УДК 621.396.6.019.3

O. V. IVANCHENKO, Y. L. YAVKUN, A. A. TOMASHPOLSKAYA

Sevastopol national technical university, Ukraine

OPTIMIZATION OF THE TEST CONTROL OF SOFTWARE CRITICAL APPLICATION

Lessons learned from the analysis of accidents and disasters involving different critical infrastructures (CI), indicate about low level of fault tolerance software (SW) of the CI's management systems. In order to address this gap is proposed optimization procedure based on the correct distribution of the test resource of software of the infrastructure creation's management system. Further prospects of using the proposed models can be relating with optimizing test resource consumption of critical applications software sub-modules and also can be using to specify requirements for functional safety of different critical infrastructures management systems.

Keywords: *fault tolerability, software module, test duration, Markov models.*

Introduction

The need to ensure effective use of critical infrastructures (CI) for its intended purpose has high requirements for fault tolerance of software modules control systems. Achievement of positive results in this area of functional use of critical applications software (SW) is impossible without serious financial costs of test control (check) software.

According to current standards of functional safety trials ensuring [1, 2] an acceptable level of risk, which determines the possibility of failsafe operation of the software module is given at the design stage of an infrastructure management formation. However, such an approach to the critical software applications fault tolerance assessment doesn't allow developers to implement test control (TC) of the corresponding software, spending the minimum amount of computing resource. Application of known software reliability models only partially solves the problem, because in most cases, their practical use is hampered by introducing a large number of restrictions and assumptions, which significantly affects the adequacy of the developed model range [3].

The aim of the article is to optimize the duration of the software (SW) of the CI's management systems (MS) test control through the implementation of fault tolerance mechanism of an appropriate software module.

1. Statement of the researches results

Suggested approach is based on the implementation of the principle of analytical and stochastic modeling (ASM) for cases we are concerned about [2]. Ground segment management system spacecraft (SC) is considered as a studied SC MS. Its architecture is shown in Figure 1.

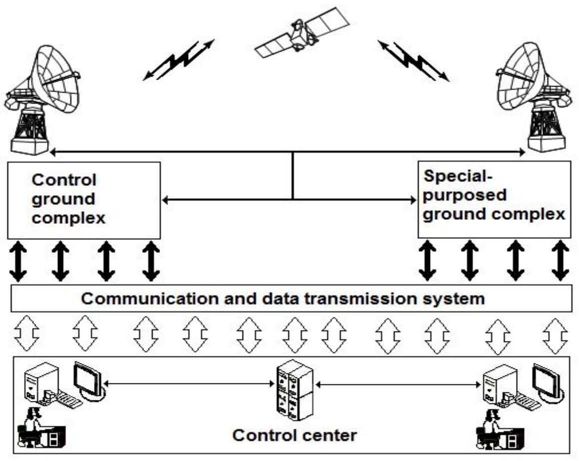
Control ground complex
Special-purposed ground complex
Communication and data transmission system
Control center

Fig. 1. Architecture of the SC MS ground segment

According to the established level of formalization let's represent the process of SC MS functioning in 2-phase process, each phase of which is characterized by the following modes of operation: a) the first phase – standby of SC MS intended use mode; b) the second phase – SC MS intended use mode.

The duration of the corresponding time intervals will be considered as the main characteristics of the operation modes: $t \in (0; t_{sb})$ – intended use standby interval; $\tau \in (t_{sb}; t_{sb} + \Delta t_{iu})$ – intended use interval.

Analysis of operating experience and SC MS intended use testifies the desirability of introducing the following limitations and assumptions:

1) intervals t and τ are predictable nonrandom quantities, because SC MS mainly exploited and used as intended in a planned manner;

2) exponential distribution with parameters $\lambda = 1/T_o$ and $\mu = 1/T_r$ (where T_o – average time between failures, T_r – average recovery time) is

considered as the basic model of CI MS reliability;

3) readiness of a control system in standby mode at a $t \in (0; t_{sb})$ time is determined by the dependability and recoverability of the SC MS technological equipment [4];

4) transition to the second phase is carried out after the completion of SC MS informative and technical state (ITS) monitoring;

5) control system reliability at the intended use interval depends mainly on the fault tolerability of the SC CS software module [5].

Features of operation management system CS at the standby interval of using $t \in [0; t_{ож}]$ point to the following ITS: S_1 – operating state of SC management system; S_2 – fault state corresponds to the destruction of the main body and (or) partial destruction of the reserve; S_3 – state of returning corresponds to the reduction of SC MS after crash (return to operating state using backups and (or) history); S_4 – state of SC management system ITS control; S_5 – state of calculated performance targets solutions corresponds to the presence in the system of additional indestructible array of operating states.

We can build a reference model as a continuous time Markov chain and perform calculations of reliability spacecraft management system based on its application. According to the illustrated graph (Fig. 2), we write the system of differential equations of Kolmogorov-Chapman in the form [2]

$$\begin{cases} \frac{dP_1(t)}{dt} = (-\lambda_{12} - \lambda_{14} - \lambda_{15})P_1(t) + \lambda_{31}P_3(t) + \\ \qquad + \lambda_{41}P_4(t) + \lambda_{51}P_5(t); \\ \frac{dP_2(t)}{dt} = \lambda_{12}P_1(t) - \lambda_{23}P_2(t) + \lambda_{32}P_3(t); \\ \frac{dP_3(t)}{dt} = \lambda_{23}P_2(t) - (\lambda_{31} + \lambda_{32} + \lambda_{34})P_3(t); \\ \frac{dP_4(t)}{dt} = \lambda_{14}P_1(t) + \lambda_{34}P_3(t) - (\lambda_{41} + \lambda_{45})P_4(t); \\ \frac{dP_5(t)}{dt} = \lambda_{15}P_1(t) + \lambda_{45}P_4(t) - \lambda_{51}P_5(t); \\ \sum_{i=1}^{5} P_i(t) = 1. \end{cases} \quad (1)$$

Solve the problem using the Runge-Kutta method, for the initial conditions at the time $t = 0$, when $P_1(0) = 1$, $\forall P_i(0) = 0$, where $i = 2,...,5$. Results of calculations of stationary readiness coefficient $K_{RDN}(t) = P_1(t)$ of SC mangement system as a function $K_{RDN}(T_o, t)$ for average recovery time $T_в = 0{,}5$ hr are shown in Figure 3.

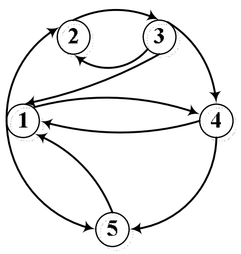

Fig. 2. Graph of transitions for the first phase of SC MS operation

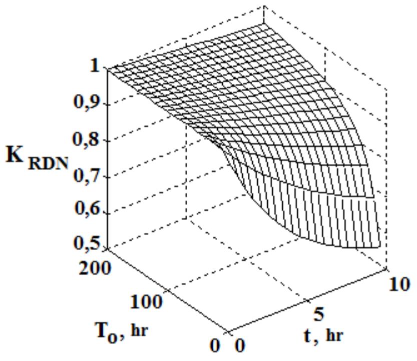

Fig. 3. Dependence of $K_{RDN}(T_o,t)$

At the next stage we construct a Markov model of critical applications software reliability, which adequately describes the processes occurring in the SC management system in the second phase of its operation. For clarity, we get a view on a critical application written in the high level programming language C, used in the European Space Agency (ESA). The software module provides a user interface of ESA spacecraft control center ESA (Figure 1) and is used to generate the data file in accordance with a predetermined user format corresponding to the specific characteristics of the configuration of the array of antenna systems (AS). This software includes about 10,000 lines of C code [5]. Software module consists of three main sub-modules:

1) parsing sub-module (PSM);

2) computing sub-module (CSM);

3) formatting sub-module (FSM).

Using architectural-oriented approach described in [5], we can determine the probability of the correct functioning of critical software applications as

$$E[R] = \left[\prod_{i}^{n-1} R_i' \right] R_n W', \quad (2)$$

where R_i' – probabilistic indicator of i-th component of the software correct functioning; R_n – probabilistic indicator of n-th component of the software correct functioning (i.e., the probability rate of completion of the program module operation); W' – probability indicator of the operating system (OS) correct functioning.

In (2) value is determined with implemented in AS MS mechanism of fault tolerance of software. In addition, in further calculations OS regarded as a system that is absolutely reliable and functioning properly. So, we can take $W' \approx 1$. Simulation results obtained using a specialized software utility SREPT (Software Reliability Estimation and Prediction Tool), are presented in Table 1.

Table 1

Main time and system characteristics of software

Time characteristics of software, hr		
Duration of custom functions realization (for C sub-modules)		
C1	C2	C3
0,01128	0,00248	0,0001251
Duration of system calls realization (for OS)		0,10589
User and system characteristics of the software		
Average number of custom functions (for C Module)		407
Average number of system calls (for OS)		5442

At the next stage we construct a model of software fault tolerance used in the future to develop a system of tests for critical applications software module. Logical basis of the model is the statement: "The more testing efforts we spend, the higher the level of software reliability". As for the main analytical dependence, it can be represented by the objective function $T = f(\lambda)$, where T – testing time, λ – the desired value of the failure rate. The search for optimal combinations of tests can later be carried out with reference to individual elements, basing on providing the required level of reliability $K_{RDN_{rq}}$ of CI MS in general, as $K_{RDN_{MS}} \geq K_{RDN_{rq}}$, where $K_{RDN_{MS}}$ – value of the complex index of reliability, corresponding to the actual level of reliability of critical infrastructure control system. Then the optimization criterion can be written as:

$$\xi = \begin{cases} T = \sum_{i=1}^{n} f_i(\lambda_i) \to T_{min}; \\ K_{RDN_{MS}} \geq K_{RDN_{rq}}; \end{cases} \quad (3)$$

$$K_{RDN_{MS}} = K_{RDN}(t)E(R), \quad (4)$$

where $i = 0,...,n-1, n$ – number of software components with a total testing time T;

λ_i – the failure rate of the components of the program module.

To display changes of the reliability of each i-th component is suggested to use software model of Goel-Okumoto [5], for which the failure rate is determined in accordance with the expression

$$\lambda(T) = E_E e^{-\beta T}, \quad (5)$$

at the same time the objective function according to the relation (3) can be written as

$$T = \sum_{i=1}^{n} \left(\frac{1}{\beta_i \ln(\kappa_i N_i / \lambda_i)} \right), \quad (6)$$

where $E_E = \kappa N$ – the expected value of the failure (crash) rate of software per finite time of its execution T;

β – the rate of decrease of failure (crash) rate due to the implementation of software fault tolerance mechanism;

κ – proportionality coefficient that establishes the connection between the number of failures and software run time.

Table 2 presents the results of statistical estimation of Goel-Okumoto model parameters obtained in [5] on the stage of the client use software modules, the architecture of which includes a three basic sub-module C1, C2, C3 (sub-modules PSM, CSM, FSM).

Table 2

Statistical evaluation of Goel-Okumoto model parameters

C1		C2		C3	
N	κ	N	κ	N	κ
13	0,0546	11	0,0946	5	0,0534

As known [5], the basis of the mechanism of fault tolerance CI MS software modules make restarts components, ie applications with retrying and redundancy. Generally, program modules include primary copies and backups. Accordingly, the fault tolerance mechanism implemented in accordance with the scheme shown in Figure 4.

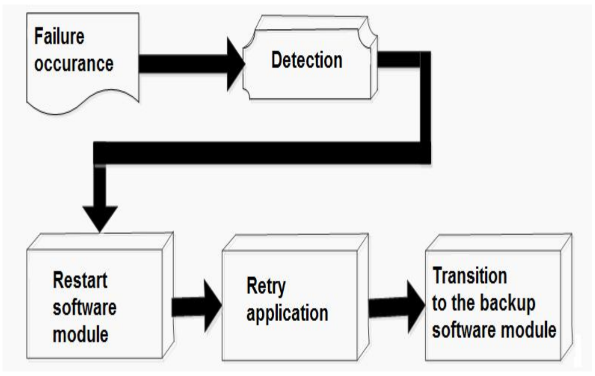

Fig. 4. Scheme of the mechanism of fault tolerance of critical applications software implementation

To obtain analytical expressions in accordance with the scheme shown in Figure 4, consider the following sequence of events:

1) DETFailed – discovery of the failure (crash) after its occurrence failed;

2) RESFailed – restart of a software module after occurring undetected failure (crash) failed;

3) RETFailed – impossible to perform the restart of the application after a fault (crash) has occurred, its detection and execution to restart the software module failed;

4) FOFailed – impossible to make the transition to the backup software module after a fault (crush) has occurred and restart of the software module and the application failed.

Provided that the developer did not perform critical application debugging or any changes of software sub-modules on their main exploitation stage it would be wise to consider the failures intensity λ_i for different components as constants. Then, the reliability indicator i-th component can be determined by the following relation:

$$R_i = \exp\left\{-\int_0^{\tau_i} \lambda_i(t)dt\right\} = \exp\{-\lambda_i \tau_i\}, \qquad (7)$$

where τ_i – time implementing custom functions for i-submodule C.

According to the relation (7) the probability of failure of i-th components of the module is defined as $F_i = 1 - R_i = 1 - \exp\{-\lambda_i t\}$, ie operating time between failure (TTF) is distributed exponentially.

According to [5] the general expression for determining the level of the reliability of the C module with a fully implemented fault tolerance mechanism (Figure 4) can be written as follows:

$$R_C = 1 - [P_{E1} \cdot EXP(\lambda) + P_{E2} \cdot ERLANG(\lambda, 2) + \\ + P_{E3} \cdot (\gamma \cdot ERLANG(\lambda, 2) + \gamma \cdot HYPO(\lambda_1, \lambda 2))], \qquad (8)$$

where $\gamma = 1$, if the backup version is the same with the main, otherwise $\gamma = 0$.

If the fault tolerance mechanism wasn't fully implemented, then $P_{RESFailed} = P_{RETFailed} = P_{FOFailed} = 1$ and $P_{E1} = 1$, but $P_{E2} = P_{E3} = 0$. Then

$$R_C = 1 - EXP(\lambda). \qquad (9)$$

If implemented in a software module reset or restart operation is without switching to a backup version, i.e., $P_{FOFailed} = 1$, $P_{E3} = 0$, then

$$R_C = 1 - [P_{E1} \cdot EXP(\lambda) + \\ + P_{E2} \cdot ERLANG(\lambda, 2)]. \qquad (10)$$

Finally, if the software module has a backup version without restarting operations sold or re-start the application, i.e. $P_{RESFailed} = P_{RETFailed} = 1$, $P_{E2} = 0$, then

$$R_C = 1 - [P_{E1} \cdot EXP(\lambda) + P_{E3} \cdot (\gamma \cdot ERLANG(\lambda, 2) + \\ + \gamma \cdot HYPO(\lambda_1, \lambda_2))]. \qquad (11)$$

If we substitute (11) in equation (2), then we can get the following expression:

$$E[R] = \left[\prod_i^{n-1} R'_{C_i}\right] R_{C_n} W'. \qquad (12)$$

Next, using as input data as the temporal characteristics of the software module (Table 1), and the statistical evaluation of the Goel-Okumoto model parameters (Table 2), substituting (19) into (8), we determine the resulting value of the stationary readiness coefficient $K_{RDN_{MS}}$. Graph of dependence of $K_{RDN_{MS}}(T_o, t)$ for said original data represented in Figure 5.

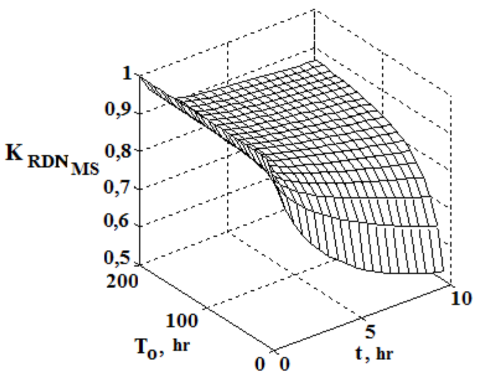

Fig. 5. Dependence of $K_{RDN_{MS}}(T_o, t)$

The results (Fig. 5), Pareto optimal, may be used to determine the rational duration of the test control of CI MS software with safe, fault-tolerant operation of process equipment as well as critical applications software. Further prospects of using the proposed models can be relating with optimizing test resource consumption of critical applications software sub-modules and also can be using to specify requirements for functional safety of different MS CI.

References

1. *CASE-assesment of critical software systems [Text] / V. S. Kharchenko, K. I. Netkacheva, A. O. Orekhova, O. M. Tarasyk, A. V. Gorbenko, V. V. Sklyar, E. V. Brezhnev, E. V. Babeshko, O. A. Illiashenko // Safety V. Kharchenko (edit) // Three-volume edition. – Department of Education and Science, Youth and Sports of Ukraine, National aerospace university named after N. Zhukovsky "KhAI", 2012. – Vol. 3. – 301 p.*

2. *Critical Infrastructures Safety: Mathematical and Engineering Methods of Analysis and Assurance [Text] / V. Kharchenko (edit). – Department of Education and Science of Ukraine, National aerospace*

university named after N. Zhukovsky "KhAI", 2011. – 641 p.

3. Stringfellow, C. An empirical method for selecting software reliability growth models [Text] / C. Stringfellow, A. Amschler Andrews // Empirical Software Engineering. – 2002. – Vol. 7. – P. 319 – 343.

4. Theoretical bases of designing information and control systems of spacecraft [Text] / V. V. Kulba, E. A. Micrin, V. V. Pavlov, V. N. Pavlov, V. N. Platonov, Ins-ut of managements problems named after V. A. Trapeznikova RAS. – M. : Science, 2006. – 579 p.

5. Pietrantuono, R. Software Reliability and Testing Time Allocation: An Architecture-Based Approach [Text] / R. Pietrantuon, S. Russo, Kishor S. Trivedi // IEEE Transactions on Software Engineering. – 2010. – Vol. 36, №. 3. – P. 323 – 337.

Поступила в редакцию 3.03.2014, рассмотрена на редколлегии 25.03.2014

Рецензент: д-р техн. наук, проф. Б. М. Конорев, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Харьков, Украина.

ОПТИМІЗАЦІЯ ТЕСТОВОГО КОНТРОЛЮ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КРИТИЧНОГО ДОДАТКА

О. В. Іванченко, Ю. Л. Явкун, О. А. Томашпольська

Уроки, які були засвоєні під час аналізу аварій та катастроф за участю різних критичних інфраструктур (КІ), свідчать про низький рівень відмовостійкості програмного забезпечення (ПЗ) систем управління КІ. Для усунення цієї проблеми запропоновано оптимізаційну процедуру, яка базується на коректному розподіленні тестового ресурсу ПЗ системи управління інфраструктурного утворення. Подальші перспективи застосування розроблених моделей пов'язані з оптимізацією витрат тестового ресурсу підмодулів зі складу програмного забезпечення критичних додатків і можуть бути використані для обґрунтування вимог до функціональної безпеки систем управління критичних інфраструктур.

Ключові слова: відмовостійкість, програмний модуль, тривалість тестування, марківські моделі.

ОПТИМИЗАЦИЯ ТЕСТОВОГО КОНТРОЛЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КРИТИЧЕСКОГО ПРИЛОЖЕНИЯ

О. В. Иванченко, Ю. Л. Явкун, А. А. Томашпольская

Уроки, извлеченные в ходе анализа аварий и катастроф с участием различных критических инфраструктур (КИ), свидетельствуют о низком уровне отказоустойчивости программного обеспечения (ПО) систем управления КИ. Для устранения этого пробела предлагается оптимизационная процедура, основанная на корректном распределении тестового ресурса ПО системы управления инфраструктурного образования. Дальнейшие перспективы использования разработанных моделей связаны с оптимизацией расхода тестового ресурса подмодулей из состава программного обеспечения критического приложения и могут быть использованы для обоснования требований функциональной безопасности систем управления критических инфраструктур.

Ключевые слова: отказоустойчивость, програмный модуль, продолжительность тестирования, марковские модели.

Иванченко Олег Васильевич – канд. техн. наук, доцент, доцент кафедры кибернетики и вычислительной техники, Севастопольский национальный технический университет, Севастополь, Украина, e-mail: vmsu12@gmail.com.

Явкун Юрий Леонидович – старший преподаватель кафедры кибернетики и вычислительной техники, Севастопольский национальный технический университет, Севастополь, Украина, e-mail: yavkun@mail.ru.

Томашпольская Александра Анатольевна – студентка кафедры кибернетики и вычислительной техники, Севастопольский национальный технический университет, Севастополь, Украина, e-mail: sashaa.tomash@gmail.com.