Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 89 (2016) 359 – 368
Procedia Computer Science

Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)

Fuzzy Logic based Software Reliability Quantification Framework: Early Stage Perspective (^{FL}SRQF)

Syed Wajahat Abbas Rizvi[a,*], Vivek Kumar Singh[b] and Raees Ahmad Khan[c]

[a]Babu Banarasi Das University, Lucknow, India
[b]Babu Banarasi Das BBD National Institute of Technology and Management, Lucknow, India
[c]Dr. Bhimrao Ambedkar University, Lucknow, India

Abstract

Today, the influence of information technology has been spreading exponentially, from high level research going on in top labs of the world to the home appliances. Such a huge demand is compelling developers to develop more software to meet the user expectations. As a result reliability has come up as a critical quality factor that cannot be compromised. Therefore, researchers are continuously making efforts to meet this challenge. With this spirit, authors of the paper have proposed a highly structured framework that guides the process of quantifying software reliability, before the coding of the software start. Before presenting the framework, to realize its need and significance, the paper has presented the state-of-the-art on software reliability quantification. The strength of fuzzy set theory has been utilized to prevail over the limitation of subjectivity of requirements stage measures. Salient features of the framework are also highlighted at the end of the paper.

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016

Keywords: Fuzzy Logic; Reliability Quantification Framework; Software Defects; Software Metrics; Software Reliability.

1. Introduction

The role of software has been increasing in our life day by day. Earlier it was limited to desktops only, but now has reached to the devices that can easily accommodate in our pockets. Nobody can think about a life without the devices controlled by software[1]. Such dependence as well as trust on software compels the software industry to be more conscious and attentive while developing software, so that delivered software became successful in their operational life[2]. On the other hand, it has also been noticed that, in industry most of the development activity is carried out in labor-intensive manner[3]. System developers are also struggling to deliver software with acceptable level of quality, within given resources and time. Such a pressure on the software professionals cannot be ignored as one of the key factor for software whose reliability is not up to mark[4].

A lot of unfortunate events had already occurred in the defense and health sectors due to the unreliability of corresponding software applications[5]. After realizing reliability as one the key quality attribute, its prediction cannot be delayed or ignored. Therefore, there is an emergent need to ensure reliability of developing software as early as

*Corresponding author. Tel.:91-749-992-0199.
E-mail address: swabbasrizvi@gmail.com

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016
doi:10.1016/j.procs.2016.06.083

possible. So that developers can take suitable corrective measure before they start writing the actual code. In the last two decades, a large number of models for predicting the reliability have been proposed. But still, this domain of software engineering has been attracting more researchers to contribute further. It is evident from the review of the literature that reliability has been estimated or predicted through a variety of techniques like, formal methods, neural network, cause-effect graph analysis, multiple linear regressions, fuzzy logic, and many more. But prediction at early stage was rarely discussed[6]. The researchers had put their best efforts, but still there are a number of theoretical and practical issues noticed in many studies, that undermine the strength as well as validity of most models. Appropriate validation process is a necessity for the success of every effort. But majority of the models lack quality validation.

Even though, it is a universally accepted statistic that 70–80% of all the faults in software are get introduced during the requirements phase, this phase of the SDLC had not been given importance while quantifying the reliability. Majority of existing reliability models are applicable only in the later stages of development, and helping developers either by the end of coding phase or in the testing stage[7]. That becomes too late for developers to take corrective measure to improve its reliability. Despite the obvious variety of software reliability quantification, a prospect to design a comprehensive framework that can be followed by industry personnel and researchers to quantify reliability on the basis of requirement and design stage measures appears highly advantageous and significant. The remaining of the paper is structured as follows; Section 2 presents a comprehensive review of the literature and concludes with a summary of critical findings. Section 3 describes different phases of the proposed reliability quantification framework. A summary of the key features of the framework is listed in Section 4, while the paper finishes off in Section 5.

2. Related Work

Research in a specific dimension needs a highly structured review and study of literature related to that theme. A critical review of the literature make available information regarding, what has been done so for in the area, leading to a significant exploration. Comprehensive and careful reviews of the researchers also endorse better understanding of the selected topic, approach, procedure, method and algorithms and facilitate to frame useful hypothesis[8].

Software reliability quantification has attracted immense interest from researchers as well as software practitioners since the early 1990's. Traditional methods for quantifying the software-reliability such as reliability growth models estimates reliability on the basis of the defects observed during validation testing, where operational patterns represent how actually the product would be used. However, quantifying software reliability in an early stage has been a tricky research topic that many researchers have attempted to resolve with limited success[9]. Unfortunately, absence of failure data in early stages of software makes it challenging to measure the reliability. So there are just a few attempts, addressing the concept of early software reliability assessment or prediction.

During the review of literature it is commonly observed that software metrics has been playing a prominent role in fault identification[10–13]. In a study[14] authors had focused on the selection of an appropriate metrics suite to develop a prediction model. The study also demonstrates that how this selection impact on the fault prediction accuracy. While, Maa *et al.*,[15] had proposed a reliability prediction model, that highlighted the potential of requirement and design metrics in defect prediction at the early stage of development. In another effort, the concept of bayesian networks was used by Okutan and Yildiz,[16] to publicize the relationship between software metrics and defect proneness. In[10] authors supported the role of method-level metrics in predicting defects of application software. While the efforts had done by Radjenovic *et al.*,[12] drawn attention towards the potential of Chidamber and Kemerer's object-oriented metrics in predicting defects. The study also concluded that these metrics are not only the most frequently used metrics but also used twice than other conventional metrics. Another work in the area of defect prediction[17] has considered the role of process maturity with software metrics, while developing a defect prediction model. The study had developed fuzzy profiles for different metrics followed by the fuzzy inference process, but the criteria behind these profiles were not justified properly. In a study Olga Georgieva *et al.*,[18] have used the fuzzy logic approach for measuring software reliability, and concluded that participation of fuzzy logic has overcome the limitations of probability based reliability models. Another fuzzy based model proposed by Yadav *et al.*,[6] that predicts the residual faults during the testing stage. Another well known work done by Pandey and Goyal,[19] where, a data mining technique (classification) was used with fuzzy logic to categorized software modules as fault prone or not. While, the research[20] had demonstrated, how fuzzy logic can solve the modeling issue of reliability? The study had developed a fuzzy based reliability growth model

that estimate software defects at the testing stage during development. Khalsa[21] had also proposed an approach that make use of fuzzy sets to identify software modules with larger defect density in the design stage of SDLC. While Adaptive-Network based Fuzzy Inference System approach had been followed by Yaun and Zhang[22] to develop a reliability model based on Fuzzy-Neural hybrid network.

After briefly describing a variety of research works, some of the pertinent and resent studies have been described in the following paragraphs with proper detail followed by corresponding critical findings.

Jaiswal and Giri, (2015) developed a Reliability estimation model of component-based software system. Beside this author had also developed a model that computes reusability in terms of understandability, variability, portability, maintainability and flexibility[23].

Critical Findings:

- One important finding is the weight that the author had used for understandability, variability, portability, maintainability and flexibility to compute the reusability. All the five factors were multiplied by a fixed value (i.e. 0.2). This is not justified as each of these factors may have different magnitude of influence on reusability. Although, the study may use the multiple linear regression to get better values in this case.
- The study had not described the development as well as validation process properly. It is unclear how accurate the reliability prediction given by this approach would be.
- Author did not perform the correlation analysis among the initially identified factors, to discard those that provide redundant information (i.e. measures similar property). Applying Pearson's correlation test with suitable significance level can do this. For each couple of highly correlated factor, only one of them will be selected.

Yadav and Yadav (2015) proposed a model that calculates the number of software defects at the end of testing phase. The proposed model had considered requirement analysis, design, coding and testing metrics. These metrics are assessed in linguistic terms and fuzzy inference system had been employed to develop the model. The proposed model predicated the defect density at the end of testing phase using the metrics and fuzzy inference system[24].

Critical Findings:

- The model would be an early predictor of reliability and more effective as well as useful if it considers requirement and design metrics only. Because prediction at early stage (design), using only requirements and design measures can help developers to arrest some of the defects to propagate in subsequent phases and produce more reliable software with lesser number of defects.
- Model is only computing the number of defects and not quantifying reliability.
- The study only provides the number of defects but not providing any suggestive guidelines to control or reduce the number of defects.

Anil and Namrata (2015) proposed a Reliability estimation model of object-oriented software in design phase. The model computes reliability in terms of effectiveness and functionality. Prior to develop reliability model, study had developed separate models for effectiveness as well as functionality. All the three models have many serious technical issues that question on their validity as well as on the entire study itself[25].

Critical Findings:

- There are many factors that are more significantly impact on reliability than effectiveness and functionality. But overlooking them and considering effectiveness and functionality without any quantitative ground is not justified.
- The most critical point is that the equation of the developed reliability model (equation 4 in [25]) shows that effectiveness and functionality negatively impacting the reliability value (i.e. both have negative coefficient), which is not true. Because each of these are positively correlated with the software reliability.
- Similarly the significance (p value) of the 'encapsulation' in Table 2 (i.e. 0.783), discourage its involvement in the 'effectiveness estimation model'.
- All the three ANOVA tables (no. 4, no. 7 and no. 11 in [25]) contain wrong values. The value of F is the ratio of mean sum of squares, but there the values are totally meaningless.
- Developing a model using just five records, questions on the validity of the model.

- It can be noticed from the tables that the number of records (sample size) used to develop the model are just 5, that is too small (i.e. Value of N is 5 in ANOVA table).

Amitabha (2012), proposed and implemented a reliability estimation framework for object-oriented design. Focus of the study was to compute complexity of object-oriented design, followed by reliability computation in terms of complexity. The study had used multiple linear regression to quantify complexity and reliability[26].

Critical Findings:

- The thesis had developed two multiple regression models (i) Complexity Estimation Model (CEM) and (ii) Reliability Estimation Model (REM).
- The study had highlighted that the Inheritance impacted positively on complexity (means as inheritance increases the complexity of the OO design will also increases), but the proposed Complexity model (CEM) does not support this, as "IMc" has a negative coefficient that will impact on the OOD complexity inversely.
- Similarly, the paper had also emphasizes that the Encapsulation is inversely proportional to the deign complexity (means as encapsulation increases the complexity of the OO design will go down), but the proposed Complexity model (CEM) does not support this, as "EMc" has a positive coefficient means the proposed model will increase the OOD complexity as the encapsulation increases.
- The study had not justified the goodness or statistical significance of neither of the model. It is not clear how efficiently these models are quantifying their respective dependent variables (Complexity and Reliability).
- In the Complexity Model the significance of individual independent variable was not shown, that is required to justify their participation as independent variables in the complexity model. (t Test should be used for this.)

Wende Kong, (2009) in his Ph.D., proposed an approach to predict the reliability at the end of the requirements phase, on the basis of SRS document. Focus of the study was on the correctness and completeness of the SRS. The author had used the Cause-Effect Graph Analysis for predicting the reliability. The study mathematically formalized the cause effect graph, and applied it on SRS to identify its faults, subsequently fault tree was built through the identified SRS faults. In order to analyze the fault tree Binary Decision Diagram (BDD) approach, along with an algorithm were used to quantifying the impact of the detected requirements faults on software reliability[27].

Critical Findings:

- The process of identifying SRS faults is totally manual, requires domain knowledge and understanding of the system under study along with inspector's creativity, experience and even intuition.
- Without prior and comprehensive knowledge of the system, the faults found through CEGA may not be correct and the final reliability estimation may not be very meaningful.
- Approach is very costly and time-consuming, specially, to construct an initial Cause Effect Graph (CEG) from a given informal specification.
- Not every aspect of a software system will be specifiable by a CEG, because a CEG can only capture functional requirements specified in the SRS. CEG analysis could not detect hidden requirements.
- Validation process was not up to the mark. It is unclear how accurate the reliability prediction given by this approach would be.
- Scalability is also one the issue, for large SRS it is very difficult to build and analyze the CEG.

Hooshmand and Isazadeh, (2008) proposed an approach for early software reliability assessment based on software behavioral requirements. Viewcharts has been used to specify the behavior description of software systems. The author had also used the concept of Markov chain with viewchart, in order to determine the rate of system's transition among its different states. The study further predicated some states, for each of the system's view, those may cause system failures, and assess software reliability as the union of the probabilities of these failure states[28].

Critical Findings:

- Drawing viewchart from the system specification is a manual task and needs to be done by a person having proper awareness about the different dimensions of system's behavior.

- The study had not specified any rule or guidelines for drawing the viewchart specification from the corresponding system behavior.
- To calculate the rate of system state transition, a prototype of the system needs to be build based on its viewchart specifications. Besides that the prototype will be executed with some input from the corresponding operational profile. This makes the approach quite complicated and expert specific, especially at the requirement stage.
- The approach also has the scalability issue, for systems of significant size developing the viewchart specification would be a challenging job.
- As the reliability assessment is totally based on the union of the probabilities of failure states, therefore for each of the view identifying and introducing the probable events those may cause a system failure, needs the comprehensive knowledge about the different behaviors of the system.

2.1 Summary of review findings

After reviewing a variety of studies on reliability quantification, it is the time to sum up the findings, and suggest the way to reach to a feasible solution. Following is the summary of critical observations noticed during the review:

- No consensus or standard steps/procedure among researchers for predicting software reliability.
- Most of the studies that incorporated Multiple Linear Regression had not bothered about multicollinearity and autocorrelation at all.
- Some of the studies have been suffering from severe technical shortcomings that compel to deduce that those researchers had not put their sincere effort.
- Dataset used for empirical analysis were inappropriate in size and also lacks quality data.
- Some researchers performed quantification quite well, but did not provide suggestive measure and guidelines to be followed for controlling the unreliability.
- One of the observations that cannot be overlooked is the need of timely identification and subsequent fixation of residual defects so that reliable software could be delivered in time.
- The best time to detect and arrest faults is the requirements and design stages. To accomplish this task researchers are bound to use quality measures based on these stages. But usually most of metric values in early stages are subjective as their sources are the opinions of domain experts.
- Therefore, to deal with such intrinsic subjectivity and vagueness, fuzzy techniques have come up as a dependable tool in capturing and processing these early stage metric values.
- There are just a few attempts where fuzzy techniques were used to quantify the reliability. But the key concern is the time and the stage of SDLC. These models are helping developers either by the end of coding phase or in the testing stage. These feedbacks make it too late to improve the existing product towards a more reliable one.

Before concluding this section, it is needed to suggest a solution that will overcome the problems or limitations identified and highlighted in the above points. Therefore, in the next section the researcher is going to present a roadmap in the form of a prescriptive framework.

3. Fuzzy Logic based Reliability Quantification Framework

After realizing the need and significance of the framework for quantifying reliability as discussed in the previous section, this section presents an integrated and prescriptive Fuzzy Logic based Software Reliability Quantification Framework (^{FL}SRQF). The proposed framework, depicted pictorially in Fig. 1, has been structured in a way that it could be easily implemented by industry personnel as well as researchers. The reliability quantification process, prescribed in the framework, is comprised of eight phases namely Conceptualization, Identification, Association, Quantification, Corroboration, Analysis, Assessment and Amendment and Packaging. All these phases are comprehensively described as follows:

3.1 Conceptualization

It is the primary step to device a comprehensive solution for an important problem. Consequently more and more brainstorming is required in this phase. Significance of this step lies in the fact that it provides the basis for sprouting preliminary set of specifications to succeeding steps of development.

Assess Need and Significance: Recognizing software reliability as a critical quality factor, its timely prediction is extremely important. The proposed framework will provide a road map for quantifying reliability in early stage of development that will help not only researchers, but also industry personnel.

Explore Advantage at Early Stage: In general an accurate estimate of reliability can be obtained through software reliability models only in the later phases of software development like testing[29]. Predicting the software reliability early would be useful for software designers since it provides vital information to take decision on design and resource allocation and thereby facilitates efficient and effective development process towards developing a reliable product[30]. Therefore, it is reasonable to develop models that more accurately arrest the faults as early as possible, before they propagate undetected to later stages and cause severe and unrecoverable damage.

Assess the Contribution of Fuzzy Logic: Techniques based on fuzzy set theory have been emerging as robust optimization techniques that can decipher highly complex, nonlinear, correlated and discontinuous problems[31]. As most of the early stage software metrics are not very comprehensible, based on expert's opinion and involve high and complex dependencies among themselves. Therefore fuzzy logic inference systems have found useful in capturing and processing subjective information in terms of early stage software metrics.

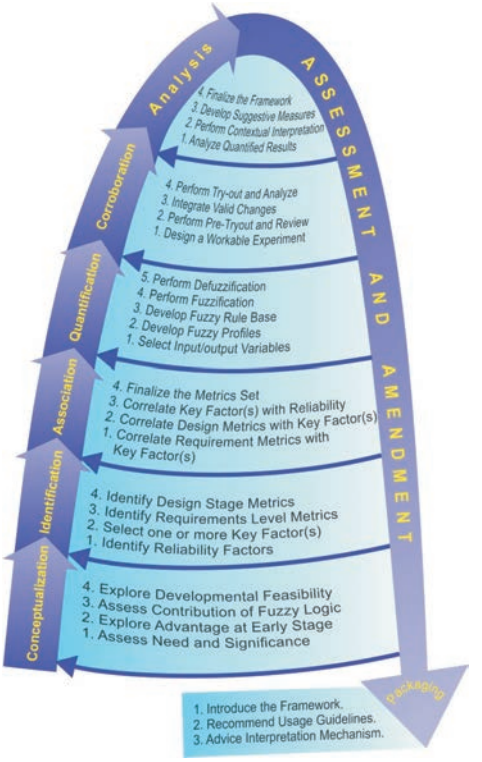

Fig. 1. Fuzzy Logic based Software Reliability Quantification Framework (^{FL}SRQF).

Explore Developmental Feasibility: It is evident from the review of the literature that no such fuzzy logic based framework exists that quantified reliability of a software on the basis of requirement and design measures. This fact further strengthens its significance as well as developmental feasibility.

3.2 Identification

In order to reach an appreciable solution, it is needed to visualize the actual problem intelligibly, and identify the factors that are related directly or indirectly to the problem as well as its solution[32]. There is no doubt, that quantified reliability will not have significant value if its underlying factors are not identified appropriately.

Identify Reliability Factors: Conservatively, most of the researchers had identified timing and the failure rate as the factors that affect reliability, but focusing only on these two is not enough[33]. Therefore this sub-section of the framework, suggest the researcher to explore other factors also, those impact reliability more significantly.

Select one or more key Factors: Although literature has been highlighting a variety of factors those may affect software reliability either positively or negatively. Considering all these factors simultaneously in a study would not be feasible, therefore framework advices to concentrate on few of them.

Identify Requirements and Design Level Metrics: As the motive behind the development of this framework is to quantifying software reliability early in the development life cycle, therefore after selecting the key factor(s), the next step is to identify those requirements and design constructs that may affect the key factor(s) identified in the previous section.

3.3 Association

It is true that the quantified reliability value will not have any significance until the underlying components don't have proper and justified relationship with each other. Therefore, the objective of this phase in the proposed framework is to align all the components together by justifying their role in the framework. Further this phase rationalize the association among the various artifact identified in the previous phase.

Correlate Requirements and Design Metrics with key Factor(s): As the proposed framework considering the requirements and design metrics, as the basic building blocks, for quantifying reliability, therefore before finalizing the metrics it is expected to justify how these are impacting the selected key factors positively or negatively.

Correlate key Factor(s) with Reliability: After justifying the association of identified requirements and design stage metrics with key factor(s), the next concern is to rationalize how these factors are related with Reliability of the software. So, that correlation of identified metrics with software reliability could be justified.

Finalize the Metrics Set: After ensuring the above two correlations this sub section finalizes the requirements and design metrics to be used as input and output variables in the fuzzy inference process to quantifying reliability.

3.4 Quantification

As the focus of this framework is to quantify the reliability with an early stage perspective, and most of the early stage metrics are subjective in nature, therefore, to deal with such lack of objective data and uncertainty this framework has focuses on the fuzzy inference system. The sub tasks for this phase are described as follows:

Select Input and Output Variables: Identification of suitable input (independent) and output (dependent) variables plays a significant role in the development of the Fuzzy Inference System. This step will use the metrics finalized in the last sub-section of the association phase as input and output variables during fuzzy inference process.

Develop Fuzzy Profiles: Developing fuzzy profiles of identified input and output variables is one of critical step to integrate human knowledge with engineering systems[34]. Therefore for implementing fuzzification membership functions are derived for identified variables and subsequently represent them with appropriate linguistic variables like very low (VL), low (L), medium (M), high (H), and very high (VH).

Develop Fuzzy Rule Base: The next step after defining the fuzzy profiles is to specify fuzzy rules. These rules are specified as IF-THEN conditional statement. The quality of these rules is very significant, so that the fuzzy system can imitate the conclusions close of an actual expert[35].

Perform Fuzzification: Generally, in real world scenario, values of most of the variable are crisp in nature. Therefore, to get the desired output through fuzzy reasoning it is needed to transform these crisp values into fuzzy. The process of mapping classical crisp set values into equivalent fuzzy set values is fuzzification[36].

Perform Defuzzification: The output of the fuzzification process is a linguistic variable and most of the applications required crisp output. Therefore, a process is needed that convert these fuzzy conclusion into crisp values, such process is referred as defuzzification. There exists a variety of defuzzification methods like Mean of maximum, Bisector, Largest of maximum, Center of Gravity and Smallest of maximum[36].

3.5 Corroboration

The primary question for any newly developed model is its validity. As pointed out in the second section of this paper that validity was a major concern in earlier models, therefore to fill this gap proposed framework emphasizes on quality validation. It has suggested that, while implementing framework, the reliability assessment model should be corroborated theoretically as well as empirically through pre-tryout followed by the final try-out.

3.6 Analysis

Quantifying and validating the software reliability is not sufficient, until it should be accompanied by valuable suggestions. The following sub sections describing that developing suggestive measures along with the guidelines for improving the reliability is a key task of this phase of the framework.

Analyze Quantified Reliability and Metrics: After successfully validating the reliability model, this is the phase of the framework where obtained quantified values will be assessed and analyzed collectively as well as individually to know how different early stage constructs influence the reliability separately and/or jointly.

Perform Contextual Interpretation: Different contextual interpretations and facts, resulted from the analysis of quantified values should also be discussed analytically.

Develop Suggestive Measures: Developing the suitable reliability improvement guidelines are the preventive measures that are advised to be taken in advance. These guidelines will assist to regulate the values of the requirements and design metrics, and improve the reliability of the developing software.

Finalize the Framework: The proposed framework would be finalized, after incorporating the appropriate suggestive revisions, came out of the above quantitative analysis, to achieve better level of the reliability. These revisions will definitely proved to be significant in making the finally delivered software more reliable.

3.7 Assessment and amendment

The motivation behind the inclusion of this phase is to make the framework more flexible, progressive and implementable. This phase provides the opportunity to assess the current status and perform the needed amendment (if any) in any of the earlier phase. These revisions followed by assessments facilitate to improve the reliability quantification process proposed in the framework.

3.8 Packaging

Finally comes the conclusive phase of this Fuzzy Logic based Reliability Quantification Framework. It is the phase where the developed model is prepared along with the desirable accessories those makes this reliability quantification model a ready to use artifact.

4. Framework's Key Features

After describing all the phases of the framework along with their sub-sections, this part of the paper lists some of the salient features of the framework as follows:

- The framework is quite prescriptive in nature, and will definitely facilitate industry professionals as well as researchers to quantify software reliability in the early stage of development, and subsequently decrease the probability of software's unreliability.
- Based on the analysis of quantified values the framework assists developers by providing them an opportunity, to improve requirements and design related internal characteristics ahead of writing the final code.
- Consideration of the requirements phase along with the design provides this framework an edge over other frameworks or approaches those are based on only design phase, because ignoring or overlooking requirements deficiencies and only concentrating on making the design constructs superior will not seems good enough.
- In order to overcome the limitations of subjective values of requirements metrics, the framework has utilized the strength of fuzzy inference process in its quantification phase.
- To ensure the validity of the developed model, the validation phase suggests a systematic methodology through pre-tryout followed by integrating valid Changes and then to perform final try-out.
- The 'assessment and amendment' phase of the framework further strengthens it practicality as well as viability by keeping the doors of improvement open for any of the earlier phases.
- In most of the cases, developed models only provide quantitative values but neither provides suggestions on how to make improvement, nor the precautions on how to avoid abnormalities. Therefore, to fill this gap framework recommends to provide needed suggestive measures based on the results and contextual interpretations.
- Apart from the above, reassessment of previously developed or underdevelopment reliability quantification models could be done as per the guidance of the proposed framework.
- Beside this, as far as further research is concern, the framework may open fresh avenues for the researchers, doing research on reliability estimation.

5. Conclusions

The study has highlighted the weaknesses of earlier software reliability prediction efforts, and subsequently proposed a structured framework that may overcome the inadequacies of earlier studies and quantifies the reliability, on the basis of the requirement and design phase measures, before the coding starts. Salient characteristics of the framework have also listed just before this section. As far as implementation of the framework is concern, it is in progress and will emerge as future work.

References

[1] M. R. Lyu, Software Reliability Engineering: A Road Map, *Future of Software Engineering*, pp. 153–170, (2007).
[2] S. R. Dalal, M. R. Lyu and C. L. Mallows, *Software Reliability*, John Wiley & Sons, (2014).
[3] S. W. A. Rizvi, V. K. Singh and R. A. Khan, The State of the Art in Software Reliability Prediction: Software Metrics and Fuzzy Logic Perspective, *Advances in Intelligent Systems and Computing*, Springer, vol. 433, pp. 629–637, (2016).
[4] K. S. Kumar, Early Software Reliability and Quality Prediction, Ph.D. Thesis, IIT Kharagpur, Kharagpur, India, (2009).
[5] E. E. Ogheneovo, Software Dysfunction: Why Do Software Fail?. *Journal of Computer and Communications*, vol. 2, pp. 25–35, (2014).
[6] D. K. Yadav, S. K. Chaturvedi and R. B. Misra, Early Software Defects Prediction using Fuzzy Logic, *International Journal of Performability Engineering*, vol. 8(4), pp. 399–408, (2012).
[7] H. B. Yadav and D. K. Yadav, Early Software Reliability Analysis using Reliability Relevant Software Metrics, *International Journal of System Assurance Engineering and Management*, pp. 1–12, (2014).
[8] S. Duraisamy, Software Quality Assessment in Object Oriented Design Ph.D. thesis, Alagappa University, India, (2008).
[9] Y. Jiang, B. Cukic and T. Menzies, Fault Prediction using Early Lifecycle Data, In *(ISSRE-07) Proceeding of* 18th *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pp. 237–246, (2007).
[10] C. Catal, Software Fault Prediction:A Literature Review and Current Trends. *Expert System with Applications*, vol. 38(4), pp. 4626–4636, (2011).
[11] C. Catal and B. Diri, A Systematic Review of Software Fault Predictions Studies, *Expert System with Applications*, vol. 36(4), pp. 7346–7354, (2009).
[12] D. Radjenovic, M. Hericko, R. Torkar and A. Zivkovic, Software Fault Prediction Metrics: A Systematic Literature Review, *Information and Software Technology*, vol. 55(8), pp. 1397–1418, (2013).
[13] O. Mizuno and H. Hata, Yet another Metric for Predicting Fault-Prone Modules, *Advances in Software Engineering Communications in Computer and Information Science*, Springer, vol. 59, pp. 296–304, (2009).
[14] P. He, B. Li, X. Liu, J. Chen and Y. Ma, An Empirical Study on Software Defect Prediction with a Simplified Metric Set, *Information and Software Technology*, vol. 59, pp. 170–190, (2015).
[15] Y. Maa, S. Zhua, K. Qin and G. Luo, Combining the Requirement Information for Software Defect Estimation in Design Time, *Information Processing Letters*, vol. 114(9), pp. 469–474, (2014).
[16] A. Okutan and O. T. Yildiz, Software Defect Prediction using Bayesian Networks, *Empirical Software Engineering*, vol. 19(1), pp. 154–181, (2014).
[17] A. K. Pandey and N. K. Goyal, *Early Software Reliability Prediction*, Springer, India (2013).
[18] O. Georgieva and A. Dimov, Software Reliability Assessment via Fuzzy Logic Model, In *(ICCST-11) Proceedings of the 12th International Conference on Computer Systems and Technologies*, pp. 653–658, (2011).
[19] A. K. Pandey and N. K. Goyal, Predicting Fault-Prone Software Module Using Data Mining Technique and Fuzzy Logic, *International Journal of Computer and Communication Technology*, vol. 2(3), pp. 56–63, (2010).
[20] S. Aljahdali, Development of Software Reliability Growth Models for Industrial Applications Using Fuzzy Logic, *Journal of Computer Science*, vol. 7(10), pp. 1574–1580, (2011).
[21] S. K. Khalsa, A Fuzzified Approach for the Prediction of Fault Proneness and Defect Density, In *Proceedings of World Congress on Eng.*, vol. 1, pp. 218–223, (2009).
[22] D. Yuan and C. Zhang, Evaluation Strategy for Software Reliability based on ANFIS, In *(ICECC-11) Proceedings of the IEEE International Conference on Electronics and Communications and Control*, pp. 3738–3741, (2011).
[23] G. P. Jaiswal and R. N. Giri, A Fuzzy Inference Model for Reliability Estimation of Component Based Software System, *International Journal of Computer Science and Technology*, vol. 3(3), pp. 177–182, (2015).
[24] H. B. Yadav and D. K. 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).
[25] K. Anil and D. Namrata, Reliability Estimation of Object-oriented Software: Design Phase Perspective, *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4(3), pp. 573–577, (2015).
[26] A. Yadav and R. A. Khan, Reliability Quantification of Object-Oriented Design: Complexity Perspective, In *(ICCSEA 2012), Proceedings of the* 2nd *International Conference on Computer Science, Engineering and Applications*, New Delhi, India; vol. 1, pp. 577–585, (2012).
[27] W. Kong, Towards a Formal and Scalable Approach for Quantifying Software Reliability at Early Development Stages, Ph.D. Thesis, University of Maryland, (2009).
[28] A. Hooshmand and A. Isazadeh, Software Reliability Assessment Based on a Formal Requirements Specification, In *Proceedings of the Conference on Human System Interactions*, Publisher IEEE Krakow, Poland, pp. 311–316, (2008).

[29] S. Mohanta, G. Vinod, A. Ghosh and R. Mall, An Approach for Early Prediction of Software Reliability, *ACM SIGSOFT Software Engineering Notes*, vol. 35(6), pp. 1–9, (2010).
[30] S. Mohanta, G. Vinod and R. Mall, A Technique for Early Prediction of Software Reliability based on Design Metrics, *International Journal of System Assurance Engineering and Management*, vol. 2(4), pp. 261–281, (2011).
[31] M. R. Lyu and X. Cai, *Fault-Tolerant Software*, Encyclopedia on Computer Science and Engineering, Benjamin Wah (ed.), Wiley, (2007).
[32] S. Aljahdali and N. C. Debnath, Improved Software Reliability Prediction through Fuzzy Logic Modeling, In *(ICIASSE-13) Proceedings of the 13th International Conference on Intelligent and Adaptive Systems and Software Engineering*, Nice, France, pp. 17–21, (2004).
[33] S. W. A. Rizvi, V. K. Singh and R. A. Khan, Revisiting Software Reliability Engineering with Fuzzy Techniques, In *(IndiaCom–2016) Proceedings of the Third IEEE International Conference on Computing for Sustainable Global Development*, Published by IEEExplore, New Delhi, India, 16–18 March (2016).
[34] C. Kai-Yuan, System Failure Engineering and Fuzzy Methodology: An Introductory Overview, *Fuzzy Sets and Systems*, vol. 83(2), pp. 113–133, (1996).
[35] O. P. Yadav, N. Singh, R. B. Chinnam and P. S. Goel, A Fuzzy Logic based approach to Reliability Improvement during Product Development, *Reliability Engineering and System Safety*, vol. 80, pp. 63–74, (2003).
[36] T. J. Ross, Fuzzy Logic with Engineering Applications, Third Edition, John Wiley and Sons, (2010).