

Software Reliability Growth Model`s Assumptions in Context of the Secondary Faults

Dmitry. A. Maevsky^{1,*}, Elena J. Maevskaya¹, Lyudmila N. Shapa¹

¹Odessa National Polytechnic University, Ukraine

Dmitry.A.Maevsky@opu.ua, E.J.Maevskaya@opu.ua, l.n.shapa@opu.ua

Abstract. The article describes the results of analysis of four general assumptions of all existing software reliability growth models: the assumptions about similarity of testing and operation conditions, immediate detected fault elimination, the assumption that secondary faults are not brought in and the one of mutual independence of faults. Under the real conditions of Software testing and operation none of these assumptions is executed. Because of this fact, all existing models possess low modeling accuracy. The authors have shown that the removal of only one of these assumptions – the one that secondary faults are not brought in – allows to automatically remove all other assumptions. A new Software reliability model created with considering the possibility to bring in secondary faults has demonstrated stably better results as compared to the traditional ones. This model is based not on the theory of probability, but on the theory of nonequilibrium processes.

Keywords: Software Reliability, Reliability modelling, Assumptions, Faults, Secondary Faults, Software Reliability Growth Models

Key Terms: MathematicalModel, SoftwareEngineeringProcess

1 Introduction

The history of software reliability theory began more than 60 years ago. The first paper, in which the words “Software” and “Reliability” are used together in a single word-combination, is likely the one written in 1965 [1]. Although the idea that the reliability growth is possible in complex systems was given much earlier – in 1956 [2]. Over this period of time a lot has been done in the software reliability theory, e.g. a great number of so-called “Software Reliability Grows Models” – SRGM –was created. We cannot know the exact amount of these models because of plenty of their modifications. According to the evaluations of many authors, the number has reached 60 units [3, 4]. All of these models in their essence try to describe the same physical process – the one of software reliability growth on account of the program fault detection. The cause of creation of such a great number of models is connected with the fact that none of them can describe the process with the accuracy enough for practical goals of all kinds of software. From the viewpoint of the resources consumed the creation of more and more new models actually reflected the process of replication,

In this fragment only common assumptions, which as one can see are present in all reliability models, are shown. The main difference of the models is the assumptions of parameters of fault detection process, which as we mentioned above are not the subject of the given paper. The first column of table 1 contains the codes of the common SRGM assumptions:

- I.1. Software testing and reliability assessment is performed in actual operating conditions.
- I.2. Faults are removed immediately.
- I.3. New faults are not brought in the process of debugging.
- I.4. All faults are occurred independently from each other.

Let us analyze each of these assumptions in detail

2.1 Assumption I.1. Conditions of Testing and Operating

The requirement to test software under the conditions of its running is likely connected with transferring the reliability evaluations, which are obtained at the testing stage, to the running stage. This requirement was completely met only during reliable model development – up to the middle 90th. However, we can argue that at present it (requirement) is not being met. Because software testing industry was actually created along with software development industry. Now the testing is performed by special groups of testers according to the special techniques and special software. Wherein the approaches to testing process and operation process are fundamentally different. The goal of testing is to find as large number of faults as possible for the shortest period of time. In the process of operation the fault detection relatively rare is the goal. Because by the beginning of software running the majority of faults as a rule are already found and corrected. And the faults detected in the process of operation are considered to be an undesirable result, and are not corrected immediately but for a long period of time. That is why in the course of testing the fault detection intensity is much (some orders) higher than during operation. But in all of the reliability models the following correlation is supposed: the higher the fault detection intensity the lower the reliability. So, at present the reliability data indicated at the running stage must not be generally transferred to the testing stage.

However, all the above mentioned considerations do not actually limit the reliability evaluation made with the help of reliability models at the running stage. The point is that any software being thoroughly tested all the same possesses a number of undetected faults at the moment of beginning of running. And if they are detected in the process of operation one can evaluate the reliability indexes according to the speed of their detection made with the help of models at the stage of running exactly.

So, we can rightly argue that assumption 1.1 is always performed.

2.2 Assumption I.2. Faults are Removed Immediately

The assumption of the immediate detected fault removal for all of the reliability models is connected with the fact that the function of dependence of the number of faults on time is monotonously decreased in these models. If to assume that the detected fault continues existing in program this function monotonousness is violated. Certainly, this assumption does not actually provide the fault correction at the moment it is detected, i.e. instantly. A fault can be corrected later, but the most important point is the interruption of the software testing and running in correcting. If the program is not executed with the help of the computer then from the position of fault detection process it does not exist. Thus, the software running time is discrete. And the resumption of testing and running processes after the fault correction provides this assumption performance.

2.3 Assumption I.3. New faults are not Brought

Of all the reliability model assumptions this one is the most exigent. The point is that the fault correction process requires the change of software code sections. The size of the changed sections is sometimes very large. Wherein there is always probability to bring new faults in correcting the previous ones. In the software reliability theory the faults brought again are called secondary ones. None of the existing reliability models has the possibility to take into account the secondary faults. And numerous attempts to account for secondary defects in the existing SRGM have shown that this is not always possible [8, 9, 10]. The low accuracy of simulation of the fault detection process is closely connected with this since one of the model assumptions is practically never carried out.

Hereinafter we will try to remove this restriction and understand how this procedure will influence on the remaining assumptions.

2.4 Assumption I.4. Independence of Faults from Each Other

The last fourth assumption is the one, which provides the detection of one of the faults independently from the detection of another one. That is from the viewpoint of Probability Theory we have a number of independent events [11]. This actually means that the fact of detection of one of the faults influences on the possibility or impossibility to detect other faults in no way. In practice, these events are really more often independent. However, there are cases when detection and removal of one of the faults give the fundamental opportunity to detect the other faults, which were hidden before. Here is the simplest example: as a result of an error in statement "IF" (assume that the equation is true in all cases) the control can be never given to one of its branches. In this case, the software code error of this branch can be detected by no means, since this code has been never executed. As a result of correction of this error the control is given to the appropriate code allowing to detect its faults. In the mentioned example the independence condition is violated but this can be taken into account by no reliability model.

2.5 Analysis of Assumptions: Intermediate Conclusions

The analysis of assumptions of the most popular reliability models shows that none of four assumptions common for all models is not executed in practice. Even this very conclusion allows to hazard a conjecture that none of the models is able to provide sufficient accuracy in the description of the fault detection process in time. The conclusions made are proven in [12] in which the reliability evaluation accuracy is analyzed with the help of nine most popular models. None of the models shows the high accuracy of reliability evaluation for none of 50 researched data set. E.g. the mean value of standard deviation (SD) in all time series is 4,1 in Musa model to 26,5 in Schneiderwind. The maximal SD values are 246 in Musa model to 996 in Duan model. Such a low accuracy of reliability evaluation is likely associated with the fact that the base model assumptions are not executed. Among all the assumptions the most unreal one is the following: in correcting the faults new secondary faults are not brought in. However hereinafter we will show that if the assumption, that secondary faults are not brought in, is removed the other three assumptions can be automatically abandoned.

3 Model in which Secondary Faults are Taken into Account. Hypothesis

So, all existing SRGM do not provide the necessary evaluation accuracy since their base assumptions are not practically executed: the detected faults are not eliminated at once, and in the course of their eliminating the new secondary faults can be brought in. In addition, the events, which occur in detecting the faults, are dependent. Suppose that there is no assumption about bringing the secondary faults in some hypothetical reliability model. That is, we can think that in correcting any detected fault any amount of secondary faults can be brought in software. Let us analyze in what way this fact will affect all other assumptions.

3.1 Assumption about Immediate Fault Elimination

The assumption that a fault is eliminated at once does not correspond to reality. At the testing stage first a list of detected faults is just made and then software is sent to developers to correct them. At the stage of operation the time interval between fault detection and correction is even greater. To correct a fault detected by a user except for the procedure of its correction the changed software code distribution among a big number of users is necessary. If software is widely-spread as for example in the case with Windows operation system the number of users can be several million. That is why the information of all detected faults is first accumulated over long period of time and then eliminated. A user as a rule gets system updating once every few months.

In removing the assumption the supposition of the immediate fault elimination can be also removed. Suppose that some fault is detected but not corrected and users con-

tinue applying the program. It means that under the real operation conditions after a fault is detected but not corrected the total number of faults in program does not change.

But the same case – invariability of the total number of faults – is possible in a slightly different situation namely when detected fault is immediately eliminated but in the process of its elimination one secondary fault is brought in. Thus, the bringing of this secondary fault is equivalent to the situation when the detected fault was not corrected. The total number of faults in program remains unchangeable – one fault was corrected and one fault was brought in.

So, the removal of the assumption that secondary faults are not brought in makes the assumption about the immediate correction of detected fault automatically unnecessary. We can argue that the assumption about possibility to bring in secondary faults covers the assumption about necessity of the immediate elimination of the detected.

3.2 Assumption about the Mutual Independence of Faults

As we demonstrate in point 2.4 there are cases in practice when the assumption about mutual independence of fault detection is not executed. However, if to suppose that secondary faults can be brought in then this assumption is not mandatory. And really, the situation mentioned in point 2.4 when elimination of one fault gives the opportunity to execute the software code branch unavailable earlier can be modeled by bringing in the secondary faults. The faults existing physically in this branch of the software code were not detected in the program and did not influence on its testing. After all a fault can be manifested only during program execution. So, if there is no possibility to execute the code of such “hidden” branch there is no principal possibility to detect the faults in it. From the viewpoint of a tester such kind of faults is absent in the program. However, in opening the branch unavailable earlier all these faults hidden earlier appear one-time and are brought in the program. This is tantamount to the situation when in detecting and correcting one fault (branch “opening”) several secondary faults are brought in the program.

We can argue that the assumption about possibility to bring in secondary faults as in the previous case covers the assumption about mutual independence of faults.

The latter assumption becomes just unnecessary if we admit the opportunity that secondary faults can be brought in.

Thus, from the analysis of the major assumptions of the software reliability models follows that in removing assumption 1.3 providing that secondary faults are not brought in:

- Assumption I.1 about coincidence of conditions of program testing and executing is not mandatory that is why it can be removed;
- Assumption I.2 about immediate correction of detected faults is covered by the supposition, that secondary faults can be brought in, and can be removed;
- Assumption I.4 about mutual independence of faults is covered by the supposition, that secondary faults can be brought in, and can be removed.

So, there is an opportunity to remove all major assumptions, which are seldom executed in practice. But on the other hand, if at least one assumption of some model is not executed it calls in question the usage of this model itself! We can argue that it is this fact that conditions a low accuracy of all SRGM existing at present. And it is the situation when some certain assumptions are not executed in some separate instances that we can explain the fact that a model demonstrating a high accuracy in one program system loses it (accuracy) in another. This fact is known as a fact of absence of universal software reliability model. And to receive a universal model describing fault detection process equally accurately the only assumption should be removed – the one that secondary faults cannot be brought in. We will show hereinafter how to remove this assumption.

4 Model Taking into Account the Secondary Faults. Implementation

A software reliability model, in which the assumption that secondary faults are not brought in is absent, was first described in [13]. In this model – The Software System Dynamics Model (SSDM) – just one important assumption is actually made. Unlike all existing reliability models in the SSDM the fault detection process in software is supposed to be a non-equilibrium one of transferring the faults outside the program. If a detected fault is corrected then it is considered as a process of fault removing out of the program system owing to the fault stream directed outside the program. Then the fact of bringing the secondary faults in the program can be considered as a result of the action of one more stream directed from outside into the program.

In SSDM the intensity (rate) of these streams are supposed to be subjected to the general laws of transferring. Thus, the SSDM naturally takes into account the presence of secondary faults. The equations of changes of the amount of faults in the system in time obtained in [13] allow to conduct the verification of this model. The verification results [12] show that the SSDM for each of 50 researched program systems demonstrates stably better results than other popular reliability models. This is not surprising since the SSDM does not possess the assumption that secondary faults are not brought in. And as the authors have mentioned above removing this assumption one can remove all other artificially created assumptions.

5 Conclusions

So, from the above we can make the following conclusions.

1. The existence of the large number software reliability models, each of which actually describes one and the same physical process, is connected with the low accuracy of modeling and absence of a universal model.
2. The low accuracy of existing reliability models is connected for each of them with the presence of assumptions, which are as a rule violated under the real conditions of testing and running.

3. Among all the major assumptions of all reliability models the one that secondary faults are not brought in the program is the most unreal.
4. In the paper we have shown that assumption about secondary faults is a key one covering all the rest assumptions. In removing this one all other assumptions can be automatically removed.
5. This assumption removal has allowed to create a new universal reliability model, which is equally well applied to all the researched programs.
6. In order to make possible the removal of the assumption that secondary faults are not brought in we had to get away from the main concept of all existing reliability models – the one that fault detection process is a random process.
7. The alternative concept in a new universal model is a deterministic one, which supposes that fault detection process in Software is a particular case of general processes of transferring.

References

1. Barlow, R., Scheuer, E.: Reliability Growth during a Development Testing Program. *Technometrics*, 8(1), 53-60. (1966)
2. Weiss, H.K.: Estimation of Reliability Growth in a Complex System with a Poisson-Type Failure. *Operation Research*, 4(5), 532-545. (1956)
3. Yakovyna V., Nytrebych O.: Discrete and Continuous Time High-Order Markov Models for Software Reliability Assessment. In: Batsakis, S. et al. (eds.), Proc. 11-th Int. Conf. ICTERI 2015, Lviv, Ukraine, May 14-16, 2015, CEUR-WS.org/Vol. 1356, 419 – 431. (2015)
4. Lyu, M. R.: *Handbook of Software Reliability Engineering*. McGraw-Hill, New York, U.S.A. and IEEE Computer Society Press, Los Alamitos, California, U.S.A. (1996)
5. Drozd, J., Drozd, A., Antoshchuk, S., Kharchenko, V.: Natural Development of the Resources in Design and Testing of the Computer Systems and their Components. In: 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 233-237. Berlin, Germany. (2013)
6. Drozd, J., Drozd, A.: Models, methods and means as resources for solving challenges in co-design and testing of computer systems and their components. In: 9 th International Conference on Digital Technologies (DT'2013), 176-180. Zhilina, Slovakia. (2013)
7. Kharchenko, V.S., Tarasyuk, O.M., Sklyar, V.V., Dubnitsky, V.Y.: The method of software reliability growth models choice using assumptions matrix. In: Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC 2002), 541-546. (2002)
8. Odarushchenko, O.M., Rudenko, O.A., Kharchenko, V.S.: The secondary defects in models of reliability software. *Mathematical Machines and Systems*, 1, 205-217. (2010)

9. Odarushchenko, O.M., Rudenko, O.A., Kharchenko, V.S.: Method of software reliability estimation taking into account secondary faults. *Radioelectronic and computer systems, NAU KhAI*, 7(59), 294-300. (2012)
10. Zeephongsekul, P., Xia, G., Kumar, S.: Software-reliability growth model: Primary-failures generate secondary-faults under imperfect debugging. *IEEE Transactions on Reliability*, 43(3), 408-413. (1994)
11. Shcherbakova, G., Krylov, V., Pisarenko R.: Parameters prediction in technical diagnostics systems with Adaptive clustering in the space of the wavelet. In the experience of designing and application of CAD systems in microelectronics: X Int. conf. CADSM'2013, 322-326. (2013)
12. Maevsky, D. A., Maevskaya, E. J., Jekov, O. P., Shapa, L. N.: Verification of the Software Reliability Models. *Reliability: Theory & Applications*, 9(3), 14-23. (2014)
13. Maevsky, D. A.: A New Approach to Software Reliability. *Lecture Notes in Computer Science, Software Engineering for Resilient Systems*, No 8166, Berlin: Springer, 156–168. (2013)