Lyfar V, Ivanov V., Baturin O., Gerasymenko K.

SETTING UP THE GOALS OF RESEARCH TASKS IN DETERMINING SAFETY INTEGRITY LEVELS OF BASIC SOFTWARE AND HARDWARE COMPLEXES IN PROCESS CONTROL SYSTEMS

This article analyses scientific and technical problem of estimating the level of security completeness of basic software and hardware complexes of the control system. In the article authors examined the possibility of developing information technology, methods and models of the decision support system. In the article the authors highlighted requirements for the elements of a complex control electronic system, approaches and models of qualitative and quantitative determination of reliability indicators of software.
Key words: *Safety integrity level, reliability of software, electronic programmable devices, reliability, safety.*

1. Importance and relevance of the research.

When developing control computer systems implemented in the automated process control systems (APCS) of high-risk industries, the requirements for safety integrity level (SIL) and the reliability of software and hardware are regulated. The SIL is determined by the integral integrity level (Safety integrity level SIL). Carrying out SIL analysis for electronic / electrical / programmable electrical (E / E / PE) systems is an important difficulty both in the determination of safety functions, due to problems in determining the severity of the negative consequences for failures of E / E / PE elements and in the quantitative definition of performance indicators of elements and components of the control system. This particularly applies to software systems that support the operation of such systems. At the moment, software reliability assessment is carried out at best by rank methods without any quantitative criteria, which excludes the possibility of comparative analysis of alternative software developments. In addition, the rank methods presented in [3] are designed as an assembly of requirements and recommendations for hybrid control systems and contain significant subjective approaches that interfere with the creative development of software.

Therefore, it is important to obtain a reliable estimate of the overall reliability of computing systems at the present stage, taking into account the reliability of not only the hardware of the computer system (the level of equipment and any associated control system is most optimally determined by analyzing the types and consequences of failures [1-4] but also the reliability of its software.

One of the modern scientific and technical directions of studies of software reliability is the assessment of the level of safety completeness of basic software and hardware complexes of automated process control systems (APCS).

2. Problem statement.

Let`s consider the requirements for software on the example of the standard [3]. The structure of the standard is shown in Figure 1. In Appendices A and B, requirements are set for protection against software failures. Appendix D contains the requirements for the Operating Manual (Safety Manual) regarding the features of the software. According to IEC 61508-3, an urgent recommendation is labeled HR. For example, from the level of SIL3 (safety integrity level) and higher for some life cycles of the development of basic software and hardware systems of the Automated Control System.

Table - Example of methods / tools according to IEC 61508-3

Method / Tool	Life Cycles for the Development of Basic Software and Hardware Complexes of Automated Control Systems
Semiformal methods	Specification of software security requirements Software design and development: software architecture design
Direct traceability between security requirements and security software requirements	Specification of software security requirements
Reverse traceability between security requirements and perceived security needs	Specification of software security requirements Software design and development: software architecture design
Automated means of developing specifications for the support listed above, suitable methods / tools	Specification of software security requirements Software design and development: software architecture design
Detecting and diagnosing errors	Specification of software security requirements Software design and development: software architecture design
Gradual disabling of functions	Specification of software security requirements Software design and development: software architecture design
Choosing the appropriate programming language	Specification of software security requirements Software design and development: software architecture design
A subset of a language	Specification of software security requirements Software design and development: software architecture design
Certified products and certified translators	Specification of software security requirements Software design and development: software architecture design

Computer aided design tools	Specification of software security requirements Software design and development: software architecture design
Programming with protection	Specification of software security requirements Software design and development: software architecture design
Modular approach	Specification of software security requirements Software design and development: software architecture design
Design and coding standards	Specification of software security requirements Software design and development: software architecture design
Functional testing and testing using the "black box" method	Design and development of software: verification and integration of software modules Integration of programmable electronic devices
Performance testing	Design and development of software: testing and integration of software modules
Model-based testing	Design and development of software: testing and integration of software modules
Testing the interface	Design and development of software: testing and integration of software modules
Test management and automation tools	Design and development of software: testing and integration of software modules

It must be noted that the one of the most important distributions of models and methods for estimating reliability parameters for the phases of the life cycle of the development of basic software and hardware complexes of automated process control systems are the coding phase and the testing phase [5].

1. Encoding. At this stage, the real code of the program is written, usually in a high-level language. Sometimes it is possible to use low-level programming languages to achieve high performance or to interface with non-standard hardware. The code is constantly analyzed for errors.

2. Testing. This phase is one of the most important and laborious. It can take from 30 to 60% of the time and material resources of the project.

For the methods presented, it is much more relevant, demandable and possible to develop models and approaches to obtaining, in addition to qualitative and also quantitative indicators of reliability, which more objectively show the compliance of the software.

3. Analysis of data and methods.

The coding phase of the software. The number of errors in the executable code directly depends on the errors made at the coding stage. One of the most commonly used parameters for evaluating the correctness of software is the error density. The initial error density is proposed to be estimated as [5]

$$D = C\, F_{ph}\, F_{pr}\, F_m\, F_s \qquad (1)$$

Where F_{ph} - is the test phase coefficient; F_{pr} - coefficient of command programming; F_m - the coefficient of experience and "maturity" of the software development process; F_s - is the structuring factor; C - constant, which determines the number of errors / KLOC (errors per thousand lines of source code). The coefficients F_{pr}, F_m, F_s and C depend only on the skill and experience of the development team. When evaluating the coefficient of command programming (F_{pr}), the error density depends on specific people, their experience in writing programs and debugging. Following settings can be accepted: "High", "Medium", "Low". Numerical indicators are defined and asked by the expert. For the coefficient of experience and "maturity" of the software development process (Fm), the following parameter values are accepted: "Level 1", "Level 2", "Level 3", "Level 4", "Level 5". Numerical indicators are defined and asked by the expert. The structuring factor (F_s) allows you to take into account the dependence of the error density on the programming language (the ratio of the number of code in assembler and the language of high level):

$$F_s = 1 + 0{,}4a \qquad (2)$$

where a - is the ratio of the amount of code in the Assembler and the high-level language. It is assumed that the code in assembler can contain 40% more errors. An obvious drawback is the lack of methods for quantifying the estimation of the coefficients $F_{pr,}$ F_m.

The software testing phase. The testing phase is the longest and can take up to 60% of the total project. During testing, the greatest number of errors in the software is detected. Let`s consider some of the most commonly used models which are used in software reliability testing [7,8].

Evaluation of the reliability of the software for the work

To predict the reliability of software in this model, data are used about the number of errors eliminated during the process of linking programs into the software system and during its debugging. Based on these data, the parameters of the reliability model are calculated, which can be used to predict reliability indicators in the process of using software in analogy with non-recoverable technical objects. In the model under consideration, it is assumed that in successive runs

of the program, the input data sets are random and are selected in accordance with the distribution law corresponding to the actual operating conditions. The model is based on the following assumptions: • at the initial moment of program layout in the software system, there are E_0 errors in them. When adjusting programs, new errors are not introduced; • the total number of machine instructions in programs is constant; • the intensity of program failures is proportional to the number of errors left in it after debugging during the time.

$$\lambda = [E_O - E_c(\tau)]\, C \tag{1}$$

Where $E_c(\tau)$- is the number of errors eliminated during the debugging time. C is the proportionality factor to be determined. Thus, the model has two different time values: the debugging time, which can be measured in months, and the program time after debugging t or the total running time of the program. The debugging time includes the time spent on detecting errors, eliminating them, checking checks, etc. The value of the failure rate is considered constant during the entire operation time $(0,t)$ and changes only when errors are detected and corrected, while the time t again is counted from zero. In view of the assumptions made for a fixed probability of the absence of errors in the program, during the operating time $(0,t)$ will be:

$$\mathrm{P}(t,\tau) = \exp\{(-C[E_O - E_c(\tau)]t)\}, \tag{2}$$

Jelinski – Moranda Model

This model is based on the following assumptions:

• the operating time before the next failure is distributed according to the exponential distribution law;

• the intensity of program failures is proportional to the number of errors remaining in the program;

• program failure occurs when one program error occurs and its operation is restored when this error is fixed.

Then, according to these assumptions, the probability of failure-free operation of the program on the *i* - m work interval after the next recovery will be:

$$\mathrm{P}(t_i,\tau) = exp(-\lambda_i\, t_i),$$
$$\text{где } \lambda_i = C_D[E_O - (i-1)] \tag{3}$$

failure rate of the program at the *i* - m work interval, which starts after the elimination of the *(i –1)* – m failure; C_D coefficient of proportionality; E_0 - the number of software errors in the software at the time of its operation. The disadvantage is that if the C_D value is not accurately determined, the intensity of the program failures can become negative, which leads to a meaningless result. In addition, it is assumed that the correction of detected errors does not introduce new errors, which is not always satisfied.

Mills model

Mills model (refers to static models and differ models discussed above, especially in that it does not take into account the time of error) provides entering into the program before testing a certain number of known (artificial) errors. Errors are introduced randomly and recorded in the protocol of artificial errors. It is assumed that all errors (both natural and artificially introduced) have the same probability of being found in the testing process.

The program is tested for some time and statistics about the detected errors are collected. Let after the testing we found n_c with our own program errors and n_u of artificially introduced errors. Then the initial number of errors in the program E_0 can be estimated from the Mills formula

$$E_0 = n_c \frac{E_u}{n_u} \tag{4}$$

where Eu is the number of artificially introduced errors.

The Mills model allows us to solve the inverse problem-testing the hypothesis of the initial number of errors E_0. We suppose that the program initially (at the time of testing) contains *K* errors, $E_0 = K$. We introduce artificially into the program E_u errors and test it until all artificially introduced errors are detected. Let it be found that there are no own program errors. Probably that there were initially *K* errors in the program can be calculated from the ratio

$$P(E_0 = K) = \begin{cases} 0, & n_c > K \\ \dfrac{E_u}{E_u + K + 1}, & n_c \le K \end{cases} \tag{5}$$

The disadvantages of this model are:

• the need to introduce artificial errors (this process is poorly formalized);

• a fairly wide assumption of the value of *K*, which is based solely on the intuition and experience of the person conducting the assessment, the model assumes a large influence of the subjective factor.

Conclusions

The described approaches and models have a number of advantages and disadvantages. One of the most important drawbacks is the lack of a common methodology allowing the selection and use of methods for quantitative analysis and assessments of the level of software safety integrity and methods and models that could be used in decision support information technology to develop requirements for reliability indicators for E / E / PE and software control systems. It is necessary to develop such methodologies, methods and models with the help of which it would be possible to develop a generalized reliability model of the software and hardware complex, to implement an automated analysis of the failure combinations of both hardware and software components of the control system.

Література

1. Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью = Ч. 1. Общие требования : Functional safety of electrical, electronic, programmable electronic safety-related systems. Part 1. General requirements: национальный стандарт Российской Федерации ГОСТ Р МЭК 61508-1-2007 / Федеральное агентство по техническому регулированию и метрологии. – М.: Стандартинформ, 2008. - V, 44 с.
2. Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью = Ч. 2. Требования к системам : Functional safety of electrical, electronic, programmable electronic safety-related systems. Part 2. Requirements for systems: национальный стандарт Российской Федерации ГОСТ Р МЭК 61508-2-2007 / Федеральное агентство по техническому регулированию и метрологии. – М.: Стандартинформ, 2008. - V, 58 с.
3. Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью = Ч. 6. Руководство по применению ГОСТ Р МЭК 61508-2-2007 и ГОСТ Р МЭК 61508-3-2007 : Functional safety of electrical, electronic, programmable electronic safety-related systems. Part 6. Guidelines on the application of GOST R IEC 61508-2-2007 and GOST R IEC 61508-3-2007 : национальный стандарт Российской Федерации ГОСТ Р МЭК 61508-6-2007 / Федеральное агентство по техническому регулированию и метрологии. - Москва : Стандартинформ, 2008. - V, 62 с.
4. Функциональная безопасность в непрерывных производствах. Руководство по безопасности процессов / IEC 61511:2004 Functional Safety – Safety Instrumented Systems for the Process Industry Sector/ национальный стандарт Российской Федерации ГОСТ Р МЭК 61511-1-2011 / Федеральное агентство по техническому регулированию и метрологии. – М.: Стандартинформ, 2013. – V, 66 с.
5. Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью = Ч. 3. Требования к программному обеспечению: Functional safety of electrical, electronic, programmable electronic safety-related systems. Part3. Software requirements (IDT): национальный стандарт Российской Федерации ГОСТ Р МЭК 61508-3-2007 / Федеральное агентство по техническому регулированию и метрологии. - Москва : Стандартинформ, 2007. - V, 62 с.
6. Ковалев И.В. Анализ проблем в области исследования надежности программного обеспечения: многоэтапность и архитектурный аспект / – Вестник СибГАУ. 2014. № 3(55)
7. Шкляр В.Н. Надёжность систем управления: учебное пособие / – Томск: Изд- во Томского политехнического университета, 2009. – 126с.
8. Осипенко, Н.Б. Основы стандартизации и сертификации программного обеспечения; М-во образ. РБ, Гомельский государственный университет им. Ф. Скорины. - Гомель: ГГУ им. Ф. Скорины, 2007. – 137с.

References

1. Funktsional'naya bezopasnost' sistem elektricheskikh, elektronnykh, programmiruemykh elektronnykh, svyazannykh s bezopasnost'yu. Obshchie trebovaniya: [Functional safety of electrical, electronic, programmable electronic safety-related systems]. Part1. General requirements: natsional'nyy standart Rossiyskoy Federatsii GOST R MEK 61508-1-2007 / Federal'noe agentstvo po tekhnicheskomu regulirovaniyu i metrologii. – Moscow: Standartinform, 2008. - V, p. 44 (in Russian)
2. Funktsional'naya bezopasnost' sistem elektricheskikh, elektronnykh, programmiruemykh elektron-nykh, svyazannykh s bezopasnost'yu. Trebovaniya k sistemam: [Functional safety of electrical, electronic, pro-grammable electronic safety-related systems]. Part 2. Requirements for systems: natsional'nyy standart Rossiyskoy Federatsii GOST R MEK 61508-2-2007 / Federal'noe agentstvo po tekhnicheskomu regulirovaniyu i metrologii. – Moscow: Standartinform, 2008. - V, p. 58 (in Russian)
3. Funkcional'naya bezopasnost' sistem ehlektricheskih, ehlektronnyh, programmiruemyh ehlektronnyh, svyazannyh s bezopasnost'yu. Rukovodstvo po primeneniyu GOST R MEHK 61508-2-2007 i GOST R MEHK 61508-3-2007: [Functional safety of electrical, electronic, programmable electronic safety-related systems]. Part 6. Guidelines on the application of GOST R IEC 61508-2-2007 and GOST R IEC 61508-3-2007 : nacional'nyj stan-dart Rossijskoj Federacii GOST R MEHK 61508-6-2007 / Federal'noe agentstvo po tekhnicheskomu regulirova-niyu i metrologii. Moscow: Standartinform, 2008. - V, p.62 (in Russian)
4. Funkcional'naya bezopasnost' v nepreryvnyh proizvodstvah. Rukovodstvo po bezopasnosti processov / IEC 61511:2004 [Functional Safety – Safety Instrumented Systems for the Process Industry Sector] nacional'nyj standart Rossijskoj Federacii GOST R MEHK 61511-1-2011 / Federal'noe agentstvo po tekhnicheskomu regulirovaniyu i metrologii. – Moscow: Standartinform, 2013. – V, p.66 (in Russian)

5. Kovalev I.V. Analiz problem v oblasti issledovaniya nadezhnosti programmnogo obespecheniya: mnogoetapnost' i arkhitekturnyy aspekt / – Vestnik SibGAU. 2014, No. 3(55) (in Russian)

6. Rukovodstvo po funkcional'noj bezopasnosti dlya sistem, svyazannyh s bezopasnost'yu, i drugih primenenij s urovnem SIL2, SIL3 v sootvetstvii so standartami MEHK 61508 MEHK 61511 / GM [International Technology for safety] / Via San Fiorano 70, 20058 Villasanta (MI) Italy, 2013. – D100, p.77 (in Russian)

7. Osipenko, N.B. Osnovy standartizatsii i sertifikatsii programmnogo obespecheniya M-vo obraz. RB, Gomel'skiy gosudarstvennyy universitet im. F. Skoriny. - Gomel': GGU im. F. Skoriny, 2007. – pp. 137 (in Russian)

8. Shklyar V.N. Nadezhnost' sistem upravleniya [The reliability of control systems]: uchebnoe posobie / – Tomsk: Izd-vo Tomskogo politekhnicheskogo universiteta, 2009. – p. 126 (in Russian)

Висвітлена та проаналізована науково-технічна проблема оцінки рівня повноти безпеки базових програмно-апаратних комплексів АСУТП, можливість розробки інформаційної технології, методів та моделей системи підтримки прийняття рішень, щодо розробки вимог до елементів складної керуючої електронної системи. Сформульовані задачі досліджень для вирішення проблем оцінки показників надійності апаратного та програмного забезпечення базових комплексів. Розглянуті підходи та моделі якісного та кількісного визначення показників надійності програмного забезпечення.
***Ключові слова**: Safety integrity level, надійність програмного забезпечення, електронні програмовані пристрої, надійність, безпека.*

This article analyses scientific and technical problem of estimating the level of security completeness of basic software and hardware complexes of the control system. In the article authors examined the possibility of developing information technology, methods and models of the decision support system. In the article the authors highlighted requirements for the elements of a complex control electronic system, approaches and models of qualitative and quantitative determination of reliability indicators of software.
***Key words:** Safety integrity level, reliability of software, electronic programmable devices, reliability, safety.*

Довідка про авторів:
Volodymyr Lyfar, Head of Department of Programming and Mathematics, Ph.D. in Technical Sciences, Associate Professor, Volodymyr Dahl East Ukrainian National University
+38 097-547-03-97
E-mail: lyfarva61@ukr.net

Vitalii Ivanov, Ph.D. in Technical Sciences, Department of Programming and Mathematics , Volodymyr Dahl East Ukrainian National University
+ 38095-801-07-99
E-mail: vetgen75@gmail.com

Oleksandr Baturin, Senior Lecturer of Department of Programming and Mathematics, Volodymyr Dahl East Ukrainian National University
+38050-919-21-84
E-mail: aibaturin1973@gmail.com

Kostyantyn Gerasymenko, Deputy Director for Automation Systems in Power Engineering in SRPA "Impulse"
+38 (06452) 6-01-94
E-mail: gerasymenko_ke@imp.lg.ua