Sensitivity of Reliability Growth Models to Operational Profile Errors

A. N. Crespo(*), P. Matrella(+) and A. Pasquini(+)

(*) Dept. of Comp. Science, UNICAMP, Campinas, Brazil
(+) Div. Robotic & Information Tech., ENEA, Via Anguillarese 301, 00060 Roma, Italy

Abstract

The estimation of the operational profile is one of the key factors during the use of software reliability growth models. But, the operational profile can be very difficult to estimate in particular applications such as the one of software used for process control. In other cases, a single operational profile may not be sufficient to describe the use of the product by a number of different customers. An operational profile may also change during the development of software or during its operational usage. All these cases may lead to errors in the estimation of the operational profile. The paper describes an empirical evaluation of the sensitivity of reliability growth models to errors in the estimation of the operational profiles. Some reliability growth models are applied during the testing phase of a software system. The particular characteristics of the case study allow the measurement of the actual reliability growth of the software and its comparison with the estimations provided by the models. Measurement and comparison are repeated for different operational profiles giving information about the effect of a possible error in the estimation of the operational profile. Results show that errors in the operational profile estimation do not heavily affect reliability estimates and that their influence is strongly dependent on the accuracy with which the software system has been tested.

1 Introduction

One of the main assumptions of reliability growth models is that the software under evaluation is tested in a manner that reflects its actual operational usage, that is, the test profile must be as close as possible to the operational profile. Then, reliability growth models require the development of an operational profile for the software under evaluation [4], [5] and their predictive quality is affected by the ability of estimating the correct operational profile. Some authors suggested a detailed procedure to develop a trustworthy operational profile and reported their successful experience in doing that [11]. But, most of those experiences, are from applications where the operative usage of the software is predictable because is related to identifiable events due to human activity (for example software for real-time communication systems). In other applications, such as the control of a chemical or nuclear process, different software components are activated by a complex pattern of events that are due to the characteristics of the process controlled and whose frequency can hardly be estimated a priori. In some cases very critical software components are activated by physical events whose frequency during the software operative usage is totally unknown. A typical example are the software components that performs failure detection and containment functions in the flight control system of a spacecraft: some of those components are activated by hardware malfunction such as the erasure or corruption of the RAMs content [2], [16]. In other cases, a single operational profile may not be sufficient to describe the use of the product by a number of different customers. But, the effort required to derive operational profile information for each group of users is usually extremely high. The usual solution is to adopt an approximate operational profile that represents a rough average of the operational profiles of the different customers [7]. In addition to the difficulty of estimating a trustworthy operational profile in some kind of applications, there could be an additional problem in using it in reliability growth models. Software systems may evolve during development because functions are added or modified. Also the way in which a system is used evolves with the operative usage, then the operational profile may change after the software reliability estimation [13]. All the mentioned reasons may easily lead to the use of an erroneous operational profile. In such a case the sensitivity of reliability growth models to errors in the estimation of the operational profiles is of great concern. This sensitivity has been studied in [3] and [12]. In the first work the authors, using a reliability estimation tool called TERSE, model a program as graph of arcs and nodes with associated transition probability.

0-8186-7707-4/96 $5.00 © 1996 IEEE

Errors in the estimation of the operational profile are then simulated modifying the arc transition probabilities. The authors conclude that inaccuracies in operational profile estimates may result in significant errors in reliability estimates and that models more robust to errors in operational profiles are required. In the other work the author takes an analytical approach. He defines the operational profile as the set of the occurrence probabilities of the software functions. Then, the sum of the occurrence probabilities is equal to 1 and an error in one occurrence probability causes countervailing errors in other occurrence probabilities. On the basis of this observation, the author suggests that multiple errors in the operational profile will also tend to have countervailing rather than cumulative effect on the failure intensity of software. He concludes that current models are very robust with respect to errors in the estimation of the operational profile. In particular he quantifies this robustness: errors in the estimation of the occurrence probabilities can be five time the percent error that is acceptable for measuring failure intensity. To contribute to this debate this paper reports the results of an empirical approach that is based on a case study. Some reliability growth models are applied during the testing phase of a software system. The particular characteristics of the case study allow the measurement of the actual reliability growth of the software and its comparison with the estimations provided by the models. Measurement and comparison are repeated for different operational profiles giving information about the effect of a possible errors in the estimation of the operational profile. Section 2 describes the approach used and the steps in which the case study has been organised. Section 3 describes the practical realisation of these steps. Finally, section 4 discusses the results obtained.

2 Description of the approach

2.1 Definitions

P = program

P_k = version of P containing K faults

n = number of faults contained in P at the beginning of the testing phase ($P=P_n$)

F_k = kth fault that is detected in P_k

T_0 = time in which the test of P_n begins

T_k = time in which F_k is detected

OP_e = operational profile for the program P estimated at the beginning of the test phase ($P=P_n$)

OP_h = hth operational profile for the program P

m = total number of software functions that are present in the program P

$fop_{h,i}$ = occurrence probability of the ith software function according to OP_h, with $\sum_{i=1}^{m} fop_{h,i} = 1$

TC_h = set of test cases for the program P generated randomly on the basis of the operational profile OP_h

IC_h = set of input cases for the program P generated randomly on the basis of the operational profile OP_h

GE = Geometric model

LV = Littlewood and Verrall quadratic bayesian model

MB = Musa basic execution time model

ML = Musa log poisson execution time model

BB = Brooks and Motley binomial model

PB = Brooks and Motley poisson model

$D(OP_q,OP_h)$ = distance between the operational profiles OP_q and OP_h measured as:

$$D(OP_q, OP_h) = \frac{1}{2}\sum_{i=1}^{k}\left|fop_{q,i} - fop_{h,i}\right|$$

$\hat{R}(P_K, j, OP_h, z)$ = reliability estimated for P_k based on the data collected till the detection of the fault j with $j \in \{1,...n\}$, using the operational profile OP_h and the reliability growth model z with $z \in$ {GE, LV, MB, ML, BB, PB}

$R(P_k,OP_h)$ = reliability of P_k with operational profile OP_h

2.2 Approach used

Testing is the execution of a program P in a controlled and systematic way to demonstrate the presence of the required functions and the absence of unwanted effects. During testing, the program P is subject to a fault removal process. Thus, the program evolves towards more "purified" versions as shown in fig. 1. During the fault removal process, the interfailure data T_0, T_1,..., T_m are recorded. If the set of test cases used to test P is generated randomly on the basis of OP_e, the analysis of the interfailure data allows to estimate $\hat{R}(P_K, m, OP_e, z)$ $\forall$ k $\in\{0,...,n\}$. An evaluation of the predictive quality of the model z with $z \in$ {GE, LV, MB, ML, BB, PB}, requires the knowledge of the actual reliability $R(P_k,OP_e)$ $\forall$ $k\in\{0,...,n\}$. An evaluation of the sensitivity of the models to operational profile errors requires the assumption that the estimation of the operational profile OP_e was inaccurate and that the right operational profile is a different one, let say OP_1. In addition it requires knowledge of the actual reliability $R(P_k,OP_h)$ $\forall$ $k\in\{0,...,n\}$ and $\forall$ $h\in\{e,1\}$. The comparison of the reliability with OP_e and with OP_1 provides information about the sensitivity to an error that is given by $D(OP_e, OP_1)$. If the same information are available for $OP_2, OP_3,...,OP_q$, where: $D(OP_e,OP_1)<D(OP_e,OP_2)<...<D(OP_e,OP_q)$ one can obtain sensitivity for increasing errors in the estimation of the operational profile. The actual reliability can be measured in terms of probability of failure per execution using the following equation [15]:

$$(1) \qquad R(P_k, OP_h) = 1 - \lim_{n_t \to \infty} \frac{n_f}{n_t}$$

where:

n_f = number of executions of input cases generated randomly on the basis of OP_h in which a failure occurred; n_t = total number of execution of input cases generated randomly on the basis of OP_h.

During the development process, P_0 is not available and therefore the actual reliability can never be measured. But, the current software engineering techniques, and, in particular, the use of configuration management techniques and tools, allow to keep track of the evolution of a program during its lifecycle and of the faults that have been found during its testing and operational usage. Then, if one refers again to fig. 1, at a time T_k for some k $\in \{1,...,n\}$ these techniques can provide record of the faults F_r $\forall$ $r \in \{1,...,k\}$. The approach followed in this case study is to use a program P that has reached the version P_0 and for which all the faults F_r $\forall$ $r \in \{1,...,n\}$ are known. We put back in P_0 all these faults obtaining P_n. Then we start a new fault removal process like the one shown in fig. 1 but with P_0 available. The case study is based on the following steps:

a) Identification of a suitable program P_0 for which all faults F_r are known and re-insertion of the faults in the program;
b) Estimation of an operational profile OP_e for the program version P_n and generation of TC_e;
c) Fault removal process for P_n based on the use of TC_e. The output of this process are the set P_k $\forall$ k : n-m $\le$ k $\le$ n and the times $T_0, T_1, ...,T_m$;
d) Estimation of $\hat{R}(P_K, m, OP_e, z)$ $\forall$ k : n-m $\le$ k $\le$ n and $\forall$ z $\in$ {GE, LV, MB, ML, BB, PB};
e) Selection of the models offering the best predictive accuracy;
f) Measurement of $R(P_k,OP_e)$ $\forall$ k : n-m $\le$ k $\le$ n;
g) Generation of OP_h where $h \in \{1,...,q\}$ and $D(OP_e,OP_1)<D(OP_e,OP_2)<...<D(OP_e,OP_q)$
h) Measurement of $R(P_k,OP_h)$ $\forall$ k : n-m $\le$ k $\le$ n and $\forall$ $h \in \{1,...,q\}$;

From step b) till e) we reproduce the normal steps of a testing and reliability estimation activity, while steps f) is only possible because P_0 is available. In these steps we implicitly assume that OP_e is correct. Steps b) to e) do not change even if we assume that the operational profile estimation was inaccurate. The consequence of this inaccuracy is that an erroneous OP_e is used in generating TC_e and in estimating $\hat{R}(P_K, m, OP_e, z)$. In this case, the actual reliability does change and will have to be measured using the actual operational profile that is different from OP_e. The aim of the case study is to evaluate the effect on reliability growth models of increasing errors in the operational profile. Then, in steps g) and h), we measure the actual reliability using different operational profiles at increasing distance from OP_e, each time assuming that the operational profile used is the correct one. As a result of the case study we will be able to:

- Compare $\hat{R}(P_K, m, OP_e, z)$ $\forall$ k : n-m $\le$ k $\le$ n and $\forall$ z $\in$ {GE, LV, MB, ML, BB, PB} with $R(P_k,OP_e)$ $\forall$ k : n-m $\le$ k $\le$ n;
- Compare $\hat{R}(P_K, m, OP_e, z)$ $\forall$ k : n-m $\le$ k $\le$ n and $\forall$ z $\in$ {GE, LV, MB, ML, BB, PB} with $R(P_k,OP_h)$ $\forall$ k : n-m $\le$ k $\le$ n and $\forall$ $h \in \{1,...,q\}$;

3 Description of the case study

3.1 Step a)

The program chosen for the case study calculates the parameters for an array of antennas used in space applications. It was developed for the European Space Agency (ESA) in the C language and consists of almost 10.000 lines of code (6.100 executables) divided into several modules. During the integration testing and operative usage of the program, 33 faults were discovered and recorded. These faults are classified in tab. 1 on the basis of [9]. The program has been extensively used after the last fault removal without having new failures. We used this program as P_0 in our case study. The 33 faults were re-inserted in P_0 generating P_{33}.

3.2 Step b)

The operational profile has been defined in [11] as the set of the occurrence probabilities of the software functions. This case study used the same definition. To estimate an operational profile OP_e the possible sub-functions of the program were identified. A graph was generated capturing the connectivity of these sub-functions. Each node in the graph represented a sub-function. Two nodes A and B were connected if control could flow from sub-function A to sub-function B. There was a unique start and end node representing sub-functions at which execution began and terminated, respectively. A path through the graph from the start node to the end node represents one possible software function. To estimate the occurrence probability of the software functions each arc was assigned a transition probability, i. e. the probability of control flowing between the nodes connected by the arc. For example, if node A is connected to nodes B, C and D, and if the probabilities associated with arcs A-B, A-C, and A-D are, respectively, 0.3, 0.6 and 0.1 then after the execution of sub-function A the program will perform sub-functions B, C or D respectively with probability 0.3, 0.6, and 0.1. There was a total of 236 nodes. Transition probabilities were determined by interviewing the program users. The set of test cases TC_e was obtained generating test cases randomly on the basis of OP_e.

3.3 Step c)

This step of the case study was performed using an automated toolset developed for the experiment. This toolset identified and removed the faults from the faulty versions of the program by running the set of test cases TC_e and using P_0 as an oracle. The execution of TC_e led to the identification of 28 faults and then, to the program versions P_{33},..,P_5. The others 5 faults remained undiscovered.

3.4 Step d)

The reliability estimation was based on the interfailure data collected till the detection of failure no. 28 and was based on the use of the tool Statistical Modelling and Estimation of Reliability Functions for Software (SMERFS) Revision 3 [8]. This tool implements 6 "time between failure" and 5 "interval data" models, but, the limited number of failures in our application and the characteristics of the interfailure data reduced the number of models applicable to the following six:

- Geometric Model (GE in the following)
- Littlewood and Verrall Bayesian Quadratic Model (LV)
- Musa Basic Execution Time Model (MB)
- Musa Log Poisson Execution Time Model (ML)
- Brooks and Motley Binomial Model (BB)
- Brooks and Motley Poisson Model (PB)

3.5 Step e)

It is widely accepted that no model can be considered appropriate in all applications [1]. Best models for having future predictions from specific data must be selected by analysing the accuracy of past predictions on the same data. SMERFS provides four analysis techniques (accuracy, bias, noise and trend) for the execution time models and one (accuracy) for the interval data models. Use of these techniques has been proposed in [10] and their implementation in SMERFS is described in detail in [8]. Tab. 2 shows the result of the analysis. The table shows also the goodness-to-fit of the models using the Chi-square for interval data model and the Kolmogorov distance for the execution time models. The best results for the interfailure data available were obtained by GE, LV, and PB and these models were used for the remaining part of the case study.

3.6 Step f)

The actual reliability for a given program P_k and the operational profile OP_e was measured using equation (1). An automated toolset run the set of input cases IC_e and used P_0 as an oracle. The size of the input data set IC_e was determined using a convergence criterion set to 10^{-5} for the reliability computation.

3.7 Step g)

To simulate the effect of error in the operational profile, we generated three different operational profiles at increasing distance from OP_e. Criteria to select the software function occurrence probabilities for each operational profile were:

- to satisfy the condition $\sum_{i=1}^{m} fop_i = 1$;
- to obtain new operational profiles at regularly increasing distances from OP_e.

The satisfaction of the latter criterion was not always possile because of the costraints imposed by the first criterion and by the characteristics of the graph described in Subsection 3.2. The resulting distances between the operational profiles used in the case study and OP_e are:

Operational profiles used in the case study	Distance from OP_e
OPe	0
OP_1	0.26
OP_2	0.54
OP_3	0.76

3.8 Step h)

The actual reliability for a given program P_k and a given operational profile OP_h was measured using equation (1). An automated toolset run the set of input cases IC_h and used P_0 as an oracle. The size of the input data set IC_h was determined using a convergence criterion set to 10^{-5} for the reliability computation.

4 Results

4.1 Presentation of the results and discussion

Figure 2 compares the reliability growth estimation provided by the models GE, LV and PB with the actual reliability measured using equation (1) and operational profile OP_e. In other words, $\hat{R}(P_K,28,OP_e,z)$ where $z \in \{GE, LV, PB\}$ and where $K \in \{33,..,5\}$ is compared with $R(P_k,OP_e)$ where $K \in \{33,..,5\}$. Use of equation (1) ensure that, in our case study, we compare the estimations with the actual reliability of the software (with the approximation due to the use of the convergence criterion) and not with the reliability occasionally experienced during testing as one would do during a conventional goodness-to-fit evaluation activity. As shown in fig. 2, the actual reliability may occasionally decrease during testing. Since no new faults are introduced during our fault removal process, the decrease in reliability is due to fault masking [6]. Figure 2 gives a qualitative idea of the predictive accuracy of the models if the operational profile OP_e is a perfect estimation of the actual operational profile. The estimation provided by the GE model fits well with the actual reliability over the whole measure range, the LV model fits better after the execution of the first $5*10^2$ test cases, while all the models give reasonably accurate estimations after $5*10^3$ test cases. Figure 3 compares the reliability growth estimation provided by the models GE, LV and PB with the actual reliability measured using equation (1) and operational profile OP_1. In other words, $\hat{R}(P_K,28,OP_e,z)$ where $z \in \{GE, LV, PB\}$ and where $K \in \{33,..,5\}$ is compared with $R(P_k,OP_1)$ where $K \in \{33,..,5\}$. In this way fig. 3 gives a qualitative idea of the predictive accuracy of the models GE, LV, and PB if there is an error in the estimation of the operational profile OP_e that is measured by the distance $D(OP_e, OP_1)$. Figure 4 and 5 give the same information for a distance that is respectively $D(OP_e, OP_2)$ and $D(OP_e, OP_3)$. Table 3 shows the predictive accuracy of the models for the four operational profiles using the Chi-square statistics. A first analysis of the results shown by the plots evidences that:

- The predictive accuracy of the models, is not heavily affected by errors in the estimation of the operational profile. In general, if a model estimate fits with the actual reliability for a correct estimation of the operational profile, then it fits well even if the estimation of the operational profile is erroneous;
- Both, the models estimation accuracy and their sensitivity to errors in the estimation of the operational profile seem to be strongly related with the number of test cases executed. Models show an acceptable predictive accuracy after the execution of a number of test cases ranging between $5*10^2$ and $5*10^3$. The sensitivity of models to errors in the estimation of the operational profile decreases significantly after the execution of a number of test cases between 10^3 and 10^4. One of the possible reasons is that testing reaches a high level of coverage after the execution of this number of test cases. When a high level of coverage is reached, changes in the distribution of new random test cases do not affect the test ability of exposing new faults.
- Models tend to overestimate reliability and this overestimation increases with increasing errors in the estimation of the operational profile.

4.2 Limitation of the study

This case study reproduces part of a software development project in realistic conditions and in a controlled environment to determine the relationship between some aspects of the reliability estimation activity. In principle, a case study cannot give considerable confidence that the same results will apply in different development projects and reliability estimation activities, because there is no replication [17]. However, this case study has been realised with the assumption that the results are of some relevance even out of its specific context. The assumption is supported by the following observations:

- the program used is a real program of typical size for this kind of applications;
- the program language is one of the most used;
- faults re-inserted are the real faults discovered during the integration testing and operative usage of the program; their density and types are the typical ones reported in the literature for the same stage of the development process [14].

5 Conclusions

The paper described a case study aimed at evaluating the sensitivity of reliability growth models to errors in the estimation of the operational profiles. Three models (Geometric, Littlewood and Verrall Bayesian Quadratic, Brooks and Motley Poisson) offered an acceptable fit with the failure data available and were used for the evaluation. Results show that the predictive accuracy of the models, is not heavily affected by errors in the estimation of the operational profile. But, the type of experiment (a single case study with no replication) limits the applicability of these results to different development projects and reliability estimation activities. Results also show a strong relation between the number of test cases executed and the sensitivity of models to errors in the estimation of the operational profile. This relation, that has not been emphasised in previous researches, could explain the apparent contradiction between the results obtained in [3] and in [12]. Reference [12] addresses field failure intensity, that is, failure intensity experienced in operation, when presumably software has been carefully tested. If this condition is satisfied the results of this work, with the limits implicit in a case study, confirm the qualitative conclusions of reference [12].

Acknowledgements

We would like to thank the Company IDS and the ESA for providing the program used for the experiment. We also would like to thank all the unknown reviewers that provided comments to improve the paper. Unfortunately, the length constraints imposed by the Publisher did not allow the inclusion of some of the clarification and extensions suggested. An extended journal version of the paper, presenting additional results and a more detailed results analysis, will take care of most of those comments.

References

[1] A. A. Abdel-Ghaly, P. Y. Chan and B. Littlewood, "Evaluation of competing software reliability prediction", IEEE Transactions on Software Engineering, Vol. SE-12, No. 9, 1986.

[2] D. B. Benson, "Magellan spacecraft will need frequent guidance from Earth", ACM Software Engineering Notes, Vol. 15, No. 2, 1990.

[3] M. H. Chen, A. P. Mathur, and V. Rego, "A Case Study to Investigate Sensitivity of Reliability Estimates to Errors in Operational Profile", Proc. of the Fifth Int. Symposium on Software Reliability Engineering, IEEE Press, 1994.

[4] R. Cheung, "A User Oriented Software Reliability Model", IEEE Trans. on Software Engineering, Vol. SE-6, No. 3, 1980.

[5] A. Currit, M. Dyer and H. Mills, "Certifying the Reliability of Software", IEEE Trans. on Software Engineering, Vol. SE-12, No. 1, 1986.

[6] F. Del Frate, P. Garg, A Mathur and A. Pasquini, "On the correlation between code coverage and software reliability", Proc. of the "Sixth International Symposium on Software Reliability Engineering", IEEE Computer Society, 1995.

[7] W. W. Everett and M. Tortorella, "Stretching the paradigm for software reliability assurance", Software Quality Journal, Vol. 3, No. 1, March '94.

[8] W. H. Farr, O. D. Smith, "Statistical Modeling and Estimation of Reliability Functions for Software - User's Guide", NSWC DD TR 84-373, 1993.

[9] IEEE Computer Society, "IEEE 1044 - Standard Classification for Software Errors, Faults and Failures", IEEE Computer Society, 1994.

[10] B. Littlewood, A. A. Abdel-Ghaly, and P. Y. Chan, "Tools for the Analysis of the Accuracy of Software Reliability Predictions", Software Systems Design Methods, ed. J. K. Skwirzynski, NATO ASI Series, Vol. F22, Springer and Verlag, 198, pp.299-333.

[11] J. D. Musa, "Operational Profiles in Software Reliability Engineering", IEEE Software, Vol. 10, no. 2, March 1993.

[12] J. D. Musa, "Sensitivity of Field Failure Intensity to Operational Profile Errors", Proc. of the Fifth Int. Symposium on Software Reliability Engineering, IEEE Press, 1994.

[13] J. D. Musa, "Adjusting Measured Field Failure Intensity for Operational Profile Variation", Proc. of the Fifth Int. Symposium on Software Reliability Engineering, IEEE Press, 1994.

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

[15] E. Nelson, "Estimating Software Reliability from Test Data", Microelectronics and Reliability, Vol. 17, Pergamon Press, New York, 1978.

[16] A. Pasquini, "Reliability Growth Modelling of Software for Process Control Systems", Proc. of the International Software Symposium on Software Reliability, IEEE Computer Society Press, 1994.

[17] S. L. Pflegger, "Experimental Design and Analysis in Software Engineering", to appear in Annals of Software Engineering.

Table and Figures

Fault class	Fault type Subtype (when necessary)	No. of faults in the progr.
300	Logic omitted or incorrect	
311	Forgotten cases or steps	2
314	Unnecessary function	1
316	Missing condition test	4
317	Checking wrong variable	2
320	Computational problems	
321	Equation insufficient or incorrect	10
330	Interface incorrect or incomplete	
333	Module mismatch	3
340	Data handling problems	
341	Data initialised incorrectly	1
342	Data accessed or stored incorrectly	10
350	Data problems	0
360	Documentation problems	0
	TOTAL	33

Tab. 1 - Faults re-inserted in the program

Model	Accuracy	Bias	Noise	Trend	Goodness to fit test		
					Kolmogorov Distance	Chi-square Statistic	Signific. at 0.05
GE	78.69	0.54333	7.2834	0.30127	0.09166		no
LV	83.323	0.59371	3.687	0.31591	0.12478		no
MB	N. C.	N. C.	N. C.	N. C.	0.43031		yes
ML	N. C.	N. C.	N. C.	N. C.	0.13543		no
BB	124.95	N. A.	N. A.	N. A.		26.83	yes
PB	24.667	N. A.	N. A.	N. A.		1.827	no
N. A. = Statistics not available for interval data analysis							
N. C. = No convergence for these data and model							

Tab. 2 - Fit of the models with the real data

Model	Chi-square stat. for OP_e Error = 0	Chi-square stat. for OP_1 Error $=D(OP_e,OP_1)$	Chi-square stat. for OP_2 Error $=D(OP_e,OP_2)$	Chi-square stat. for OP_3 Error $=D(OP_e,OP_3)$	Level of signific. at 0.05
GE	1.827	2.083	2.817	3.912	37.7
LV	3.261	3.484	4.165	5.153	36.4
PB	3.439	3.56	4.063	4.834	37.7

Tab. 3 - Fit of the models in case of erroneous operational profile

P_n	→	P_{n-1}	→ ... →	P_{n-m}	→ ... →	P_0
T_0	→	T_1	→ ... →	T_m	→ ... →	T_n
Beginning of the testing phase		Detection and removal of the fault F_1		Detection and removal of the fault F_m and release of the program	The fault removal process continues occasionally during program operation and maintenance	Detection and removal of the fault F_n

Fig. 1 - Fault removal process of P

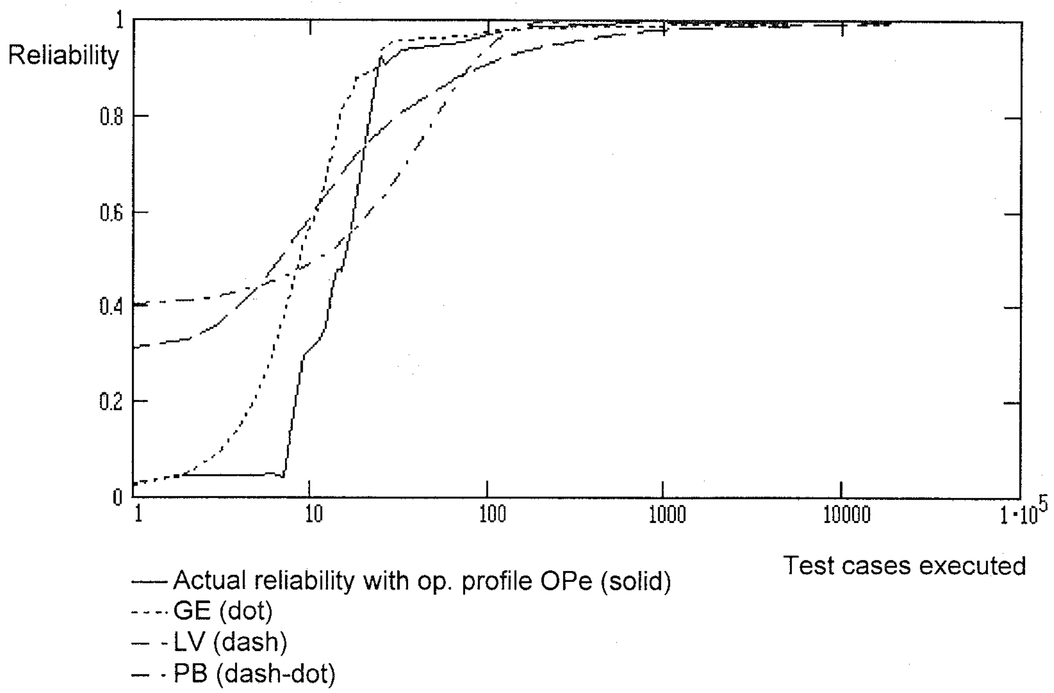

Fig. 2 - Actual reliability and model estimations for OP_e

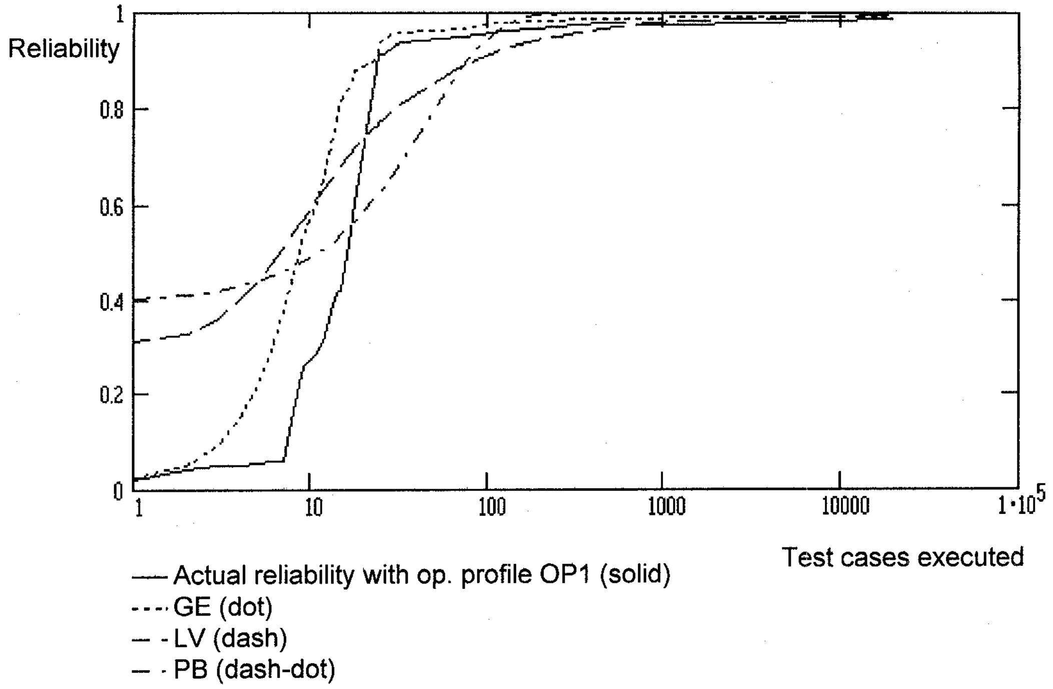

Fig. 3 - Actual reliability and model estimations for OP_1

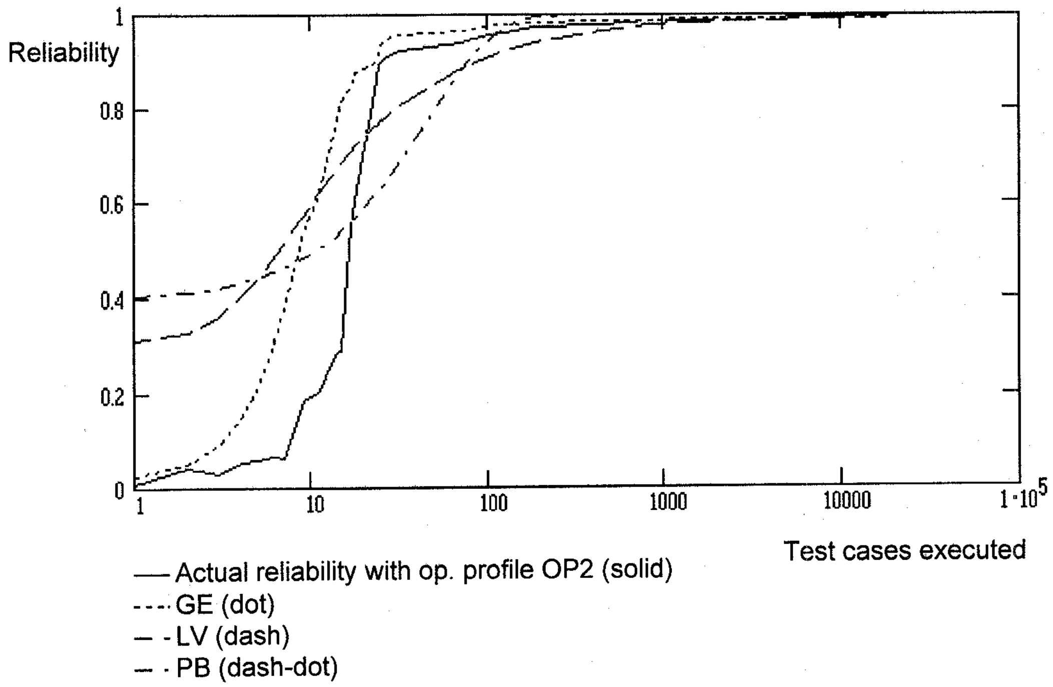

Fig. 4 - Actual reliability and model estimations for OP_2

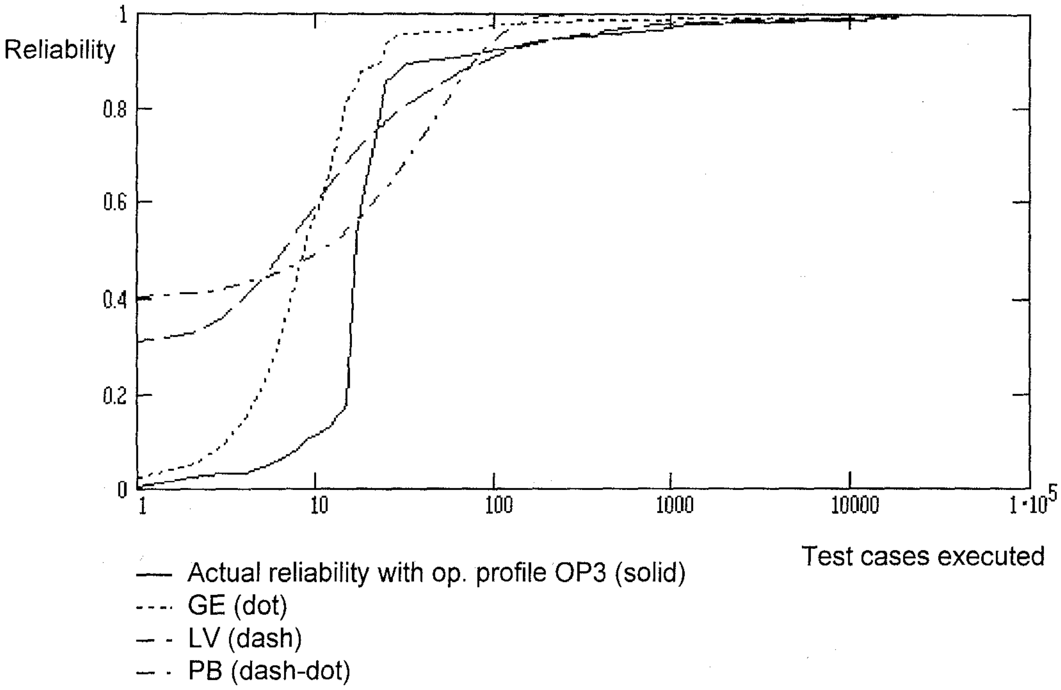

Fig. 5 - Actual reliability and model estimations for OP_3

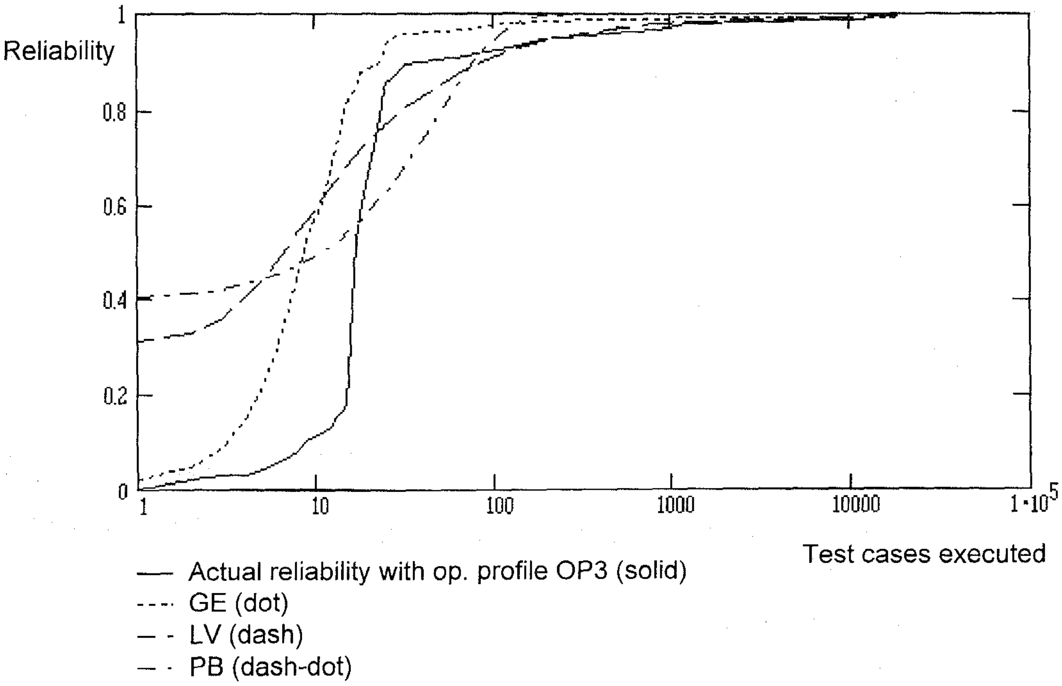

Fig. 6 - Actual reliability and model estimations for OP_3