Software Reliability Measurement Base on Failure Intensity

By Bambang Krismono Triwijoyo

WORD COUNT 3294
TIME SUBMITTED 09-AUG-2022 08:51PM
PAPER ID 89375733

2017 3rd International Conference on Science in Information Technology (ICSITech)

Software Reliability Measurement Base on Failure Intensity

Bambang Krismono Triwijoyo, Ford Lumban Gaol, Benfano Soewito, Harco Leslie Hendric Spits Warnars
Binus Graduate Program Bina Nusantara University, Jl. Kebon Jeruk No27
Jakarta, Indonesia
STMIK Bumigora Mataram Indonesia
bambang.triwijoyo@binus.ac.id, fgaol@binus.edu, bsoewito@binus.edu, shendric@binus.edu

***Abstract*—Software reliability is an important factor in software quality measurement, which is measured by the probability of an error-free software within the operating period within a given time period and environment. Software reliability measurements are performed at every stage of software development process to evaluate whether the software reliability requirements has been fulfilled. In this paper proposed the new methods to measuring the software reliability based on categorize faults. We use J.D Musa-III failure datasets are divided into 5 modules to measure software reliability using our method. Base on J.D Musa-III datasets we got the value of reliability is 0.7416 or 74%. The software reliability can be measured using this method and the future work is to categorize the failure of the software based on the source of its failure.**

Keywords—software reliability; matrix; measured; fault

I. INTRODUCTION

The most important thing in the effectiveness of software project management is the accuracy in estimating software development efforts [1]. Flexibility of service, end-user personalities and shorter software development time is a major challenge in software project management [2]. Software failure is critical in software development, where the failure is independent of the hardware specifications used for execution. Software failure is caused by the occurrence of design errors that occur when given input into code during execution [3]. Software reliability is a measure of software quality and can provide feedback to the software development team for evaluation tools. Software reliability model is generally divided into two categories, namely black-box model and white-box model [4]. The distinguished between the two models is in estimating software reliability, the white-box model considers the structure of the software while the black-box model does not consider the software structure [5].

One of the most common things in the software development process is the number of design errors or defects known as bugs. Software errors occur when given certain inputs resulting in software behavior deviations from an estimated failure rate. Software errors are assumed to be completely fixed if they can be detected through the testing process. If a new error does not arise during the fixing process, then the reliability of the software can improved. Estimates of software reliability can be performed using statistical models for software failure identification by counting the number of failures per specific time period and the length of time between failures. In applications that are very critical of the safety level, then software reliability becomes very important. It is important to determine the time and resources used in software testing, to achieve the expected level of reliability. Models depicting the growth of software reliability have been proposed by [3].

II. SOFTWARE RELIABILITY GROWTH MODELS (SGRM)

Prediction of software reliability is very important, Statistical models can be used to aid in performing reliability predictions. Initially, the model that can describe the growth rate of reliability is used on hardware, it is then applied to the software. One of the differences between hardware and software is that hardware may be aging while software does not. Limitations in predicting software reliability is made by reference to the production of the software, it is impossible to estimate the consequences of such errors, either the failure or the failure resulting from software period of usage [6-8]. The "bathtub" curve describes the standard behavior in the software. (see Fig. 1, and 2).

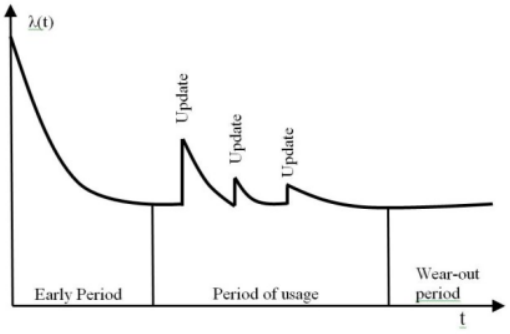

Fig. 1. Bathtub curve of default behavior (software) [3].

The curve has the following 3 periods:

- Early period (early failures),
- Period of usage (failure during usage),
- Wear-out period (late failures).

Early failures

Failure in the early stages of software implementation can be eliminated by the testing process. On the other hand

978-1-5090-5864-8/17/$31.00 ©2017 IEEE

hardware failures may occur in the production process or malfunctions of materials and hardware operations. While the failure in the software occurs due to errors in software programming. The software is an integral part of the hardware, so errors that occur in hardware cause software failure. So transition failures can happen from hardware to software. [6, 9].

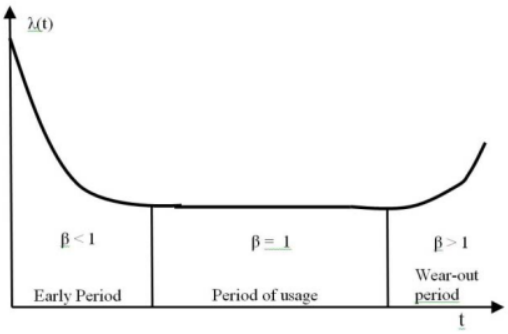

Fig. 2. Bathtub curve of hardware behavior [3].

Failures during usage:

In hardware there is almost no natural use during the implementation phase, so the failure rate almost constant. The cause of failure on pure hardware occurs by chance, whereas software failure rates during the usage stage increase and decrease over time because of the continuous updates during the use of the software.

Late failures:

Hardware failure occurs due to obsolescence but upgrades may be available during its service life. Failures occurs mostly due to the natural aging process as well as signs of fatigue resulting in increased hardware failure rates. In software the fact that the failure rate due to the aging process as well as fatigue does not occur so it remains constant. Some commonly used terms associated with Software Reliability Growth Models (SGRM) are presented in Table I.

TABLE I. TERMINOLOGY COMMON TO SRGMS [6]

Term	Explanation
M(t)	The number of failures experienced by time t.
μ(t)	Average Function for SRGM. This is the expected number of failure times as predicted by the model, where μ (t) = E [M (t)].
λ(t)	The intensity of failure, the representative of the mean, where λ (t) = μ '(t).
Z(Δt/t_{i-1})	Software hazard level, is the probability density probability at t_{i-1} + Δt for noise failure (i-1) in t_{i-1}.
Z(t)	The danger level per error, is the probability of an error, which has not been activated so far, will cause instantaneous failure when enabled. This is assumed to be constant (φ).
N	Initiate the number of errors in the software before testing.

In general the data is supplied to Software Reliability Growth Models (SGRM) are either times between failures {Δt_1, Δt_2, Δt_3, …} or the times at which failure occurred {t_1, t_2, t_3, …}. The model presented here generally assumes the independence between failures [6]. This model implies that the hazard level can be reduced by correcting any errors that occur of the new time between failures by a constant λ > 0. This idea is depicted in Fig. 3.

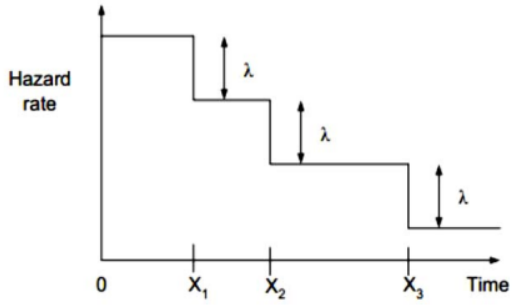

Fig. 3. Hazard rate [10].

III. METHODS AND RESULTS

We implemented the Krini approach [10] and the Chwala algorithm [8] to measure the reliability of the software. Fig. 4, shows an approach for a procedure to predicting software reliability.

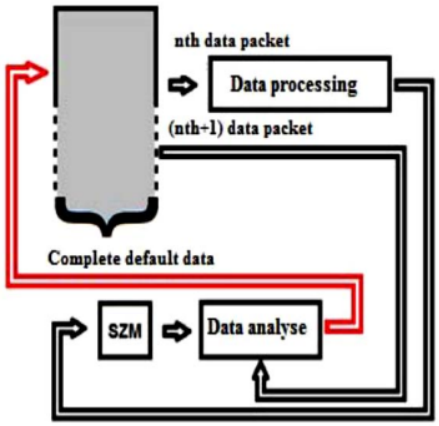

Fig. 4. The approach for a predicting reliability models [10].

Data processing blocks are used as the basis for measuring data values. We recommend that not all available data be evaluated at the same time. After the first data has been analyzed, the data can be compared with the actual second data. Model validity can be determined by analysis by comparing several different models [11-13]. In this model approach the longer errors in the system have not been found then the more dangerous. An unlikely error may be harmless, but it is necessary to use an appropriate distribution model to predict reliability in a practical and realistic way.

This is an improvement of the reliability prediction model if the source of error can be considered (see Fig. 5).

Determination of each source of software reliability error can be done. This is an advantage by doing the appropriate distribution by selecting a software failure group. After that prediction can be done with the help of a single prediction [10].

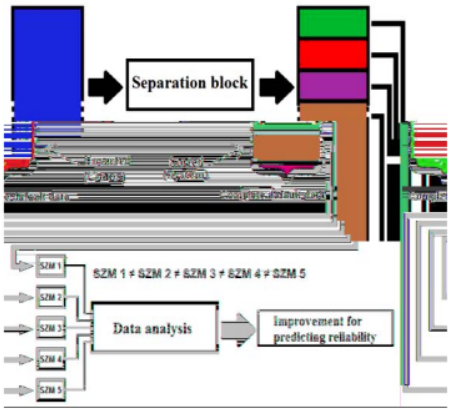

Fig. 5. Improvement for Prediction of reliability [10].

Many different probabilities concerning every failure group may be made. This approach may lead to a more exact prediction of reliability [14-15]. Software reliability for a software system depends on following attributes [16-17].

Type of fault

Software fault itself divided under different categories based on the influence of fault on software system. A fault can be warning, bug or the failure.

Application Type

The type of application affected due to software failure, determine the software failure level. If affected is a game app, then the system criticality level is low, but if the affected due to a software failure is a business application or a real time application, then obviously the criticality level of software failure is high.

Associated Module

The linkage between software modules and software errors is also the reason for identifying software errors. The entire system will be severely affected in the event of a high critical condition in the module associated with the system.

Identification and categorization of errors based on the severity of errors is required in designing reliable software. In doing the first job analysis done is to divide the module related to the software system into several module groups. Initially the errors of each module are identified, then lists the test cases relating to each error in the module. We apply the priority metric approach recommended by Chawla [16]. Table II shows Indicates several types of software failures along with their respective priorities used in this work.

TABLE II. PRIORITY OF FAULT [16]

Module ID	Function	Priority
1.	Graphical Evaluation	Low
2.	Backup Database	High
3.	Data Recovery	High

Algorithm of associated faults

We use algorithms to define software failures related to each module, as recommended by [16] with the following process stages.

Determine the software system associated with the software module.
Determine the failure associated with each software module.
Determine the category of software failure based on the criticality level of software failure.
Specify the priority of each module based on the criticality level of software failure.

```
Reliability:=1;
For i:=1 to Length (Modules)
  {   For j:=1 to Length (Faults)
      {   If (Associated (Module (i), Fault (j)):=True)
          {  K:=Criticality (Fault (j));
             Reliability=Reliability-K;
          }
      }
  }
Return Reliability;
```

To implement the algorithm of associated faults with each module we used J.D Musa-III failure dataset [18] it contains 164 values fault time series in seconds. In this case we divided faults base on categories by randomly into five modules to measure software reliability. Tables III to Table VIII shows the fault data series, it contains Fault Number (FN), Fault Time in seconds during period of usage (FT), Inter Fault Time (IFT) and Total Inter Fault (TIF) where:

$$IFT = | FT_i - FT_{i+1}| \quad (1)$$

$$TIF = \Sigma_{i=1} IFT \quad (2)$$

and Mean Absolute Error (MAE) [19] are given below:

$$MAE = (\Sigma_{i=1} (| FT_i - FT_{i+1}|)) / n \quad (3)$$

TABLE III. FAULT OF MODULE #1

FN	FT	IFT
1	640	0
2	640	2240
3	2880	2770
4	110	21970
5	22080	38574
6	60654	8491
7	52163	39617
8	12546	11762
9	784	9409
10	10193	2352
11	7841	23524
12	31365	7052
13	24313	274577

FN	FT	IFT
14	298890	297610
15	1280	20819
16	22099	2949
17	19150	16539
18	2611	36559
19	39170	16624
20	55794	13162
21	42632	224968
22	267600	180526
23	87074	62532
24	149606	135206
25	14400	20160
26	34560	5040
27	39600	294795
28	334395	38380
29	296015	118620
30	177395	37227
31	214622	58222
32	156400	155760
TIF		**2178036**
MAE		**68063.63**

TABLE IV. FAULT OF MODULE #2

FN	FT	IFT
1	320	1119
2	1439	7561
3	9000	6120
4	2880	2820
5	5700	16100
6	21800	5000
7	26800	86740
8	113540	1403
9	112137	111477
10	660	2040
11	2700	26093
12	28793	26620
13	2173	5090
14	7263	3602
15	10865	6635
16	4230	4230
17	8460	6345
18	14805	2961
19	11844	6483
20	5361	1192
21	6553	54
22	6499	3375
23	3124	48199
24	51323	34313
25	17010	15120
26	1890	3510
27	5400	56913
28	62313	37487
29	24826	1529
30	26355	25992
31	363	13626
32	13989	1069
33	15058	17319
34	32377	9255
35	41632	37472
36	4160	77880
37	82040	68851
38	13189	9763
39	3426	2407
40	5833	5513
TIF		**799278**
MAE		**19981.95**

TABLE V. FAULT OF MODULE #3

FN	FT	IFT
1	10506	166734
2	177240	64247
3	241487	98459
4	143028	130536
5	273564	84173
6	189391	16591
7	172800	151200
8	21600	43200
9	64800	237600
10	302400	449788
11	752188	665788
12	86400	14400
13	100800	81360
14	19440	95760
15	115200	50400
16	64800	61200
17	3600	226800
18	230400	352800
19	583200	324000
20	259200	75600
21	183600	180000
22	3600	140400
23	144000	129600
24	14400	3894
TIF		**3844530**
MAE		**160188.75**

TABLE VI. FAULT OF MODULE #4

FN	FT	IFT
1	166800	156000
2	10800	256200
3	267000	232487
4	34513	26833
5	7680	29987
6	37667	26567
7	11100	176100
8	187200	169200
9	18000	160200
10	178200	34200
11	144000	495200
12	639200	552800
13	86400	201600
14	288000	287680
15	320	57280
16	57600	28800
17	28800	10800
18	18000	70640
19	88640	343360
20	432000	427840
21	4160	960
22	3200	39600
23	42800	800
24	43600	33040
25	10560	104640
26	115200	28800
27	86400	28800
28	57600	28800
29	28800	403200
30	432000	86400
31	345600	230400
32	115200	70706
33	44494	122306

FN	FT	IFT
TIF		4922226
MAE		149158.36

TABLE VII. FAULT OF MODULE #5

FN	FT	IFT
1	86400	23700
2	110100	81300
3	28800	14400
4	43200	14400
5	57600	410400
6	468000	482400
7	950400	550000
8	400400	483400
9	883800	610200
10	273600	158400
11	432000	432000
12	864000	661400
13	202600	800
14	203400	74280
15	277680	172680
16	105000	475080
17	580080	3953880
18	4533960	4101960
19	432000	979200
20	1411200	1238400
21	172800	86400
22	86400	1036800
23	1123200	432000
24	1555200	777600
25	777600	518400
26	1296000	576000
27	1872000	1536400
28	335600	586000
29	921600	625585
30	296015	740785
31	1036800	691200
32	1728000	950400
33	777600	720000
34	57600	40320
35	17280	69120
TIF		24305290
MAE		694436.86

Fig. 6, illustrated the fluctuation of IFT module 1 to module 5. Base on this graph module 1 is have most high IFT fluctuations occurs between fault number 17 and fault number 21.

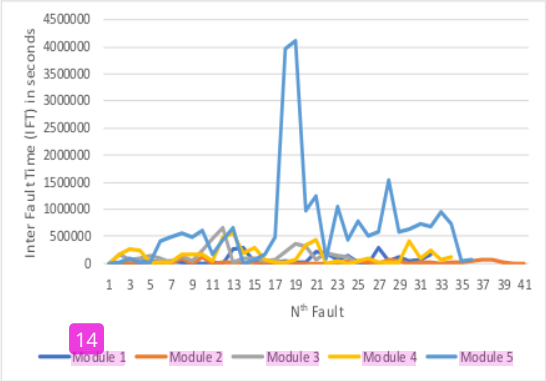

Fig. 6. Graph Inter Fault Time module 1 to module 5.

The next step is determined the priority value of each module in interval value 0 to 1. The priority value is determined by how big the impact of errors on the system software. Table VIII shows the matrix of reliability under the fault based prioritization, in this case the data recovery functions have the high priority (0.3), database backup (0.2) and evaluation of graphical windows have lowest priority (0.1). The total of priority value for all module is should be 1.00. In the actual case the priority value of each function types are determined by the software developer, because they are most know exactly the priority of each type of function.

TABLE VIII. MATRIX OF RELIABILITY BASED ON PRIORITY [16]

Module ID	Type of Function	Prio-rity	Number of fault	MAE	Fault Critically
1	Evaluation of Graphical Window	0.1	32	68063.63	0.0062
2	Database Backup	0.2	40	19981.95	0.0037
3	Data Recovery	0.3	24	160188.75	0.0440
4	Evaluation of Graphical Window	0.1	33	149158.36	0.0137
5	Data Recovery	0.3	35	694436.85	0.1908
				1091829.54	**0.2584**

Finally we calculated the Fault Criticality (FC) of each module is given below:

$$FC = (MAE * TF) / TMAE \quad (4)$$

FC is fault criticality of each module, MAE is mean absolute error of each module, TF is type of function of each module, TMAE is the total of mean absolute error and TFC is the total of fault criticality.

Base on Algorithm to define Reliability of the software from associated faults of each module [16], the value of Reliability (R) of that case is:

$$R = 1 - TFC \quad (5)$$

Then the Reliability of software base on prioritization is 1 – 0.2584 = 0.7416 or 74%, it means that the reliability of software in this case is 74% (in 100% scale) or we can categorize in Good enough.

IV. Conclusion

The software reliability measurement model is one of the key components of software security functions, so research in this area is important to continue to develop better models. We has been proposed a new approach of software reliability matrix models base on fault analysis. We divided the failure data into three groups and five modules, once group for once type of module functions. Base on J.D. Musa-III datasets we got R= 0.7416 or 74%, it means that the reliability of software in this case is 74%. The software reliability can be measured using this matrix. Our future work is categorize software failure based on the source, so expect to apply the appropriate model for each source of failure.

References

[1] Suharjito, S. Nanda, and B. Soewito, "Modeling software effort estimation using hybrid PSO-ANFIS". *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Lombok, 2016, pp. 219-224.

[2] Rahmansyah, Ryan, and F.L. Gaol. "Service oriented architecture governance implementation in a software development project as an enterprise solutions". *Journal of Computer Science* 9, no. 12 (2013): 1638.

[3] J. Börcsök. "Electronic Safety Systems, Hardware Concepts, Models, and Calculation". *Heidelberg*, Germany, 2004.

[4] Pai, and J. Ganesh. "A survey of software reliability models." *arXiv preprint arXiv*:1304.4539 , 2013.

[5] K. Iskandar, F.L. Gaol, B. Soewito, and H.L. Hendric Spits Warnars. "Software size measurement of knowledge management portal with use case point". *The international conference on Computer, Control, Informatics, and its Applications (IC3INA 2016)*, 3-5 Oct 2016, Tangerang, Indonesia.

[6] J.D. Musa. "More Reliable Software Faster and Cheaper". *Software Reliability Engineering*, United States of America, 2004.

[7] N.F. Schneidewind. "Minimizing risk in Applying Metrics on Multiple Projects", *Proc. IEEE Int. Symp*, 1992. pp. 173-179.

[8] J. Börcsök, and O. Krini. "Principle Software Reliability Analysis with different Failure Rate Models". *IEEE Int. Symp. ICAT*, 2009.

[9] B. Lindqvist, and A. Doksum. "Mathematical and Statistical Method in Reliability". *Covent Garden*, London 2003.

[10] J. Krini, and J. Börcsök. "Basic concept for the selection of an appropriate Software Failure Prediction Model". *University of Kassel, Wilhelmshöhe*. 2015

[11] M. Lyu. "IEEE Recommended Practice on Software Reliability". *Standards Committee of the IEEE Reliability Society*, 2008.

[12] X. Zhang, X. Teng, and H.Pham: "Considering Fault Removal Efficiency in Software Reliability Assessment", *IEEE* vol.33, 2003.

[13] N.F. Schneidewind, "Analysis of processes in computer software," *Sigplan Notices*, Vol. 10, 1975. pp.337-346, 1975.

[14] M. Lyu. "Handbook of Software Reliability", *McGraw-Hill*, New York, 1996.

[15] A. Birolini, "Reliability Engineering: Theory and Practice", *Springer Verlag*, Heidelberg, 2007.

[16] G. Chawla, and S.K. Thakur. "A Fault Analysis based Model for Software Reliability Estimation", *International Journal of Recent Technology and Engineering*, 2013. pp.121–123.

[17] A.L. Goel, and K. Okumoto. "Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures", *IEEE Trans. Rel., R-28(3)*, 1979. pp. 206-211.

[18] J.D. Musa, A. Iannino, and K. Okumoto, "Software Reliability: Measurement, Prediction, Application" ,*McGraw-Hill*, 1987.

[19] Ehrlich, K.Willa, A. Iannino, B.S. Prasanna, J.P. Stampfel, and J.R. Wu. "How faults cause software failures: implications for software reliability engineering." *In Software Reliability Engineering, 1991. Proceedings., 1991 International Symposium on*, pp. 233-241. IEEE, 1991.

Software Reliability Measurement Base on Failure Intensity

ORIGINALITY REPORT

14%
SIMILARITY INDEX

PRIMARY SOURCES

1	Jamal Krini, Josef Borcsok. "Contribution to reducing the critical faults in critical Software Systems", 2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT), 2015 Crossref	117 words —	3%
2	Ossmane Krini, Josef Borcsok. "New approach to determine the critical number of failure in software systems", 2011 XXIII International Symposium on Information, Communication and Automation Technologies, 2011 Crossref	78 words —	2%
3	Safrizal, Harco Leslie Hendric Spits Warnars, Ford Lumban Gaol, Edi Abdurachman. "Use case point as software size measurement with study case of Academic Information System", 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), 2017 Crossref	39 words —	1%
4	ieeexplore.ieee.org Internet	36 words —	1%
5	www.cse.cuhk.edu.hk Internet	26 words —	1%
6	ijece.iaescore.com Internet	25 words —	1%

7	eprints.binus.ac.id Internet	24 words — 1%
8	www.studymode.com Internet	21 words — 1%
9	ijcotjournal.org Internet	20 words — 1%
10	scholar.binus.ac.id Internet	19 words — 1%
11	Krini, Jamal, Abderrahim Krini, Ossmane Krini, and Josef Borcsok. "Extended approach to selecting a project-specific reliability growth model", 2016 39th International Convention on Information and Communication Technology Electronics and Microelectronics (MIPRO), 2016. Crossref	16 words — < 1%
12	P.K. Kapur, Hoang Pham, A. Gupta, P.C. Jha. "Software Reliability Assessment with OR Applications", Springer Science and Business Media LLC, 2011 Crossref	15 words — < 1%
13	www.ijsr.net Internet	12 words — < 1%
14	jultika.oulu.fi Internet	11 words — < 1%
15	S. CHATTERJEE, R. B. MISRA, S. S. ALAM. "Prediction of software reliability using an auto regressive process", International Journal of Systems Science, 1997 Crossref	8 words — < 1%
16	pt.scribd.com Internet	

8 words — < 1%

17 Krini, Ossmane, and Josef Borcsok. "New scientific contributions to the prediction of the reliability of critical systems which based on imperfect debugging method and the increase of quality of service", 2012 IX International Symposium on Telecommunications (BIHTEL), 2012.
Crossref

7 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF