Advanced Models for Software Reliability Prediction

Zigmund Bluvband, PhD, ALD ltd.

Sergey Porotsky, PhD, ALD Software ltd.

Michael Talmor , M.Sc., RAFAEL Ltd.

Key Words: software reliability, analytical models, cross-entropy

SUMMARY

This article describes the advanced parametric models for assessment and prediction of software reliability, based on statistics of bugs at the initial stage of testing. The parametric model approach, commonly associated with reliability issues, deals with the evaluation of the amount of bugs in the code. Computed parameter values inserted into the model allow to estimate:

(a) number of bugs remaining in the product, and

(b) time required to detect the remaining bugs.

Many models are developed for similar purpose: Duane Reliability Growth Model, Goel Model, Weibull Model, Classical S-shaped Model, Ohba S-shaped Model, etc. Taking into account some detailed, but practical, aspects of the software testing process, a few Advanced Models were developed and usefully implemented by the authors. The proposed models are sensitive to the situations typical for the early stages of Software development. As a result, one deals with the essentially non-linear, multimodal goal function to define the optimal value as the estimation of the unknown control parameter. To support the optimization of such complex models, the Cross-Entropy Global Optimization Method is proposed. Some authentic numerical examples are considered to demonstrate the efficiency of the proposed models.

1 INTRODUCTION

One of the main questions in the Reliability Software Analysis and Test Planning is: “When will the Software product be ready for release?” The answer to this question comes from the Quantitative Analysis, based on applicable parametric models.

In the Software Reliability analysis the following time-series are used:

- "Cumulative Effort of testers from start of testing" (CE), as an input indicator
- "Cumulative Amount of Bugs" (CB), as an output one

From the bugs’ trend perspective, our model applies a nonlinear approximation for the cumulative data under analysis to estimate the number of bugs remaining in the product, and the time required to detect the remaining bugs.

.

2 CUMULATIVE EFFORT AS MAIN INPUT INDICATOR

Number of days from start of testing is typically used as an Input parameter in Software Reliability analysis. Application of this parameter is reasonable, when one assumes that amount of testers and their effort approximately don't change during the testing process. In this paper it is proposed to use the "Cumulative Effort of testers from start of testing” (person*days) instead of "Number of days". Nevertheless the standard worksheet reports don’t contain the detailed required information: how long every tester worked for the particular project during his working day (week). In that case it is not known whether the software has been tested by the tester or the bugs have not been detected during the test.

That’s why the following Control Parameter is proposed:

- TP (Time Pause) - Significant Value of Non-Working Time

The assumption is: if a tester during a Time period longer than TP didn't insert records on bugs found, then he/she didn't work on a given project during this Time period.

Assume, that TP = 7 days. If single tester hasn’t record bugs during period of 5 days, we will take into account his effort (5 person*day) for Cumulative Effort calculation. Therefore, we suppose that the tester has really worked on this project without recording bugs each day, but once a week (it is typical situation of real statistics!). Nevertheless, if there are no bug records during two weeks (more than TP selected value = 7 days), we will assume, that he/she hasn’t work on this project these two weeks.

Certainly, this assumption is an integrated rough approximation over a group of testers, i.e. it is recommended to determine the TP value for the entire software project.

Using this approach, one can estimate the Cumulative Effort for each single tester and then to get the general Cumulative Effort for the entire project.

Value of TP essentially influences for the "Cumulative Effort" depending on "Quantity of testing days". For example, we will get the following curves (see Fig. 1):

- ______ for TP = 7 days
- - - - - - - for TP = 14 days
- ……… for TP = 28 days
- _._._._. without defining and using the TP (TP = infinite)

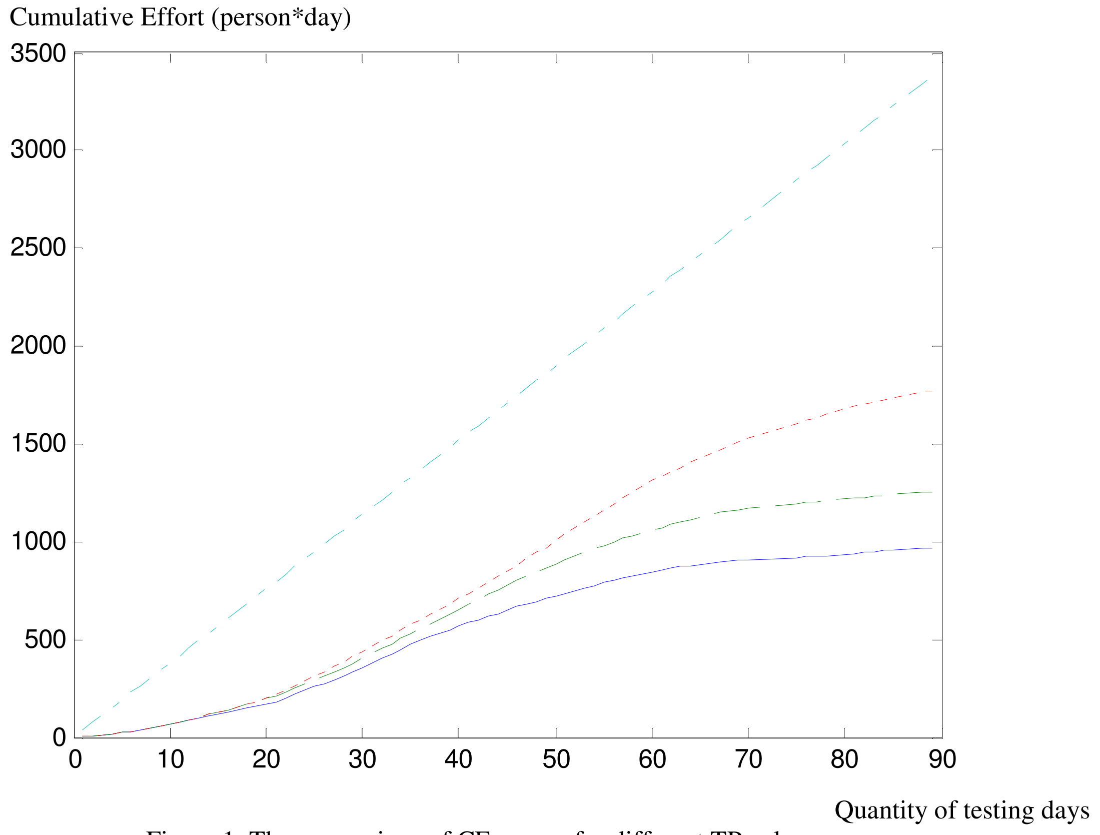

Figure 1. The comparison of CE curves for different TP values

For each single project study it is recommended to select the Optimal Value of TP (as well as all other control parameters) by means of the least-squares method – to minimize the difference between the cumulative bug distribution curves (analytical and the statistical).

3 PARAMETRIC MODELS FOR BUG AMOUNT PREDICTION

Many models are developed for Bug Amount Prediction: Duane Reliability Growth Model (that assumes infinite amount of failures that can occur in infinite time), Goel Model, Weibull Model, Classical S-shaped Model, Ohba S-shaped Model (that assume finite amount of failures, which can occur in infinite time), etc [1]. In the multiple projects the authors worked on, the modified Ohba S-shaped model was the most suitable for Software Reliability estimation:

$$CB = N * \frac{1-\exp(-b*CE)}{1+C*\exp(-b*CE)} \tag{1}$$

Where:

- N is the expected total amount of bugs to be eventually detected;
- CB – amount of the Cumulative Bugs;
- CE –amount of the Cumulative Effort;
- N, b, C - unknown control parameters.

This model is much more flexible one, but it does not take into account some specific features of Software Testing Process:

- Rate of bugs' discovery sometimes essentially depends on tester familiarity with the project – i.e. generally speaking, the more the tester experience is in this project, the higher the rate of bugs' discovery is.
- Rate of bugs' discovery sometimes essentially depends on the CE rate of the tester – generally, the bigger the CE Rate is, the higher the rate of bugs' discovery is.

To take into account some detailed aspects of the software testing process, authors have developed following Advanced Model:

$$CB = N * \frac{1-\exp(-b*CEM)}{1+C*\exp(-b*CEM)} \tag{2}$$

Where for the time-series index i:

$$CEM(i+1) = CEM(i) + \Delta CEM(i) \tag{3}$$

$$\Delta CEM(i) = \Delta CE(i)*(CE(i)^{\mathbf{f}})*(Mean\_Derive(i)^{\mathbf{p}}), \tag{4}$$

$\Delta CE(i) = CE(i+1) - CE(i)$,

- p and f are additional unknown control parameters,
- CEM is a Modified Cumulative Effort,

(5)

Mean _ Derive(i) is the Mean Value of the CE derivative:

$$\text{Mean\_Derive}(i) = \frac{\sum_{j=1}^{M} \text{Derive\_CE}(i-j+1)}{M} \tag{6}$$

Where:

- M = interval of averaging of Derivative of CE (it is also control parameter)

$$\text{Derive\_CE}(k) = \frac{\text{CE}(k) - \text{CE}(k-1)}{t(k) - t(k-1)} \tag{7}$$

- t(k) – day number at the time-series at index k.

To evaluate model parameters based on input statistics, one can use the different methods:

- Maximum Likelihood Estimation (MLE), based on Probability Density Function
- Least Squares Method, based on Cumulative Density Function
- Method of moments
- Etc.

In our case we don't use suspensions (censored times), so it is possible to use Least Squares method. We have to select values of our control parameters by means of minimization of the sum of Least Squares of measured and calculated values of CB.

4 PARAMETRIC MODELS FOR BUG RATE LOW VALUES PREDICTION

The described approach is applicable only for non-rare bugs. But Software standards dictate strict requirements to the software safety within the system application. For example, the corresponding claim limit for SIL 3 (Safety Integrity Level) in IEC61508 [2] is defined as 10^{-6} failures (bugs) per hour.

For systems with so high requirements for software reliability we could not support rest of the bug amount, as in the above model. It is rather necessary to ensure that in the field the Bug Rate will be less than some pre-defined value. For example, for the above SIL 3 requirement, one should finish the test at the rate equal or less 10^{-6} *K,
Where K is the test-to-field bug rate ratio (K ≈ 30-50).

Therefore, in the case of K = 50, the end-of-test bug rate requirement will be about $10^{-6} *50 = 5*10^{-5}$.

Is it possible to check and define this end-of-test time point manually, by means of direct calculation of bug rate based on measurement data? Unfortunately, the answer is – NO.

But for the question: Is it possible to check and define the end-of-test time by means of calculation/prediction of bug rate based on some software reliability analytical models? The answer will be, in most cases, - YES.

To illustrate this fact, consider following example. Suppose, that group of 4 - 8 testers, working 6 - 10 hours per day, discover 20 critical bugs, i.e. bugs of urgent severity during 3 months. Question – is it tested enough time or one should continue to test and find bugs of urgent severity? Suppose, we have following statistics about discovered bugs of urgent severity type (see Table 1 and Figure 2 below). Last row (Suspended/censored time) corresponds to the last test period without failure.

It is very difficult to estimate current Bug Rate, based only on measurements. For example, on the base of 5 last rows, one can get the Rate ≈ 4/(5400 – 1481) = 0.001 bugs/hour, which is very far from the required value of $5*10^{-5}$ bugs/hour.

To evaluate Bug Rate, we will use parametric models. Consider, for example, the Goel model:

$$CB = N*(1-\exp(-b*CE)) ,$$

$$Rate = N*b \exp(-b*CE) \tag{8}$$

Where:

- CE is Cumulative Effort (input parameter, i.e. variable)
- CB is Cumulative Amount of Bugs,
- Rate is Bug Rate
- N is the expected total amount of bugs to be eventually detected
- N and b are unknown control parameters.

In our case we essentially use suspensions (last row of the Table 1), so one should use MLE method.

Table 1. Cumulative Bug and Cumulative Effort Values

Cumulative Bug (CB)	Number of working day	Cumulative Effort (CE) from testing start, person*hours
1	1	1
2	2	60
3	2	87
4	4	168
5	6	258
6	7	269
7	8	309
8	9	423
9	10	474
10	10	505
11	13	600
12	16	731
13	17	808
14	20	927
15	21	1017
16	24	1173
17	30	1481
18	44	1906
19	44	1918
20	59	3063
S (no bug)	100	5400

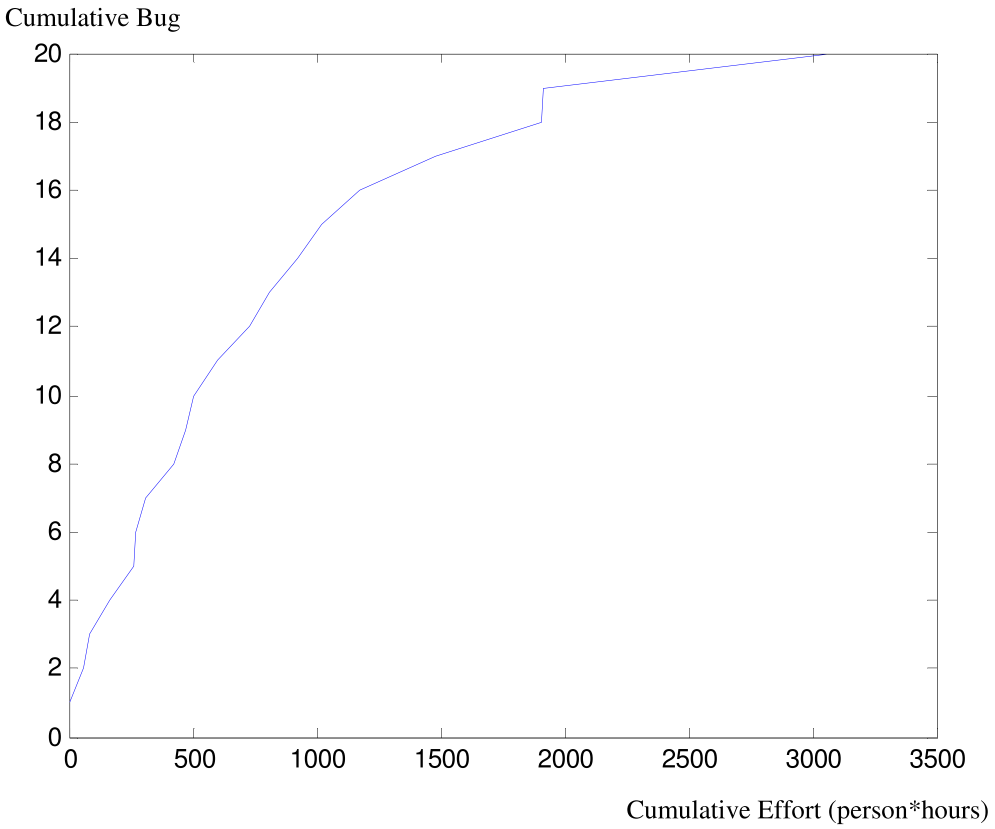

Figure 2. Amount of Cumulative Bugs Vers Cumulative Effort

By means of MLE, one can obtain the following values of model parameters:

N = 20.03, b = 0.00123.

Basing on these values, one can calculate Bug Rate for CE = 5400. The calculated value is $3*10^{-5}$, so it is possible to stop the testing.

These models can be used not only for current situation analysis (is Software ready for release?), but also for the prediction. If the answer for the question ”is Software ready for release?” is currently negative, it is necessary to predict the expected time of end-of-testing - to satisfy the required value of the "Bug Rate" in the field: and to ask "When will Software be ready for release?"

Suppose, that last row (Suspense/Sensor Time) has value 3200 (instead of early used 5400).

20	59	3063
S (no bug)	65	3200

By means of MLE solving, one can obtain the following:

N = 20.6, b = 0.0011

Basing on these values, one can calculate Bug Rate for CE = 3200. Calculated value of the Bug Rate is 0.0007, which is not enough to stop testing.

How to predict, when the test can be stopped?

Given there will not be bugs detected in the farther test continuation, the simplest approach typically leads to use of the above obtained parameters (N = 20.6, b = 0.0011) and to find appropriate value of CE, at the point where the corresponding Bug Rate will be less than $5*10^{-5}$.

Unfortunately, this approach isn't correct, because the above obtained N and b values correspond to the old value of suspended (last not-failure) time.

In this case it is necessary to calculate the new appropriate values of N and b – for each assumed end-of-test's CE.

So, one rather should find new values of parameters N and b, especially for the current value of suspended CE.

For that purpose additional control parameter CE_Critical is introduced, and then the following (a little more complex) task should be solved:

To find {CE_Critical, b, N} s.t. Rate = $5*10^{-5}$, where Rate = N*b*exp(-b*CE) and parameters' values b, N are maximizing the MLE function of above table with un-known value of CE_Critical instead of last row.

This is 3-parametric non-linear optimization task and after its solution one can get the following values:

CE_Critical = 4880, b = 0.00122, N = 20.05,

Rate = $5*10^{-5}$

So, the testing should be continued until the CE will reach the value of 4880 person*day (if, certainly, until this moment new bug will not be detected - in this case it is necessary to re-calculate the model parameters).

5 GLOBAL OPTIMIZATION APPROACH

After the model type building it is essential to define values of un-known (control) model parameters. Global Optimization of non-linear function is a common approach for such problems. For example, concerning problem of Parameters Estimation, a Linear Regression model can support only a few cases (e.g., Duane Model). In non-linear regression case, one has to search parameters by means of non-linear multi-modal and non-convex, global optimization. In this case the task is to get the value of **Z**, which provides min G(**Z**) under constraints Low[i] <= z[i] <= High[i], i = 1…K,

where:

- **Z** = {z[1],…,z[i],…z[K]} is a vector of parameters
- K is amount of parameters
- Low[i] is Low Boundary of Parameter i value (i = 1…K)
- High[i] is High Boundary of Parameter i value (i = 1…K)

G is some Goal Function depended on vector Z (in this case it is sum-of-difference-squared between the model-generated cumulative bug distribution curve to the actual cumulative bug distribution curve – see chapter 3; MLE function – see chapter 4).

To solve this task, two different approaches can be used:

- Write and transform derivatives of Goal Function for each single situation, to solve system of non-linear equations, according to situations, to compare different obtained solutions, etc.
- Use "direct search methods", provided universal search of Global Minimum (without analytical definition of derivatives).

For the first approach one should define complex analytical expressions for derivatives for each single situation.

It is proposed to use the second (universal) approach.

For Global Optimization Task one should use some Random Search oriented method – Cross-Entropy Optimization [3]. The method derives its name from the cross-entropy (Kullback-Leibler) distance - a well known measure of "information", which has been successfully applied in diverse fields of engineering and science. Initially the Cross-Entropy method was developed for discrete optimization, but later was successfully extended for continuous optimization [4].

6 CONCLUSION

This paper presents two advanced analytical models for obtaining accurate results for software reliability prediction. First model takes into account some specific features of software testing process and it is based on well-known S-shaped Ohba model. However this advanced model is applicable only for non-rare bug testing. For the rare bug rate prediction other model is proposed, based on introduction of the additional control parameter - "last suspended time".

These two models essentially use control parameters and so there is a need for optimization – Least Squares for the first model and MLE for the second model. Cross-Entropy Global Optimization method has been used and so it is recommended both for the first and the second models.

REFERENCES

1. M. R. Lyu, Handbook of Software Reliability Engineering. McGraw-Hill, 1996
2. 61508 - Functional Safety of Electrical / Electronic / Programmable Electronic Safety-Related Systems, International Electrotechnical Commission (IEC), 1999
3. R. Y. Rubinstein and D. P. Kroese. "The Cross-Entropy Method: A unified approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning". Springer-Verlag, 2004
4. D. P. Kroese, S. Porotsky and R. Y. Rubinstein. "The Cross-Entropy Method for Continuous Multi-Extremal Optimization". Methodology and Computing in Applied Probability, 2006, 8(3) : 383–407.

BIOGRAPHIES

Zigmund Bluvband, Ph.D.
A.L.D. Ltd.
52 Menachem Begin Road
Tel-Aviv, Israel 67137

e-mail: zigmund@ald.co.il

Zigmund Bluvband is the President of Advanced Logistics Developments Ltd. His PhD (1974) is in Technical Sciences. He is a Fellow of ASQ and ASQ–Certified Reliability Engineer, Quality Engineer, Quality Manager and Certified Six Sigma Black Belt. Z. Bluvband has accrued more than 30 years of industrial and academic experience. Z. Bluvband was the President of the Israel Society for Quality (1989 – 1994). He has published more than 70 papers and tutorials, ten patents and four books. In 2006 Dr. Zigmund Bluvband has been honored with the IEEE Reliability Society Lifetime Achievement Award. In 2009 Dr. Z. Bluvband was elected as an IAQ (International Academy for Quality) Academician.

Sergey Porotsky, Ph.D
ALD Software Ltd.
52 Manachem Begin Road,
Tel-Aviv, Israel 67137.

e-mail: sergey.porotsky@ald.co.il

Sergey Porotsky is a Chief Scientist of ALD Software Ltd. He is an expert in analytical and numerical methods of Applied Mathematics. Dr. Porotsky is involved in design and optimization algorithms for reliability analysis of complex systems - Monte Carlo, Markov Chains, FTA methods. While with Moscow Scientific and Research Center of Computers, he developed analytical methods for Computer Systems and Networks Evaluation. His PhD is in Computer Science (1985) and his Dr. Sc. degree is in Operation Research (1992).

Michael Talmor , M.Sc.
Head of the Reliability Center, RAFAEL Ltd.
14A Savion St., Quriat Yam, 29500, Israel

e-mail: michaelt@rafael.co.il

Michael Talmor is a Head of the Reliability Center in RAFAEL Ltd. He holds MsC degrees in Electrical and Reliability Engineering. Since coming to RAFAEL he focused his efforts in Quality and Reliability of the Systems. His work includes Reliability and Availability modeling and analyses, Reliability testing and demonstration, Bayesian approach in reliability estimation, microelectronic reliability, Safety analyses, software reliability and safety, etc. He published several articles in different areas of his, more than 25 years, activity.