A Software Reliability Evaluation Based on Key Factors in Software Life Cycle

Dileep Sadhankar[1] Dr. Ashish Sasankar[2]
[1,2]Department of Computer Science & Engineering
[1,2]G.H. Raisoni Institute of Information Technology, Nagpur, India

Abstract— Evaluation of Software reliability plays significant role in the development of software, but the conventional software evaluation method generally aims at evaluation by use of failure data which is gained only after testing or handled in the last phase of the software life cycle. So people expect to get each stage's information about the software's reliability which is taken as the reference to guide the software's design, analysis and testing and so on. A software reliability evaluation method is put forward in this paper, which focuses on lots of information correlative with reliability during the entire software life cycle. At last an application is put forward to demonstrate the viability of this method.

Key words: Software Reliability, Measurement Factor

I. Introduction

Software reliability evaluation is important part of normal system reliability tests and trials, to minimize costs and to exercise the software in the system environment. This process also ensures information taken as the reference or accordance to guide the software's design, analysis and testing and so on. Lastly it will provide the quantitative estimation outcome for the issued software product.

In recent years, software reliability evaluation based on number of factors like number of defect found, failure recovery rate etc., as the main means of software reliability estimation. Numbers of software reliability growth models have been proposed [1-5]. But with the inadequacy of not very good assessment quality, many new models and technique were proposed to effectively progress the reliability evaluation performance, such as Neural-Network-based model presented by N.Karunanithi[6], chaos deduce model[7], Bayes networks model[8], fuzzy theory model[9] and so on. Newer technologies are also proposed, such as the failure data trend analysis and prediction quality improvement [10-11]. Based on the statistical theory, David et al. [12-14] proposed many software reliability assessment methods which recognized the sampling theory for software reliability evaluation. With the deficiency of only applied in the delayed phase of the software life cycle, such as testing and maintenance process, its application is mired. Whether it can be used in the early phase of software development becomes an important question. Li et al [15-17] provided the software reliability broad evaluation method from the outlook of system theory. IEEE standard 982.1 confirmed that different factors related to the product, process and resource of software in the complete software life cycle will have big influence on software reliability [18]. Reference [19] presented the findings of empirical research from 13 firms involved in software development to identify the factors that may affect software reliability. Some software engineering professional want to use frequently used software metrics to forecast software reliability in a straight way, so whether these can give new ideas for software reliability estimation?

II. Analysis of Software Reliability Evaluation Factors

Reference [19] presents the findings of empirical research from 13 companies involved in software development including AT&T, Bell Core, Chrysler and MCI International to identify 32 factors that may affect software reliability. These factors are analyzed and ranked in terms of their impact on software reliability. Based on [4] and [17], this paper presents 28 regular factors as the objectives to be further studied, which is shown in Table 1, also these factors are classified into product factors, process factors and resource factors.

1) Product factors: Product factors are software attributes which have something to do with software size, document and structure. It is a static constraint which can be collected from software design documents etc.
2) Resource factors: Resources factors can be divided into: human, reusable software component, software and hardware environment.
3) Process factors: Process factors refer attributes and behaviors in every phase of software development.

III. Evaluation Method

Here the software reliability evaluation method refers distinguished hardware reliability demonstration method [20,21], whose principle is to use factors' information to give an assessment for reliability, then according to grading scores to determine the evaluation value. The steps are given as follows: find out factors used for auditing; list the detailed contents required to audit; design professional grade table, collect information of factors and choose software specialist; arrange experts to audit and grade according to information and contents; combine the experts' grading results to obtain the integrated evaluation results.

Thus software reliability evaluation is a process of auditing and demonstration, which not only presents evaluation results but also finds the hidden defects in the process of software development and testing to guide and improve the process accordingly.

A. Determine Factors for Auditing

Factors listed in Table 1 can be considered as objectives for auditing. Before assessment, factors can be selected as the auditing set according to the actual requirement. For example, in the early stage of development, factors affect software reliability can be selected just involved in that stage.

B. Determine Contents To Be Checked

Contents to be tested are those requirements for factors involved in software design, development and testing process and characteristics of software, e.g. software reliability design method as fault-tolerance, fault-avoidance and control of software complexity. The contents should be simple and correct, distinguishing and casing all characteristics. Some examples are shown in Table 2.

C. Devise Grading Table for Expert Auditing

Based on hardware reliability qualitative demonstration method [20], the following four type grades are given as follows:

1) Excellent:

The work assigned has been done very well and fulfilled the requirement entirely even surpass the demand to some extent, which is equal to 90-100;

2) Good:

The work assigned has been done well and fulfilled the requirement apart from some mistakes with less effect on software reliability and could be corrected easily, which is equal to 75-90;

3) Normal:

The work assigned has been done and fulfilled the requirement for the most part except a lot of serious mistakes are made affecting software reliability to some extent and have to to be corrected with lots of efforts, which is equal to 60-75;

4) Bad:

The work assigned has been done poorly and not fulfilled the requirement with a lot of serious mistakes affecting software reliability rigorously, which have to be done over again to avoid more severe losses, whose score is below 60. Expert auditing grading table can be designed as Table 2.

D. Auditing and Grading:

1) Hypothesis and Notation:

Suppose m is the number of factors and n is the number of experts. E_i is the i^th considered factor, where $i = 1 \sim m$. Let p_i be the number of contents under E_i , S_{ij} be the j^th content of factor E_i , therefore $j = 1 \sim p$ .Suppose every expert has the same significant impact, then let weight for E_i is w_i and weight for every expert is 1/n. Let u_{ijk} *be* the score of S_{ij} given by E_i will is used for evaluation and based on software itself and all factors' impact; every factor is assigned a highest score as the basis to integrated marks. Let A_i be the basic mark of E_i where $\sum_{i=1}^{m} A_i = 100$. The simplest method of ascertaining basic score is in terms of weight as follows

$$A_i = W_i \times 100 \qquad (1.1)$$

Take E_i for example, expert *k* gives a mark u_{ijk} after careful auditing for every content, then expert *k* gives a total mark for E_i, i.e. U_{ik} is ：

$$U_{ik} = A_i \times \frac{\sum_{j=1}^{Pi} u_{ijk}}{p_i \times 100} \qquad (1.2)$$

By Eq. 2, one can get the total mark U_i for E_i by all experts such that

$$U_i = \sum_{k=1}^{n} \frac{1}{n} U_{ik} \qquad (1.3)$$

By Eq. 3, one can get the final grading score U_i for factor E_i

Software reliability incorporated evaluation outcome

The integrated evaluation result of software reliability R is as follows:

$$R = \sum_{i=1}^{m} U_i \qquad (1.4)$$

04-grading method is used to give the weight of factor E_i. The principle of 04-grading method is given as follows: 1)Expert grades independently without conversation amid each other; 2)When the importance of two factors matched, the comparing method is adopt and the following three steps can be used such that: i) Between two factors, the more important one gets the score of 4 and the other gets 0; ii)Between two factors, the relatively important one gets the score of 3 and the other gets 1; iii)If two factors are of the same importance, they all get 2; 3) The two factors cannot both get 4 or cannot both get 0. Examples are given as Table 4.

IV. APPLICATION

Here gives a simple application.

1) Determine factors for auditing For convenience, take 7 factors from Table 1 noted as A, B, C, D, E, F, G for example.
2) Determine contents to be checked：The contents to be checked are shown as Table 2.
3) Design grading table for expert auditing：The table is shown as Table 3.
4) Auditing factors：Suppose seven software experts are invited to audit 7 factors and the weights are calculated as Table 4. E.g. expert A gives an auditing result as which is shown in Table 5.Take factor A (software complexity) for example, the score from expert A is computed as follows, then final result is shown as Table 6.

$$U_{A1} = 20 \times \frac{90 + 85 + 80 + 80 + 90}{5} \times 100 = 17.1$$

Result of integrated reliability evaluation：In terms of the results in Table 6, by Eq.4, then

$$R = \sum_{n=1}^{m} U_i = 88.3$$

V. CONCLUSION

A software reliability assessment method based on influencing factors was kept forward by means of expert evaluation and auditing, in terms of contents to be examined, the incorporated evaluation outcome was accomplished. From the practice of application, not only assessment outcome can be achieved, but also guidance can be kept forward to get better the process of software development and testing and so on. More appropriate factors, association between classification of factors and weights, contents to be considered should be further studied in future work.

Type	No	Factor Name
Product factors	1	Software complexity
	2	Percentage of reused code and modules
	3	User's quality objective
	4	Software programming languages
	5	Nature of defects and failures
	6	programmer's skill

Resource factors	7	Development team size
	8	Programmer organization
	9	User's skill
	10	Testing tools
	11	Testing environment
	12	Hardware resource
	13	Software development environment
	14	Testing resource allocation
Process factors	15	Testing methodologies
	16	Testing coverage
	17	Test case
	18	Fault detecting and removing process
	19	Test efforts
	20	Documentation
	21	Software development design methodologies and technology
	22	Development management
	23	Work standard
	24	Relationship of detailed design to requirements
	25	Frequency of program specification and requirements change
	26	Difficulty of programming
	27	Whole schedule
	28	Programming effort

Table 1: List Of Ordinary Factors

Factor	**Contents to be checked**
software complexity	Does software have a good architecture system
	percentage of defects caused by source code size
	software McCabe's complexity
	software functionality
	code readability

Table 2: Contents Of Factors To Be Checked

Factor	**Contents to be checked**	**Grading ranks**				**Grading results**
		excellent	**good**	**normal**	**bad**	
software complexity	Does software have a good architecture system	√				90
	Percentage of defects caused by source code size		√			80
	Software McCabe's complexity		√			80
	Software functionality		√			85
	Code readability	√				90

Table 3: Grading Tables For Factors Contents

Factor	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Grading score**	**Weights**
A		4	4	3	3	2	1	17	0.20
B	0		3	2	2	0	0	7	0.08
C	0	1		1	2	0	0	4	0.05
D	1	2	3		2	1	0	9	0.11
E	1	2	2	2		1	0	8	0.10
F	2	4	4	3	3		1	17	0.20
G	3	4	4	4	4	3		22	0.26
Amount								84	1

Table 4: Statistical Results By 04-Grading Method

Factor	**Contents to be checked**	**Grading ranks**				**Grading results**
		Excellent	**Good**	**Normal**	**Bad**	

software complexity	Does software have a good architecture system	√				90
	percentage of defects caused by source code size		√			80
	software McCabe's complexity		√			80
	software functionality		√			85
	code readability	√				90

Table 5: Grading Table Of Expert A

Factor	A	B	C	D	E	F	G	Grading score	Weights
A	17.1	17.8	18.8	18.1	18.7	17.6	15.9	17	20
B	6.5	6	6.3	6.8	7.1	5.7	6.7	7	8
C	4.3	3.9	4.1	4.4	3.8	4.1	4.3	4	5
D	9.2	9.9	9.8	9.4	10.1	9.2	10	9	11
E	9	9.1	8.8	8.6	9.5	9	8.8	8	10
F	16.4	18.2	18.9	17.3	19.1	18.4	17.4	17	20
G	21.5	23.6	23.2	24.6	22.7	23.2	24.8	22	26
Amount								88.3	100

Table 6: Grading Results Of All Experts

VI. REFERENCES

[1] R. Z. Xu, M. Xie and R. J. Zheng. Software Reliability Model and Application. Beijing: Tsinghua University Press, 1994(in Chinese).

[2] S. Yamada Osaki. Software reliability growth modeling: Models and applications. IEEE Trans. Software Engineering SE-11(12), 1431-1437, 1985.

[3] J. D. Musa, A. Iannino and K. Okumoto. Software Reliability: Measurement, Prediction, Application. Mc-Graw-Hill, New York, 1987.

[4] H. Pham. Software Reliability. Springer-Verlag, Singapore, 2000.

[5] S. Yamada. Software reliability models In Stochastic Models in Reliability and Maintenance. Springer-Verlag, Berlin, 2002, 253-280.

[6] N. Karunanithi, D.Whitley and Y. K. Malaiya. Prediction of Reliability Using Connectionist Models. IEEE Transactions on Software Engineering, 18(7), July 1992, 563 574.

[7] F. Z. Zou. Software reliability theoretical analysis and computation. Ph.D thesis of Wuhan University of Hydraulic and Electric Engineering, 1999(in Chinese).

[8] C. G. Bai. Software reliability research based on Bayes networks. Ph.D thesis of Zhejiang Univers, 1999(in Chinese).

[9] J. H. Guo, X.Z.Yang and H.W.Liu. Software Reliability Nonlinear Modeling and Its Fuzzy Evaluation. 4th Wseas Int. Conf. on Non-linear analysis, non-linear systems and chaos, Sofia, Bulgaria, October 27-29, 2005,49-54.

[10] B. Littlewood. Forecasting Software Reliability. CSR Technical Report, 1989.

[11] S. Brocklehurst, P. Y Chan and B. Littlewood. Recalibrating Software Reliability Models. IEEE Trans.on Software Engineering, SE-16(4), 458-470, Apr.1990.

[12] L. P. David, A. John, S. P. Kwan. Evaluation of Safety critical Software. Communication of ACM , 1990(6).

[13] D. L. Parnas, G.J.K. Asmis, J. Madey. Assessment of Safety-critical Software in Nuclear power plants. Nuclear safety, 1991(2).

[14] W. H. Howden. Good enough versus High Assurance Software Testing and Analysis Methods. In Proc. of IEEE HASE, 1998.

[15] H. F. Li, M. Y. Lu, Z. X. Wang and Z. Li. Framework for software reliability comprehensive evaluation based on grey system theory, Journal of Beijing University of aeronautics and astronautics ,34(11), 2008(in Chinese).

[16] B. Liu, M.Y. Lu and L. RUAN. Early Software Reliability Prediction: an Approach Based on Fuzzy Neural Network Journal of Beijing University of aeronautics and astronautics, 2001, 27(2) (in Chinese).

[17] T. J. Wang and M. Li. A Fuzzy Comprehensive Evaluation Model for Software Reliability. Computer engineering and applications, 2002, 38(20) (in Chinese).

[18] IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software, IEEE, 1988.

[19] X. M. Zhang, Hoang Pham. An analysis of factors affecting software reliability. The journal of systems and software, 2000, 50, 43-56.

[20] RMS qualitative demonstration method. National defense technology report, 2004(in Chinese).

[21] Qiuying Li and Haifeng Li, International Conference on Computer and Software Modeling,2011

