Modular Integrated Probabilistic Model of Software Reliability Estimation

Roman Yu. Tsarev, Alexey S. Chernigovskiy, Elena N. Shtarik and Andrey V. Shtarik
Siberian Federal University, Department of Informatics, Krasnoyarsk, Russia
E-mail: tsarev.sfu@mail.ru

Mustafa S. Durmuş
Pamukkale University, Department of Electrical and Electronics Engineering, Denizli, Turkey
E-mail: msdurmus@pau.edu.tr

Ilker Üstoglu
Yildiz Technical University, Department of Control and Automation Engineering, Istanbul, Turkey
E-mail: ustoglu@yildiz.edu.tr

Keywords: software reliability, reliability estimation, mean time to failure, mean time to repair, availability, multiversion software

Received: October 22, 2015

A modular integrated probabilistic model of software reliability estimation and an algorithm of its application for estimation of software reliability with different architecture such as multilevel, multiversion, distributed and object-oriented ones are presented in the article. The modification of this model is given there for the object-oriented multiversion software with the distributed architecture. The procedure of its estimation is perfected to improve the quality of the reliability prediction. The description of the developed program system based on the modular integrated probabilistic model of reliability estimation of the object-oriented multiversion software with the distributed architecture is presented in the article. The analysis of relation of software reliability parameters to the component count, conditional and unconditional probability of the failure appearance in components and temporary components characteristics is done there as well.

Povzetek: Opisan je modularni verjetnostni model za oceno zanesljivosti programske opreme.

1 Introduction

The interest to the software reliability estimation has arisen at the same time as the software origin. It has been caused by the natural need to get traditional probabilistic software reliability estimation as one of the computer system components. Originally the approach to the computer system parts reliability estimation was a little different from the hardware reliability estimation and it consisted in application of well-known statistical methods of classical reliability theory in a new technological branch which laid the corner stone of the individual trend like the software reliability theory [22]. However, as far as computing machinery was developed it became obvious that software was not only the part of the computing system.

In the modern conditions of digital technology development the software discontinued to be a part of the one computing system as it used to be, it began to be used on hundreds and thousands of similar computers (basically, on personal ones) [16]. It is obvious that the problem of assurance of the stable programs functioning, identification and correcting the failures in programs sharply exists for software developers nowadays.

Over previous decades, lots of approaches, models and methods of software reliability research have been created [3], [4], [5], [19]. However, any unified approach to the solution of this problem has not been proposed yet and, apparently, it will not happen in the near future. Nevertheless, developing difficult programs systems, their creators are trying to get software reliability estimation [8], [17], [20]. One of the most effective approaches consists in sequential estimation of the programs reliability at every stage of their development [10], [19]. The main difficulty in using statistical methods is the absence of the sufficient amount of the input data. The detection of errors dynamics should be thoroughly registered and processed. Another important problem is a grain size of element's computing reliability [7], [14]. Defining all the paths of program execution during information processing as it sometimes offers is virtually unreal even for an easy program. According to this, the elements' computing reliability detailing (they are theoretically called program modules) should be limited by the completed program formations, which are connected to each other, compose more complicated unit (complex) which reliability holds our interest [6], [11], [12]. In this case it is acceptable that the computing machinery, the operating system and the programming environment are absolutely reliable. Of interest is only

the reliability of functioning of special software tools which solve the main system problem [21].

As the result of the analysis of many researchers' works [2], [5], [9], [13], [15], [16] in the field of software reliability research, three basic problem groups can be distinguished. They are:

- the absence of the unified methodology of high-reliable software system development;
- the absence of the unified methodology of high-reliable software system testing;
- the absence of the unified approach in software systems reliability estimation and analysis.

One of solutions of the previous problems is the usage of the software reliability estimation models presented in this paper. The generic modular integrated probabilistic model of software reliability estimation and its modification for the multiversion software with the distributed architecture are adapted to the modern analysis and software development methods; in particular the option of application of the models for the software building following the object-oriented approach is presented there.

2 Methodology

2.1 The generic modular integrated probabilistic model of software reliability estimation

The following generic modular integrated probabilistic model of software reliability estimation has been developed to evaluate the reliability parameters of the software.

It is obligatory to satisfy the condition for this model:

$$\sum_{i=1}^{F} PU_i = 1,$$

where F is a number of software architecture components; PU_i is probability of using component i, i = 1, …, F.

The mean time to repair is calculated as follows:

$$\begin{aligned} MTTR = \sum_{i=1}^{F}[PU_i \times PF_i \times [TA_i + TC_i + TE_i + \\ + \sum_{j=1, j\neq i}^{F}[PL_{ji} \times [TA_j + TC_j + TE_j + \\ + \sum_{l,j\in D}[PL_{lj} \times (TA_l + TC_l + TE_l)]]] + \\ + \sum_{k,i\in D} PL_{ki} \times [TA_k + TC_k + TE_k + \\ + \sum_{m=1, m\neq j}^{F}[PL_{mk} \times [TA_m + TC_m + TE_m + \\ + \sum_{l,m\in D} PL_{lm} \times (TA_l + TC_l + TE_l)]]]]]. \end{aligned} \quad (1)$$

where M is a number of the software architecture levels; PF_i is theoretical probability of component i failure, i = 1, …, F; PL_{ij} is conditional probability of component i failure under component j failure, i = 1, …, F, j = 1, …, F; TA_i is relative time of the access to component i, i = 1, …, F; TC_i is relative time of failure's analysis in component i, i = 1, …, F; D_{mj} is disjoint sets of component j at level m, m = 1, …, M, j = 1, …, F; TE_i is relative time of failure recovery in component i, i = 1, …, F.

The mean time to failure is calculated as follows:

$$\begin{aligned} MTTF = \sum_{i=1}^{F}[PU_i \times (1 - PF_i) \times [TU_i + \\ + \sum_{j=1, j\neq i}^{F}[(1 - PL_{ji}) \times [TU_j + \sum_{l,j\in D}[(1 - PL_{li}) \times TU_l + \\ + \sum_{k,i\in D}[(1 - PL_{ki}) \times [TU_k + \sum_{m=1, m\neq i}^{F}[(1 - PL_{mk}) \times [TU_m + \\ + \sum_{l,m\in D}[(1 - PL_{lm}) \times TU_l]]]]]]]]], \end{aligned} \quad (2)$$

where TU_i is relative time of using component i, i = 1, …, F.

The software availability ratio is calculated as follows:

$$S = MTTF/(MTTF + MTTR).$$

The software reliability is computed as follows:

$$R_S = \sum_{i=1}^{F} PU_i \times R_i \text{, where } R_i = 1 - \prod_{k\in Zi} PF_{ik}. \quad (3)$$

where R_i is component i's reliability, i = 1, …, F; Z_i is a set of component i's versions, i = 1, …, F.

The cost of software development is calculated as follows:

$$C_s = \sum_{i=1}^{F} \sum_{j\in Zi} C_j,$$

where C_i is the cost of component i's development, i = 1, …, F.

2.2 The algorithm of using the generic modular integrated probabilistic model of software reliability estimation

The algorithm of software reliability parameters' evaluation with the help of the developed generic modular integrated probabilistic model is described below.

Algorithm 1: software reliability parameters' evaluation with the help of the developed generic modular integrated probabilistic model

1) Divide the estimating software into modules, define the modules' scopes, their characteristics and interaction order.
2) Define the number of architecture levels. If the architecture is multilevel, it is necessary to pass to step 4 or follow step 3 if it is not.
3) Eliminate D_{mj} from the model in formulas (1) and (2). Next, pass to step 4.
4) Define the number of versions. If the architecture is multiversion, it is necessary to pass to step 6 or follow step 5 if it is not.
5) Eliminate Z_i from the model in formula (3). After that, pass to step 6.

6) Define if it is possible to eliminate failures. If it is not, pass to step 8 or, if it is so, follow step 7.
7) Eliminate TC_i, TE_i from the model in formula (1). Then, pass to step 8.
8) Get summarized expressions *R*, *MTTR*, and *MTTF*, solving formulas (1), (2), and (3).

There is a flowchart of the algorithm of the generic modular integrated probabilistic model of software reliability estimation in Figure 1.

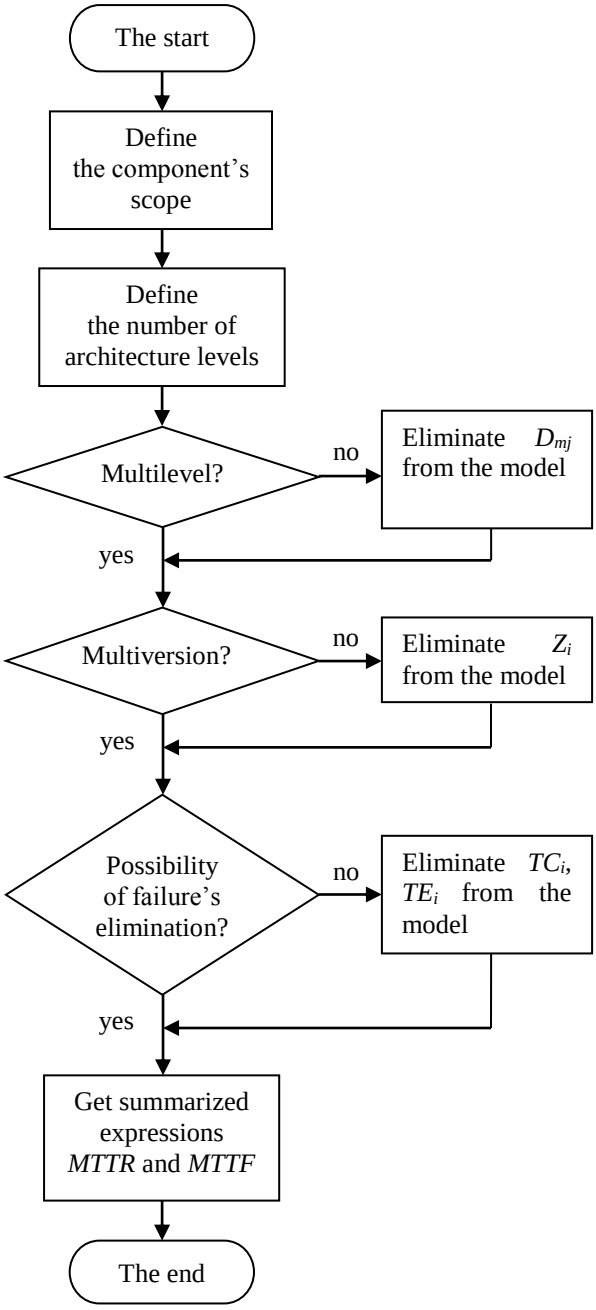

Figure 1: Flowchart of the algorithm of the generic modular integrated probabilistic model of software reliability estimation.

2.3 The modular integrated probabilistic model of reliability estimation of the object-oriented multiversion software with the distributed architecture

The difficulty in the usage of the generic modular integrated probabilistic model at the step of designing the software architecture is that all required parameters are not always known. If the component reliability is unknown beforehand, it can be estimated only at the coding stage. More exact information about reliability can be obtained at the module testing stage. The probability of using the component and component's failure can be gained after software testing. Parameters such as access, analysis and recovery component time for the distributed multiversion software can be estimated after testing, so it is not ruled out that the structure formation of the architecture of the projectable software can be at the conceptual phase. It is possible to build a class hierarchy and method's tree for the object-oriented software. In general, at this step it is necessary to set the parameters which have to be estimated at the following stages.

According to the object-oriented approach computational process is a consecutive calling sequence of class methods. The number of architecture levels equals 1 for this variant. Such parameters as access, analysis and recovery time are parts of the distributed multiversion software.

Let us examine the modification of the generic modular integrated probabilistic model for the instance of the object-oriented multiversion software with the distributed architecture in detail.

The software architecture is a set of class hierarchies for the object-oriented approach. Every class is a set of properties (variables) and methods (functions) of the object.

The process is a set of transitions from one class method to different class method [1], [9]. It is obligatory to satisfy the condition for this model:

$$\sum_{i=1}^{F} PU_i = 1,$$

where F is a general component (class) count in the software architecture, PU_i is a probability of component i's usage, $i = 1, \ldots, F$.

The reliability of the multiversion component depends on the reliability of each version and meta-class which implements the multiversion approach:

$$R_i = (1 - \prod_{K \in Z_i} PF_{ik}) R_{mul},$$

where R_i is a reliability of component i, $i = 1, \ldots, F$; Z_i is a variety of component i's versions, $i = 1, \ldots, F$; R_{mul} is a reliability of the meta-class which implements the multiversion approach. Let us mention that the meta-class should not be considered as the architecture's component and it should be eliminated from computing *MTTR*, *MTTF*, and R_s.

The mean time to repair is calculated as follows:

$$MTTR = \sum_{i=1}^{F}[PU_i \times PF_i \times [TA_i + TC_i + TE_i +$$

$$+ \sum_{j=1, j\neq i}^{F}[PL_{ji}[TA_j + TC_j + TE_j]]]].$$

The mean time to failure is calculated as follows:

$$MTTF = \sum_{i=1}^{F}[PU_i \times (1 - PF_i) \times$$

$$\times [TU_i + \sum_{j=1, j\neq i}^{F}[(1 - PL_{ji}) \times TU_j]]].$$

The software availability ratio is computed in the following way:

$$S = MTTF/(MTTF + MTTR).$$

The software reliability is calculated as follows:

$$R_s = \sum_{i=1}^{F} PU_i \times R_i \text{, where } R_i = 1 - \prod_{k\in Zi} PF_{ik}.$$

As the suggested approach does not take into account the conditional probability of the failure in components, the following model modification was used in the implementation of the model:

$$R_s = \sum_{i=1}^{F} PU_i \times R_i \times \prod_{j=1, j\neq i}^{F}[PU_j \times (1 - R_j \times PL_{ji})].$$

3 Results and discussion

Let us study the program realization of the system of the reliability estimation of the object-oriented multiversion software with the distributed architecture based on the presented model.

3.1 The system of the reliability estimation of the object-oriented multiversion software with the distributed architecture

The modular integrated probabilistic model of reliability estimation of the object-oriented multiversion software with the distributed architecture has been realized as the program system in C# language.

The operational system's function is:

the system user's provision of the information about the projectable software reliability parameters;

the definition of the likehood degree of the modular integrated probabilistic model of software reliability estimation in comparison with the real software.

The primary performing functions are:

the definition of the reliability parameters of the projectable software by means of the modular integrated probabilistic model;

the definition of the reliability parameters of the projectable software by means of estimation of its simulator's behaviour;

the visualization of components' behaviour of the software simulator in the time.

A great number of the system functions forms the structure from five blocks (Figure 2):

the data reduction provides data input and presentation in the form which is convenient for the user;

the modular integrated probabilistic model makes it possible to define the reliability parameters of the projectable software;

the simulator duplicates the behavior of the projectable software following the data which are obtained from the block of data reduction during the specified number of cycles;

the simulator monitoring is done for the statistics' gathering of the simulator work and definition of the reliability parameters of the projectable software following the collected data;

the output is done to lead the results of system work.

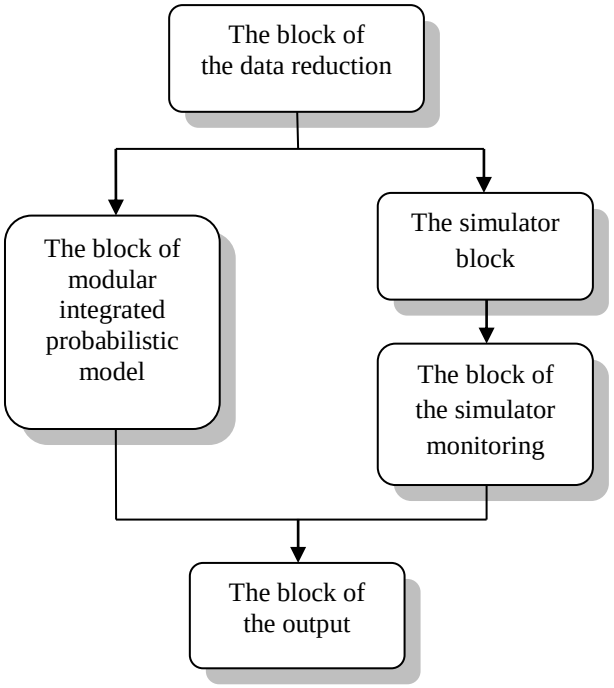

Figure 2: The structure of the system of the object-oriented multiversion software reliability estimation.

The subsystem "The block of the data reduction" serves for the solution of the following tasks:

data editing;

checkout of the correction of the posted data.

The statistical data about the structure of the projectable software are imported into the table with the clipboard or directly by the user.

The visualization of the array of software parameters and its components is performed as the table. In case of having a mistake in edited data the system user will be informed about it by means of the message "An error".

The subsystem "The block of the modular integrated probabilistic model" is basic in the system structure and serves for definition of the reliability parameters of the projectable system by means of using the modular integrated probabilistic model of estimation of object-oriented multiversion software with the distributed architecture.

The input data of the block of the modular integrated probabilistic model is a result of the subsystem "The block of the data reduction" works.

The subsystem "The simulator block" is basic in the system structure and serves for the imitation of the projectable software work in compliance with the data received from the subsystem "The block of the data reduction". The imitation of software execution continues during the time interval specified by the user. During the work of this block it is supposed that each component of the projectable software is invoked to execute the probability PU_i during the time equal TU_i. At the same time during the execution of the component the failure will be made with the probability PF_i, which time equals the sum of TA_i (the access time of the component i), TC_i (the analysis time of the failure in the component i) and TE_i (the time of failure's elimination in component i). Defining the failure, the probability PL_{ij} (the probability of the failure in the component i during the failure of the component j) is also considered.

The subsystem "The block of simulator monitoring" is assigned for the statistics information gathering about simulator work and defining the reliability parameters of the projectable software on basis of this statistics.

The subsystem "The block of output" serves to lead the results of the system work. It displays the operating schedule of the simulator for the user and the results in the work of the block of simulator monitoring and the block of the modular integrated probabilistic model.

3.2 The analysis of the modular integrated probabilistic model of reliability estimation of the object-oriented multi-versioned software with the distributed architecture

The analysis of the modular integrated probabilistic model of reliability estimation of the software includes the analysis of the model behaviour subject to the software components number, conditional and unconditional probability of the failures in the components, and also the relation of software reliability parameters to time characteristics of the components.

Let us guess that the software consists of homogeneous components with the following characteristics: the probability of using $PU_i = 1$, unconditional probability of the failure $PF_i = 0.1$, conditional probability of the failure $PL_{ij} = 0$ for all j, the access time $TA_i = 5$ cycles, the analysis time $TC_i = 7$ cycles, the clearing rime of the failure $TE_i = 10$ cycles, the average time of the using components $TU_i = 30$ cycles. The time of imitation is 1200 cycles.

The analysis of the software reliability relation to the component count has detected the different behavior pattern of reliability parameters in the modular integrated probabilistic model of software reliability estimation from the component count. Thus, for example, the relation of the mean time to repair $MTTR$ and of the reliability R to the component count F has a linear form (Figure 3 and 4). At the same time, the relation of the meaning of the mean time to failure to the component count has a nonlinear form (Figure 5).

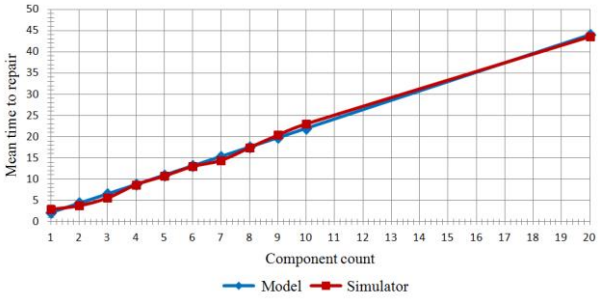

Figure 3: The relation of the mean time to repair to the component count.

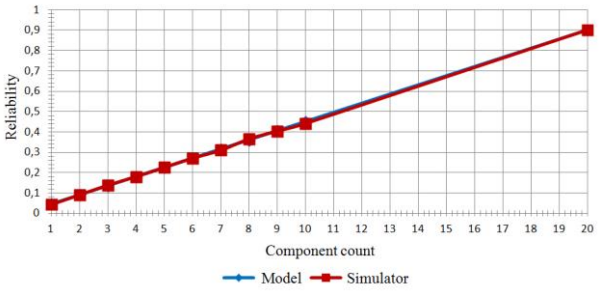

Figure 4: The relation of the software reliability to the component count.

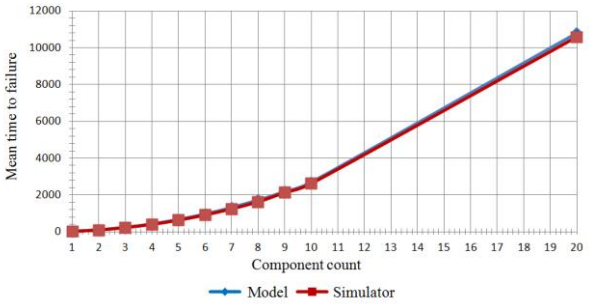

Figure 5: The relation of the mean time to failure to the component count.

Analyzing the relation of software reliability parameters to the committed component count $F = 10$ from the quantity of the unconditional probability of the failure in the software components, the linear growth of the mean time to repair time (Figure 6), the scaling-down of the mean time to failure and the reliability have been detected (Figure 7 and 8).

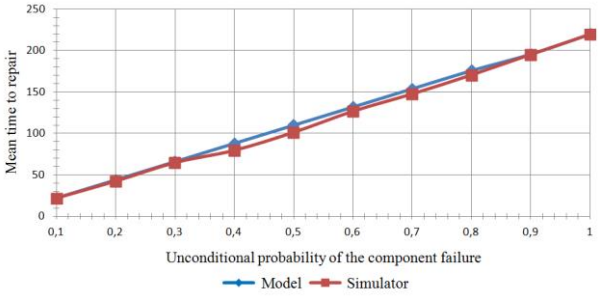

Figure 6: The relation of the mean time to repair to the quantity of unconditional probability of the failure in software components.

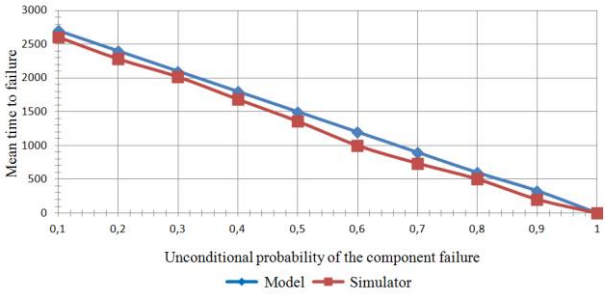

Figure 7: The relation of the mean time to failure to the quantity of the unconditional probability of the failure in the software components.

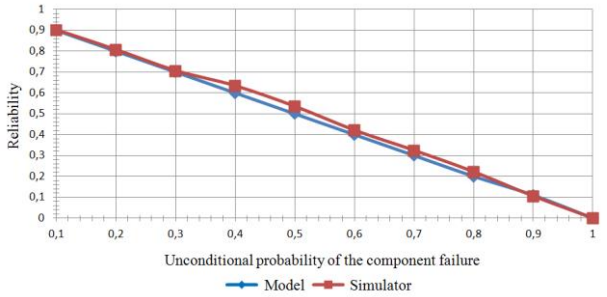

Figure 8: The relation of the reliability to the quantity of unconditional probability of the failure in software components.

Analyzing the relation of software reliability parameters to the committed component count $F = 10$ from the value of the conditional probability of the failure in the software components, the scaling-down of the mean time to failure *MTTF* is marked due to increasing of unconditional probability of the failure in the component (Figure 9). The scaling-up of the mean time to repair *MTTR* occurs during the augmenter of the unconditional probability of the failure in software components (Figure 10). At the same time, the relation of the probability point of the meaning of the software reliability *R* to the value of conditional probability of the failure in the component is absent (Figure 11).

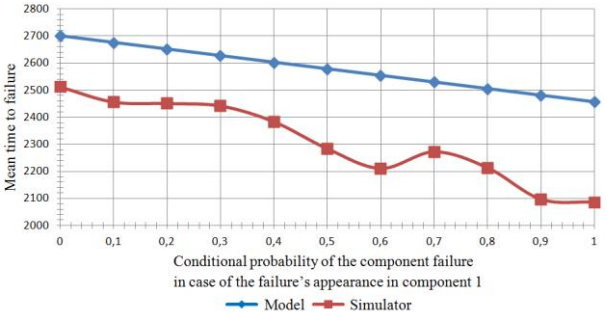

Figure 9: The relation of the mean time to failure to the conditional probability of the failure in the component in case of the failure's appearance in component 1.

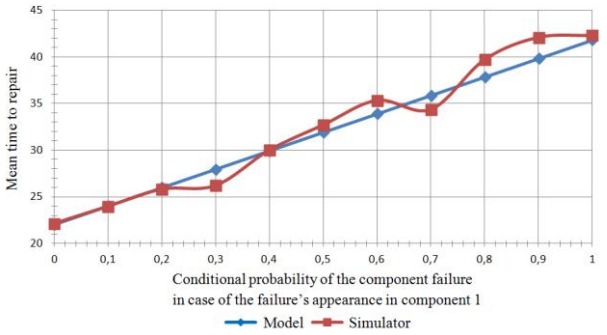

Figure 10: The relation of the mean time to repair to the conditional probability of the failure in the component in case of the failure's appearance in component 1.

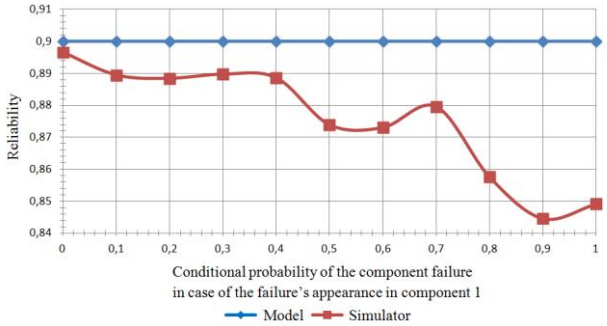

Figure 11: The relation of the software reliability to the conditional probability of the failure in the component in case of the failure's appearance in component 1.

As this exponent of the software reliability as the probability of no-failure operation does not take into account conditional probability of the failure in components, let us use its modified evaluation (10) to increase the quality of the forecast. The result is shown in Figure 12.

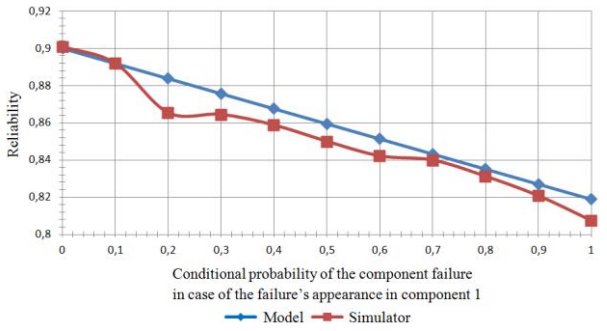

Figure 12: The relation of the software reliability to conditional probability of the failure in the component during the failure's appearance in component 1.

In Figure 13 there is a relation of the average time of usage of the component from the component count included in the software structure to the average time between the failures which equals 675 cycles.

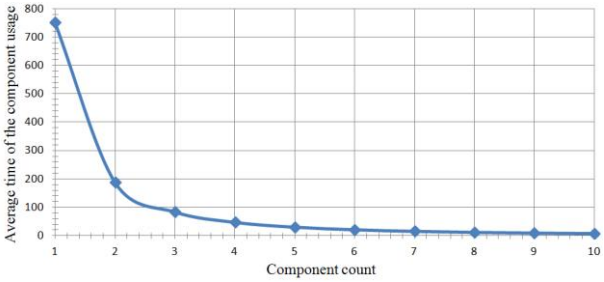

Figure 13: The relation of the average time of the usage of the component to the component count in the system.

In Figure 14 there is a relation of the mean time of the recovery of the component from the software component count to the average time of the system recovery which equals 11 cycles.

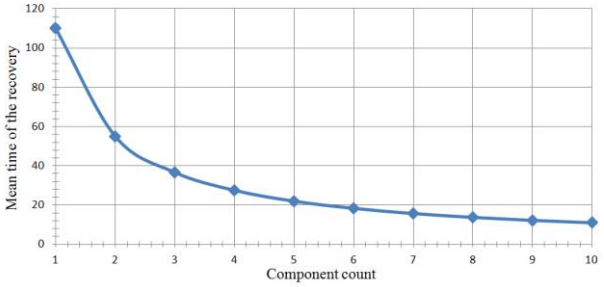

Figure 14: The relation of the mean time of the recovery after the failure of the component to the component count in the system.

During the analysis a backward exponential relation of the average time of the component recovery to the software component count has been detected.

The analysis has shown a high forecast accuracy of the meanings of the reliability parameters in the modular integrated probabilistic model of reliability estimation for the systems with a low intermodule relation. The degradation of the forecast accuracy has been detected for the systems with a high intermodule relation who is specified by a lack of attention to conditional probability of the component's failure and partial ignorance of the intermodule communications and the depth of system components integration. The presented modification of reliability calculation for the modular integrated probabilistic model permits to expand a model range of application and to improve the quality of forecasting.

4 Conclusion

The presented generic modular integrated probabilistic model of reliability estimation of software permits to do sums of assessment of the software reliability parameters of different architecture: multilevel, multiversion, distributed, object-oriented ones. The authors have offered the algorithm of the developed model application for the software reliability estimation with specified software architecture.

In the work the modification of generic modular integrated probabilistic model for the case of the object-oriented multiversion software with the distributed architecture has been analyzed in detail.

The developed system on the basis of the presented modification of the generic modular integrated probabilistic model for the case of the object-oriented multiversion software with the distributed architecture provides end-to-end solution of the following problems: the system user's support in reliability parameters of the projectable software and the definition of the adequacy degree of modular integrated probabilistic model of software reliability estimation towards the real software.

The research has confirmed high performance of the modular integrated probabilistic model of software reliability estimation which is characterized by the weak dependence between the modules. The nonlinear relation between the quantities of the average time of using the component, the average time of recovery after the component's failure and the number of the components in software, and also behaviour pattern of the model during the change of the quantities in conditional and unconditional probability of the failure in software components have been detected. It has been revealed experimentally that the mean time to failure and the mean time to repair linearly depend on unconditional probability of the failure in the components of software.

5 Acknowledgment

The reported study was funded by RFBR according to the research project №16-57-46016 CT_a and by TUBITAK according to the research project №215E196.

References

[1] Abdallah, C., Hafida, B. (2010). A new architectural approach for dynamic adaptation of components-based software using multi agent system. *Control Engineering and Applied Informatics*, vol.12, no.4, pp. 43-50.

[2] Avizienis, A., Laprie, J.C., and Randell, B. (2001). *Fundamental Concepts of Dependability*, Research Report no. 1145, LAAS-CNRS.

[3] Avizienis, A., Laprie, J.C., Randell, B. and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, vol.1, no.1, pp. 11-33.

[4] Benso, A., Di Carlo, S. (2011). The art of fault injection. *Control Engineering and Applied Informatics*, vol.13, no.4, pp. 9-18.

[5] Boehm, B. (2011). *The Future of Software Engineering*, Springer Berlin Heidelberg.

[6] Golubev, I.M., Tsarev, R.Ju., Semenko, T.I. (2005). N-version software systems design. *11th International Scientific and Practical Conference of Students, Postgraduates and Young Scientists; "Modem Techniques and Technologies", MTT 2005 - Proceedings*, IEEE, Tomsk, Russian Federation, pp. 147-149.

[7] Huang, C.-Y., Hung, T.-Y. (2010). Software reliability analysis and assessment using queueing models with multiple change-points. *Computers and Mathematics with Applications*, vol.60, no.7, pp. 2015-2030.

[8] Huang, G., Mei, H., and Yang, F. (2006). Runtime recovery and manipulation of software architecture of component based systems. *Automated Software Engineering*, vol.13, no.2, pp. 257-281.

[9] Huang, C.-Y., Lin, C.-T. (2006). Software reliability analysis by considering fault dependency and debugging time lag. *IEEE Transactions on Reliability*, vol.55, no.3, pp. 436-450.

[10] Kang, W.-H., Kliese, A. (2014). A rapid reliability estimation method for directed acyclic lifeline networks with statistically dependent components. *Reliability Engineering and System Safety*, vol.124, pp. 81-91.

[11] Kulyagin, V.A., Tsarev, R.Y., Prokopenko, A.V., Nikiforov, A.Y., Kovalev, I.V. (2015). N-version design of fault-tolerant control software for communications satellite system. *2015 International Siberian Conference on Control and Communications, SIBCON 2015 - Proceedings*, IEEE Inc., Omsk, Russian Federation, pp. 1-5.

[12] Landon, J., Özekici, S., Soyer, R. (2013). A Markov modulated Poisson model for software reliability. *European Journal of Operational Research*, vol.229, no.2, pp. 404-410.

[13] Lee, W.S., Grosh, D.L., Tillman, F.A., Lie, C.H. (1985). Fault tree analysis, methods, and applications - a review. *IEEE Transactions on Reliability*, vol.34, no.3, pp. 194-203.

[14] Li, X., Xie, M., Ng, S.H. (2010). Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points. *Applied Mathematical Modeling*, vol.34, no.11, pp. 3560-3570.

[15] Myers, G.J., Hocker, D.G. (1981). Use of software simulators in the testing and debugging of microprogram logic. *IEEE Transactions on Computers*, vol.C-30, no.7, pp. 519-523.

[16] Okamura, H., Dohi, T., Osaki, S. (2012). Software reliability growth models with normal failure time distributions. *Reliability Engineering and System Safety*, vol.16, pp. 135-141.

[17] Park, G.-Y., Jang, S.C. (2014). A software reliability estimation method to nuclear safety software. *Nuclear Engineering and Technology*, vol.46, no.1, pp. 55-62.

[18] Rekab, K., Thompson, H., Wu, W. (2013). A multistage sequential test allocation for software reliability estimation. *IEEE Transactions on Reliability*, vol.62, no.2, pp. 424-433.

[19] Rekab, K., Thompson, H., Wu, W. (2013). An efficient test allocation for software reliability estimation. *Applied Mathematics and Computation*, vol. 220, pp. 94-103.

[20] Toader, C. (2010). Increasing reliability of web services. *Control Engineering and Applied Informatics*, vol.12, no.4, pp. 30-35.

[21] Tyagi, K., Sharma, A. (2012). A rule-based approach for estimating the reliability of component-based systems. *Advances in Engineering Software*, vol.54, pp. 24-29.

[22] Zheng, C., Liu, X., Huang, S., Yao, Y. (2011). A parameter estimation method for software reliability models. *Procedia Engineering*, vol.15, pp. 3477-3481.