Multi Up-gradation Software Reliability Growth Model Considering the Joint Effect of Testing and Operational Phase

Raksha Verma
Department of Mathematics,
Sunrise University,
Alwar-301001, Rajasthan

R. S. Parihar
Department of Mathematics
Sunrise University,
Alwar-301001, Rajasthan

Subhrata Das
Department of Operational Research
University of Delhi, Delhi 110007

ABSTRACT

Software companies are coming with multiple add-ons to survive in the purely competitive environment. Each succeeding up-gradation offers some performance enhancement and distinguishes itself from the past release making it more prone to failures. For developing highly reliable software it is important to understand the manner in which faults might encounter. Majority of researchers have focused on understanding the fault removal phenomenon during testing phase but few have also focused on operational phase. With the aim of catering more realistic scenario for comprehending the fault removal process for successive release, both the faults of new release along with remaining bugs of its preceding release has been considered; wherein it is assumed that remaining faults of previous release can be debugged in its operational phase together with testing phase of newer version. Convolution of probability distribution function has been considered for capturing the effect of faults removed in testing (new release) and operational phase of just previous release. Further, two different cases are formulated depending upon the failure distribution being followed for testing as well as for operational phase. The proposed cases are validated on real data set.

Keywords

Fault Removal Phenomenon (FRP), Multi Up-gradation, Operational Phase, Testing Phase.

1. INTRODUCTION

Many a times it is difficult to be sure about the performance of the software system unless it is assumed that software system can run successfully without any failure which may bring it down. Researchers [7] have defined software reliability as the probability of failure free operation of software for a specified period of time under the stated condition of environment. There exist several tools to measure the reliability of the software, one such approach is the employment of software reliability growth models (SRGMs). Broadly, SRGMs are modeled to have deep insight about the characteristics of how and why does a software fails and also attempt to determine the software reliability in quantitative terms. Many researchers have tried to work in modeling the fault removal phenomenon either through exponential or S-shaped growth pattern. Many categories of models exists in literature capturing varied aspects viz. GO model, Kapur & Garg Model and Yamada model etc. [4], [6], [16]. One of the recent trend seen in the field of software industry is to provide upgraded versions of the software with some added functionalities. Firms do so in order to maintain their competitive edge in the market, satisfy the growing need of consumers and also to be the first in bringing new version with added functionalities.

Up-gradation involves the replacement of existing version with newer version of the same software product but with additional features. Generally, software is upgraded to improve their characteristics but some software up-gradations can be risky by increasing the fault content. Several examples can be quoted in which software up-gradations were hazardous:

- The U.S. federal government opened a new health insurance exchange web site in October 2013, during first few months of operation there were several reported problems arising due to insufficient time given to in-house testing of software [5].
- One of a largest bank of Europe upgraded their software in June 2012, which resulted in breakdown of their website due to which millions of customers were unable to access their own money. It is being believed that this crash occurred because of poor testing [5].
- Some recent issues were reported in July 2011 in Asia regarding the computerized testing and grading system which resulted in incorrect allocation of marks foe thousands of students [5].
- In August 2013, Asian brokerage's securities order system which resulted in more than $3 billion of incorrect trading orders [5].

In recent era, up-gradation has become an indispensable practice in which functionalities or features are added to meet the expectations of users. Sometime firms upgrade in order to survive the competition which might increase their market hold. Also from above example it is quite clear that firms need to upgrade their offerings but when it does not turn hazardous or increase the eventual faults count in the software. Major attention should be given to manner in which faults are identified and removed so eventual reliability can be increased. Sometimes the remaining fault of earlier versions can become active in succeeding version and can be risky as a result it is important to understand the manner in which remaining faults will be tackled. A lot of work has been done in the field of multi generation starting from the very early concept in which the overall reliability was computed based on the effect of faults from all preceding releases of the software [8]. Later it was felt that the count of faults from all preceding releases is not an appropriate measure and its numeric value is not very significant in affecting the reliability of the software [13]. Later on, researchers realised that testing team may not always be able to debug the fault perfectly leading to imperfect scenario which was modeled by Kapur et al. [9] under the consideration of faults from just previous release. Other researchers also worked on incorporating the concept of stochasticity in multi up-gradations [11]. Kapur et al. [10] worked on fault severity modeling i.e. simple and hard faults modeling. Several practitioners realised the importance of multi up-gradations and have worked further on elaborating the effect of imperfect debugging and stochastic differential equation into severity of faults; to model more realistic scenarios [1], 12]. Not only the faults encountered under testing phase were given importance but researchers also explore the fault removal phenomenon both under testing as well as the operational phase [2], [3].

In this paper, emphasis has been given to mathematically model, the faults which are removed in each version of the software keeping in mind that new add-ons are risky and will increase the faults in the software. Also, the remaining faults of just previous release can be removed both in testing phase of its succeeding release and its operational phase through a joint fault removal process. The conjoint effect employs the more realistic context of fault evaluation by taking into consideration both; the testing phase faults (for the new release) and the reported faults from operational phase.

Rest of the paper is structured as follows: Section 2 describes the mathematical framework for proposed methodology. Model validation and Conclusion are supplemented in Section 3 & 4. Further the list of references is given at the end.

2. MODELING FRAMEWORK

In general sense, each upgraded version of the software is assumed to be accompanied with new applications in order to attract a large pool of users which assist the firm in attaining competitive edge over other competing software. It is a general perception that adding new functionalities might increase fault content in the software. Thus there are two set of faults one due to add-ons and other comprise of the remaining faults of preceding release. Now it is a major concern about how these faults will be removed. There are several chances that some of the bugs in the previous release are removed directly by the testing team of existing release and some are removed in the operational phase of succeeding release. This paper focuses upon modeling the fault removal phenomenon for multiple releases of the software by inculcating the removal process of remaining faults which are removed during the testing as well as operational phase.

Prior to study the mathematical structure, some notations and assumptions are discussed which form the basis of our proposal.

2.1 Notations

$F_i(t)$	Probability distribution functions for fault removal process in testing phase. $(i=1,2,3,4)$
$G_i(t)$	Probability distribution functions for fault removal process in operational phase. $(i=2,3,4)$
$m_i(t)$	Expected number of faults removed by time t. $(i=1,2,3,4)$
a_i	Initial fault content for i^{th} release $(i=1,2,3,4)$
b_i	Rate of fault removal for i^{th} release $(i=1,2,3,4)$

2.2 Assumptions

The basic assumptions of the model are as follows:

1. Fault removal process is modeled by Non homogeneous poisson process (NHPP).
2. The number of bugs discovered at any instant of time is directly dependent on the remaining number of bugs in the system.
3. The count of faults in the software is finite.
4. The detected faults can be removed instantly, as soon as it occurs.

First Release of the Software

Based on the following set of assumption, it has been considered that the testing for the first release starts at time $t=\tau_1=0$ and the testing process for first release has been continued till time $'t=\tau_2'$ and there are chances that some bugs will remain in the software as no software can be bug free [2]. Thus the expression representing the fault debugged can be given as follows:

$$m_1(t)=a_1.F_1(t)\ ; \qquad (\tau_1=0)<t<\tau_2 \tag{1}$$

where $F_1(t)=1-e^{-b_1.t}$

Second Release of the Software

One basic reason for constant addition of functionalities and features by the firms is the competition between software firms and the need to survive in the market. When the first version of the software is in the operational phase, there are reports from users regarding the issues faced and this is how firms get information about any failure that is occurring while the software was under usage. New functionalities and fixing of bugs are done in accordance to feedback or reports received by the company. It is being assumed that adding some features results in the change of source code which might increase fault count in the software. Here, the differentiation between the faults removal phenomenon of the new version and remaining faults of just preceding release has been presented. In the course of testing of new version, there are two situations by which remaining faults of just previous release can be tackled viz. some faults will be removed by the testing team of current release and some faults will be debugged in the operational phase through the reports received from the users. To model such situation, a joint rate which unites both fault removal phenomenon using Steiltjes convolution approach has been considered [7]. Thus, the faults which will be debugged in the second release when it’s testing process started at time point $'t=\tau_2'$ are given by equation (2)

$$m_2(t)=a_2.F_2(t-\tau_2)+a_1(1-F_1(\tau_2-\tau_1)).\,F_2^*(t-\tau_2);\quad \tau_2<t<\tau_3 \tag{2}$$

where $F_2^*(t-\tau_2)=F_2\otimes G_2(t-\tau_2)$ show the impact of both testing and operational phase based on the conjoint effect of two distribution function $F_2(t-\tau_2)$ being fault removal process under the testing phase of second release where as $G_2(t-\tau_2)$ defines the removal process in operational phase of first release; convolution probability function has been used which has a following mathematical representation [14]:

$$(F\otimes G)(t)=\int_0^t F(t-x).g(x)dx=\int_0^t G(t-x).f(x)dx \tag{3}$$

Third Release of the Software

On similar basis as in second release, it has been assumed that faults are generated due to new add-ons, which implies that there will bugs due to addition of new features and also there will be some remaining faults of just previous release. Thus under the testing phase of third release, faults generated will be removed with FRP $F_3(t-\tau_3)$ and the faults from previous version will be removed with FRP $F_3 \otimes G_3(t-\tau_3)$. Equation (4) gives the count of overall faults removed for third release when testing phase begins at $'t=\tau_3'$.

$$m_3(t) = a_3.F_3(t-\tau_3) + a_2(1-F_2(\tau_3-\tau_2)).F_3^*(t-\tau_3); \quad \tau_3 < t < \tau_4 \tag{4}$$

where $F_3^*(t-\tau_3) = F_3 \otimes G_3(t-\tau_3)$ show the impact of both testing and operational phase based on the conjoint effect of two distribution function $F_3(t-\tau_3)$ being fault removal process under the testing phase of third release where as $G_3(t-\tau_3)$ defines the removal process in operational phase of second release.

Fourth Release of the Software

Further, a case when the new features are added in the software for the third time has been explained. Then the mean value function for overall fault removal process when testing starts at $'t=\tau_4'$ is given as follows:

$$m_4(t) = a_4.F_4(t-\tau_4) + a_3(1-F_3(\tau_4-\tau_3)).\ F_4^*(t-\tau_4); \quad \tau_4 < t < \tau_5 \tag{5}$$

where $F_4^*(t-\tau_4) = F_4 \otimes G_4(t-\tau_4)$ show the impact of both testing and operational phase based on the conjoint effect of two distribution function $F_4(t-\tau_4)$ being fault removal process under the testing phase of fourth release where as $G_4(t-\tau_4)$ defines the removal process in operational phase of third release;

The process of adding new functionalities is an ongoing process. These add-ons keep on happening till software is there in the market. This phenomenon helps in improving the value of software and also helps in increasing the reliability of the product as more and more faults are removed when testing and integration of code is done. Then the mean value function for n^{th} version is given as follows:

$$m_n(t) = a_n.F_n(t-\tau_n) + a_{n-1}(1-F_{n-1}(\tau_n-\tau_{n-1})). \\ F_n^*(t-\tau_n); \qquad \tau_n < t < \tau_{n+1} = T \tag{6}$$

The above presented structure is empirically tested for four releases of the software and under two different scenarios whose equations are given in model-I and model-II.

Model-I: In this case it has been considered that the remaining faults of just previous release will be removed with the joint rate of its operational as well as the testing phase of its succeeding release. The FRP for testing phase follows exponential pattern where as FRP for operational phase is constant i.e. $F(t) \sim exp(b_i)$ and $G(t) \sim 1(t)$.

$$m_i(t) = a_i.\left(1-e^{-b_i.(t-\tau_i)}\right) + a_{i-1}.\left(1-\left(1-e^{-b_{(i-1)}.(\tau_i-\tau_{i-1})}\right)\right). \\ \left(1-e^{-b_i^*(t-\tau_i)}\right); \ (i=2,3,4) \tag{7}$$

Model-II: In second case it has been considered that the remaining faults of just previous release will be removed with the joint rate of its operational as well as the testing phase of its succeeding release. The FRP for testing phase as well as operational phase follows exponential pattern i.e. $F(t) \sim exp(b_i)$ and $G(t) \sim exp(b_i)$.

$$m_i(t) = a_i.\left(1-e^{-b_i.(t-\tau_i)}\right) + a_{i-1}.\left(1-\left(1-e^{-b_{(i-1)}.(\tau_i-\tau_{i-1})}\right)\right). \\ \left(1-\left(1+b_i^*.(t-\tau_i)\right).e^{-b_i^*(t-\tau_i)}\right) \ ; \ (i=2,3,4) \tag{8}$$

where b_i represents the rate of removal of testing team in current release and b_i^* shows the rate of removal of debugging team with the joint effect of testing team and operational team. Equation (7) & (8) demonstrate the mathematical form of fault removal phenomena for multi releases of a software with the impact of fault testified from operational phase of previous release and testing phase of current release. Further, in next section equation (1), (7) and (8) are analyzed on software fault data.

3. MODEL VALIDATION

In this study, the Statistical Analysis System (SAS) software has used for estimating the parameters of equations [14]. The present study is analyzed on the data available for Tandem Computers [15]. From the analysis, the value of total number for release 1 is 130.202 and the rate of debugging is 0.083. The estimated values of each release for model-I and model-II are given in Table 1 & 2 and set of computed comparison criteria for all four releases are given in Table 3. The deviation between estimated and actual value of removed faults is being given in Figure (1 to 4) for all four versions of the software. From table 3, it can be observed that the values of R^2 is closer to 1, which is quite significant for the proposed cases. In model-I, it is observed that the rate by which faults have been removed in current release is same as the rate by which the left over faults have been removed conjoint effect of testing team and operational team because operational team acting constantly.

Table 1: Parameter Estimates of Model-I for Different Releases $(i=2,3,4)$

Parameters	Release 2	Release 3	Release 4
a	158.278	92.42	40.89
b_i	0.061	0.52	0.037
b_i^*	0.061	0.52	0.037

Table 2: Parameter Estimates of Model-II for different releases $(i = 2,3,4)$

Parameters	Release 2	Release 3	Release 4
a	146.397	84.025	60.42
b_i	0.062	0.030	0.054
b_i^*	0.240	0.302	0.032

Table 3: Comparison Criteria for four releases

Criterion	Release 1	Release 2		Release 3		Release 4	
		Model-I	Model-II	Model-I	Model-II	Model-I	Model-II
SSE	232.3	442	323.9	249.4	134.8	85.01	87.17
MSE	12.91	26.002	20.243	22.67	14.98	5.001	5.128
R^2	0.986	0.982	0.987	0.951	0.974	0.976	0.975

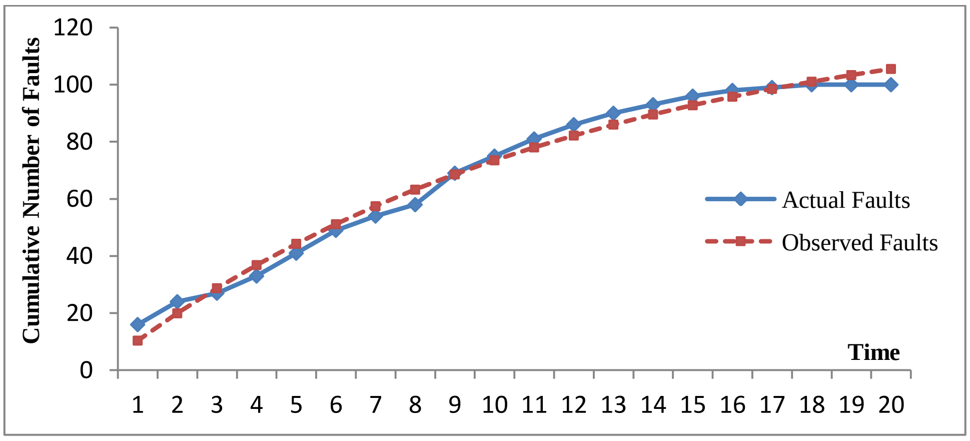

Fig 1: Goodness of Fit curve for Release 1

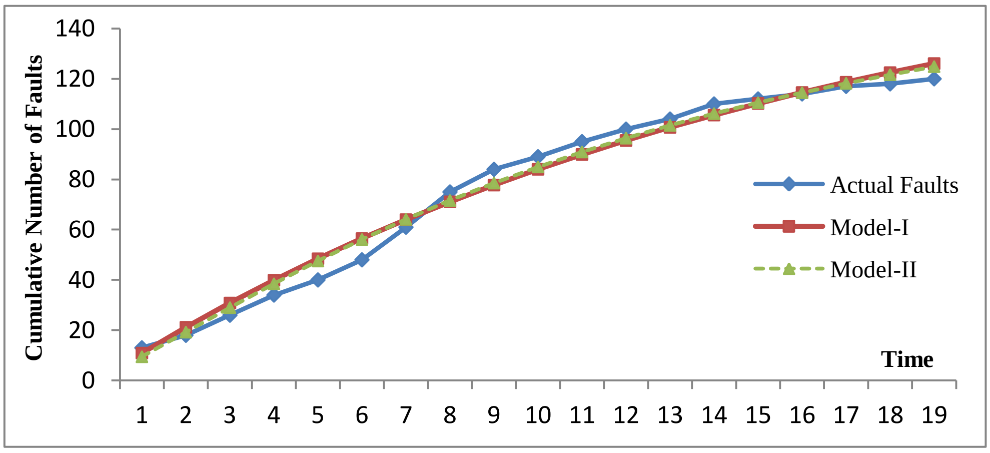

Fig 2: Goodness of Fit curve for Release 2

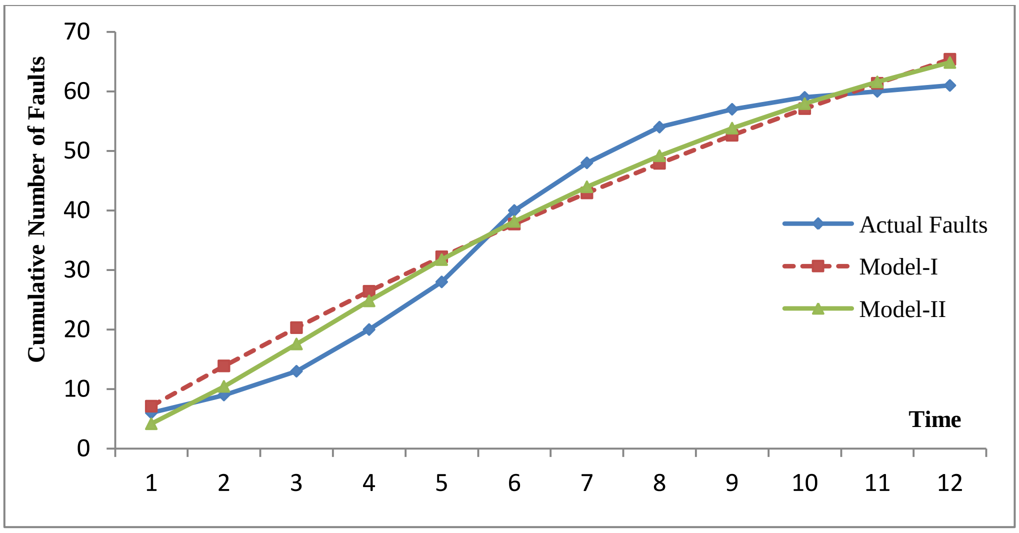

Fig 3: Goodness of Fit curve for Release 3

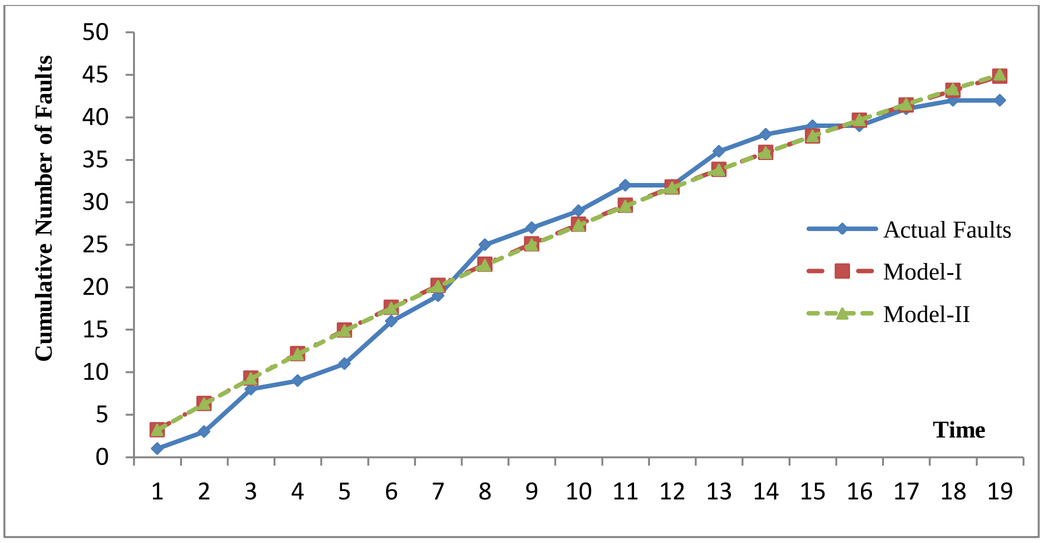

Fig 4: Goodness of Fit curve for Release 4

4. CONCLUSION

The proposed model is based on the assumptions that the overall fault removal of the new release depends on the reported faults from the just preceding release of the software which are removed both in its operational as well as the testing phase of its newer version and on the faults generated due to addition of some new functionalities to the existing software system. The remaining faults of just previous release are debugged through a joint rate modeled via the use of convolution approach and two different cases have been discussed. In the first case it is assumed that the fault removal phenomenon follows constant and exponential pattern for testing as well as operational phase, respectively whereas in second case both distributions follow exponential pattern with same rate. The proposal has been validated on failure data of tandem computers for four different releases and the result so obtained justifies the concept. In future the concept can be extended to study the fault removal phenomenon for successive releases of the software under various environment viz. stochastic environment, imperfect environment etc. Also release time problem can also be formulated to determine the time at which firm should release their software in the field for usage.

5. REFERENCES

[1] Aggarwal, A.G., Kapur, P.K., and Garmabaki, A.S. 2011 “Imperfect Debugging Software Reliability Growth Model for Multiple Releases” Proceedings of the 5th National Conference on Computing for Nation Development-INDIACOM, New Delhi.

[2] Anand A., Singh O., and Das S., 2015 “Fault Severity based Multi up-gradation Modeling Considering Testing and Operational Profile”, International Journal of Computer Applications (0975 – 8887), Volume 124 – No.4.

[3] Garmabaki, A. H. S., Aggarwal, A.G., Kapur, P. K., and Yadavali, V. S. S., 2014 , “ The Impact of Bugs Reported from Operational Phase on Successive Software Releases”, International Journal of Productivity and Quality Management, volume 14, number 4, pp 423-440.

[4] Goel, A.L., Okumoto, K.., 1979, “Time dependent error detection rate model for software reliability and other performance measures”, IEEE Transaction Reliability, R-28(3), 206-211.

[5] Software QA and Testing Frequently-Asked-Questions, Part 1 [online] http://www.softwareqatest.com/ qatfaq1.html.

[6] Kapur P.K. and Garg R.B., 1992, "A software reliability growth model for an error removal phenomenon", Software Engineering Journal, (7), 291-294.

[7] Kapur P. K., Pham H., Gupta A., and Jha P. C., 2011, "Software Reliability assessment with OR application", Springer, Berlin.

[8] Kapur P.K, Tandon A., and Kaur G., 2010, "Multi Up-gradations Software Reliability Model", ICRESH, 468-474.

[9] Kapur P. K., Singh O., Garmabaki A. and Singh, J., 2010, "Multi up-gradation software reliability growth model with imperfect debugging", International Journal of systems Assurance Engineering and Management, 1(4),299-306.

[10] Kapur P. K., Anand A., and Singh O., 2011, "Modeling Successive Software Up-Gradations with Faults of Different Severity", Proceedings of the 5th National Conference on Computing For Nation Development, ISSN 0973-7529 ISBN 978-93-80544-00-7.

[11] Singh O., Kapur P.K., Anand A. and Singh, J., 2009, "Stochastic Differential Equation based Modeling for Multiple Generations of Software", Proceedings of Fourth International Conference on Quality, Reliability and Infocom Technology (ICQRIT), Trends and Future Directions, Narosa Publications, pp. 122-131.

[12] Singh O., Kapur P.K., and Anand A., 2011, "A Stochastic Formulation of Successive Software Releases with Fault Severity" Industrial Engineering and Engineering Management, 136-140.

[13] Singh O., Kapur P.K., Khatri S.K., and Singh J.N.P., 2012, "Software Reliability Growth Modeling for Successive Releases", proceeding of 4th International Conference on Quality, Reliability and Infocom Technology (ICQRIT), PP 77-87.

[14] SAS Institute Inc., 2004, "SAS/ETS user's guide version 9.1", Cary, NC: SAS Institute Inc..

[15] Wood A., 1996, "Predicting Software Reliability", IEEE Computer (11) 69-77.

[16] Yamada S., Ohba M., Osaki S. 1983 "S-shaped software reliability growth modelling for software error detection", IEEE Transaction Reliability, R-32(5), 475-484.