Software Reliability

Mubariz Khalilov
Baku State University. Baku, Azerbaijan
khalilov@bsu.az

***Abstract*— We study a dynamic model test the software reliability. It is assumed that prior to testing in a SS (software system) has Eτ errors. During the time test of τ is detected εc error based on a team in machine language.**

Keywords*— *software reliability; dynamic model test; the specific number of errors

In the early 70's, many experts have come to the conclusion that necessity widespread industrial (engineering) methods in the field of building programs, based on regimentation and automation of technological processes. Thus, standardization in the field of building programs also became a necessity. Indicators of quality software products depending on the nature of tasks to assess the quality of products can be classified according to the following criteria: characterized by the properties, mode of expression, characterized by a number of properties, place in a procedure of evaluation, the stage of determining the values of parameters [1].

The solution to any problem, the implementation of any function conferred on a computer that is running in the network or locally, possibly in the interaction of hardware and software. Therefore, when analyzing the reliability performance of the computer defined functions should be considered a single set of hardware and software.

By analogy with the terminology adopted for describe a reliable indicator of software refers to property of this software to perform specified functions, while preserving its characteristics within the prescribed limits under certain conditions[4].

Software reliability is determined by its high reliability and recoverability. Reliability software is a property continue to operate when it is used for information processing in the IS. Dependability of software is estimated the probability of operation without failure under certain conditions of the external environment within a specified period.

Reliability software can be characterized by the average time of occurrence of failures in the operation of the program. It is assumed that the computer hardware is in good condition. From a reliability standpoint, the fundamental difference between software from the hardware is that software does not wear out and their failure is not possible due to breakage. Consequently, the characteristics of the software operation depends only on its quality, dictated by the development process[2]. This means that software reliability is defined by its perfection and depends on the availability of errors in it introduced during its creation. In addition, the manifestation of errors associated with the fact that at some point in processing time may come previously not encountered a data set that the program is unable to properly process. Therefore, the input data to a certain extent affect the functioning of the software.

The main reasons that cause disruption of the normal functioning of the software are:

- Errors that are hidden in the program;

- warping of the input information;

- Incorrect user actions;

- Failure of hardware IS(Information systems), which implemented the computational process.

The consequence of errors in the program is its failure. The consequences of failures software can be divided into:

- The complete cessation of performing the functions of the program;

- Short-term disturbance in the course of processing IS.

The severity of the consequences of failure software at the estimated relations IS between the recovery time after the failure of the program and the dynamic characteristics of objects, using the results of this program[5].

Emergency shutdown of the application software is easily identified, because the operating system displays a message containing the alarm code. Typical reasons for the appearance of a crash are errors in the performance of the macro, incorrect use of access methods, security breach memory, lack of memory resources, misuse of macros, the emergence of software interrupts that do not specify a handler, and other causes.

There are some models of software reliability. Among them, the dynamic Schumann's model for software testing more efficient.

Baseline data for the Schumann's model, which refers to the dynamic models of discrete time, collected in the process of testing the SS during the fixed or random time intervals. Each interval - this is the stage at which the sequence tests and fixed some bugs.

Schumann's model can be used in some way organized the testing procedure. Using model of Shuman suggests that testing is done in several stages. Each stage represents the program

designed to complete a complex test data. The detected errors are logged (prepare statistics of error), but not corrected[3].

On completion stage on the basis of data collected about the behavior of the SS at the next stage of testing can be used Shumam model of quantitative indicators of reliability. After that fixes bugs found in the previous step, if necessary, corrected test kits and a new phase of testing. When using model Shumam assumes that the initial number of bugs in the program is constantly in the process of tests for can be reduced to the extent that the errors been revealed as a mistake and corrected. New errors are not made adjustments. The rate of detection of errors is proportional to the number of remaining errors. The total number of machine instructions in a single phase of testing time.

It is assumed that prior to testing in a SS E_τ errors. During the time test of τ error is detected based on a command in machine language.

Thus, the specific number of errors on one machine instruction, remaining in the system after t time of testing is *equal*:

$$\varepsilon_r(\tau) = \frac{\varepsilon_m}{I_T}\varepsilon_c(\tau), \quad (1)$$

where I_T is total number of machine team, which is assumed constant in the testing phase.

The authors suggest that the value of failure rate Z(t) is proportional to the number of errors remaining in the SS after the time spent on testing τ:

$$Z(t) = C\varepsilon_t(\tau), \quad (2)$$

where *C* is a constant; *t* is time of SS without failure.

Then, if the time of the SS without giving 1 a measured from the point $t = 0$ and τ is fixed, the reliability function, or probability of failure of the time interval from 0 to t, is *equal*:

$$R(t,\tau) = \exp\{C[E_T/I_T - \varepsilon_c(\tau)]t\}; \quad (3)$$

$$t_{cp} = \frac{1}{C[E_t/I_T - \varepsilon_c(\tau)]}. \quad (4)$$

Of the quantities appearing in formula (3) and (4), are not known at the initial value errors in the SS (E_T)) and the proportionality coefficient is proportional to - C. To determine them, resorted to the following reasoning. During process of testing, information collected information of the time and the number of errors for each run, i.e. total test time τ is made up of time of each run:

$$\tau = \tau_1 + \tau_2 + \tau_3 + \ldots + \tau_n. \quad (5)$$

Assuming the rate of errors on a constant and equal to λ, it is possible to calculate the number of errors per unit time:

$$\lambda = \frac{\sum_{i=1}^{k} A_i}{\tau}, \quad (6)$$

where *Ai* is the number of errors on the i-th run .

$$t_{cp} = \frac{\tau}{\sum_{i=1}^{k} A_i}. \quad (7)$$

With data for two different things and testing τ_a and τ_b, which are chosen randomly with the requirement to $\varepsilon_c(\tau_b) < \varepsilon_c(\tau_A)$ can be associated with equation (4) and (7):

$$\frac{1}{\lambda_{\tau_A}} = \frac{1}{C[E_T/I_T - \varepsilon_c(\tau_A)]} \quad (8)$$

$$\frac{1}{\lambda_{\tau_B}} = \frac{1}{C[E_T/I_T - \varepsilon_c(\tau_B)]}. \quad (9)$$

By calculating the relation (8) and (9), we obtain:

$$E_T = \frac{E_T[\lambda\tau_b/\lambda\tau_A\varepsilon_C(\tau_A) - \varepsilon_c(\tau_b)]}{(\lambda\tau_b/\lambda\tau_A) - 1}. \quad (10)$$

Substituting this estimate parameters of E_T, to the expression (8), we obtain an estimate for the second unknown parameter:

$$C = \frac{\lambda\tau_A}{[E_T/I_T - \varepsilon_C(\tau_A)]} \quad (11)$$

Having unknown Eτ and C, we can calculate the reliability of the program by the formula (3).

Function of the frequency during the 1st testing interval remains constant and is equal to:

$$Z(t) = (E_T - n_i), t \geq 0, i = 1,2,\ldots m. \quad (12)$$

Known model parameters E_T and C authors offer calculate from the following relations:

$$\frac{\sum_{i=1}^{m} m_i}{C} - \sum_{i=1}^{m}(E_T - n_{i-1})\tau = 0, \quad (13)$$

$$C\sum_{i=1}^{m}\tau_i - \sum_{i=1}^{m}\frac{m_i'}{(E_T - n_{i-1})} = 0, \quad (14)$$

where τ_i is time of the *i*-th run (*i*-th time interval); m_i' is the number of runs, which resulted in the denial of the *i*-th interval (the number of errors in the *i*-th interval); *m* is total number of test intervals; n_i is total number of errors detected (not included) to the *i*-th interval.

All these data can be obtained during testing. Calculated values of the parameters showers Eτ and *C*, we can define parameters. Number of remaining errors in the SS:

$$N_T = E_T - n; \quad (15)$$

reliability:

$$R(t) = \exp(-C(E_T - n)t), \quad t>0. \qquad (16)$$

The advantage of this model compared to other conclude that can correct mistakes, making changes to the text of the program during testing without breaking the process into stages to meet the requirement of constancy of the number of machine instructions.

In this paper the dynamic model checking software reliability. It is assumed that prior to the test in the SS has Em errors. During the time of testing revealed τ, ε_c errors in the calculation of the team in the native language. Thus, the specific number of errors on one machine team remaining in the system after m time of testing, as well:

$$\varepsilon_r(\tau) = \frac{\varepsilon_m}{I_T}\varepsilon_c(\tau)$$

where I_T is the total number of machine instructions, which is assumed to be constant in the testing phase.

REFERENCES

[1] Myers G. Reliability of software. Mir. M., 1980. 360 p.

[2] LISayev V.V. Reliability of software, SINTEG. M.,1998. 232 p.

[3] Myshenkov K.S., Novitsky V.O., A. Kuzmin A.G., Vasiliev A.G., Trofimov S.A., Drozdkov A.N. Automated Information System, Certificate of official registration of programs for computer, № 2001611176, ROSPATENT. M., 12.09.2001.

[4] Myshenkov K.S. The method of designing automated information systems manage undertaking.

[5] Abstracts of the 2nd International Conference on "Managing the properties of grain in the technology of flour, cereals and animal feed", MGUPP., M., 2000, pp.109-112.