ANALYSIS OF THE CURRENT STATUS OF THE THEORY AND PRACTICE OF ASSESSING THE RELIABILITY OF SOFTWARE OF AUTOMATED INFORMATION AND CONTROL SYSTEMS

Yusupbekov N. R.
Tashkent State Technical University named after Islom Karimov

Gulyamov Sh. M.,
Tashkent State Technical University named after Islom Karimov

Mirzaev D. A.
Tashkent State Technical University named after Islom Karimov

Kuziyev Z. J.
Tashkent State Technical University named after Islom Karimov

ABSTRACT

The relevance and relevance of the problem of increasing the reliability of assessing the reliability indicators of the operation of information and control systems software at different stages of their life cycle through the use of adequate models, methods and analysis tools is shown. An analytical analysis of known approaches to assessing and predicting the reliability indicators of computer systems software has been carried out, their advantages and disadvantages have been established, which will make it possible to determine the reserves for increasing their reliability when evaluating modern complex and multicomponent software.

Keywords: information management systems, software reliability, computing systems, testing, product life cycle.

INTRODUCTION

Modern trends in the development and improvement of automated information and control systems are stimulated by scientific research aimed at information support for the life cycle of process and production control systems, the development of reliable software tools and methods that ensure a high level of quality of dynamic control objects. These systems are software and hardware systems, the process of functioning of which consists in the interaction of software and hardware, of particular importance for the design, production and operation of which is the assessment and provision of specified indicators of their reliability.

Back in the 60s of the last century [1], B.V. Gnedenko emphasized: "on the one hand, the growing complexity of systems leads to a decrease in their reliability, and on the other hand, more stringent requirements are put forward for the reliable operation of these systems." The current stage of development of industrial automation systems is characterized by a clearly expressed contradiction between the responsibility and complexity of the corresponding software, on the one hand, and methods and means for assessing and predicting its reliability, on the other. This contradiction can be eliminated by increasing the degree of adequacy of software reliability models at all stages of its life cycle. At the same time, at each stage of the life cycle, the most adequate model of the reliability of the software of information and control systems should be used by clarifying and increasing the degree of detail of information about the object of study [2-4].

Modern approaches to assessing the reliability of information management systems software

Let us turn to the basic concepts of the theory of reliability of technical and technological systems. A computer program is understood as a sequence of operations that a computer can perform, and a software tool is an interconnected set of programs, procedures, rules and data related to the operation of the software to maintain operability and turn output data into the desired result under specified conditions for a specified period of time. This group of properties includes: reliability, which is characterized by the frequency of failures due to errors and imperfections of the software system; resistance to anomalies, which determines the ability of the software to perform its functions in abnormal conditions (failures and failures of hardware, operator errors and errors in output data, etc.); recoverability, which determines the ability to restore the level of quality of operation and data after failures; accuracy, which is characterized by the similarity of data processing results with true or theoretically correct values; reactivity, which is characterized by the ability to turn input data (requests) into the desired result in a timely manner. All these characteristics form one of the groups that determine the quality of software in conjunction with functionality, usability, rationality, maintainability and portability [5].

The functional concept of the theory of reliability - failure - can be designated as an event, after the occurrence of which the characteristics of a technical (or technological) object go beyond the permissible limits. In this case, the discrepancy between the results may be due to hardware defects or due to errors in the software. A software error is a record of a program element or software documentation text, the use of which leads or may lead to incorrect results [6].

Thus, unlike hardware, software failures can be caused by an incorrect algorithm, its incorrect implementation, incorrect software documentation, which will result in incorrect user actions. At the same time, software failures may depend on the data that the program is currently processing. In addition, software failures can also be caused by hardware failures due to external factors (ionizing radiation, temperature, etc.) and, in some cases, failures can be eliminated by rebooting the software system.

In general, failures are divided into sudden and gradual [8,9]. Gradual failures occur with a gradual change in the parameters that determine the quality of the product (mainly as a result of aging or wear), when these parameters go beyond the established tolerances. Such failures are not typical for software. Sudden failures are determined by a sharp change in the parameters that determine the quality of the product. In addition, computer technology is characterized by failures, that is, failures that self-repair [8].

Reliability criteria and indicators are used to quantify reliability [9]. The reliability indicator is the numerical value of the criterion, which is specified in the technical requirements for the product, calculated during the design process, evaluated during the testing and operation of a technical object.

The mathematical apparatus of the theory of reliability considers the failure of an element as a random event, in the time ξ before its occurrence - as a random, probabilistic value. The main characteristic of the reliability of an element of any system, including a non-renewable one, is the distribution function of the duration of its uptime F (t) = P ($\xi<t$), on the basis of which the following reliability indicators can be obtained [10] for non-renewable systems: P (T) - the probability of failure-free operation during the time t; Q (T) - 1 P (T) the probability of failure during the time t; T1 - (mean time between failures); F (T) - distribution density of the duration of no-failure operation; λ (T) – failure rate at time t.

The failure rate λ (t) is the high reliability of the elements of complex systems. This probability of occurrence of failures λ (T) is the probability of propagation to the probability of non-failure operation of the object F (T). Statistically tolerable failures are sets of engineering samples that have given up time to the average number of samples that work properly.

There are the following dependencies between the reliability indicators [11]:

$$(t) = f(t)P(t), \tag{1}$$

$$(t) = e - \int \lambda(t)dtt0, \tag{2}$$

$$(t) = \int f(t)dt\infty t, \tag{3}$$

$$T1 = \int (t)dt\infty 0 = \int tf(t)dt\infty 0. \tag{4}$$

Reliability indicators of renewable elements and systems, which in the general case are software tools, can also be indicators of the reliability of non-renewable elements [12]. This occurs in cases where the system, which includes the element, cannot be repaired due to the conditions of its operation. The reliability of renewable facilities is assessed by the following indicators: T - average duration of operation between failures (mean time between failures); T_V - the average duration of recovery; ω (T) – failure flow parameter; K_G (T) is the readiness function (the probability that the system is operational at time t); KP (T) is a function expressing the probability that at the moment t the system is out of order and is being restored; K_G - availability factor (probability that the system will be serviceable during long-term operation (stationary mode); KP - downtime factor (probability that the system will be faulty during long-term operation).

The failure rate parameter ω (t) is the derivative (rate of change) of the average number of object failures at time t. This parameter is defined as the ratio of the number of failed samples of equipment to the number of samples ordered for testing, provided that the samples that failed are replaced with serviceable or repaired ones [12]. The failure flow parameter has the following properties [13]:

In the case of an exponential law of the duration of the object with the parameter λ and instant recovery ω (T) = λ;

- in case of instantaneous recovery, the limit to which the failure rate parameter tends at t is equal to the value reciprocal to the average uptime, that is, the limit ω (T) = 1 T;
- in case of instant recovery, the failure rate parameter and the time-to-failure distribution density are related by the Volterra integral equation of the second kind:

$$\omega(t) = f(t) + \int \omega(\tau)f(t-\tau)d\tau t0.$$

This equation establishes the relationship between the reliability indicators of renewable

and non-renewable technology. It allows using static data on failures of renewable equipment during its operation to determine the reliability indicators of non-renewable equipment.
Since the beginning of research into the reliability of software (SW) of computing systems and complexes, theorists and practitioners of this subject area have discussed and interpreted the issues of software reliability in terms of similarities, differences, features of engineering and modeling technology, etc. [14,15]. Since the fundamentals of engineering and technology for modeling software reliability are interested in the theory of reliability developed for complex technical systems over the past half century, a comparison of the reliability of software and hardware hard ware can help in using these techniques in software and hardware systems.
In table. Table 1 shows comparative characteristics of software and hardware reliability [16,17]. The material of technical equipment wears out over time, so calendar time is a common argument for the hardware reliability function. However, in the case of software, failure will never happen if the program is not used.

Table 1 Comparison of software and hardware reliability

Software Reliability	Software Reliability
Without taking into account the evolution of the program, the failure rate does not increase statically	The failure rate has the form of a U-shaped curve. The section of the initial operating time is similar to the stage of debugging the program
Failure does not occur if the software is not in operation	Material aging can lead to failure even if the system is not in operation.
The subject of the study is the failure mechanism	The failure mechanism can be interpreted as a black box
Processor time and program runtime are two of the most popular reliability function arguments.	Calendar time is a common argument of the reliability function
Most models are analytically derived from sentences. Emphasis is placed on model development, clarification of model assumptions, physical content of parameters	The failure data describes some distributions. The choice of an appropriate distribution is based on the analysis of failure data and on the experience of the researcher. Emphasis is placed on failure data analysis
Failures caused by incorrect logic, inadequate choice of programming language operators, or incorrect input data. It looks like a mistake in the design of a complex hardware system	Failures caused by material wear, random failures, design errors, misuse, environment, etc.
The reliability of the software can be improved by increasing the efforts to test and fix the identified bugs. Reliability tends to change constantly throughout the testing phase, either by introducing bugs into new code or by removing existing bugs.	The reliability of hardware can be improved through reliable design (system and circuit design stages), by improving the technological process and materials, and accelerated testing of the object
"Repair" of systems requires the installation of a new version of the entire software product or part of it	Hardware recovery requires renewal of cash and boundary conditions or replacement of system elements with new ones
Software modules are rarely standardized	Hardware modules are usually standardized

Therefore, in the context of software reliability, it is more appropriate to interpret time as the load on the software (or as the amount of work done by the software). Here it should be noted the importance of taking into account the metrics of program code coverage by tests and program execution paths: software failure statistics should be studied when performing a certain sample of work (tasks with different input data). Running any number of jobs with the same input (if they didn't fail for the first time) will not cause the software to fail. So, according to [18], the number and nature of failures that relate to software is a consequence of its internal errors and depends on the conditions of its application. The following units of time can be used as an argument of the software reliability function [19]:

- Execution Time – processor time; Time during which the processor is busy;
- Operation Time – time during which the software is used;
- calendar time - an indicator that is used for software that works 24 hours a day;
- launch (run) of the program (eng. Run) – the work presented to the processor;
- instruction – the number of executed code instructions;
- Path – sequences of input data execution.

Models based on run time, run time, calendar time, and time of instructions executed belong to time domain models. Run-and-path models belong to models in the input domain [21].
According to [22], the features and differences in (in terms of reliability) from traditional technical systems cannot be applied to all types of programs the concept and methods of the theory of reliability - they can only be used for software tools that operate in real time and directly interact with the external environment.
When developing and evaluating the quality of software modules, the concept of operational reliability cannot be applied if they do not use real time during information processing and do not interact with the external environment. The dominant factors that determine the reliability of programs are design and development errors. Physical destruction of program modules under external influences is of secondary importance. Relatively rare destruction of software modules and the need for their physical replacement lead to fundamental changes in the concepts of error and failure and to the need to separate them according to the duration of recovery relative to some allowable downtime of the information management system;
To improve the reliability of software systems, methods of automatically reducing the duration of recovery and converting failures into short-term failures are of particular importance by introducing temporary, software and information redundancy into software;
The unpredictability of the place, time and probability of errors, as well as their infrequent detection during the actual operation of sufficiently reliable software, does not make it possible to effectively use traditional methods for a priori calculation of the reliability indicators of complex systems, focused on stable, measurable values of the reliability of composite modules.
Traditional methods of forced testing of system reliability by physical impact on their modules cannot be applied to software tools, they should be replaced by methods for taking into account the forced influence of information flows of the external environment.
Analysis of the reliability of hardware and technical support, and in general is to solve problems. Determining from static data the values of the reliability indicators of elements (and, in particular, the failure rate $\lambda(t)$), as well as identifying the reliability indicators of technical systems (mainly functions and availability) based on failures of the system elements. To solve

the first problem, mainly well-developed methods of mathematical statistics are used [22,23], and for the second one, methods of reliability analysis are used, which are based on the use of probability theory, logical-probabilistic methods, topological methods and methods based on the theory of Markov processes, methods of static modeling, etc. [24].

A separate task facing the developers of complex technical systems is reliability design, in which the requirements for system reliability are taken into account even at the stage of its systematic design [25].

On the other hand, the need for developers in systematic, reproducible engineering approaches to the creation of reliable software products has led to the emergence of the concept of Software Reliability Engineering [9,10], which is defined as quantitative studies of the behavior of software and hardware with respect to the reliability requirements [9]. IMS software reliability engineering includes:

- measurements of software reliability, estimation and forecasting based on reliability models;
- attributes and metrics of software design and development, system architecture, software operating environment, their impact on reliability;
- application of this knowledge during the implementation of the specification and architecture design, development, testing, operation, implementation of software interface.

The development of methods and tools for reliability engineering at the moment and in the coming years of software is an urgent problem in the field of computer engineering, software engineering and related disciplines at the moment and the next decade [9,10].

CONCLUSION

Summarizing, it can be noted that solving the problem of increasing the reliability of software and hardware systems requires the development of models, methods and tools for analyzing the reliability of software tools, which are the constituent elements and nodes of such systems. In turn, the resolution of the contradiction between the complexity of modern software, on the one hand, and the high and stringent requirements for its reliability, on the other, requires the resolution of scientific and technical problems:

- Development of models and methods for evaluating software reliability indicators based on testing data, moreover, such models, in order to increase their adequacy, should take into account the complexity of software tools;
- Development of models and methods for evaluating the reliability indicators of complex software systems based on data on the behavior of the reliability of their constituent elements should take into account the interdependence of program control flows;
- Building a generalized method of reliability analysis, taking into account its complexity at different stages of the product life cycle;
- Development of decision support tools at the stages of testing and commissioning of software products, taking into account the requirements for their reliability.

LITERATURE

1. Gnedenko B.V., Belyaev Yu.K., Solovyov A.D. Mathematical methods in the theory of reliability // Publishing house “Nauka”, Moscow, 1965. – P. 524.
2. Hecht H. Can software benefit from hardware experience? // Proceedings Annual Reliability and Maintainability Symposium. – 1975. – P. 480-484.
3. Soi I.M., Gopal K. Hardware vs. software reliability – comparative study // Microelectronics and Reliability. – 1980. – Vol. 20 – P. 881-885.
4. Pham H., Pham M. Software Reliability Models for Critical Applications // EGG-2663 Technical Report. Idaho National Engineering Laboratory, EG&G Idaho Inc. – 1991. – 98 p.
5. Pham H. System software reliability // Springer-Verlag London Limited, 2006. – 440 p.
6. Lipaev V.V. Reliability of software // SINTEG, Moscow, 1998. – P. 232.
7. Polovko A.M., Gurov S.V. Fundamentals of the theory of reliability // BCHV-Petersburg, St. Petersburg. 2006. – P. 704.
8. Volochy B.Yu. Technology of modeling algorithms of behavior of information systems // Publishing house of the university “Lvov. Polytechnic, Lvov. 2004. – P. 200
9. Lyu M.R. Software Marketing Guide // Computing McGraw-Hill, New York, 1996. – 819 p.
10. Lyu M.R. Software Reliability Engineering: Roadmap // Future of Software Engineering (FOSE’07), Minneapolis, USA, May 2007. – P. 153-170.
11. Lai R., Gard M. Detailed Study of NHPP Software Reliability Models // Journal of Software. – 2012. – Vol. 7, №6. – P. 1296-1306
12. Thayer T., Lipov M., Nelson E. Software reliability // Per. from English. “Mir”, Moscow, 1981. – P. 281.
13. Malaya Y.K., Srimani P.K., Software Reliability Models: Theoretical Developments, Evaluation and Application, IEEE Computer Society Press, Los Angeles, 1990. – 372 p.
14. Lyu M.R., Reliability-oriented software engineering: design, testing and evaluation techniques, Software, IEE Proceedings, Vol. 145, №6, 1998. – p.p. 191-197.
15. Goel A.L., Software Reliability Models: assumptions, limitations and applicability, IEEE Transactions on Software engineering, Vol. SE-11, №12, 1985. – p.p. 1411-1423
16. Reussner R.H., Schmidt H.W., Parnomo I.H., Reliability prediction for component-based software architectures, Tha Journal of Systems and Software, Vol. 66, 2003. –p.p. 241-252
17. Durand J.B., Gandoin O., Software reliability modeling and prediction with hidden Marlov chains, Statistical Modeling, Vol. 5(1), 2005. – p.p. 75-93
18. Jrlinski Z., Moranda P., Software reliability research, statistical Computer Performance Evaluation, 1972. –p.p. 465-484
19. Schick G.J., Walkerton R.W., Analysis of Competing software reliability modeling, IEEE Transactions on Software Engineering, Vol. SE-4, №2, 1978. – p. 104-120
20. Moranda P.B., An error detection model for application during Software development, IEEE Transactions on Reliability, Vol. R-28, №5, 1979 – p.p. 325-329
21. Littlewood B., Software reliability modeling for modulator program structure, IEEE Transactions on Reliability, Vol. R-28, №3, 1979 – p.p. 241-246
22. Goel A.L., Okumoto K., An analysis of recurrent software failures in real-time Control system, Proceeding on Conference ACM, 1978, -p.p. 496-500.

23. Nelson E., Estimating software from test data, Microelectronic. Rel., Vol. 17, 1978. –p.p. 67-74
24. Ramamoorthy C.V., Bastani F.B., Software reliability: Status and perspectives, IEEE Transaction on Software Engineering, Vol. SE-8, 1982. – p.p. 359-371
25. Musa J.D., A theory of software reliability and its application, IEEE Transaction on Software Engineering, Vol. SE-1 (8), 1975. – p.p. 312-327

INFORMATION ABOUT THE AUTHORS

Yusupbekov Nadirbek Rustambekovich, Academician of the Academy of Sciences of the Republic of Uzbekistan, Doctor of Technical Sciences, Professor of the Department "Automation of production processes", Tashkent State Technical University named after Islam Karimov, 1000095, Uzbekistan, Tashkent, Str. University, 2 email:dodabek@mail.ru

Gulyamov Shukhrat Manapovich, Doctor of Technical Sciences, Professor of the Department "Automation of production processes", Tashkent State Technical University named after Islam Karimov, 1000095, Uzbekistan, Tashkent, Str. University, 2 email:shukhrat@mail.ru

Mirzayev Dilshod Aminovich, Doctor of philosophy (PhD) Technical Sciences, Associate professor of the department "Radio engineering devices and systems", Tashkent State Technical University named after Islam Karimov, 1000095, Uzbekistan, Tashkent, Str. University, 2 email:mdilshod@mail.ru

Kuziev Zokir Jumanazar ugli, doctoral student of the department "Radio engineering devices and systems", Tashkent State Technical University named after Islam Karimov, 1000095, Uzbekistan, Tashkent, Str. University, 2.
tel: 99-664-93-55, gmail: zakirquziyev1993@gmail.com.