Assessing the reliability of the hardware and software complex of fault-tolerant control systems

Igor Kovalev[1,2,3,4*], *Dmitry* Kovalev[2,5], *Roman* Kovalev[6], *Valeria* Podoplelova[2,7], *Vasiliy* Losev[4], *Dmitry* Borovinsky[8], *Pavel* Gofman[4], and *Mavlyuda* Gadoeva[9]

[1]Siberian Federal University, Krasnoyarsk, Russia
[2]Krasnoyarsk State Agrarian University, Krasnoyarsk, Russia
[3] Navoi State University of Mining and Technology, Navoi, Uzbekistan
[4]Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
[5]National Research University "Tashkent Institute of Irrigation and Agricultural Mechanization Engineers", Tashkent, Uzbekistan
[6]JSC "Academician M F Reshetnev Information satellite systems", Zheleznogorsk, Russia
[7]Sochi State University, Sochi, Russia
[8]FSBEE HE Siberian Fire and Rescue Academy EMERCOM of Russia, Zheleznogorsk, Russia
[9]Bukhara State University, Bukhara, Uzbekistan

Abstract. Reliability assessment of hardware and software complexes is relevant due to the expansion of automation and intellectualization for sustainable development of mining and transport systems, energy complexes and mechanical engineering. This paper presents an approach to the reliability assessment of control systems based on functional block diagrams of software and control flows. When assessing the reliability of a control system, software is considered taking into account the close relationship with hardware. The article considers control systems that assume the performance of the required function periodically and repeatedly within the transportation and technological cycle of the system. This is typical for automation of mining and transportation systems, as well as energy complexes. It is taken into account that the main function within the transportation-technological cycle does not require a complex algorithm in contrast to the operating system. Thus, the proposed reliability assessment model allows to take into account the interaction between hardware and software of automated control systems, including the variant of their fault-tolerant execution.

1 Inrtoduction

When solving problems related to automation and intellectualization for sustainable development of mining and transport systems, energy complexes and mechanical engineering, much attention is paid to the issues of ensuring the reliability of operation of their hardware and software complexes (HSC). In modern automated control systems, digital

* Corresponding author: kovalev.fsu@mail.ru

systems mainly prevail due to their higher reliability and speed compared to analog systems [1-3]. However, it is very difficult to assess the reliability of digital control systems because they are usually multilevel and include complex error handling mechanisms at different levels of the system.

Modern software is integrated into the structure of HSC and, therefore, the reliability of the complex operation largely depends on the reliability of software. At the same time, software is an additional obstacle in assessing the reliability of control systems as a whole. This primarily applies to systems requiring ultra-high reliability and operating in real time [4-6].

In addition, when assessing the reliability of digital control systems, it is necessary to consider not only software and hardware separately, but also the interaction between software and hardware. This follows from the fact that software cannot be fully analyzed separately from hardware and vice versa [7].

In the hierarchical functional view of a control system, an example of which is shown in Figure 1, software is designed to perform the functions that the control system is required to perform.

A software system consists of software modules. The software modules perform their tasks through a combination of instruction sets provided by the microprocessor [8].

Some of the hardware components such as microprocessors and memory modules are used to execute a single instruction. That is, in order for the control system to fully perform the required function, the software determines the correct sequence of utilization of the hardware resources.

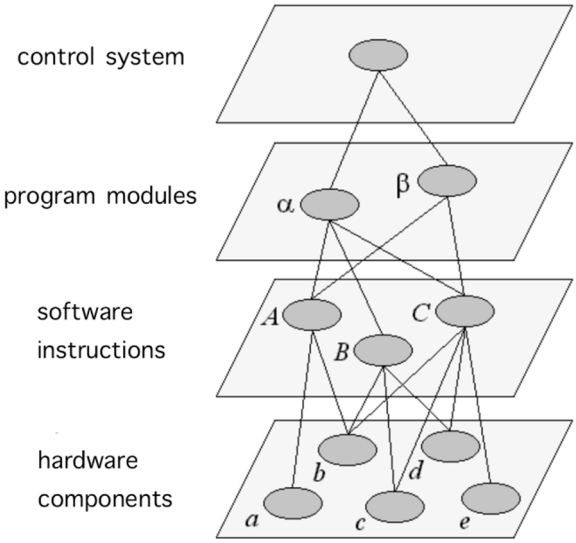

Fig. 1. Hierarchical functional representation of the management system.

A system failure thus occurs when the software fails to correctly determine the sequence of hardware resource utilization. Another case corresponds to a situation where an error occurs in one or more of the hardware resources being used, even though the software has determined the correct sequence of hardware resource utilization.

2 Materials and methods

2.1 Types of possible errors in control systems

The article considers control systems that assume the execution of the required function periodically and repeatedly within the transportation and technological cycle of the system [9]. This is characteristic of the automation of mining and transportation systems, as well as energy complexes. The main function within the transportation-technological cycle does not require a complex algorithm in contrast to the operating system. Therefore, functional block diagrams are mainly used for designing and coding software applications of control systems. In this approach, software is designed with a limited number of functional blocks (modules) such as adder, subtraction block, comparer and multiplier. These are then compiled directly into executable code without the need for further transformations by designers or programmers.

We will assume that software designed using the above procedure may contain the following errors:

- errors occurring directly in the program specification;
- errors occurring during the transformation of the specification into a functional block diagram;
- errors in individual functional blocks.

Hardware at the operational stage may have the following three types of errors:

- random errors;
- periodic errors
- permanent errors.

This approach is based on the concept proposed for assessing the reliability of control and measuring complexes [10] and information and control systems [11].

3 Results and discussion

Taking into account the functional representation of control systems proposed in the work (Figure 1) and taking into account the features of automation of mining, transport systems and energy complexes, we will next consider three possible options for the interaction of hardware and software components within the system. At the same time, we will take into account that the transport and technological cycle includes periodically and repeatedly repeated functions characteristic of systems of these production technologies.

So, in this section we will consider and discuss three main varieties of control systems, which differ by the presence or absence of fault tolerance mechanism, as well as the existence of errors in the software. Depending on these factors, the reliability model of the hardware and software complex of the automated control system will be built.

3.1 Software without errors and fault tolerance mechanism

For the case where the software has no errors and no fault tolerance mechanism (FTM), we will assume that system failure does not occur when the system is idle. For this type of system, the following statement is true. If the failure rate (FR) of hardware components is constant and the software is error-free, then an expression for the FR of a single instruction can be obtained as follows (equation 1). That is, the FR for one instruction is defined as the sum of the FR of the hardware components used by the instruction multiplied by the time required to process the instruction:

$$\lambda_{inst}^{j} = m^{j} \sum_{i} \lambda_{HW}^{i} \quad (1)$$

where m^j – time required to process the *j*-th instruction, λ_{HW}^i – failure rate of the *i*-th hardware component.

Based on the structure in Figure 1, where program modules and software instructions are at different levels of the hierarchy, we will establish the relationship between failures of instructions and software modules. The FR of one software module is determined as follows:

$$\lambda_{mod}^k = \sum_i p_i n^{ik} \lambda_{inst}^i \quad (2)$$

where *pi* – probability of the software branch and n^{ik} – total number of instructions *i,* used in the *k*-th program module. The probability pi is given by the operating profile of the software.

Then, taking into account equations 1 and 2, we write the following equation that determines the FR of the system:

$$\lambda_{sys} = \sum_k \lambda_{mod}^k \quad (3)$$

The considered case is typical for traditional development of hardware and software systems. Debugging of the complex is carried out in full and a conclusion is made that there are no errors in the software. In this regard, methods that ensure software fault tolerance are not used. However, anomalies in the functioning of the system hardware can give rise to situations where software branches (and corresponding instructions) are used that were not included in the testing pool. This can lead to malfunctions of the hardware and software complex and failures of the control system.

3.2 The software contains errors and there is no FTM at the board level

As noted earlier, software is not always perfect. It may include “sleeping” errors, have branches of program code execution that are not included in the testing pool, etc. Therefore, software-induced failure also needs to be considered and included in the control system reliability model. To this end, we expand equation (3) to take into account possible software failures:

$$\lambda_{sys} = \sum_k \lambda_{mod}^k + \lambda_{SW} \quad (4)$$

where λ_{SW} – is the software failure rate, which is determined by experimental software testing, based on the operational profile of the control system software [12].

3.3 The software contains errors and there is a FTM at the board level

The hierarchical functional representation of a control system, shown in Figure 1, allows the application of various fault tolerance methods at any level of the hierarchy, both in software modules and in hardware components. For example, typical hardware fault tolerance techniques are:

- error detection;
- correction codes for memory;
- parity bits for data buses;
- schemes with self-control;
- control timer.

One of the methods or a combination of them can not only detect errors, but also restore the system after they are detected. Since the hardware and software complex as a whole is considered, these methods help to increase the reliability of a multi-level control system. In order to reflect the use of these hardware fault tolerance methods, we transform equation (4) to the following form:

$$\lambda_{sys} = (1 - C) \sum_k \lambda_{mod}^k + \lambda_{SW}$$

where *C* is the coverage ratio, defined as follows:

$$C = \frac{\text{number of errors eliminated by FTM}}{\text{number of errors that occurred in the system}}.$$

For modern hardware and software systems with a complex structure and a large number of program code instructions, there is no general approach to representing C in analytical form. It is noted that this is practically impossible. As a rule, the value of C is determined based on experiments with artificial introduction of errors into the control system [13]. This is done at the system design stage and requires additional time resources and labor.

4 Conclusion

The proposed approach is intended to assess the reliability of control systems that involve performing the required functions periodically and repeatedly within the framework of the transport and technological cycle of the system. As noted earlier, this is typical for the automation of mining and transport systems, as well as energy complexes. That is, the main assumption when constructing a reliability model was that the system under study performs basic functions periodically and repeatedly. Potentially, the reliability model discussed in this work can be extended to the class of information systems. But this generalization requires additional research. In order for this reliability model to be used in the general case, it requires modification in order to generalize the scope of its application.

References

1. M.V. Saramud et al., J. Phys.: Conf. Ser. **2388**, 012048 (2022). https://doi.org/10.1088/1742-6596/2388/1/012048
2. M. Saramud, V. Losev, M. Karaseva, AIP Conf. Proc. **2969**, 020011 (2024). https://doi.org/10.1063/5.0182143
3. I.V. Kovalev, V.V. Losev, A.O. Kalinin, Modern Innovations, Systems and Technologies **3(2)**, 0243-0253 (2023). https://doi.org/10.47813/2782-2818-2023-3-2-0243-0253
4. I.N. Kartsan, Modern Innovations, Systems and Technologies **3(4)**, 0322-0331 (2023). https://doi.org/10.47813/2782-2818-2023-3-4-0322-0331
5. D.I. Kovalev, P.K. Zaitsev, Informatics. Economics. Management **2(3)**, 0201-0209 (2023). https://doi.org/10.47813/2782-5280-2023-2-3-0201-02096
6. Hamdioui, et al., Proceedings - Design, Automation and Test in Europe, DATE, 129-134 (2013) https://doi.org/10.7873/DATE.2013.040
7. P. Kuznetsov, Y. Tynchenko, V. Kolesnik, Informatics. Economics. Management **1(1)**, 0217-0228 (2022). https://doi.org/10.47813/2782-5280-2022-1-1-0217-0228
8. J.G. Choi, P.H. Seong, Reliability Engineering & System Safety **91(3)**, 261-269 (2006). https://doi.org/10.1016/j.ress.2005.01.005
9. D.I. Kovalev, V.A. Podoplelova, T.P. Mansurova, Informatics. Economics. Management **1(1)**, 0110-0120 (2022). https://doi.org/10.47813/2782-5280-2022-1-1-0110-0120
10. P. Gee-Yong, S. Jang, Nuclear Engineering and Technology **46**, 55-62 (2014). https://doi.org/10.5516/NET.04.2012.067
11. B. Littlewood, P. Popov, S. Lorenzo, Safety Science **40(9)**, 781-796 (2002). https://doi.org/10.1016/S0925-7535(01)00084-4

12. D. Tang, H. Hecht, *An approach to measuring and assessing dependability for critical software systems,* Proceedings of the 8 International Symposium on Software Reliability Engineering, Albuquerque, NM, USA, Nov. 1997, pp. 192-202 (1997). https://doi.org/10.1109/ISSRE.1997.630864
13. B.W. Johnson, Fault-tolerant Computing Systems **214**, 1-5 (1989). https://doi.org/10.1007/978-3-642-75002-1_5