Kabak I.S.

OPTIMIZATION OF COSTS FOR DEVELOPMENT AND OPERATION OF COMPLEX SOFTWARE OF INFORMATION AND CONTROL SYSTEMS

The problems related to information solutions deployment in the education system are connected with the development of complex software that solves both information and management tasks. The development of such software involves significant material costs. Beside the software development costs, the expenses related to elimination of the consequences of software failures must be taken into consideration as well. All those costs should be evaluated at the stage of the development of performance specifications for software. This article deals with the evaluation and minimization of the above costs.

Keywords: *modeling, software dependability.*

In a market economy, product quality is of primary importance. Quality involves a number of parameters, including dependability. Manufacturing quality products requires dependable automation systems. Automated control systems (ACSs) include hardware and software. ACS software dependability is a comprehensive property whose main component is reliability. Reliability is a property that characterizes the ability to retain the availability in given operating modes and conditions. ACSs of technical systems are constantly growing in importance. They are used in industrial facilities, transportation, aerospace, defense and other industries.

Today's ACSs of technical systems are complex equipment systems that include computers, data networks, controllers, sensors, executive mechanisms and other devices. As an essential component they include complex software. The operational quality and dependability of the whole automated system largely depends on the quality and dependability of the software. Compliance with the requirements for the quality and dependability of the software must be ensured at all stages of an ACS's lifecycle. Evaluation of the dependability of ACS software involves collecting and processing of failure data both during software development and operation. ACS software failures are primarily caused by coding errors.

The purpose of this paper is to reduce the cost of software development and operation of equipment controlled by such software during business operations, personnel training and research.

In order to reach the set goal the following tasks must be solved:

- evaluate the dependability of complex software, mean time to failure;
- based on the software mean time to failure, identify the expected total losses caused by system failures;
- generate the model of dependence of a complex software development cost on its dependability (mean time to failure);
- define an economically feasible level of complex software dependability.

Evaluation of software development cost

The development of software includes a number of stages, including the development of performance specifications and algorithmic presentation, offline and complex debugging, acceptance testing. Depending on the type of program and design tools, programming languages and techniques used, the respective cost components differ. However, in the development of complex control and information management hardware and software systems that include costly equipment, the component relating to the complex debugging is the largest. The complex debugging is often combined with acceptance tests, therefore it can be considered that this is the final stage of the development process. The decision regarding the end of development and commissioning of the finished software can only be taken at this stage.

Complex debugging involves not only local resources, but also additional remote equipment. Beside the cost of personnel performing the search and elimination of software errors, depreciation of local computer and communications equipment, the expenses related to manufacturing equipment rental, as well as the cost of computer and other equipment operation, cost of intermediate products and tools used in non-production activities, remuneration to remote employees, electricity and heating should be taken into consideration as well.

As an example, let us consider the project of development of software for remote training of process automation and mechanical engineers. Table 1 shows the weekly cost of complex debugging during 10 weeks. The information includes all costs including equipment depreciation.

Table 1. Costs of complex debugging of software of the flexible industrial training system (MSTU STANKIN) in conventional units

1st week	2nd week	3rd week	4th week	5th week	6th week	7th week	8th week	9th week	10th week
0.98	1.02	1.01	0.99	0.98	1.01	1.02	0.99	0.99	1.01

The information given in Table 1 lets us suggest a linear dependence of debugging cost from its duration (time of debugging in weeks). Using the chi-square method, let us verify the legitimacy of that suggestion. The processing of the information given in Table 1 confirms the hypothesis with probability 99.5 %. Therefore, the assumption of the linear dependence of the rising cost of complex debugging on its duration is confirmed. Those costs are shown in Fig. 1 with the dash line:

$$C^1 = c \cdot t, \tag{1}$$

where c is the average cost of software development per time unit; t is the duration of debugging.

Each software failure entails additional financial losses caused by interruptions in the manufacturing process and correction of errors. In order to identify the losses caused by software failure it is required to know the dependence of the failure rate from the duration of debugging. Such dependence is considered in the author's papers [1-3].

The flow of software failures is considered to be ordinary. Failure rate matches the failure flow parameter [1]. The average number of failures with recovery per time unit matches the failure rate. The failure rate was identified based on the software dependability model [1]:

$$H(t) = \sum_{i=1}^{N} a_i \cdot \exp(-b_i \cdot t), \tag{1}$$

where N is the number of various groups of operators or classes with identical dependability characteristics; a_1, a_2, a_3, …, a_N , b_1, b_2, b_3, …, b_N are the coefficients of the model equation.

Let us specify the average software failure cost C_1 and deduce the formula that associates the losses caused by software failure C_0 with the failure rate $H(t)$:

$$C_0 = C_1 \cdot H(\mathrm{t}). \tag{2}$$

By inserting the value $H(t)$ from formula (1) we will finally deduce:

$$C_0 = C_1 \cdot \sum_{i=1}^{N} a_{i\cdot} \cdot \exp(-b_i \cdot t). \tag{3}$$

In Fig. 1 the curve C_0 is drawn with the solid line.

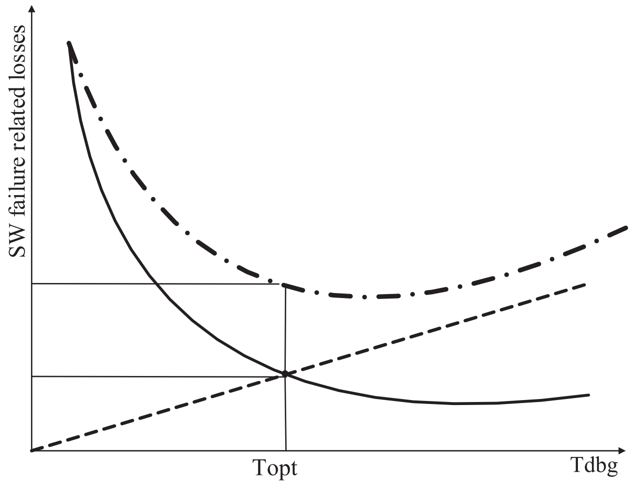

Fig. 1. Total software development cost

The total cost of software debugging can be defined as the sum of C^1 and C_0:

$$C = C^1 + C_0 = c \cdot t + C_1 \cdot \sum_{i=1}^{N} a_{i\cdot} \cdot \exp(-b_i \cdot t) \tag{4}$$

In fig. 1 total costs C are shown with the heavy dash-and-dot line.

Function C has a minimum, which is shown in fig. 1. Thus, an optimal software debugging time exists (T_{dbg}) that minimizes total financial losses.

Statement of problem for optimization of software dependability and selection of method

In [1] the authors provide a formula that associates the software failure rate with its structure, failure rates of its modules and probability of no-failure of the system's modules. According to that formula, the software failure rate is:

$$H\left(t_1, t_2, \ldots, t_N\right) = \sum_{i=1}^{N} v_i \cdot h_i\left(t_i\right) \quad (5)$$

where $h_i(t_i)$ is the failure rate of a module of the i-th group in case of its debugging during time t_i, v_i is the frequency factor that takes into consideration the system's structure and probability of no-failure of its modules.

Dependence $H(t)$ is considered in [2] that refers to the results of mathematical modeling of dependability. It has been found that the value of function $H(t)$ is determined from formula:

$$H\left(t\right) = -\sum_{i=1}^{N} v_i \cdot \frac{1}{\tau_i} \cdot \ln\left[1 - \left(1 - \alpha_0^{-h_i^0 \cdot \tau_i}\right) \alpha_i^{\left(\frac{t}{\sum_{i=1}^{N} v_i \cdot \tau_i}\right)}\right], \quad (6)$$

where v_i is the probability of no-failure of a module of the ith type; t_i is the time of its operation; $б_i$ is a complex coefficient that takes into consideration the debug time and several other parameters.

Dependence $H(t)$ of failure rate from the software operation time is a non-linear characteristic.

Let us assume that time T_0 has been given for software debugging. This time will be the total of all debug times of modules. The debug time of each module is a nonnegative quantity:

$$\begin{cases} T_0 = \sum_{i=1}^{N} t_i \\ t_1 \geq 0, t_2 \geq 0, \ldots, t_N \geq 0, \end{cases} \quad (7)$$

By modifying the value of debug time of module i t_i while observing the set of constraints (7) we will deduct various values: functions $H(t_1, t_2, ..., t_N)$.

Let us set the problem of minimizing the value of software failure rate $H(t_1, t_2, ..., t_N)$ (6) while observing the constraints (7).

Given the non-linearity of functions $h_i(t_i)$ the optimization task can be solved based on the nonlinear mathematical programming methods.

Most non-linear programming methods have no restrictions of the number of optimizable variables. As N grows, the labor intensity of the calculations increases as well, which complicates their practical application. Existing software of automated control systems consists of 100 – 500 or even more program modules. With the N values that high, many nonlinear programming methods become inefficient.

Let us consider a case when the software consists of a large number of program modules characterized by a small amount of code and simple structure.

In order to optimize such software's dependability it is suggested to use an algorithm based on the Bellman principle [8, 9].

Solution of the optimization task based on the Bellman principle

The software failure rate depending on the debug time can be evaluated using formula [2]:

$$h\left(t\right) = -\frac{1}{\tau} \cdot \ln\left[1 - P^0 \cdot \alpha_i^{\frac{t}{\tau}}\right] \quad (8)$$

Let us expand the logarithmic function $\ln\left[1 - P^0 \cdot \alpha_i^{\frac{t}{\tau}}\right]$ as a power series, use it in formula (8) and deduce:

$$h\left(t\right) \approx \frac{1}{\tau} \cdot \sum_{j=1}^{\infty} \frac{\left(P^0 \cdot \alpha^{\frac{t}{\tau}}\right)^j}{j}. \quad (9)$$

Note that value P^0 is quite small and amounts to 0.1 – 0.03. Value б does not exceed 1. Raising to the jth power will make the product $\left(P^0 \cdot \alpha^{\frac{t}{\tau}}\right)$ rapidly diminish. The second element of the power series will be at least 100 times, while the third at least 15000 times smaller than the first element.

We can afford using just the first element of the series without significantly loosing in accuracy:

$$h\left(t\right) = \frac{P^0 \cdot \alpha^{\frac{t}{\tau}}}{\tau}.$$

Let us reduce the right part of the last equation to an exponential form:

$$h\left(t\right) = \frac{P^0}{\tau} \cdot \exp\left(-\frac{\ln\left(\frac{1}{\alpha}\right)}{\tau} \cdot t\right).$$

In order to simplify further calculations let us indicate:

$$\frac{P^0}{\tau} = \beta,$$

$$\frac{\ln\left(\frac{1}{\alpha}\right)}{\tau} = \gamma.$$

Then we will deduce:

$$h_i(t) = \beta_i \cdot \exp(-\gamma_i \cdot t). \quad (10)$$

Let us substitute (10) into (5):

$$H(t_1, t_2, \ldots, t_N) = \sum_{i=1}^{N} v_i \cdot \beta_i \cdot \exp(-\gamma_i \cdot t). \quad (11)$$

It is required to find the minimal failure rate (11) for the set of constraints (7). In order to optimize the failure rate function imposed by formula (11), let us use the dynamic programming method, i.e. generate a series of recurrence formulas:

$$f_k(T_0) = \min_{0 \le t \le T_0} \left[f_{k-1}(T_0 - t) + v_k \cdot \beta_{ki} \cdot \exp(-\gamma_k \cdot t) \right], (12)$$

$$f_1(T_0) = v_1 \cdot \beta_1 \cdot \exp(-\gamma_1 \cdot t). \quad (13)$$

In formula (12), the minimal sum-function is calculated, one of the functions being exponential. It can be shown that functions $f_k(t)$ at all allowable values of k will be exponential as well. This assertion is evidenced in [1].

The minimization required to perform calculations using formulas (12) and (13) is somewhat complicated. In order to proceed to the next step, i.e. increment of value k, it is required to identify function $f_k(t)$ within the interval $0 \le t \le T_0$.

In order to obtain the recurrence calculation formulas, let us differentiate formula (12) in square brackets (where $k = 2$) and equate the result of differentiation to zero, as it is normally done when solving extremum problems in classic mathematical analysis.

Having solved the resulting equation with respect to variable t_1 and substituted it into expression (12) we will obtain the analytic expression $f_2(t)$. Let us write the expression for $f_2(t)$ as (13) and identify the respective value of new coefficients with exponent and argument t.

The validity of the above mathematical operations is evidenced and the detailed development of the recurrence formulas is given in [1].

In order to evaluate the practical results of software dependability optimization, failure rates of existing software were analyzed. The values of failure rate were evaluated for two methods:

- for traditional debug time distribution, when all program modules are debugged to the same value of failure rate;

- for optimal debug time distribution that is calculated using formulas (11) – (13).

The comparison of the two debug time distribution methods has shown that optimization reduces the debug time by 9 – 20% while keeping the same level of dependability as without the optimization.

Conclusions

The paper has set and solved the task of rational planning of software debugging time. Rational planning includes two optimization tasks: definition of the minimal time fund of complex software debugging aimed at reaching the specified failure rate and the task of reaching the specified failure rate of software system within the allowed development time. The failure rate is presented as a function of many variables from the debug time and failure rates of individual software modules.

It is shown that most nonlinear programming techniques are not applicable to optimization of FMS software dependability.

The Bellman's principle can be used under condition of insignificant simplification of the mathematical model of software dependability without a significant loss of accuracy. Using that principle, recursion formulas have been determined that allow solving both of the above tasks.

References

1. **Kabak I.S., Rapoport G.N.** Software dependability evaluation based on its mathematical model // Matters of creation of flexible automated manufacturing facilities. Edited by Makarov, I.M., Frolova, K.V., Belianina, P.N. – Moscow. Nauka. – 1987. – P. 236-245.

2. **Kabak I.S.** Mathematical model for forecasting and evaluation of software dependability // Herald of MSTU STANKIN, 2014, Issue No. 1 – P. 123 – 126.

3. **Kabak I.S., Sukhanova N.V.** On modeling and dependability evaluation of complex software systems // Herald of the Kabardino-Balkarian State University. – 2012, Issue No. 5. – P. 74 – 76.

4. **Grossman K., Kaplan A.A.** Nonlinear programming based on unconstrained minimization. Novosibirsk: Nauka. – 1981. – 84 p.

5. **Polak E.** Computational methods in optimization. A unified approach. Moscow: Mir. – 1974. – 376 p.

6. **Cea J.** Optimization. Theory and algorithms. Moscow: Mir. – 1973. – 244 p.

7. **Fiacco A., McCormick G.** Nonlinear programming. Sequential unconstrained minimization techniques. Moscow: Mir. – 1972. – 240 p.

8. **Bellman R.E., Dreyfus S.E.** Applied dynamic programming. Moscow: Nauka. – 1965. – 460 p.

9. **Bellman R.** Dynamic programming: translation from English. Moscow: Inostrannaya literatura. – 1960, 400 p.

10. **Kolmogorov A.N., Fomin S.V.** Elements of the function theory and functional analysis. Moscow: Nauka. – 1976. — 544 p.

11. Application for invention 2013133304 Russian Federation Method and apparatus for technical diagnostics of complex manufacturing equipment based on neural networks [Text]/ **Solomentsev Yu.M., Sheptunov S.A., Kabak I.S., Sukhanova N.V.**: applicant and patent holder

Federal State Public Scientific Establishment Institute of Information Technology in Design and Engineering of the Russian Academy of Sciences (IKTI RAN. – No. ; application on 18.07.2013; notice of favorable result of examination as to form

12. Patent for utility model 75247 Russian Federation MPK 7 G06F15/16 Modular computer system [Text]/ **Kabak I.S., Sukhanova N.V.**: applicant and patent holder **Kabak I.S., Sukhanova N.V.** – No. 2008106859; application on 26.02.2008; published on 27.07.2008, Bul. No. 21 – 2 p.: drawings

13. Patent for invention 2398281 Russian Federation MPK 7 G06N 3/06 Multilayer modular computer system [Text]/ **Solomentsev Yu.M., Sheptunov S.A., Kabak I.S., Sukhanova N.V**.: applicant and patent holder Federal State Public Scientific Establishment Institute of Information Technology in Design and Engineering of the Russian Academy of Sciences (IKTI RAN. – No. 2008143737; application on 07.11.2008; published on 27.08.2010, Bul. No. 24 – 8 p.: drawings.

14. Application for invention 2417442 Russian Federation MPK 7 G 06NMet7/02 Method for construction of fuzzy logic systems and apparatus for its implementation [Text]/ **Solomentsev Yu.M., Sheptunov S.A., Kabak I.S., Sukhanova N.V.**: applicant and patent holder Federal State Public Scientific Establishment Institute of Information Technology in Design and Engineering of the Russian Academy of Sciences (IKTI RAN. – No. 200810203; application on 19.12.2008; published on 27.04.2011, Bul. No. 12- 8 p.: drawings

15. **Kabak I.S.** Neural network model for forecasting and evaluation of software dependability // MSTU STANKIN Herald, 2014, Issue No. 1 – P. 107 – 111.

16. **Kabak I.S.** Development of large hardware and software neural networks for control systems. // Aviatsionnaya promyshlennost. – 2012, Issue No. 4. – p. 57 – 61