Int J Syst Assur Eng Manag
DOI 10.1007/s13198-014-0325-3

ORIGINAL ARTICLE

Early software reliability analysis using reliability relevant software metrics

Harikesh Bahadur Yadav · Dilip Kumar Yadav

Received: 6 June 2014

Abstract The early software reliability analysis is very useful for improving the quality of software at reduced testing effort. Software defect density indicator predicted in the early phases (requirement analysis, design and coding phases) provides an opportunity for the early identification of cost overrun, software development process issues and optimal development strategies. Failure data is not available in the early phases of the software development life cycle (SDLC). However, qualitative values of software metrics are available in the early phases of SDLC. Therefore, in this paper, a model is proposed to predict the software defect density indicator of early phases of SDLC using fuzzy logic and the reliability relevant software metrics of early artifacts. The proposed model is applied on twenty real software projects. It is observed that the requirement analysis phase defect density indicator value is relatively greater than that of the design and coding artifacts. The model is validated with the existing literature. Validation result is satisfactory.

Keywords Software reliability · Software metrics · Defect density indicator · Fuzzy logic · Software defect

1 Introduction

Now a day, software has become the integral part of most of the complex applications and people are working under direct or indirect influence of software. Therefore, it is very important to ensure the reliability of the software system. The reliability of a software system depends upon the number of residual defects. A defect is the product anomaly (IEEE 1988). A general method to measure the reliability of software is to reveal the presence of defects in it, and usually the metric used for it is defect density (DD). The DD is defined as the total number of defects divided by the size of the software (IEEE 1990). The software defect density indicator metric provides the information regarding the reliability improvement during development phases.

Software reliability is an important factor of software quality. Software reliability is the probability that software will not cause any failure of a system for a specified period of time under the specified conditions (IEEE 1990). Reliability is requested to be assured in almost all safety–critical system. Software reliability model was designed to quantify the likelihood of software failure (IEEE 1988; Lyu 1996). The termination of the ability of a functional unit to perform its required function called failure (IEEE 1990). Software reliability plays an important role in the early software development phases (Musa et al. 1987). Lots of study in the past has been made for software reliability estimation and prediction (Lyu 1996; Pham 2007). (Gaffney and Davis 1988; Gaffney and Pietrolewiez 1990) proposed a phase based model for predicting reliability by using the faulty statistics. Rome Laboratory developed a model for early software reliability prediction (McCall et al. 1992; Friedman et al. 1992). The model is mainly based on the software requirement specification and data collected by the organization. Agresti and Evanco (1992) proposed a model to predict defect density on the basis of process and product characteristics. (Smidts et al. 1998) developed a reliability prediction model based on the

H. B. Yadav (✉) · D. K. Yadav
Department of Computer Applications, National Institute of Technology, Jamshedpur 831 014, India
e-mail: yadavaharikesh@gmail.com

D. K. Yadav
e-mail: dkyadav1@gmail.com

Published online: 06 December 2014

requirements change request during the SDLC. The traditional models for software reliability prediction are neither universally successful in predicting reliability behavior, nor generally tractable to users (Cai et al. 1991). The majority of models is based on probabilistic approach.

The causal model for defect prediction with Bayesian net is developed by (Fenton and Neil 1999; Fenton et al. 2007; Fenton et al. 2008). The main feature is that it does not require detailed domain knowledge and it combines both qualitative and quantitative data. (Mohanta et al. 2010, 2011) proposed a model to predict the reliability of object-oriented systems during the early stages of the product development based on bottom-up approach. In this approach, the reliability of the overall system is estimated based on operational profile and reliabilities of classes. (Octane and Yildiz 2014) proposed a novel method using Bayesian networks to explore the relationships among software metrics and defect proneness.

(Pandey and Goyal 2009) have proposed an early fault prediction model using process maturity and software metrics. They have considered the fuzzy profiles of various software metrics in different scale and have not explained the criteria used for developing these fuzzy profiles. The method level metrics are used in most of the fault prediction models. Yadav et al. (2012) proposed a software defect prediction model in which they had considered only the uncertainty associated over the assessment of software size metric and three metrics of requirement analysis phase. (Catal and Diri 2009; Catal 2011) provided a systematic review of various software fault prediction studies with a focus on metrics, methods and datasets. (Radjenovic et al. 2013) reported that the process metrics are successful in finding the faults. (Can et al. 2013) suggested a model for software defect prediction in which they used the benefit of the non-linear computing capability of support vector machine and parameters optimization capability of particle swarm optimization. Recently, (Maa et al. 2014) analyze the ability of requirement metrics for software defect prediction during the design phase.

The most of software reliability models are based on failure data. However, failure data are not available in the early phases of SDLC. There are many factors which affect the software reliability in SDLC. Thirty-two factors are identified which have an impact on the software reliability (Zhang and Pham 2000). In another study, (Li et al. 2000; Li and Smidts 2003) identified thirty software metrics which influence the software reliability.

In fact, most of the software metrics are associated with uncertainty. The smaller size of software testing data, unrealistic assumptions, and the fact that some measures cannot be defined precisely, are the key reasons that a fuzzy logic approach should be developed for predicting the software reliability at the early phase of the SDLC.

The rest of the paper is organized as follows: the proposed model and methodology is discussed in Sect. 2. In Sect. 3, a case study is presented. Results and validation are discussed in Sect. 4 and 5 respectively. Conclusion is presented in Sect. 6.

2 Proposed model and methodology

In the proposed model, defect density indicator of early phases of SDLC is predicted based on the measures present in the early phases of SDLC. Therefore, proposed model leverages the top most reliability relevant metrics (Li et.al. 2000; Li and Smidts 2003) from early phases of SDLC. In the requirement analysis phase, the defect density indicator is predicted using requirement fault density (RFD), requirement stability (RS), and review, inspection and walk through (RIW) software metrics.

The defect density indicator predicted at the end of requirement phase (RPDDI) is taken as input in the design phase along with cyclomatic complexity (CC) and design review effectiveness (DRE) to predict the defect density indicator at the end of the design phase. Similarly, the defect density indicator predicted at the end of the design phase (DPDDI) taken as input in coding phase along with the programmer capability (PC) and process maturity (PM) (Fig. 1). At the end of coding phase, we will get the total number of defects predicted for the software before testing phase using coding phase defect density indicator (CPDDI).

The following steps are involved in this proposed model

A. Selection of software metrics
B. Define the membership function of each input and output variable

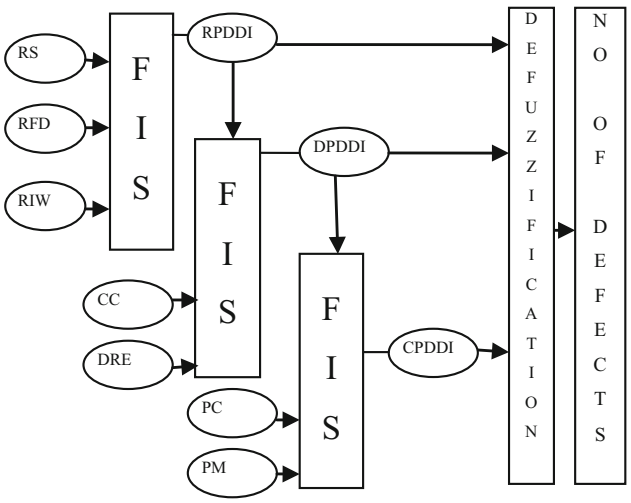

Fig. 1 Proposed model architecture

C. Design fuzzy rules
D. Perform fuzzy inference, and defuzzification.

2.1 Selection of software metrics

Software metrics that are considered in the proposed model are explained as follows:

2.1.1 Requirement phase software metrics

(i) *Requirement stability* (*RS*) Requirement stability is inversely proportional to requirement change request. The requirement change may happen at any time during a software project development. Studies have exposed that more than half the errors are due to imprecisely defined requirements during software development.
(ii) *Requirement fault density* (*RFD*) This metric measures the fraction of faulty requirements specification documents. Requirement fault density provides an indicator of the software quality of developing software during the requirement analysis phase.
(iii) *Review, inspection and walk-through* (*RIW*) This metric purify the software product and can be applied at various points during software project development. The goal of the review process is to ensure that the software requirement specification is feasible, complete, consistent and accurate. From a quality point of view, it is very important metrics.

2.1.2 Design phase software metrics

(i) *Cyclomatic complexity* (*CC*) The measurement of Cyclomatic complexity by McCabe (Kan 2002) was intended to specify a program's understandability and testability. It can be used to indicate an upper bound in the model for estimating the number remaining software defects.
(ii) *Design review effectiveness* (*DRE*) Design defects are usually found by a design review process during the software project development. The goal of design review is to make sure that the design meets the stakeholder's requirements or to find whether design requires modification.

2.1.3 Coding phase software metrics

(i) *Programmer capability* (*PC*) Software complexity depends on the experience of the staff and their intelligence. An experienced and sound technical background programmer will develop quality software with the least number of defects.
(ii) *Process maturity* (*PM*) In Software Company, capability maturity model (CMM) plays a key role in defining software development process improvement. CMM has five levels. Software defect density reduces as one proceeds from one CMM level to next CMM level.

2.2 Define the membership function of each input and output variable

There are many methods of membership value assignment such as: rank ordering, intuition, inference, etc. (Yadav et al. 2012; Ross 2004; Yadav et al. 2012; Yadav and Yadav 2013; 2014; Yadav et al. 2011; Verma et al. 2007). In the intuition method, fuzzy profile is derived from the ability of humans to develop a fuzzy profile through their own innate intelligence and understanding. In the inference method of fuzzy profile development, one uses knowledge to perform deductive reasoning. The assessing preference of a single individual, a committee, a poll, and other opinion methods can be used to assign membership values to a fuzzy variable in the rank ordering method. Membership functions for all the input and output software metrics which are considered in the proposed model should be defined by domain experts. Developing a fuzzy profile of selected software metrics with the help of domain expert knowledge is one of the basic steps in the design of a problem which is to be solved by fuzzy set theory. There are no standard guidelines or rules that can be used for the appropriate membership function construction technique. Another problem that makes membership function construction an important task is the lack of consensus on the definition and interpretation of membership functions. The majority of the methods is application domain dependent and complex. It is impractical to use different membership function construction technique for different application problem. It is not impossible, come up with a single membership construction technique which will work for most application problems.

Membership function can have a variety of shapes like polygonal, trapezoidal, triangular, and so on (Ross 2004; Yadav et al. 2012; Yadav and Yadav 2013; 2014). However, triangular and trapezoidal shapes provide a convenient representation of domain expert knowledge and it also simplifies the process of computation (Kaya and Alhajj 2003). In the proposed model membership function of all the input and output metrics are defined with the help of domain experts. In this model triangular and trapezoidal membership are considered for representing the linguistic state.

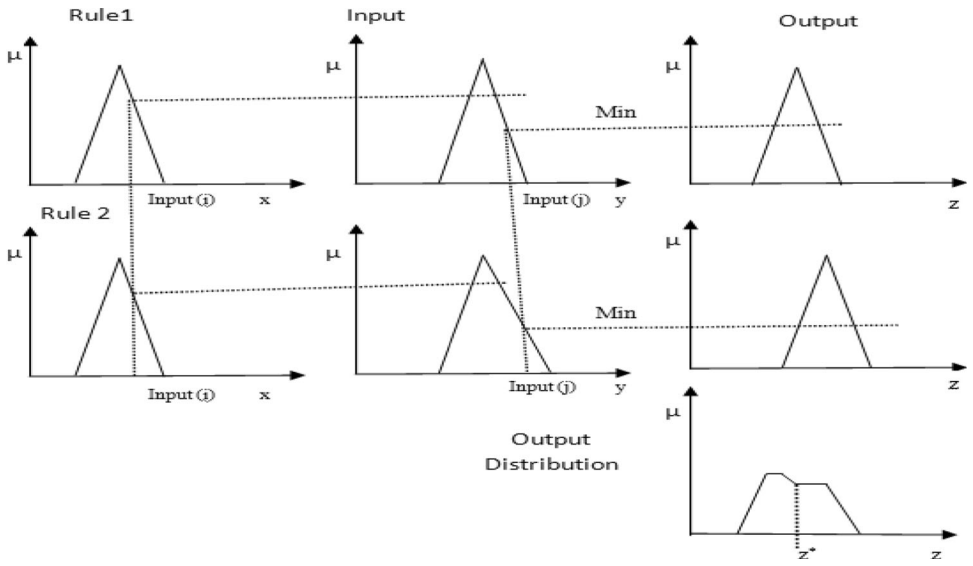

Fig. 2 Process of fuzzy inference and defuzzification

2.3 Design fuzzy rules

In this step fuzzy rule is defined in the form of IF–THEN conditional statement.

IF A is X
THEN B is Y

IF part of the rule is known as the antecedent and THEN part is consequent (Zadeh 1989). Fuzzy rules that are required for the prediction of defects of software projects are defined using human intuition. Instead of considering the entire set of input variables at the same time, software metrics involved from phase to phase reduces the required number of rules. Considering all the selected software metrics one at a time, it is required to define large numbers of rules for the prediction of software defects. However, instead of using generalized fuzzy inference methods proposed model considers cascading the input variables. Therefore, less numbers of rules alone are required.

2.4 Perform fuzzy inference, and defuzzification

Fuzzy inference engine evaluates and combines the result of each fuzzy rule. Fuzzy inference engine maps, fuzzy set into a fuzzy set. A fuzzy Max–Min operator is used for this step. In many applications, the crisp value needs to be obtained as an output. The defuzzification method such as centroid, max–min and bisection etc. maps, fuzzy set into crisp value (Ross 2004). The process of fuzzy inference and defuzzification is shown in Fig. 2. Centroid method of defuzzification is used to calculate the value of z* in this model.

3 Case studies

3.1 The data set used

In order to validate the proposed model, twenty real software project data sets (Fenton et al. 2008) are used for case studies and that is reproduced in Table 1.

3.2 Model illustration: case study 1

In this case study, software project one has been considered to explain the proposed approach. Following are the steps for finding the defect density indicator and total number of residual defects for software project one before the testing phase.

3.2.1 Selection of software metrics

The selected software metrics and their fuzzy range and values for early phases of the SDLC are shown in Tables 2, 3 and 4.

3.2.2 Define the membership function of input and output variable

Membership functions for individual software metrics are illustrated in this section. Membership functions for each input and output software metrics are shown in Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12. The proposed model consists of a set of input and output values. The range of software input and output metrics are in normalized form.

Table 1 Software projects metrics data

Case study no.	Project # (Fenton et al. 2008)	Size (KLOC)	RS	RFD	RIW	CC	DRE	PC	PM
1	1	6	L	H	VH	M	H	H	H
2	2	0.9	H	H	VH	L	H	H	H
3	3	53.9	H	VH	VH	H	H	VH	VH
4	7	21	M	L	VH	L	H	VH	VH
5	8	5.8	H	L	H	M	M	H	H
6	9	2.5	VH	M	VH	L	VH	VH	VH
7	10	4.8	H	M	H	M	H	H	H
8	11	4.4	H	H	H	H	H	H	M
9	12	19	L	M	H	H	M	M	H
10	13	49.1	L	H	M	H	H	H	M
11	15	154	VL	VH	H	H	H	H	H
12	16	26.7	M	H	H	L	H	H	H
13	17	33	M	H	M	L	H	M	M
14	19	87	M	H	H	H	H	H	H
15	20	50	VL	M	M	VH	L	VL	H
16	21	22	M	M	H	L	H	H	H
17	22	44	L	M	M	M	L	M	H
18	24	99	L	H	M	M	H	H	H
19	29	11	VH	M	VH	M	H	VH	H
20	30	1	VH	M	VH	L	H	H	H

Table 2 Requirement analysis phase software metrics

Requirement analysis phase software metrics		Fuzzy range	Value
Input metrics	Requirement stability (RS)	{0–1}	{Low, medium, high}
	Requirement fault density (RFD)	{0–1}	{Low, medium, high}
	Review, inspection and walk-through (RIW)	{0–1}	{Low, medium, high}
Output metrics	Requirement phase defect density indicator (RPDDI)	{0–1}	{Very low, low, medium, high, very high}

Table 3 Design phase software metrics

Design phase software metrics		Fuzzy range	Value
Input metrics	Cyclomatic complexity (CC)	{0–1}	{Low, medium, high}
	Design review effectiveness (DRE)	{0–1}	{Low, medium, high}
	Requirement phase defect density indicator (RPDDI)	{0–1}	{Very low, low, medium, high, very high}
Output metrics	Design phase defect density indicator (DPDDI)	{0–1}	{Very low, low, medium, high, very high}

Table 4 Coding phase software metrics

Coding phase software metrics		Fuzzy range	Value
Input metrics	Programmer capability (PC)	{0–1}	{Low, medium, high}
	Process maturity (PM)	{0–1}	{Low, medium, high}
	Design phase defect density indicator(DPDDI)	{0–1}	{Very low, low, medium, high, very high}
Output metrics	Coding phase defect density indicator (CPDDI)	{0–1}	{Very low, low, medium, high, very high}

3.2.3 Design fuzzy rules

The fuzzy rules for project one in early phases of the SDLC are shown phase wise in Tables 5, 6 and 7.

(i) *Requirements phase fuzzy rule* If RS is high, the defect will be low and if RFD is high, the defect will be higher but it is not applicable for RIW. Therefore, the fuzzy rules are interpreted in the following manner.

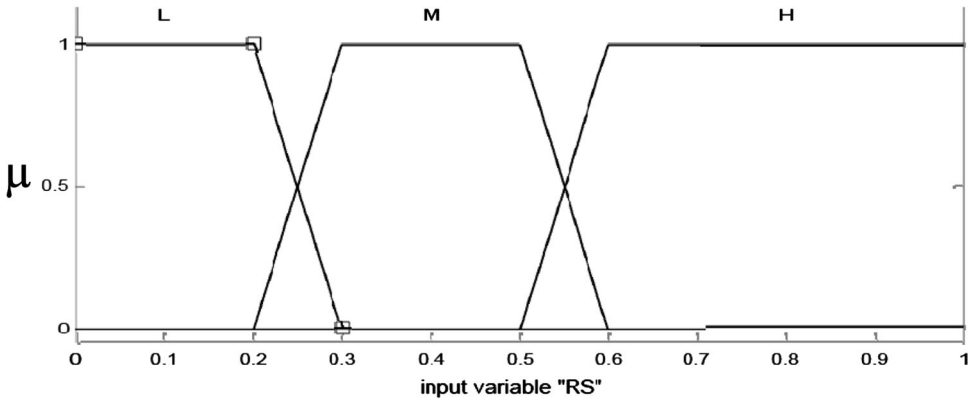

Fig. 3 Requirement stability

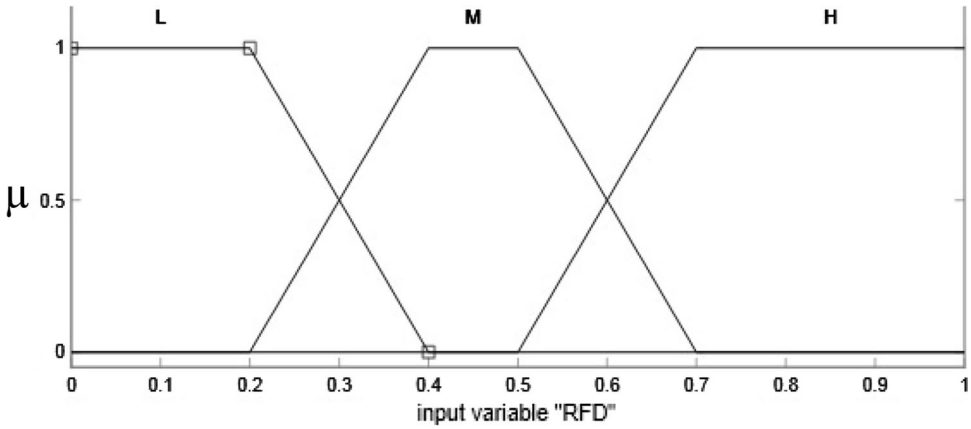

Fig. 4 Requirement fault density

(ii) *Design phase fuzzy rule* For lower value of CC, the defect will be lower, but for lower values of DRE, the defect will be higher. Therefore, the following fuzzy rules are developed.

(iii) *Coding phase fuzzy rule* If the PC and PM are high, then defect will be low in a software project. Therefore, the fuzzy rules are developed as follows:

3.2.4 Perform fuzzy inference, and defuzzification

The defect density indicator value is obtained using fuzzy inference tool of MATLAB at the end of requirement analysis phase, design phase and coding phase. The Result of case study one is shown in Table 8.

4 Prediction result

The prediction results for 20 case studies are shown in Table 9. Table 9 shows the actual defects, predicted defects and defects predicted by (Yadav et al. 2012) and (Fenton et al. 2008). Defects of software projects are obtained based on defect density indicator in the coding phase of the respective project, which has been compared with the similar results done by Fenton, et al. (2008) and (Yadav et al. 2012).

We can observe from Fig. 13 that the maximum number of defect density occurs in requirement analysis phase, which also effect later on in the design phase and coding phase. It is also observed that the software metrics that are responsible for the defect density present in the initial phases of SDLC need to be considered with more attention than the metrics that become available in the later phases of SDLC. Early software defect density indicator prediction could improve the reliability of a software project and helps software managers to achieve reliable software within time and costs.

In case study 8, 15 and 18, defects in the design phase are higher than the requirement analysis phase. Design phase metrics are critical in these projects. Similarly, in case study 1, 2, and 13 coding phase metrics require high consideration along with design phase metrics.

5 Model validation

5.1 Evaluation measures

To validate the prediction accuracy of the proposed model commonly used and suggested evaluation measures have

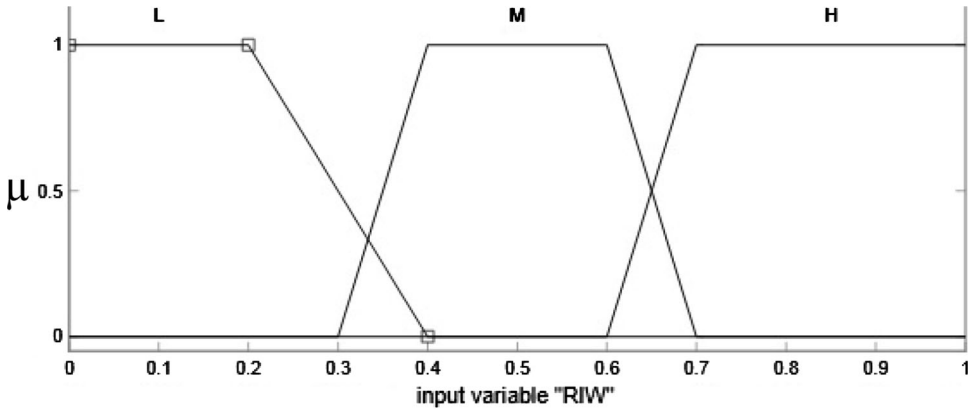

Fig. 5 Review, inspection and walkthrough

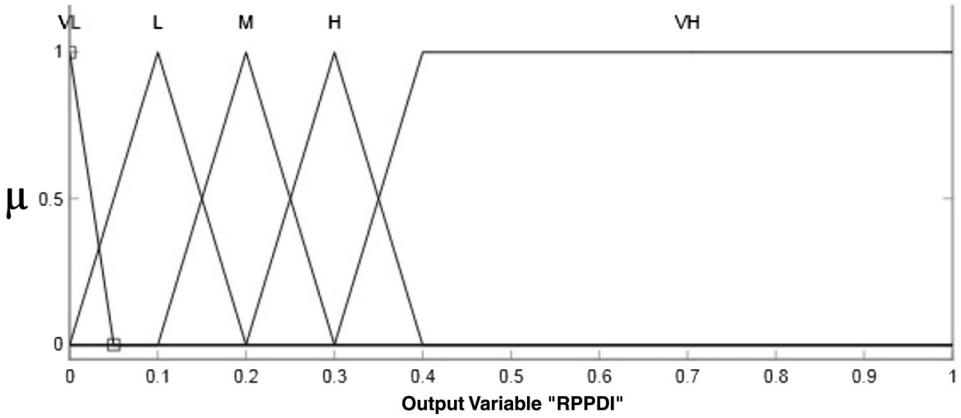

Fig. 6 Requirement phase defect density indicator

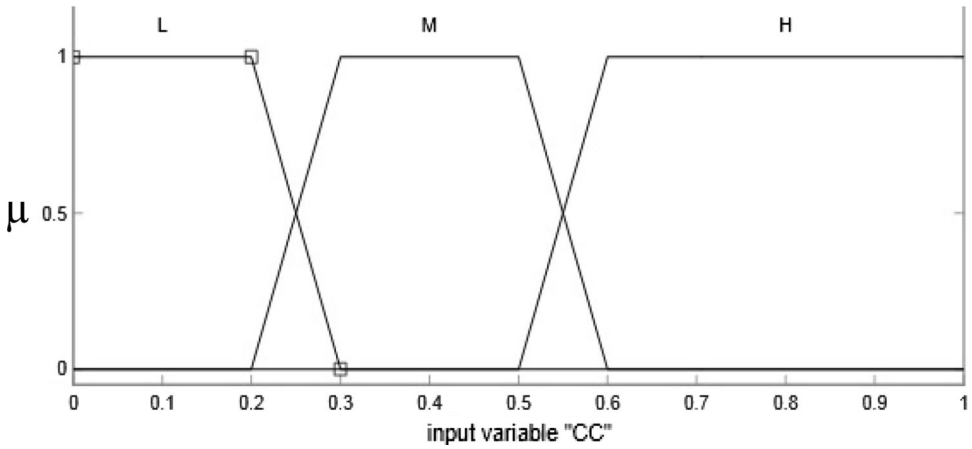

Fig. 7 Cyclomatic complexity

been taken (Fenton et al. 2008; Yadav et al. 2012; Chulani et al. 1999; Kitchenham et al. 2001).

(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 the Z = estimate/actual

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}$$

where, y_i is the actual value and $\hat{y}_i$ is the estimated value of a variable of interest.

(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 the relative error measure is also considered which is as follows:

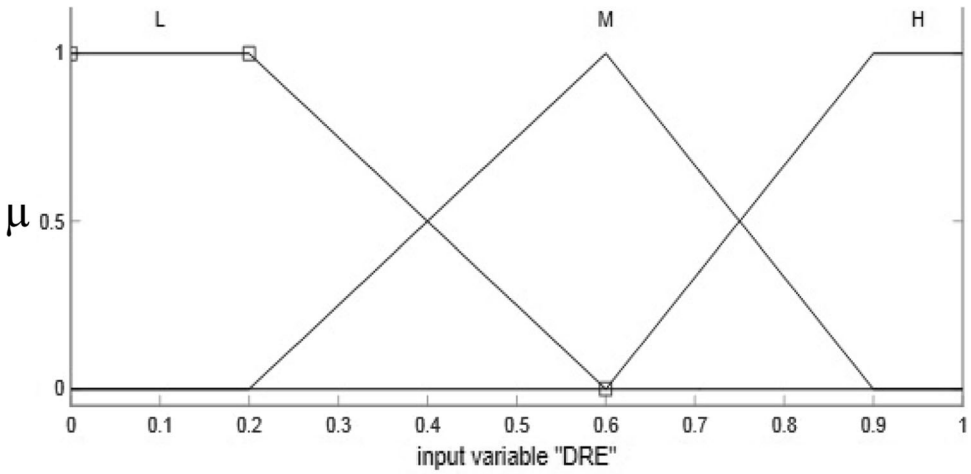

Fig. 8 Design review effectiveness

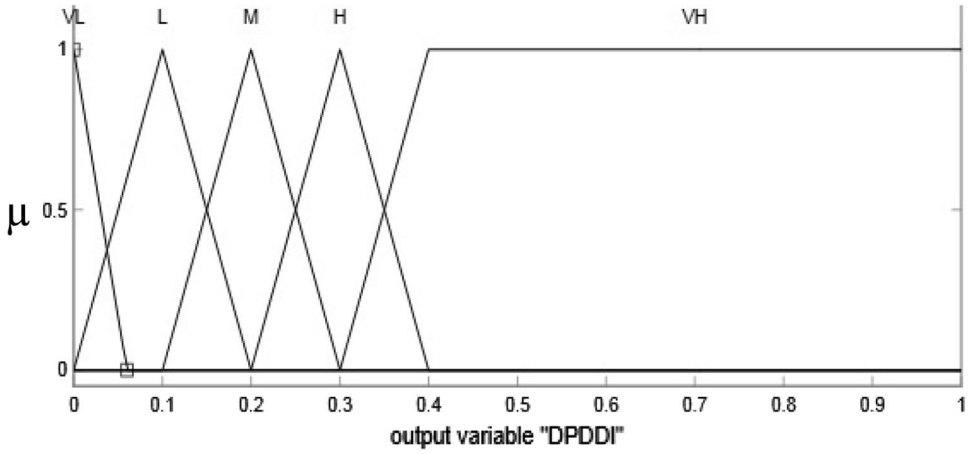

Fig. 9 Design phase defect density indicator

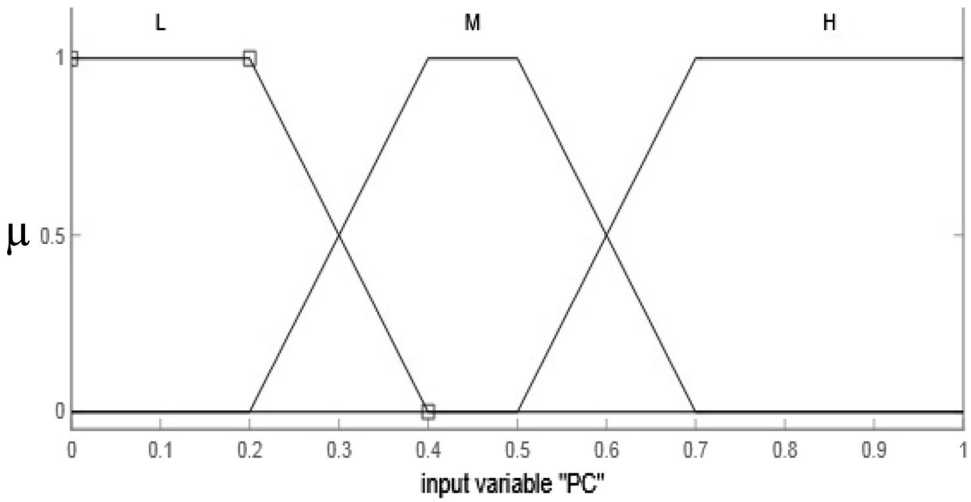

Fig. 10 Programmer capability

$$\text{BMMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{\text{Min}(y_i, \hat{y}_i)}$$

The lesser value of MMRE and BMMRE indicates better accuracy of prediction.

5.2 Validation results

The proposed model is validated using actual defects, and the predicted result of Yadav et al. (2012) and (Fenton et al. 2008). Fenton proposed a Bayesian Net

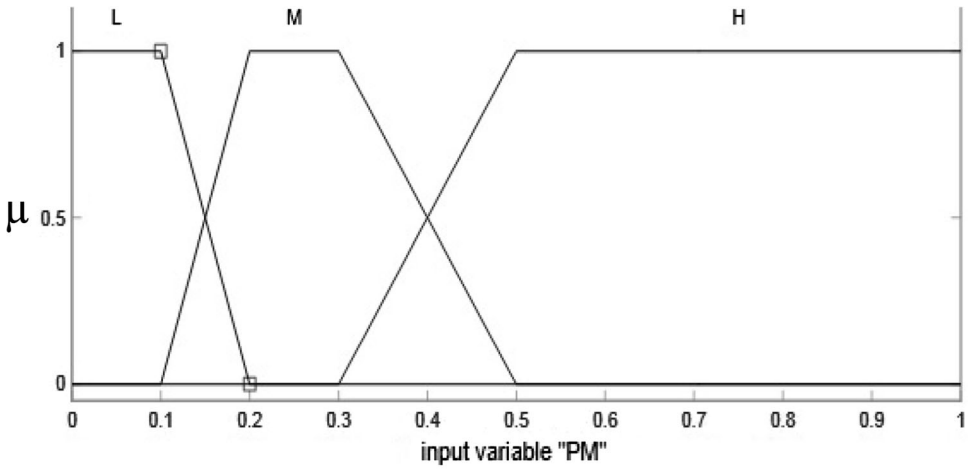

Fig. 11 Process maturity

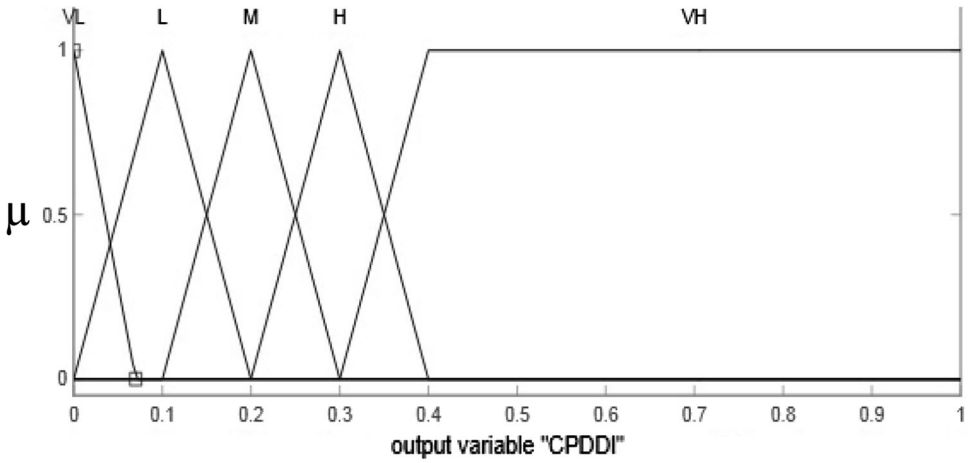

Fig. 12 Coding phase defect density indicator

Table 5 Requirements phase fuzzy rule

Rule no.	Fuzzy rule
1	If RS is L and RFD is L and RIW is L then RPDDI is L
2	If RS is L and RFD is L and RIW is M then RPDDI is VL
…….	……..
26	If RS is H and RFD is H and RIW is M then RPDDI is L
27	If RS is H and RFD is H and RIW is H then RPDDI is VL

Table 6 Design phase fuzzy rule

Rule no.	Fuzzy rule
1	If CC is L and DRE is L and RPDDI is VL then DPDDI is VL
2	If CC is L and DRE is L and RPDDI is L then DPDDI is VL
……..	……..
44	If CC is H and DRE is H and RPDDI is H then DPDDI is VH
45	If CC is H and DRE is H and RPDDI is VH, then DPDDI is VH

model for predicting the software defects for the same software projects.

It can be observed in Table 10 that the MMRE and BMMRE for the proposed model are 0.0687 and 0.0757, respectively. Clearly, the MMRE and BMMRE of the proposed model come out to be much lesser than that of the (Fenton et al. 2008) model and (Yadav et al. 2012) model.

It can also be observed that the predictive accuracy of the models expressed by different measures increases with the size of the project. Measures based on the relative error

Table 7 Coding phase fuzzy rule

Rule no.	Fuzzy rule
1	If the PC is L and PM is L and DPDDI is VL then CPDDI is VL
2	If the PC is L and PM is L and DPDDI is L then CPDDI is L
…….	……
44	If the PC is H and PM is H and DPDDI is H then CPDDI is L
45	If the PC is H and PM is H and DPDDI is VH then CPDDI is H

Table 8 Defect density indicator of project one

Case study no.	Project No. # (Fenton et al. 2008)	RPDDI	DPDDI	CPDDI	No. of defects = CPDDI × LOC
1	1	0.0133	0.0183	0.0222	133

Table 9 Predicted defect density indicator in requirement analysis, design, and coding Phase

Case study no.	RPDDI	DPDDI	CPDDI	Defects predicted by			Actual defects
				Proposed model	Yadav et al. (2012)	Fenton et al. (2008)	
1	0.0133	0.0183	0.0222	133	88	75	148
2	0.0167	0.0219	0.0290	26	9	52	31
3	0.0025	0.0031	0.0038	205	261	254	209
4	0.0066	0.0080	0.0087	183	204	262	204
5	0.0066	0.0080	0.0087	51	56	48	53
6	0.0044	0.0059	0.0061	15	24	57	17
7	0.0044	0.0059	0.0061	29	70	203	29
8	0.0033	0.0181	0.0183	81	64	51	71
9	0.0030	0.0042	0.0046	87	92	347	90
10	0.0014	0.0020	0.0025	125	476	516	129
11	0.0055	0.0073	0.0113	1,740	1,490	1,526	1,768
12	0.0025	0.0031	0.0038	102	130	145	109
13	0.0100	0.0146	0.0185	611	589	444	688
14	0.0030	0.0042	0.0046	400	130	581	476
15	0.0033	0.0181	0.0183	915	892	986	928
16	0.0064	0.0075	0.0081	180	214	259	196
17	0.0025	0.0031	0.0038	169	213	501	184
18	0.0033	0.0181	0.0191	1,554	1,440	1,514	1,597
19	0.0066	0.0080	0.0082	91	107	116	91
20	0.0030	0.0042	0.0046	5	5	46	5

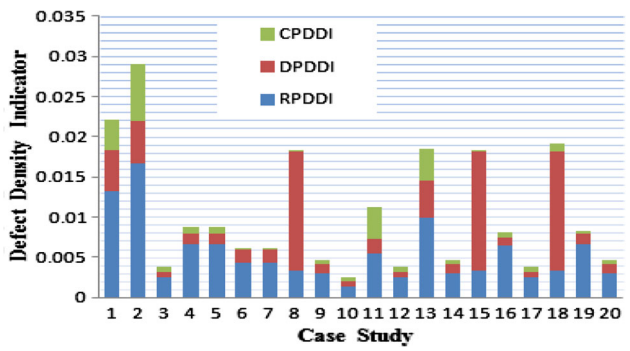

Fig. 13 Defect density indicator in early phases

(MMRE, BMMRE) decrease significantly, as project size increases for all three models.

6 Conclusion

In this paper, a fuzzy logic based model is proposed for predicting software defect density indicator at early phase of the SDLC. The proposed model considers only reliability relevant software metrics of the early phase of the SDLC. The proposed model takes into account the uncertainty associated with the reliability relevant software

Table 10 Values of model evaluation measures

Project size	MMRE			BMMRE		
	Fenton et al. (2008)	Yadav et al. (2012)	Proposed model	Fenton et al. (2008)	Yadav et al. (2012)	Proposed model
Projects < 5 KLOC ($n = 5$)	3.5024	0.5268	**0.0970**	3.5245	0.6862	**0.1065**
5KLoC $\leq$ Projects < 50 KLOC ($n = 10$)	0.9731	0.3936	**0.0654**	1.0416	0.4237	**0.0715**
Projects ≤ 50 KLOC ($n = 5$)	0.1375	0.1313	**0.0471**	0.1424	0.1404	**0.0535**
All projects ($n = 20$)	1.11377	0.3613	**0.0687**	1.4375	0.4185	**0.0757**

Bold values indicate the results of the proposed model

n number of projects, *MMRE* mean magnitude of relative error, *BMMRE* balanced mean magnitude of relative error

metrics of early phases of SDLC. The predicted defect for 20 software projects are found very near to the actual defects detected during testing. The predicted defect density indicators are very helpful to analyze the defect severity in different artifacts of SDLC of a software project. This provides a guideline to the software manager for early identification of cost overruns, schedules mismatch, software development process issues, software resource allocation and release decision making etc.

References

Agresti WW, Evanco WM (1992) Projecting software defects form analyzing Ada design. IEEE Trans Softw Eng 18(11):988–997

Cai KY, Wen CY, Zhang ML (1991) A critical review on software reliability modeling. Reliab Eng Syst Saf 32(3):357–371

Can H, Jianchun X, Ruide Z, Juelong L, Qiliang Y, Liqiang X (2013) A new model for software defect prediction using particle swarm optimization and support vector machine. Control and decision conference (CCDC), 25th Chinese, p 4106–4110

Catal C (2011) Software fault Prediction: a literature review and current trends. Exp Syst Appl 38:4626–4636

Catal C, Diri B (2009) A systematic review of software fault predictions studies. Exp Syst Appl 36(4):7346–7354

Chulani S, Boehm B, Steece B (1999) Bayesian analysis of empirical software engineering cost models. IEEE Trans Softw Eng 25(4):573–583

Fenton NE, Neil M (1999) A critique of software defect prediction models. IEEE Trans Softw Eng 25(5):675–689

Fenton NE, Neil M et al (2007) Predicting software defects in varying development lifecycles using bayesian nets. Inf Softw Technol 49(1):32–43

Fenton NE, Neil M et al (2008) On the effectiveness of early life cycle defect prediction with bayesian nets. Empir Softw Eng 13:499–537

Friedman MA, Tran PK, Goddard PL (1992) Reliability techniques for combined hardware and software system. Rome laboratory Technical Report RL-TR-92-95 1-2

Gaffney JE Jr, Davis CF (1988) An approach to estimating software errors and availability. In: Proceedings of 11th Minnowbrook workshop on software reliability, SPC-TR-88-007, version 1.0, July 26–29, Blue Mountain Lake, NY

Gaffney JE Jr, Pietrolewiez J (1990) An automated model for software early error prediction (SWEEP). In: Proceedings of 13th Minnowbrook workshop on software reliability, July 24–27, Blue Mountain Lake, NY

IEEE (1988) Guide for the use of IEEE standard dictionary of measures to produce reliable software. IEEE, New York, IEEE Std. 982.2-1988

IEEE (1990) Standard glossary of software engineering terminology. IEEE, New York, p 1–84, IEEE Std. 610.12-1990

Kan SH (2002) Metrics and models in software quality engineering, 2nd edn. Addison wesley, Boston

Kaya M, Alhajj R (2003) A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining. In: The 12th IEEE international conference on Fuzzy systems, 2003, FUZZ'03, vol 2, p 881–886

Kitchenham AB, Pickard LM, MacDonell SG, Sheppered MJ (2001) What accuracy statistics really measure? IEEE Proc Softw 148(3):81–85

Li M, Smidts C et al. (2000) Ranking software engineering measures related to reliability using expert opinion. In: Proceedings of the 11th international symposium on software reliability engineering (ISSRE), San Jose, p 246–258

Li M, Smidts C (2003) A ranking of software engineering measures based on expert opinion. IEEE Trans Softw Eng 29(9):811–824

Lyu MR (1996) Handbook of software Reliability Engineering. IEEE Computer Society Press, Los Alamitos

Maa Y, Zhua S, Qinb K, Luob G (2014) Combining the requirement information for software defect estimation in design time. Inf Process Lett 114:469–474

McCall JA, Randell W, Dunham J (1992) Software reliability, measurement, and testing. Rome laboratory Technical Report RL-TR-92-95 1-2

Mohanta S, Vinod G, Ghosh AK, Mall R (2010) An approach for early prediction of software reliability. ACM SIGSOFT Softw Eng Notes 35:1–9

Mohanta S, Vinod G, Mall R (2011) A technique for early prediction of software reliability based on design metrics. Int J Syst Assur Eng Manag 2:261–281

Musa JD, Iannino A, Okumoto K (1987) Software reliability: measurement, prediction, application. McGraw-Hill Publishers, New York

Octane, Yildiz OT (2014) Software defect prediction using bayesian networks. Empir Softw Eng 19:154–181

Pandey AK, Goyal NK (2009) A fuzzy model for early software fault prediction using process maturity and software metrics. Int J Electron Eng 1(2):239–245

Pham H (2007) System software reliability. Reliability engineering series. Springer-Verlag publisher, London

Radjenovic D et al (2013) Software fault prediction metrics: a systematic literature review. Inf Softw Technol 55(8):1397–1418

Ross TJ (2004) Fuzzy logic with engineering applications, 2nd edn. Wiley, New York

Smidts C, Stutzke M, Stoddard RW (1998) Software reliability modeling: an approach to early reliability prediction. IEEE Trans Reliab 47(3):268–278

Verma AK, Srividya A, Gaonkar RSP (2007) Fuzzy reliability engineering: concepts and applications. Narosa Publishing House, Delhi

Yadav HB, Yadav DK (2013) A fuzzy logic approach for software quality modeling. In: Proceedings of the second international conference on computing sciences (ICCS-2013), Lovely Professional University, Punjab, p 32–39

Yadav HB, Yadav DK (2014) A multistage model for defect prediction of software development life cycle using fuzzy logic. In: Proceedings of the third international conference on soft computing for problem solving (SOCPROS-2013), IIT Roorkee. Advances in intelligent systems and computing, vol 259, Springer India Publication, India, p 661–671

Yadav JY, Kharat V, Deshpande A (2011) Fuzzy description of air quality: a case study. Rough sets and knowledge technology. Lecture notes in computer science, vol 6954, p 420–427

Yadav DK, Charurvedi SK, Mishra RB (2012a) Early software defects prediction using fuzzy logic. Int J Perform Eng 8(4):399–408

Yadav DK, Chaturvedi SK, Misra RB (2012) Forecasting time-between-failures of software using fuzzy time series approach. In: IEEE Proceeding of North American fuzzy information processing society (NAFIPS), Berkley, p 1–8

Zadeh LA (1989) Knowledge representation in fuzzy logic. IEEE Trans Knowl Data Eng 1:89–100

Zhang X, Pham H (2000) An analysis of factors affecting software reliability. J Syst Softw 50(1):43–56