UDK: 004.415.5:004.416
COSATI: 09-02

Software Reliability: Models and Parameters Estimation

Vladimir Zeljković[1)]
Nela Radovanović[1)]
Dragomir Ilić[1)]

Software systems are present in many safety-critical applications such as power plants, health care systems, air-traffic, etc. They all require high quality, reliability and safety. In this paper, the software system modeling methods for estimating parameters such as failure rate and reliability are presented. The second part of the paper is a case study of a complex software system developed in the LOLA Institute for an external customer.

Key words **software, software reliability, software maintenance, reliability estimation.**

Introduction

UNTIL the late 60s, the attention of scientists and engineers was directed towards hardware reliability (mechanical, electronic systems). From the 70s, with the permanent growth of software, applications became the center of many studies. The possibility to create complex dependencies and better cost/price ratios compared to hardware led to a wide range of software applications. Today, computers are used in everyday life, in industry, banks, large systems like power distribution, traffic, water supply, etc. Computers are used even in life critical applications in hospitals; they control air traffic and airplane flight, where failures could lead to catastrophes and loss of many lives.

On the one hand, there is our increasing dependence on software; on the other hand, software systems are becoming more and more complex and thus harder to develop and maintain. Software functionality is becoming crucial from the aspects of reliability, safety of human lives and security issues as well [1,9,10].

Specification, evaluation and verification of this quality characteristic are important issues for both developers and users of the system.

Every software system has faults. In order to decrease the number of remaining faults in the system, two basic activities can be performed:
- Fault prevention
- Fault detection and removal through inspections/reviews and testing

There are numerous methods and techniques dealing with fault prevention. There are also many software-testing methods and tools and they are valuable for the process of reliability growth as well as for achieving desired software reliability [2]. Different fault detection methods are interesting because they supply valuable data that forms the basis for evaluating software reliability. In this paper, we are going to present the models of reliability estimation based on the failure data on detected faults in the system during development and/or maintenance. We are going to show how it is possible, with a certain confidence, to make estimates that match the real data.

Software reliability

Software reliability is defined as *the probability of failure-free software operation for a specified period of time in a specified environment* [3]. Therefore, the main concern is centered around software faults, their effect on the system and the remaining number of faults, system failures, the way of detecting failures, time between failures and failure rates, as well as the confidence in the performed estimates.

Suppose the software system has N errors. Each error posses its *MTTF* or the failure rate $\theta_i=1/\lambda_i$. The reliability growth method leads to the decrease of the system failure rate and the increase of the system reliability with time [2, 5, 6]. The cumulative number of errors detected and corrected increases with time. Two types of software reliability growth models are typical: concave and S - type, shown in Figure **1**.

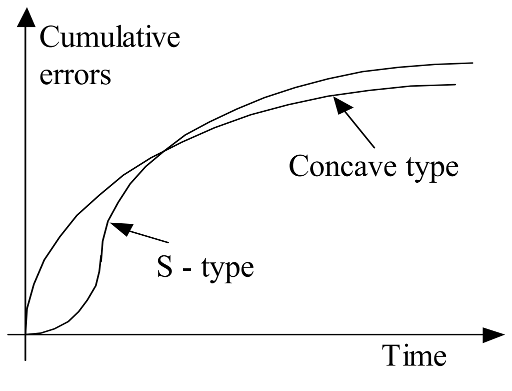

Figure 1. Concave and S – type shape of the cumulative errors growth curve

[1)] LOLA Institut, Kneza Višeslava 70a, 11030 Belgrade, SERBIA

The cumulative number of errors is a function that increases with time and asymptotically approaches the total number of errors *N*. By systematically detecting failures and removing faults, it is possible to achieve the required system *MTTF* and the failure rate $\theta_R=1/\lambda_R$, before the software is shipped to the customer. For a specific software system, it is necessary to find out the mathematical model and the parameters that fit best the software reliability growth.

Many software reliability growth models are suggested [7, 2]. Some of them are summarized in Table 1.

Table 1. Software reliability growth models

Name	Type	$m(t)$
Gompertz	S- type	$a\left(b^{c^t}\right)$
Weibull	concave	$a\left(1-e^{-bt^c}\right)$
Pareto	concave	$a\left[1-(1+t/\beta)^{1-\alpha}\right]$
Yamada Raleigh	S- type	$a\left[1-\exp\left\{-r\alpha\left(1-e^{-\frac{1}{2}\beta t^2}\right)\right\}\right]$

where $m(t)$ is the expected number of failures observed over a period of time *t*. At the beginning of testing, $m(t=0)=0$, and $m(t)\to N$ as the time $t\to\infty$.

The failure rate is determined as

$$\lambda(t)=\frac{dm(t)}{dt} \tag{1}$$

The reliability, defined as the probability that the software system will operate for a time interval *T*, after the elapsed time *t*, is

$$R(T|t)=e^{-\int_t^{t+T}\lambda(\tau)d\tau}=e^{-\{m(t+T)-m(t)\}} \tag{2}$$

where *T* is defined as the mission interval that started at *t*.

For specific software, it is necessary to estimate the parameters of the model δ_1, δ_2, ..., δ_n, from the test pairs $\{i, t_i\}$ $i = 1, 2, ..., j$.

As a measure of goodness -of -fit for different models, the least square is compared

$$LS=\sum_{i=1}^{j}\left[k_i(t_i)-\hat{m}(t_i)\right]^2 \tag{3}$$

The maximum likelihood estimates $\hat{\delta}$ follows the joint normal distribution if the number of sample size, test pairs $\{i, t_i\}$, increases [5]. The lower and upper confidence bounds on the expected number of failures are

$$m_{U/L}(t)=m\left(\hat{\delta},t\right)\pm K_\gamma\sqrt{Var\left[m\left(\hat{\delta},t\right)\right]} \tag{4}$$

where

$m_{U/L}(t)$	- Upper and lower confidence bounds on the cumulative number of errors,
$m\left(\hat{\delta},t\right)$	- Model estimation,
K_γ	- The value of $100(1-\gamma)/2$ percent of standard normal distribution,
$Var\left[m\left(\hat{\delta},t\right)\right]$	- Variance.

The variance is determined by

$$Var\left[m(\delta,t)\right]=\left[\frac{\partial m(\delta,t)}{\partial\delta_1},\frac{\partial m(\delta,t)}{\partial\delta_2},...,\frac{\partial m(\delta,t)}{\partial\delta_l}\right]\cdot Cov\left[m(\delta,t)\right]\cdot\begin{bmatrix}\frac{\partial m(\delta,t)}{\partial\delta_1}\\ \frac{\partial m(\delta,t)}{\partial\delta_2}\\ \vdots\\ \frac{\partial m(\delta,t)}{\partial\delta_l}\end{bmatrix} \tag{5}$$

defined for $\hat{\delta}_1,\hat{\delta}_2,...,\hat{\delta}_l$. The covariance matrices are

$$Cov\left[m(\delta,t)\right]=I^{-1} \tag{6}$$

where the elements of the matrices are

$$I_{ij}=E\left[-\frac{\partial^2 l(\delta)}{\partial\delta_i\partial\delta_j}\right]\ ,\ i,j=1,2,...,l \tag{7}$$

The above procedure gives the basic elements of a reliability growth analysis for any software, such as the estimate of the total number of errors, the remaining number of errors after the test time *t*, *MTTF* and $R(t)$.

Software system

A complex software system developed in the LOLA Institute for a valued external customer is presented [8]. It is a distributed real time system controlling four packing machines, enabling order management through an operator input, as well as supervision and report generation. We choose the model to determine the parameters for that model based on the collected "Failure" Data, given in Table 2.

Table 2. Failure Data for the Case Study

Date	T_i [months]	*No* **of errors in the interval**	**Cumulative errors** k_i (T_i)	**Remarks**
09/01/97	T_0			
09/12/97 09/18/97	$T_1=1$	2	2	
10/16/97 10/23/97	$T_2=2$	2	4	
11/06/97	$T_3=3$	1	5	
01/28/98	$T_4=5$	3	8	
02/09/98 02/26/98	$T_5=6$	3	11	
03/16/98	$T_6=7$	1	12	
04/04/98	$T_7=8$	3	15	*(new version installed)*
03/15/00	$T_8=31$	1	16	

First, we applied the modified G-O model, as stated

$$m(T)=\alpha\left(1-(1+\beta t)e^{-\beta T}\right)$$

The estimated parameters based on $(T_i, k_i\,(T_i))$ pairs and the maximum likelihood function, are

$\hat{\alpha}$ = 16.36 and $\hat{\beta}$ = 0.3880 with L_{min}= 7.2758

The model for the expected number of errors, shown in, is

$$m(T)=16.36\left(1-(1+0.388t)e^{-0.388T}\right)$$

The upper and lower bounds were determined for γ=0.9.

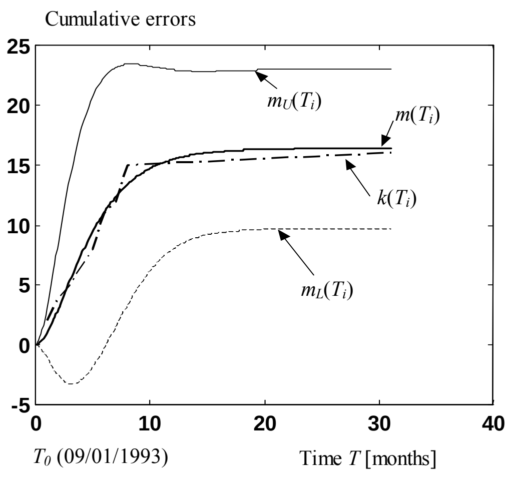

Figure 2. Failure Data, G-O model and confidence bounds

The failure intensity is presented in Figure **3**.

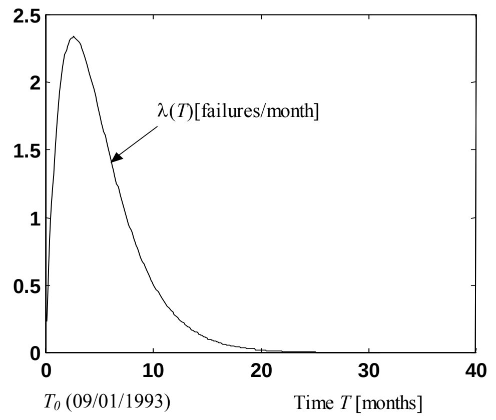

Figure 3. Failure rate – Modified G-O model

The expected number of failures after 24 months is

$$m(T=24)=E\{\hat{\alpha}-m(T=24)\}=0.0152$$

The next failure is expected in $\theta_{(T=24)}$=187.2804 months.

A probability that after 24 months the system will be operating without a failure for another month is

$$R(14)=e^{-\{m(1+24)-m(24)\}} = 0.9955$$

The G-O model matched the real data well.

Conclusion

Software reliability cannot be calculated during the design phase. If adequate data on system failures is collected throughout the project, especially during the testing/verification/validation and maintenance (operational) phase, the same models for estimating reliability parameters, such as the expected number of failures in a certain period of time, failure intensity, the expected time of the next failure, etc., could be applied to software systems as well. In the example of a complex real-time software system we have shown that modified Gompertz models give approximations, based on the failure data collected during the maintenance phase, that match the real data. In this way, it is possible to supply valuable information and confidence in the system to both the customer and the management and software development team. This case study also showed the importance of reliability estimation during the development/testing, thus proving the product readiness to be delivered to the customer as well as during the operational phase for validating the product reliability.

References

[1] Voas,J., Payne,J.: *Dependability CERTIFICATION OF Software Components*, Journal of Systems and Software 2000, 52(2-3) pp.165-172.

[2] ZELJKOVIĆ,V.: *Reliability in practice*, LOLA Institute, 2000, (In Serbian)

[3] ANSI/IEEE, *Standard Glossary of Software Engineering terminology*, STD-729-1991.

[4] GOEL,L.A.: *Software Reliability Models: Assumptions, Limitations, and Applicability*, IEEE Transaction on Software Engineering, December 1985, Vol.SE-11, No.12, pp.1411-1423.

[5] YAMADA,S., OHBA,M., OSAKI,S.: *S - Shaped Reliability Growth Modeling for Software Error Detection*, IEEE Transaction on Reliability, December 1983, Vol.R-32, No.5, pp.475-478.

[6] YAMADA,S., OSAKI,S.: *Software Reliability Growth Modeling: Models and Applications*, IEEE Transaction on Software Engineering, December 1985, Vol. SE-11, No.12, pp.1431-1437.

[7] Wood, A.: Predicting Software Reliability, COMPUTER, November 1996, pp 69-77.

[8] ZELJKOVIĆ,V., RADOVANOVIĆ,N.: *Software Reliability - A Case Study*, COMMUNICATIONS IN DEPENDABILITY AND QUALITY MANAGEMENT, An International Journal, 2000, Vol.1, pp.25-33.

[9] ZELJKOVIĆ,V., MAKSIMOVIĆ,K., KNEŽEVIĆ,J.: *Probabilistic approach to buckling of thin panels subject to combined loads*, Scientific Technical Review, ISSN 1820-0206, 2010, Vol.60, No.3-4, pp.34-37.

[10] POSAVLJAK,S., MAKSIMOVIĆ,K.: *Initial fatigue life estimation in aero engine discs*, Scientific Technical Review, ISSN 1820-0206, 2011, Vol.61, No.1, pp.25-30.

Received: 11.04.2011.

Pouzdanost softvera: modeli i parametri procene

Softverski sistemi su primenjeni u mnogim bezbednosno-kritičnim objektima kao što su elektrane, sistemi namenjeni zaštiti zdravlja, vazdušni saobraćaj. Ovi sistemi zahtevaju: visok kvalitet, pouzdanost i bezbednost. U ovom radu su prikazani softverski modeli i metod za procenu parametara, intenziteta otkaza, pouzdanosti, itd. Drugi deo rada je primer složenog softverskog sistema razvijenog u LOLA Institutu za određenog korisnika.

Ključne reči: **softver, pouzdanost softvera, održavanje softvera, ocena pouzdanosti.**

Надёжность программного обеспечения: образец и параметров оценки

Системы программного обеспечения применяются во многих безопасно- критически важных объектах, в таких как электростанции, системы предназначенные для здравоохранения и медицинского обслуживания и воздушного транспорта. Все они требуют: высокое качество, надёжность и безопасность. В этой работе представлены методы системы программного обеспечения и методы моделирования для оценки параметров, интенсивности отказов, надёжности и так далее. Вторая часть представляет собой пример сложной системы программного обеспечения, разработанного в Институте «LOLA» для внешних клиентов.

Ключевые слова: **программное обеспечение, надёжность программного обеспечения, обслуживание программного обеспечения, достоверности рейтинга программного обеспечения.**

Fiabilité logiciel

Les systèmes logiciels sont utilisés dans de nombreux cas critiques de point de vue sécurité, tels que : centrales électriques, systèmes destinés à la protection de la santé, circulation aérienne. Ces systèmes exigent une haute qualité, fiabilité et sécurité. Dans ce papier on a présenté les modèles de logiciel et la méthode pour l'estimation des paramètres, l'intensité de défaillance, la fiabilité etc. La seconde partie de ce travail représente l'exemple d'un système logiciel complexe qui a été développé au sein de l'Institut LOLA pour un utilisateur connu.

Mots clés: **logiciel, fiabilité logiciel, entretien de logiciel, estimation de fiabilité.**