Similarly, based on the above steps the proposed model has been experimented for other case studies. The resultant probabilistic values of defect severity in each phase of SDLC for all 20 real software projects are shown in Figure 3. The graphical representation of resultant value of defect severity is shown in Figure 4. From Figure 4, it can be observed that in most of the projects requirement phase and design phase is more severe than the coding and testing phase. For example the probability value of software defects is high in the requirement phase of case study 1, 2, 6, 10, 12, 16, 17, 18 and 20. Similarly the probability value of software defects is high in the design phase of case study 3, 8, 9, 11, 14, 15 and 19. In literature [8-9, 14-15] it is also found that initial phase metrics like requirement and design phase metrics are more responsible for defects present in the software. In sum we can say that software metrics that are responsible for defects present in the initial phases (requirement and design) of SDLC need to be considered with more attention than the metrics that become available in the later phases (coding and testing) of SDLC.

Case Study No.	Project No.	Size (KLoC)	Requirement Phase			Design Phase			Coding Phase			Testing Phase		
			High	Medium	Low	High	Medium	Low	High	Medium	Low	High	Medium	Low
1	1	6	50.2	30.6	19.2	25.5	40.6	33.9	20.5	33.6	45.9	19.5	33.7	46.8
2	2	0.9	24.2	30	45.8	15.6	27.2	57.3	17	29.2	53.8	18.2	31.9	49.9
3	3	53.9	24.2	30	45.9	38.6	31.5	29.9	21.1	33.3	45.7	19.6	33.7	46.8
4	7	21	14.6	24.8	60.6	13.7	25.3	61	16.4	28.5	55.1	26.4	38	35.6
5	8	5.8	14.6	24.8	60.6	26.6	41.7	31.7	20.8	34.1	45.2	33.7	42.3	24
6	9	2.5	19.6	24.6	55.8	14.5	25.8	59.7	16.6	28.7	54.7	18	31.7	50.3
7	10	4.8	19.6	24.6	55.8	22.4	36.6	41	19.4	32.3	48.3	33.4	42.2	24.4
8	11	4.4	24.2	30	45.9	38.6	31.5	29.9	24.4	34.4	41.3	23.7	37.5	38.7
9	12	19	34.5	40.1	25.4	48.4	33.3	18.4	25.7	40.9	33.4	24.9	39.6	35.5
10	13	49.1	60.1	24.4	15.5	54.6	27	18.4	26	35.9	38.2	34.9	41.9	23.1
11	15	154	50.2	30.6	19.2	50.7	29	20.3	22.5	34.3	43.2	23.4	37.3	39.3
12	16	26.7	30.1	40.4	29.5	17.3	29.5	53.3	17.6	29.9	52.5	21.9	35.5	42.6
13	17	33	32.1	45.4	22.5	17.9	30.5	51.6	31.5	41.3	27.2	51.9	27.3	20.8
14	19	87	30.1	40.4	29.5	42.4	32.6	25	21.8	34	44.2	27.4	39	33.6
15	20	50	45.5	35.1	19.4	60.8	28.9	10.3	58	24.7	17.3	59.3	23.9	16.8
16	21	22	19.5	45.1	35.4	15.9	29.1	55	17.3	29.7	53	18.3	32.1	49.6
17	22	44	45.5	35.1	19.4	43.9	39.9	16.2	25.3	41.6	33.1	28.2	40.4	31.4
18	24	99	60.1	24.4	15.5	26.5	39.6	33.9	20.5	33.5	46	33.7	42.2	24.1
19	29	11	19.6	24.6	55.8	22.4	36.6	41	19.4	32.3	48.3	19.1	33.2	47.8
20	30	1	19.6	24.6	55.8	14.5	25.8	59.7	16.6	28.7	54.7	18	31.7	50.3

Fig 3. Probability value of software defect in each phase of SDLC

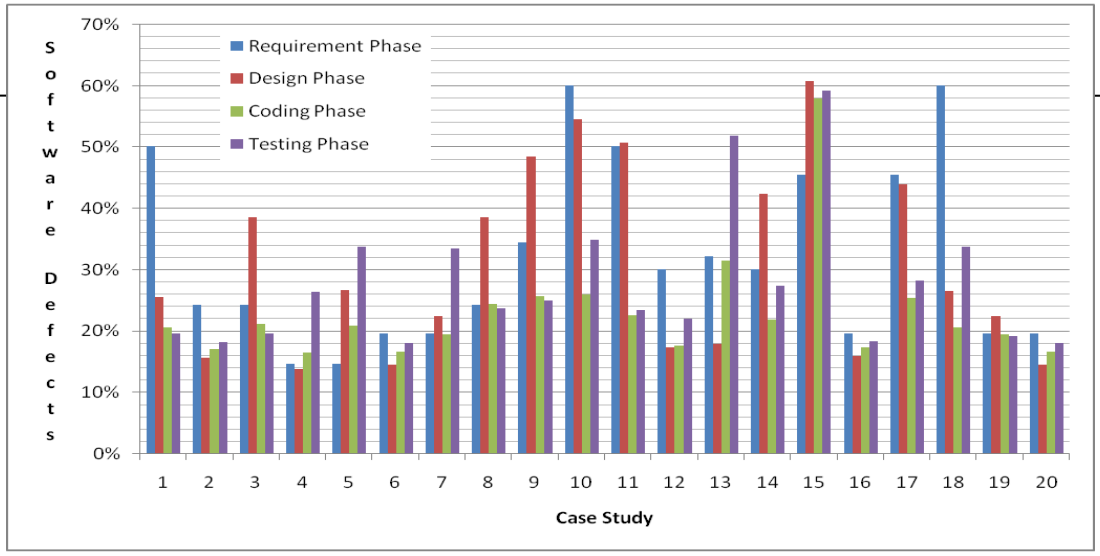

Fig 4. Graphical representation of severity of software defects in each phase of SDLC

Step 7-Estimate the residual software defects: Residual software defects are calculated using theprobabilistic values of defect severity in testing phase and the pessimistic and optimistic value of software defectsobtained from domain experts.The complete resultsare shown in Table 3.

Table 3. Predicted value of software defect

Case	**Actual**	**Defects predicted by**

Study No.	Defects	Fenton 1.7 *et al.* [6]	Pandey *et al.* [8]	Chatterjee *et al.* [15]	Proposed Model
1	148	75	56	96	138
2	31	52	6	3	29
3	209	254	211	216	195
4	204	262	113	210	199
5	53	48	54	52	54
6	17	57	--	8	16
7	29	203	26	29	30
8	71	51	41	66	68
9	90	347	176	114	88
10	129	516	337	393	133
11	1768	1526	1651	1540	1698
12	109	145	128	134	103
13	688	444	136	627	741
14	476	581	574	435	469
15	928	986	869	900	1027
16	196	259	106	198	181
17	184	501	291	220	183
18	1597	1514	--	1485	1635
19	91	116	110	88	85
20	5	46	6	8	5

4. COMPARISON AND VALIDATION OF THE PROPOSED MODEL

The performance of the proposed model has been compared with the previous work done by Fenton et al. [6], Pandey et al. [8] and Chatterjee et al. [15] which is shown in Table 5. To validate the proposed model, following commonly used and suggested evaluation measures [6, 8, 15, 24] have been taken:

i. Mean Magnitude of Relative Error (MMRE): MMRE is the mean of absolute percentage errors and a measure of the spread of the variable z, where z = estimate / actual

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|(yi - \hat{y}i)|}{yi} \quad (1)$$

ii. Balanced mean magnitude of relative error (BMMRE): MMRE is unbalanced and penalizes overestimates more than underestimates. For this reason, a balanced mean magnitude of relative error measure is also considered which is as follows:

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|(yi - \hat{y}i)|}{Min(yi, \hat{y}i)} \quad (2)$$

Where y_i is the actual value and $\hat{y}_i$ is the predicted value and $\bar{y}_i$ is the arithmetic mean of the actual values.

The computed values of MMRE and BMMRE with the help of Eq. 1 and Eq.2 is tabulated in Table 4, for the proposed model and the previous work of Fenton et al. [6], Pandey et al.[8] and Chatterjee et al. [15]. Table 4shows that the proposed model has lesser values of MMRE and BMMRE than the previous work. Hence, the prediction accuracy of the proposed model is better than the previous work.

Table 4- Compared values of model evaluation measures

Evaluation Measure	Proposed Model	Chatterjeeet al.[15]	Pandey et al. [8]	Fenton et al. [6]
MMRE	**0.044**	**0.286**	**0.431**	**1.396**
BMMRE	**0.046**	**0.749**	**0.915**	**1.437**

CONCLUSION

In this paper, BBN approach is used to construct the model. The proposed model considers the top most software metrics of each phase of SDLC. The severity of software defect at each phase of SDLC is analyzed based on the evidence of software metrics. The analyzed severity of software defects in different phases of SDLC is very useful for software project manager to take correct decision in recourse utilization. Software development team may easily detectmost defective phases of SDLC and accordingly they can take correct decision to reduce the defects level.To measure the performance level of proposed model, twenty real software projects data sets have been applied. The predicted defect for twenty software projects are found very near to the actual defects. The performance of the proposed model has been compared with the previous work [6, 8, 15]. The proposed model is very useful for software developers for developing a reliable software product at reduced cost.

REFERENCES

[1] IEEE Std. 1044-1993,"IEEE standard classification for software anomalies,"1994.

[2] P.B. Crosby, "Quality Is Free," McGraw-Hill, 1979

[3] C. Jones, "Measuring Defect Potentials and Defect RemovalEfficiency," CrossTalk, http://www.stsc.hill.af.mil/crosstalk/2008/06/0806Jones.html, June 2008.

[4] S. Amasaki, Y. Takagi, O. Mizuno and T. Kikuno, "A bayesian belief network for assessing the likelihood of fault content," Proc. of 14th international symposium on software reliability engineering (ISSRE), pp. 215–226, 2003.

[5] N. E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," IEEE Transactions on Software Engineering, vol 25(5), pp. 675-689, 1999.

[6] N. E. Fenton, M. Neil, W. March, P. Hearty, L. Radlinski and P. Krause, "On the effectiveness of early life cycle defect prediction with Bayesian Nets," Empirical Software Engineering, vol.13, pp. 499 –537, 2008.

[7] Kumar KS, Misra RB, "An enhanced model for early softwarereliability prediction using software engineering metrics," In: Proc.of 2ndinternational conference on secure systemintegration and reliability improvement, IEEE, pp. 177–178, 2008.

[8] K. Pandey, N. K. Goyal, "Multistage model for residual fault prediction," In Early Software Reliability Prediction, Springer India, pp. 59–80, 2013.

[9] Harikesh Bahadur Yadav, Dilip Kumar Yadav, “A fuzzy logic based approach for phase-wise software defects prediction using software metrics,” Information and Software Technology, vol. 63 pp. 44–57, 2015

[10] Khoshgoftaar, T. M., Allen, E. B., Jones, W. D., and Hudepohl, J. P.,“Accuracy of software quality models over multiple releases,” Annals of Software Engineering, vol. 9, pp. 103–116, 2000.

[11] X. Zhang, H. Pham, “An Analysis of Factors Affecting Software Reliability,” The Journal of Systems and Software, vol. 50 (1), pp. 43–56, 2000.

[12] M. Li and C. Smidts, “A Ranking of Software Engineering Measures Based on Expert Opinion,” IEEE Transactions on Software Engineering, vol.29, pp. 811– 824, 2003.

[13] Catal, “Software fault Prediction: A literature review and current trends,” Expert System with Applications, vol. 38, pp. 4626–4636, 2011.

[14] Y. Maa, S. Zhua, K. Qinb, G. Luob, “Combining The Requirement Information For Software Defect Estimation In Design Time,” Information Processing Letters, vol. 114(9), pp. 469–474, 2014.

[15] Subhashis Chatterjee, Bappa Maji, “A new fuzzy rule based algorithm for estimating software faults in early phase of development,” Soft Computing, DOI 10.1007/s00500-015-1738-x, 2015

[16] FV Jensen, “An introduction to bayesian networks,” Springer Verlag New York, Inc., Secaucus. ISBN 0387915028, 1996.

[17] Sucheta Nadkarni, Prakash P. Shenoy, “A Causal Mapping Approach to Constructing Bayesian Networks,” Decision Support Systems, vol. 38, Issue 2, pp. 259-281, 2004.

[18] Netica, Available at http://www.norsys.com; 2010.

[19] K. Huang and M. Henrion, “Efficient Search-Based Inference for Noisy-OR Belief Networks,” Twelfth Conference on Uncertainty in Artificial Intelligence, Portland, pp. 325-331, 1996.

[20] F.J. Díez, “Parameter adjustment in Bayes networks: the generalized noisy or-gate,” Proc. Ninth Conference on Uncertainty in Artificial Intelligence, pp. 99-105, 1993.

[21] Das, “Generating node probabilities for Bayesian networks: Easing the knowledge acquisition problem,” arXiv preprint cs/0411034, 2004.

[22] K.F.R. Liu,J.Y. Kuo, K. Yeh, C.W. Chen, H.H. Liang and Y.H. Sun, “Using fuzzy logic to generate conditional probabilities in Bayesianbelief networks: a case study of ecological assessment,” Int. J. Environ. Sci. Technol, DOI 10.1007/s13762-013-0459-x, 2013.

[23] Kumar, D. K. Yadav, “A Method for Developing Node Probability Table Using Qualitative Value of Software Metrics,” Proc. of C3IT 2015: 3rd International conference on Computer, Communication, Control and Information Technology, Hooghly, India, 7-8 Feb. 2015, Publisher: IEEE, DOI: http://dx.doi.org/10.1109/C3IT.2015.7060187

[24] C. Kumar, D. K. Yadav, “Software defects estimation using metrics of early phases of software development life cycle,” International Journal of System Assurance Engineering and Management, Springer, DOI: 10.1007/s13198-014-0326-2, December 2014.