Simulation of Software Security Tests by Soft Computational Methods

Alexey S. Markov
Bauman Moscow State Technical University
Moscow, Russia
a.markov@bmstu.ru

Georgii A. Markov
Jet Infosystems, JSC
Moscow, Russia
gm@cnpo.ru

Valentin L. Tsirlov
Bauman Moscow State Technical University
Moscow, Russia
v.tsirlov@bmstu.ru

Abstract — The worth of the using of software performance reliability assessment models at early testing stages is shown. The characteristic property of nonmonotonic increasing of performance reliability of current software systems is underscored. Ways of enhancing adequacy and fidelity of software testing simulation are exemplified. The conditions of using of soft computing for assessing the reliability and security of information systems software are shown.

Keywords — software performance reliability, software testing, reliability growth models, debugging models, fuzzy models, artificial neural networks, soft computing

I. INTRODUCTION

Approval of mandatory requirements for formalizing the results of compliance assessment of high-confidence information protection software [1] lent relevance to the use of mathematical models for assessing IS software performance reliability and safety. Although studies into mathematical models for software reliability assessment were first undertaken back in the second half of the last century [2, 3], these issues remain relevant nowadays [4-16]. All this relates both to software manufacturing technologies (for example, open source software) and to newly introduced international standards in the software engineering and information security area (ISO/IEC 15408, ISO/IEC 33001, IEC 61508, IEC 61511 etc.) [7, 17, 18]. It should be understood that the concept of reliability is equivalent to that of software safety, if it is assumed that all detected defects (errors) and vulnerabilities are not strictly identified as deliberate. Therefore, the software testing models under review herein will be conventionally termed as reliability models.

As is well known, development, selection and synthesis of mathematical models are directly dependent on the statistics gathering subsystem implemented in the software life cycle. This article investigates the possibility and applications of soft computing as part of mathematical test modeling and software maintenance.

II. DEBUGGING MODEL

There are a few taxonomies of mathematical software reliability models known at present [2, 19-22]. We believe that, from the applied perspective, it is convenient to use the classification relating to the objectives of software product/system testing stages [21], which allows 4 types of mathematical models to be singled out:

- Test planning models taking into account software complexity (program complexity models);
- Debugging models oriented toward multiple updates and input data domain coverage (data-domains models);
- Time-dependent reliability growth models intended for trial and production use (time-domains models);
- Test confidence models.

Debugging models are the most sensible to the statistics gathering system, as it is initial testing (debugging) stages that are characterized by multiple changes in test object versions, including in various media and platforms. The first offered models of the specified class are Nelson and LaPadula models plus their modifications [3, 21-26] which imply a strictly monotonous growth of software reliability. At present, this is not always consistent with current multiversion programming technologies. To rule out the above drawback, the authors gave a thorough study to software implementation of probabilistic reliability growth model allowing some additional positive and negative debugging factors to be taken into account, namely [21]:

\[ P_u = P_0 - (P_\infty - P_0) \prod_{j=1}^{u} \left(1 - \frac{A_j}{A_j + B_j}\right), \]

(1)

where: \( P_0 \) — initial reliability level, \( P_\infty = \frac{A_j}{A_j + B_j} \) — ultimate reliability level, \( 0 \leq P_0 < P_\infty \leq 1 \). \( u \) — number of completed modifications; \( A_j \) — modification “effectiveness” factor characterizing an error probability decrease due to the \( j \)-th modification; \( B_j \) —
modification “negativeness” factor characterizing reliability degradation caused by the \( j \)-th modification.

Depending on the available statistics, the test modeling/software debugging evolution may take, for example, the following paths:

- Updating a probabilistic model (1) parameters depending on the significance of a target software update class;
- Updating a probabilistic model (1) parameters on the code update complexity level;
- Bayes modifications of a model when updating the stochastic nature of its parameters;
- Developing an interval model based on the fuzzy sets theory machinery;
- Developing a model by means of artificial neural network technologies etc.

The last two modeling options are considered to pertain to soft computing, which will be discussed a bit later. Because Bayes modifications make model visualization (simplicity) parameter determination much more difficult, they were left outside this article. Yet, the above theoretical approach is considered in literature [27-30]. Let us touch briefly upon the issues of model (1) refinement by modifying model parameters.

III. ANALYTICAL MODEL REFINEMENT

As far as the updating of a probabilistic model’s parameters (1) is concerned, the most graphical model is the bigeminal debugging one taking account of opposite software modification classes, namely: correcting an error and introducing a new software update. As far as the updating of a probabilistic model when updating the stochastic nature of its parameters, the bigeminal model (2) depends on 4 parameters (\( P_0, P_\infty, a_1, a_2 \)) easy to calculate, for example, using the maximum likelihood method [21]. No restrictions are imposed on the domain of defining the modification scope factor \( k_{ij} \), to allow consistency and completeness of describing the testing and debugging process. Thus, where statistics on source code modification are available, it is easy to utilize the known complexity metrics. For machine-oriented and procedural (also partially object-oriented) languages, it is easily implemented by making use of heuristic and statistical complexity metrics (Halstead, McCabe, Chepen etc.). In case of visual programming systems, the introduction of popular complexity metrics may sometimes fall short of weary anticipations. Ultimately, they may be simply reduced to length or scope of modified fragments.

Practical modeling experience showed that adequacy and fidelity could be enhanced by taking into account the phenomenon of nonmonotonic growth of multiversion software reliability and diverse wave errors. Graphically, the software reliability variation process is represented by a step reliability growth function (Figure 1).

![Fig.1 Reliability degree variation after updates](image)

IV. USE OF SOFT COMPUTING

Soft computing is normally undertaken in apparent cases of available statistics on testing and debugging software products and systems. As mentioned before, the most popular approaches include the use of artificial neural system technologies [19, 31-33] and fuzzy set apparatus [34-37]. Given below are examples of developing such models.

A. 4.1. Using neural network technologies

A transition to neural networks stems from the difficulty of accounting updates in multiversion software systems (in particular, open source systems where programmers are distributed by time, style, technology and qualification). As a result, there are some regularities unclear at initial stages. One must realize that neural technologies can be implemented providing there are statistics representative enough in volume.

A package of computer modeling in Neuroshell2 (Pro) was chosen by a computer experiment to verify this approach [32]. As part of the study, the authors checked if modeling was possible by using allowed topologies of neural networks and selecting the optimum number of their parameters (number of neurons, layers, link types etc.). Source data and the computer experiment outcomes are presented in figures 2-5.

The study found even a simple 4-layer neural network and a network with bypass connection to have coped with the task set [31]. It should be realized that applying neural network technologies requires support by the relevant data gathering system and, naturally, accumulation of highly representative statistics.
it is convenient to pass on to the fuzzy model:

\[ \mathbf{P} = \{(P_u, \mu_P(P_u))\} , \]

where: \( P_u \) – software reliability degree in case of \( u \)-modifications; \( \mu_P(P_u) \) - membership function.

Showing the confidence level \( \alpha \) allows the desired interval fuzzy model to be obtained in general terms:

\[ \mathbf{P} = \{P_m | \mu_M(m) \geq \alpha \} . \]

The fuzzy model development is based on determining a range of fuzzy sets, such as:

- A fuzzy set \( \mathbf{A} = \{(x_i, \mu_A(x_i))\} \) representing a totality of ordered pairs of software modifications – the ground set \( X \) and the membership function \( \mu_A(x_i) \) characterizing the availability of software modification (update).
- A fuzzy set of accountable \( \mathbf{U} = \{(u, \mu_U(u))\} \), where \( \mu_U(u) \) - a membership function meaning the degree of confidence that the number of accountable modifications is equal to \( u \).

This approach, including membership function obtaining techniques, is described in detail in [21].

V. CONCLUSION

Notwithstanding the long-term investigations in the software testing, reliability and quality field, introduction and development of mathematical testing models remains sought-after. This, in particular, is specified in some international standards concerning both procedural guidelines as such and software safe life cycle procedures. At present, there is a representative set of mathematical models whose selection and synthesis depend on specific situations, including the reliability assessment of software products based on current and advanced software manufacturing systems.

The work shows the possibility of using soft computing in apparent cases of available statistics on testing current software systems. The results obtained using soft computing methods can be applied as initial when calculating parameters of analytical models.

This study is within the scope of development and introduction of a range of new standards harmonized with GOST R 56939, a national standard for safe software development [38].

REFERENCES


