

THE MODEL FOR SOFTWARE QUALITY MEASUREMENT, USING THE “GENETIC” FEATURE OF THE SOFTWARE

Jekaterina Kokina
Department of Transport Electronics and Telematics
Riga Technical University
LV-1019, Lomonosova 1,
e-mail: jkokina@yahoo.com

KEYWORD: *Software quality. “Genetic” feature. Mathematical model.*

ABSTRACT

The author of this paper believe, that software system by itself is a complex object, which could be described via the Linear Mixed Model ANOVA. It includes different parameters, which could have fixed (non-random) or stochastic nature. There is produced an idea about software quality prediction by defect number using “genetic” feature of the software (Kokina 2003). It is believed that the term of “genetic” feature is an initial software feature of the software affecting the software from the moment it was initially developed. There are presented a mathematical model for “genetic” quality of software evaluation by Best Linear Prediction (BLP) method and some results of the experiment using simulation of this mathematical model in MATLAB here (Kokina and Petersons 2004).

Finally it is presented an idea of real complex software system simulation and a reference to the produced mathematical model for evaluation of the real software “genetic” feature in this paper. Currently the author is working at this idea and software model simulation.

This paper is a report of the current stage of the exploration work in the field of the software quality evaluation using the term of the “genetic” feature and simulation of produced model and the real application system. The previous report was presented in (Kokina 2004).

INTRODUCTION

The narrowest sense of software product quality is commonly recognized as lack of “bugs” in the product. Software reliability and quality fundamentals were discussed in one on the previous publications (Kokina 2003). The software quality (SQ) could be determined as defect rate or number of defects (NOD) per unit of source code, for example, per million lines or per function point, or other units. Despite developers’ attempts at checking every detail, errors are made during development and very often cause end-release software failures. Some errors could be found in the software even after very long and good testing.

The author of this paper supposed that software „quality”, measured in NOD, is in high dependence of software birth conditions: main system requirements and system complexity, human factor of development

team - number of developers and testers, their skills, development place and tools and so on. This set of factors stipulates for the “genetic” feature of program, which characterizes initial quality of the system. Then it makes sense to make the assumption, that more trained and more equipped fathers using the best development tools develop more qualitative product, i.e. product with the minimum NOD.

The aim of the exploration work is to predict and evaluate software quality using a term of “genetic” feature of the software. We present Software Genetic evaluation model, based upon Best Linear Prediction (BLP) method. This method was proposed by Henderson and was used in biology, i.e. biometrics, originally and then in many other fields.

The target of this exploration work is try to predict and estimate the software “genetic” feature, making the analysis of a potential way to get better software reliability and quality.

THE PROBLEM DESCRIPTION

The software system under testing itself is a complex object, which may be described by a set of different parameters, which may have a fixed (non-random) or stochastic nature.

Suppose the software under testing at the time moment t is characterized by the NOD $y(t)$: the less is the NOD, the higher SQ is (Kokina 2003). Further testing allows to reveal program errors, which number become less and less during the testing process

The number of fixed and random factors affects number of defects raised during testing. The number of fixed and random factors affects NOD raised during testing. The fixed factors for instance are financial resources for development, programming language specificity of the development sphere, development deadlines and other. The random factors are software quality and the noise, representing the not considered factors

The stochastic SQ is like the „genetic” feature, affecting the software from the moment it was initially developed. We can see, that it is very difficult to generally determine, what the „genetic software quality“ is, but we strongly believe, that this parameter must be taken into account in the planning and design of the software.

MATHEMATICAL MODEL

The distribution of time relation is indirect to SQ. We suppose the tie between the controlled and measurable parameter $\mathbf{y}(\mathbf{t})$, the NOD found during the testing process in time moment \mathbf{t} , and uncontrolled random parameter \mathbf{s} , which defines the “genetic” feature of the software in our case.

This relation may be expressed via the linear mixed ANOVA model (Henderson 1984):

$$\mathbf{y} = \mathbf{X} \mathbf{h} + \mathbf{Z} \mathbf{s} + \mathbf{e} \quad (1)$$

where:

\mathbf{y} - the array of controlled parameters (NOD) with the dimension $\mathbf{n} \times \mathbf{1}$;

\mathbf{h} - the array of fixed parameters with the dimension $\mathbf{q} \times \mathbf{1}$;

\mathbf{X} - incidence matrix, consisting from $\mathbf{1}$ and $\mathbf{0}$, dimension $\mathbf{n} \times \mathbf{q}$; specifies which components of \mathbf{h} is taken into account;

\mathbf{s} - the array of uncontrolled random parameters with the dimension $\mathbf{r} \times \mathbf{1}$, having \mathbf{m}_s expectation for BLP case and zero expectation for BLUP and the variance $\mathbf{G} \times \sigma_s$.

\mathbf{Z} - incidence matrix, consisting from $\mathbf{1}$ and $\mathbf{0}$, dimension $\mathbf{n} \times \mathbf{r}$; specifies which components of \mathbf{s} is taken into account;

\mathbf{e} - the noise, having zero expectation and the variance $\mathbf{R} \times \sigma_e$.

Our task is to predict the numerical values of \mathbf{s} , knowing the values of \mathbf{y} . This is the reverse task to the classical regression problem.

It is known from the literature (Henderson 1984), that the Mixed Model Equations (MME), based on BLP and BLUP techniques should be used to calculate the values of \mathbf{s} . If we assume, that there is no correlation between \mathbf{s} and \mathbf{e} , than MME will look as follows:

$$\begin{bmatrix} \mathbf{X}^T \mathbf{X} & \mathbf{X}^T \mathbf{Z}_1 \\ \mathbf{Z}_1^T \mathbf{X} & \mathbf{Z}_1^T \mathbf{Z}_1 + \mathbf{I} \gamma \end{bmatrix} \begin{bmatrix} \mathbf{h} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T \mathbf{y} \\ \mathbf{Z}_1^T \mathbf{y} \end{bmatrix} \quad (2)$$

where $\gamma = \sigma_e^2 / \sigma_s^2$,

SQ values \mathbf{s} for BLP technique could be derived from the equation below:

$$\mathbf{s} = \mathbf{m}(\mathbf{s}) + (\mathbf{Z}^T \mathbf{Z} + \mathbf{I} \times \gamma)^{-1} \mathbf{Z}^T (\mathbf{y} - \mathbf{X} \mathbf{h}) \quad (3)$$

for BLUP technique

$$\mathbf{s} = (\mathbf{Z}^T \mathbf{Z} + \mathbf{I} \times \gamma)^{-1} \mathbf{Z}^T (\mathbf{y} - \mathbf{X} \mathbf{h}) \quad (4)$$

where

$$\mathbf{h} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (5)$$

RESULTS OF THE MATHEMATICAL MODEL SIMULATION

It was decided to approve stated above theoretical theses as applied to the problem of program «genetic» parameters definition (Kokina and Petersons 2004). We put some experiments with known beforehand results, which should give a model. Besides of that it was revealed the rate of the algorithm convergence to the known results.

During the experiment we assume, that the \mathbf{y} array consists of random values with normal distribution, which determines NOD in every of \mathbf{t}_i error revisions. The mean value of $\mathbf{y}(\mathbf{t}_i)$ decreasing with exponential distribution, as in the most of software reliability models:

$$\mathbf{y}(\mathbf{t}_i) = e^{-\mathbf{t}_i} + \mathbf{a}, \quad (6)$$

where \mathbf{a} determines some “genetic” error level, i.e. NOD, which are appearing in program in birth stage and not found during the testing process, but could appear in production after short or long time.

We believe also the matrix \mathbf{X} with dimension $\mathbf{n} = 20$ and $\mathbf{q} = 3$ as a constant, the matrix \mathbf{Z} with dimension $\mathbf{n} = 20$ and $\mathbf{r} = 4$ as a variable value. The evaluation of \mathbf{s} with dimension 4×1 were made for following cases of \mathbf{Z} matrix: 1) one parameter of \mathbf{Z} array is taken into account, i.e. one component of matrix \mathbf{Z} has value $\mathbf{1}$, but others components of matrix are $\mathbf{0}$, 2) more, than one parameters of \mathbf{Z} array are taken into account, i.e. more, than one component of matrix \mathbf{Z} have value equal to $\mathbf{1}$, but not all, others components of matrix \mathbf{Z} are $\mathbf{0}$, 3) all parameters \mathbf{Z} array are taken into account, i.e. all components of matrix \mathbf{Z} have value equal to $\mathbf{1}$. We assume various values for γ also: 1) $\sigma_e^2 = \sigma_s^2$; 2) $\sigma_e^2 < \sigma_s^2$; 3) $\sigma_e^2 > \sigma_s^2$.

During the experiment of the mathematical model simulation the only one «genetic» parameter was evaluated, witch is NOD in the software concerned with its «genetic» feature. We assume, that the array of the \mathbf{s} parameter also consists of random values with normal distribution and known expectation mean value $\mathbf{m}(\mathbf{s}) = \mathbf{a}$, we get some estimation results, shown on Figure 1 and Figure 2. For model simulation the tool Matlab 5.0 was used.

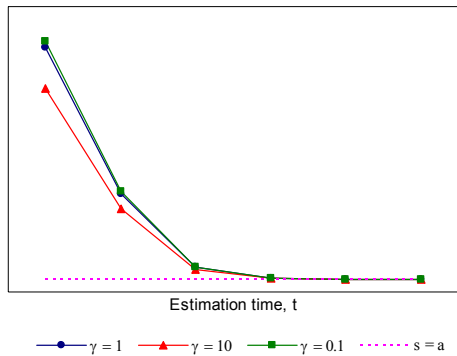


Figure 1. Uncontrolled random parameters (SQ) distribution as function of time by BLP method. One parameter of Z array is taken into account.

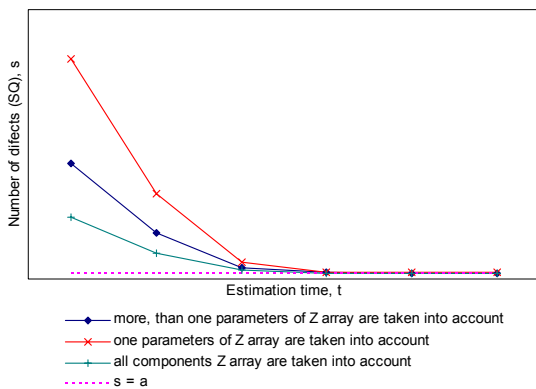


Figure 2. Uncontrolled random parameters (SQ) distribution as function of time by BLP method. $g = 1$. Z matrix has different values for array parameters

Figure 1 and Figure 2 demonstrate evaluation results of simulation model, which we get using BLP method and the formula (3). The Figure 1 is $s(t)$ - uncontrolled random parameters (SQ) distribution for case, when Z is constant, but parameter γ is a variable value. The Figure 2 is $s(t)$ SQ distribution for case, when γ is constant, but Z a variable value. We can see that both curves in the beginning of the testing process have more defects (NOD), than in the end of testing. The NOD as parameter of SQ is decreasing as an exponent and strives to a constant value $s = a$, meanwhile the actual quality of software is increasing. So we reveal, that value $s = a$ actually is the “genetic” software feature in our experiments.

The given graphical results represent the rate of the of the NOD algorithm convergence to the known

beforehand result. The more is $\gamma = \sigma_e^2 / \sigma_s^2$, relation of noise to the uncontrolled random parameters, the more is the rate of the algorithm convergence (Figure 1.). The less is the number of the s components, which has influence to SQ, i.e. “genetic” parameter, the smaller rate of NOD distribution is (Figure 2.).

In the real life software development process we do not know the expectation $m(s)$ of uncontrolled random (SQ) beforehand, that is why is could be used the other approach for “genetic” software feature’s evaluation using BLUP method, which assumes zero expectation for s value. Then the analysis of s distribution for BLUP could be such: s parameter is negative for almost all time of testing, which means bad SQ, meanwhile positive values of s define good SQ, and the value, for which s is streaming in time by exponent could be accepted as “genetic” software feature.

REAL SYSTEM SIMULATION

Two previous paragraphs discussed the mathematical model of software system for initial quality of software evaluation using system parameters in the output of the software system. This part of the paper produces a model description for real application system simulation by traditional way, i.e. using input parameters:

1. Suppose we have three urns, which simulate three real application systems under test.
2. Let’s put a random number of red balls N, which means the number of program code lines with errors, into each urn with the known expectation and the known variance.
3. Let’s put the known number of white balls M to the every urn. The white ball means the line of application code without errors. The real number of program code lines could be taken from the real software systems.
4. When red and white balls (wrong and right code lines) are distributed among the urns (software) let’s take out balls from each urn. This step corresponds to the software testing and error prevention process: if a chosen ball is white, it should be returned back into the urn, where it was initially placed; if the chosen ball is red, it should be taken out from the urn and replaced by a white ball – wrong code line is repaired to a right one. Here we believe, that: a) color of the ball is defined correctly; b) there is an error in color of ball definition: probability P_1 – color of the ball is defined rightly, P_2 - not rightly.
5. Make a prediction of number of red balls in each urn according to the BLUP method (see previous paragraphs).
6. Do the next selection (testing iteration) of balls from urns: repeat steps 4 and 5 i times.
7. Compare results of BLUP method evaluation (from 5th step) with number of red balls after i iteration.

We suppose, that after realization of this scenario number of defects in the software is very small and it corresponds to results received by BLUP before and called by software “genetic” feature.

There are many examples of software reliability research works, which show the similar NOD distributions in real life system development and testing (Wood 2003).

The described scenario realization for software system simulation is developed in C++. It shows that our supposition was right, but the task of theoretical results comparison with simulated system is not finished yet. This is the reason, why these results are not published here.

CONCLUSIONS

This exploration work shows that software quality could be predicted and evaluated using a term of “genetic” feature of the software and stated above theoretical theses. Currently the author is working in research of the theoretical approach application to the real software system using the simulation model described overhead. We believe that our SQ estimation approach of software “genetic” feature is a potential way in research of good reliability and quality for application system.

REFERENCES

- Kokina J. 2003. “Software Reliability And The “Genetic” Feature Of The Software”. *Riga Technical University Scientific Proceedings 2003.*
- Kokina J., Petersons E. 2004. “The Mathematical Model For Measurement Quality, Using The “Genetic” Feature Of the Software”. *Riga Technical University Scientific Proceedings 2004.*
- Henderson Charles R. 1984. “Application of Linear Models in Animal Breeding. University of Guelph”.
- Wood Allan P. 2003. “Software Reliability From the Customer View”. Computer, August.

AUTHOR BIOGRAPHIE



JEKATERINA KOKINA was born in Riga, Latvia and went to the Riga Aviation University, where she studied telecommunications and obtained Master degree in 1999. She worked 6 years for DataPro company and then 2 years for the MicrolinkLatvia company as a software testing specialist. From the 2002 she is a post-graduate student in Riga Technical University, working in the field of the software testing and quality with the assistance of the professor Ernests Petersons. J.Kokina participated in international RTU conferences in 2003 with a report “Software Reliability and The “Genetic” Feature of The Software” and in and 2004 with report “The Mathematical Model For Measurement Quality, Using The “Genetic” Feature Of the Software” E-mail address is: jkokina@yahoo.com.