

Antonov A.V., Zharko E.F., Promyslov V.G.

PROBLEMS OF EVALUATION OF SOFTWARE DEPENDABILITY AND QUALITY IN INDUSTRIAL AUTOMATION AND CONTROL SYSTEMS

The article describes aspects of evaluation of quality, reliability of software regarding theoretical basis, methods, main tendencies and problems in this area.

Keywords: *dependability, quality assurance, software, IACS.*

1. Introduction

The development of automation of complex technological objects, whose malfunctions lead to large economic, ecological losses, threats to health or life of people, is characterized by a tendency of developing industrial and automation control systems (IACS) realizing much more complicated algorithms of data control and analysis with usage of complex software/hardware [1]. Assurance of IACS dependability at all stages of its life cycle is based on the qualitative and quantitative analysis, which according to normative documentation, should be also conducted at all stages. The qualitative and quantitative analysis of dependability should consider two components of software/hardware system (SW/HW): hardware and software. However, the quantitative analysis of dependability for software components of SW/HW systems (programs, software (SW)), unlike the hardware, has difficulties [2]. These difficulties are due to differences in the causes of failure occurrence in hardware and software components of SW/HW. The difficulty of calculating the dependability of classical program functioning within the framework of Turing universal machine consists in the fact that the algorithm of its functioning is not stochastic. Program failure appears after imposing on the determined function, which corresponds to the program's algorithm, stochastic process describing input data for it. Determinacy of algorithm for classical programs leads to the fact that at best there is a calculated probability of failure for the system: input data + program. Calculation of probability characteristics of output process, even in case of a known function, can be difficult, and at presence of errors during its implementation it is possible to consider such task as unfeasible. The problem was realized by many experts and, as there is a requirement for evaluation of dependability of programs applied as a part of various systems, there are models and methods allowing evaluation of dependability of the program [3].

Together with dependability, sometimes substituting it, the term "quality" of software is used. Quality of software can be defined as compliance to the explicitly set functional and operational requirements, explicitly specified development standards and implicit characteristics. Qualitative and quantitative indicators of quality of programs, unlike probabilistic reliability, can be used efficiently for analysis of types and consequences of failures, comparative analysis of variants of technical solutions on provision of dependability, organization of maintenance. Qualitative and quantitative indicators of quality of programs have a doubtless practical value.

In the article, most frequent methods of calculation of software dependability are analyzed, problems with application of these methods are shown. A review of main approaches to evaluation of software quality is given.

2. Causes of failures of hardware and software

In GOST 27.002-89 the following causes of occurrence of hardware failures are highlighted:

1. Imperfection or violation of fixed rules and (or) norms of development and designing (design errors);

2. Imperfection or violation of fixed process of manufacture or repair carried out at a repair enterprise;

3. Violation of fixed rules and (or) operational conditions;

4. Natural processes of ageing, wear, corrosion and fatigue at observance of all fixed rules and (or) norms of designing, manufacture during operation.

In a flow of hardware failures the greatest weight, as a rule, has failures of the 2nd and 4th type.

The program is a collection of instructions expressed in one of the languages and recorded on a material object of long-term or temporary storage. The failure of a program's material object is a failure of hardware, on which the program is executed, or an object's failure. Program's failure appears as mismatch of value on program's output to a preset value. The informational contents of the program does not vary itself (does not fail). Therefore, for failures of programs the reasons of types 1, 2 and 3 are characteristic. The greatest share among all SW failures, as a rule, is the failures caused by the first reason. The key feature of failures of this type both for software programs (otherwise called SW errors) and for hardware consists in the fact that errors are brought in the program (hardware) accidentally, while they appear deterministically at occurrence of specific events. For programs, the moment of failure occurrence is defined by configuration and value of a set of input data, level of loading of computer resources, informational environment of the program at a stage of its execution and by similar factors.

A wide experience and theoretical basis on quantitative methods of analysis of hardware dependability has been gained. The program cannot change in time without change of properties of a material object **by itself** and its failure is the demonstration of errors contained in the program. The quantitative analysis of software failures has a number of problems:

- complexity of receipt of analytical expression for the function describing operation of the program;
- stochastic process linked to input data, level of loading of computer resources, informational environment can have complicated or unknown distribution;
- presence of software program errors has a nonlinear influence on type of the function describing operation of the program, as well as the type of this function is unknown.

The document [4] contains the classification of types of software program errors by their origin:

1. System errors during setting of purposes and tasks related to the creation of program;

2. Software programming errors in texts of programs and data descriptions (syntax errors);

3. Algorithmic errors of development under a direct formulation of requirements to program's functions and algorithmic errors of implementation of these requirements.

The first type of errors is not specific to the software and is not the subject of consideration. The overwhelming majority of errors of the second type are eliminated by means of automatic check of programs (compilers). With *algorithmic errors* the situation is different: you can be convinced that the program works correctly and there are no algorithmic errors only in the course of program *testing* (testing allows revealing all types of errors). Due to the large area of check, the test coverage for any real program is not complete, i.e. there is always a probability that there are errors in the program.

For classic software programming languages it is known that the number of errors depends on the volume of program's source code, technology of software programming, qualification of personnel participating in the program's development and resources allocated for the testing [5]. These indices can be considered as constants for the closed development teams with the established norms of development and testing.

However, nonlinear link between number of program errors and probability of their appearance during program usage leads to negative results under the attempt to use evaluation by a number of program errors for calculation of probability of its failure [4]. Despite the problems with substantiation of applicability of probability methods of software reliability evaluation, a considerable amount of methods of quantitative evaluation of reliability of programs have been developed and applied. Most frequently used methods and problems with their application are specified below.

3. Methods of analysis of reliability of software programs

There is a big variety of areas of usage of models from the point of view of modeling hardware and program failures; however, the greatest attention is given to models of evaluation of dependability of software capable to be integrated into existing complex model of calculation of dependability of a control system. In complex model, consequences of types of failures of components in digital system as a whole for the object are considered. The main methods of analysis of reliability are classified according to their main objective in accordance with the fact, how analysis of architecture of the program system is carried out:

1) Ascending method (mainly directed on research of consequences of single failures):

a) Event tree analysis (ETA) and modifications;

b) Failure mode and effects analysis (FMEA) and modifications;

2) Descending methods (directed on research of consequences of combinations of failures);

a) Failure tree analysis (FTA);

b) Markov analysis;

c) Petri net analysis;

3) Hazard and Operability study (HAZOP);

4) Statistical methods of evaluation of dependability.

These methods of analysis are applicable both for evaluation of characteristics of quality and for evaluations of quantitative characteristics during forecasting of system behavior during operation. Reliability of the result depends on accuracy and correctness of data on main events. In practice, combinations of descending and ascending analyses are used to raise integrity of analysis.

In the paper [4] the main requirements to the models used in methods of reliability calculation are selected:

1) Model should explain both already occurred failures and allow to predict failures in the future;

2) Model should be based on substantial characteristics of modeled system;

3) Model should be based on clear and authentic suppositions;

4) Model should express in numerical form dependences between failures;

5) Model should be based on simple and easily studied concept;

6) Input data required for model construction should be accepted as authentic by a considerable part of expert community;

7) Model should distinguish between single and multiple failures;

8) Model should distinguish between failures during performance of function and intermediate failure;

9) Model should allow to the user obtaining of checked data, including probability of failure and evaluation of reliability of result;

10) Model should allow analysis of failure scenarios of digital components in interaction with non-digital components;

11) Model should not use momentary information on system's status.

Table 1 shows comparison of most often used methods for evaluation of indices of reliability from the point of view of their application for program components. Data for table 1 are taken basically from [4], wherein the subjective character of data is underlined.

It is possible to see that methods of calculation of reliability given in Table 1 on the whole have the following disadvantages:

1. Incompleteness of components and their failures;

2. Absence of commonly accepted philosophical basis of software modeling of intensity and probability of failures and methods for their quantitative evaluation;

3. Weakness of evaluation of failure parameters – failure rates, distribution of failure modes and factors of common cause failure (CCF).

Doubtfulness of application of methods of reliability calculation for receipt of absolute values of reliability indices does not mean a necessity of total rejection of probabilistic methods of their evaluation. Methods can be used for analysis of types and consequences of failures of individual IACS components and for a system as a whole, as well as for analysis of its functionality. Markov methods and Petri nets are considered in [13] as the most perspective from the point of view of receipt of quantitative evaluation of software reliability and consideration of mutual influence of software and hardware components of the system.

4. Evaluations of software quality: quality models

Complexity of SW engineering and maintainability process is in many respects stipulated by special requirements presented to its quality. The base quality model can be defined as the structured set of properties, which are necessary for accomplishment of definite purposes [14]. Advantage of a base quality model consists in decomposition of such objects significant for software, as life cycle processes, software product, a number of characteristics/ sub-characteristics.

SW users feel the needs in creation of the SW quality models necessary for quality evaluation both qualitatively

Table 1. Comparative characteristic of methods*

Method \ Requirement	1	2	3	4	5	6	7	8	9	10	11
Continuous tree of events [6]	x	x	x	x	0	?	?	x	?	?	0
Dynamic tree of events [7]	x	X	x	x	x	?	?	?	x	x	0
Markov models [2]	x	x	x	x	0	?	?	x	x	x	0
Petri nets [8]	x	x	x	x	0	?	?	?	x	?	0
Methodology of dynamic data-flow graphs [9]	x	x	x	?	x	?	?	?	x	x	x
Dynamic tree of failures [10]	x	?	?	?	x	?	x	?	x	?	x
Diagram of sequence of events [11]	x	x	x	x	0	?	?	?	x	x	0
Evaluation by metrics of software [12]	x	?	0	0	?	?	x	x	0	0	x

** In the table we have the following designations: X – the property is covered, 0 – the property is not covered, ? – the coverage is doubtful.*

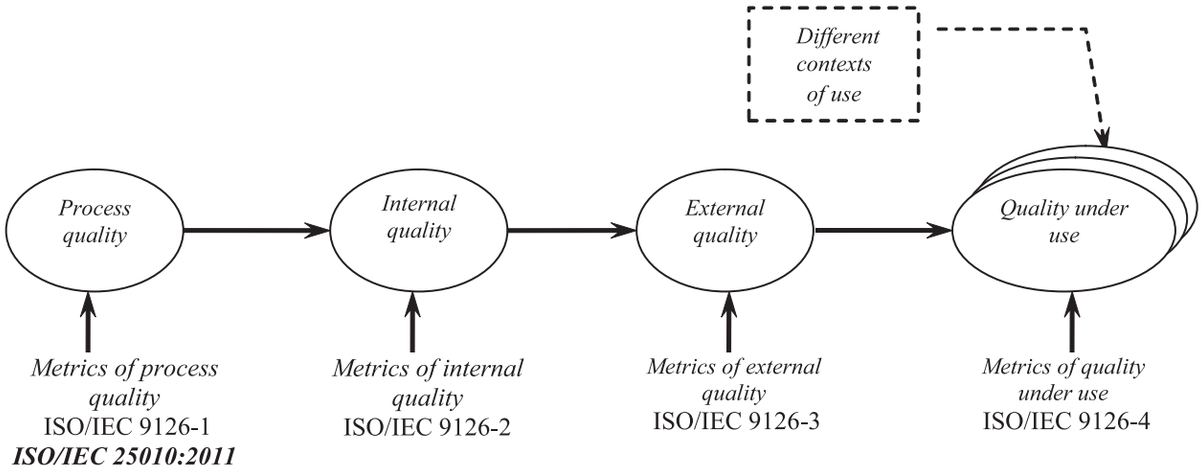

Fig. 1. Main aspects of software quality according to standards ISO/IEC 9126-1:2001 and ISO/IEC 25010:2011

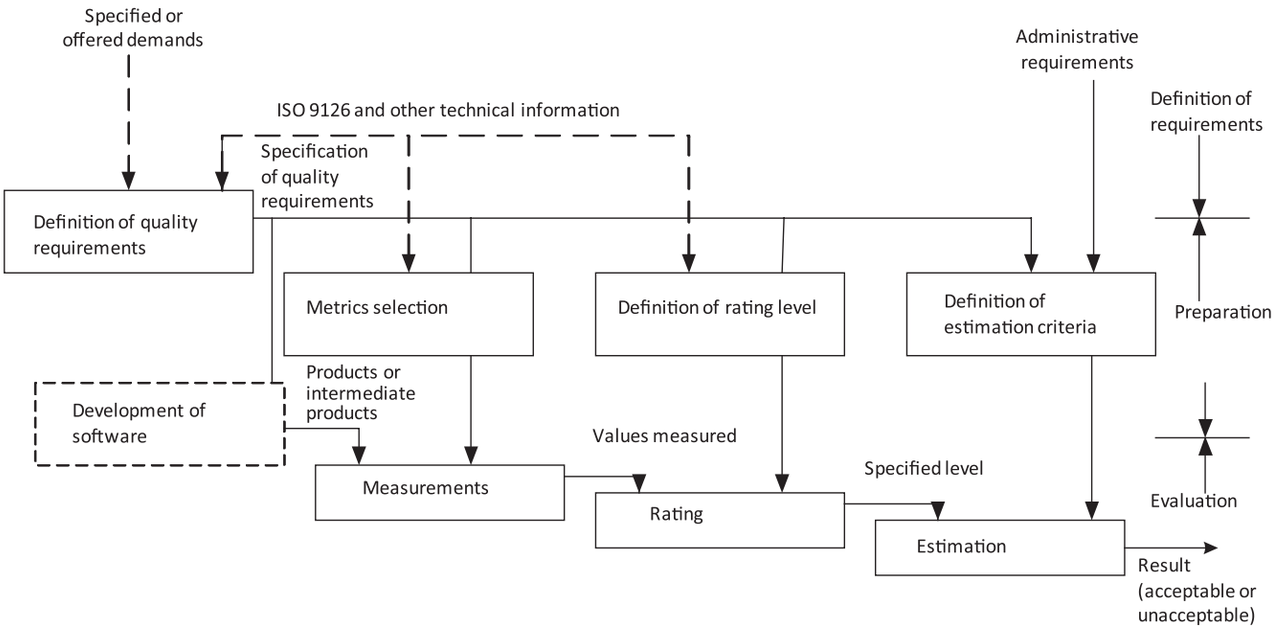

Fig.2. Factors and attributes of external and internal quality of software according to ISO/IEC 9126

and quantitatively [15]. Quality models, which are available now, in most cases are hierarchical models on the basis of quality criteria and indices (metrics) linked to them. All quality models can be divided into three categories according to the methods, on the basis of which they were developed. The first type includes theoretical models based on a hypothesis of ratios between variables of quality. Models of "data control" based on statistical analysis refer to the second type. Finally, the combined model, in which intuition of the researcher is used for definition of the necessary model type and data analysis, is used for definition of constants of a quality model. But all these models link interests of the user, i.e. system's initial properties with internal properties, which are clear to developers.

SW quality is defined in the standards ISO/IEC 9126-1:2001 and ISO/IEC 25010:2011 as any collection of its characteristics referred to possibility to meet stated or meant requirements of all interested persons.

There are distinguished concepts of internal quality linked to characteristics of SW itself without consideration of its behavior, external quality characterizing SW from the point of view of its behavior and SW quality during usage in various contexts, i.e. the quality, which is felt by users at concrete scenarios of SW work. For all these aspects of quality there are introduced metrics allowing their evaluation. Besides, for creation of reliable SW the quality of technological processes of its development is important. Mutual relations between these aspects of quality by the scheme accepted in various quality models are shown in Fig. 1.

Fig. 2 shows the model of evaluation of SW quality according to ISO/IEC 9126.

5. Conclusion

The problem of evaluation of dependability (failure-free functioning) of the program, unlike the problem of creation of qualitative program, probably, has no solution in general case within the limits of classical Turing machine and there is a quantity of fundamental problems linked to the determined character of program's functioning. The quantitative evaluation of reliability of systems based on software can be obtained only by a combination of actual data from several sources; however, even then there will be a considerable distrust to absolute digits for dependability parameters. Now there are no commonly agreed methods and data about failures for quantitative evaluation of reliability of digital systems. Probably, problem's solution lays in the field of transition from classical universal Turing machine to its modification in the form of probabilistic Turing machine or to functional programming, which are free from the above limitations, permitting formal verification of the program.

References

1. **Byvaikov M. E., Zharko E.F., Mengazetdinov N.E., Poletykin A.G., Prangishvili I.V., Promyslov V.G.** Experience of designing and implementation of system of upper unit level of NPP IACS//Automation and telemechanics. 2006. No. 5. p. 65-79.

2. **Smith D., DeLong T., Johnson B.W.** A Safety Assessment Methodology for Complex Safety-Critical Hardware/ Software Systems//International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies. Washington, DC, November, 2000

3. **Lipaev V.V.** Dependability of software. M: SINTEG, 1998.

4. **Aldernir T., Miller D.W., Stovsky M.P., Kirschenbaurr J., Bucci P., Fentiman A.W., Mangan L.T.** Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments (NUREG/CR-6901).

5. **Halstead M.H.** Elements of Software Science. New York: Elsevier, 1977.

6. **Devooght J., Smidts C.** Probabilistic Reactor Dynamics l: The theory of continuous event trees // Nuclear Science and Engineering. 1992. Vol. 111. No. 3. P. 229-240.

7. **Acosta C., Siu N.** Dynamic event trees in accident sequence analysis: Application to steam generator tube rupture // Reliab. Engng & System Safety. 1993. Vol. 41, No. 2. P. 135-154.

8. **Goddard P.L.** A Combined Analysis Approach to Assessing Requirements for Safety Critical Real-Time Control Systems // Reliability and Maintainability Symposium, 1996 Proceedings. International Symposium on Product Quality and Integrity., Annual. P. 110-115.

9. **Stamataletos M. et.al.** Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners, Version 1.1, August, 2002.

10. **Andrews J.D., Dugan J.B.** Dependency modeling using failure-tree analysis // Proceedings of the 17 International System Safety Conference, The System Safety Society, Unionville, Virginia, 1999. P. 67-76.

11. **Matsuoka T., Kobayashi M.** An analysis of a dynamic system by the GOFLOW methodology // Proc. ESREL'96/PSAM III, Crete, 1996. P. 1547-1552.

12. **Smidts C., Li M.** Validation of a Methodology for Assessing Software Quality. Report UMDRE 2002-07. February, 2002.

13. NEA/CSNI Recommendations on assessing digital system reliability in probabilistic risk assessment of nuclear power plants. 2009. 157 p.

14. **Fitzpatrick R.** Software Quality: Definitions and Strategic Issues. Staffordshire University, School of Computing Report. 1996. 35 p.

15. **Zharko E.F.** Comparison of models of software quality: analytical approach//XII All-Russia conference on control problems. VSPU-2014. Moscow, June, 16-19th, 2014: Works. M: IPU of the Russian Academy of Sciences, 2014. p. 4585-4594.