The minimization of inter-module interface for the achievement of reliability of multi-version software

I. Kovalev, P. Zelenkov, and S. Ognerubov

Abstract—**This article shows that the use of the modular principle for the stage of technical designing of multi-version software is connected to the optimization process of the structure and interaction of its components. The new problem definition of the designing of the multi-version, modular structure of software, which provides the minimization of inter-module interface, is presented. This problem definition allows us to create the multi-version set of components which are used in software in the predefined sequences as well as in the random sequences. Simultaneously, the set of versions matches the predefined conditions, providing the maximum reliability for personal computers and the minimal values of the attributes of the inter-module interface.**

Index Terms— **inter-modular interface, modular structure, multi-version software, reliability.**

I. The preconditions for the improvements for multi-version software

Reliability plays the key role in the development of the software. Many scientists all over the world are engaged in the problem of software reliability tasks [1, 2, 3]. One can pay attention to the approach connected with the development of software systems in critical spheres.

The assigning models and the methods of estimation of parameters of reliability to the phases of software development the life-cycle of systems of information control and processing, is very important during the designing of multi-version structure of software [4]. The phases which have the greatest influence on the reliability of software are the most important. They are the designing structure phase, the coding phase, the testing components phase and the testing system as a whole phase.

Using the narrow point of view, the definition of the structure of software and the other methods of information processing is a combination of properties and attributes which are designed to satisfy users. The wider definition is a model of information system which is complete enough to understand the principles of system functioning [5].

Igor Kovalev is with the Siberian State Aerospace University, Krasnoyarsk, 660014 Russia (e-mail: kovalev.fsu@mail.ru).

Pavel Zelenkov is with the Siberian State Aerospace University, Krasnoyarsk, 660014 Russia (e-mail: zelenkow@rambler.ru).

Sergei Ognerubov is with Siberian Federal University, Krasnoyarsk, 660041 Russia (e-mail: ogss@rambler.ru).

The reliability of the structure includes the reliability of individual components. The failure of an individual component leads to the inoperability of that component and potentially other components, but it does not lead to the inoperability of the whole system. System failure or the failure of its functions could increase downtime. The definition of the downtime of a system is a period of time during which the system or a part of the system cannot perform operations. System downtime leads to a significant decrease in performance, therefore reducing the downtime of the system is one of the most important goals in the structure of distributed software development process.

The main concepts of the theory of software reliability are based on concepts from the theory of hardware reliability, but there are significant differences between the principles of reliability assurance for software and other technical systems.

Experiments show that the use of redundancy in the early stages of the development process reduces (if it is possible) the probability of errors. Reliable software can be created using the detailed structure development process and the process of finding errors in components which have the greatest influence on the reliability of the system. These components are defined as the most frequently used components or as the most connected with many other components, on whom they have influence, of the same structure.

Modular organization essentially involves splitting software into complete modules, standardization of links between them and building the hierarchy of relationships between components which is defined by the order of their execution. Execution is the control transfer from the calling module to the method being called, which transfers control back after its completion. Execution of a set of modules is done by transferring control to the lower level of the hierarchy manager of modules which is included in the software.

The main methods of modern software development are based on the decomposition of functions using modular-hierarchic principles [6]. Simultaneously, there are limits for the complexity of components and links between them on every hierarchic level of software. As a result, assuming an increase in the amount of tasks, the combined complexity of the system grows significantly slower in comparison to unstructured designing.

It is obvious that the use of the modular principle on the

technical designing stage is connected with the process of optimization of the amount single components of multi-version software of processing and control information systems and the links between them. Therefore the set of tasks comprising the design of modular systems which are based on multi-version software architecture includes the optimal selection of modules of multi-version software and informational arrays, the contents of inter-module interface, and the structure of the system as a whole.

It is known, that the main criteria for designing modular systems of processing information and control on the technical designing stage [7] are the minimum complexity of inter-module interface, the minimum period of time for operations between RAM and external storage devices during processing, the minimum unused data volume during this period of time, the minimum technological complexity of data processing algorithms, which is the generalization of the criterion of "transport factor" during the process of coding algorithms for the solution of given tasks, the maximum informational performance of modular systems during processing tasks and the maximum reliability of data processing.

The procedures selection and volume of each module, the selection and volume of informational arrays, the complexity and structure of inter-module interface, the degree of redundancy of procedures and informational elements, the amount of RAM, the amount of channels for connections between RAM and external storages, the procedures execution order, the acceptable system creation expenses and period of time are used as limitations during execution of tasks creating optimal, modular systems of multi-version data processing and control as well as the tasks listed in the previous paragraph.

The initial data for process of designing tasks are technical specification and the analysis report on the current control system. Using that data, the set of processing data tasks, the set of processing data procedures, including alternate ones which are multi-versions, the set of informational elements which are connected with data processing procedures and identified by type input, intermediate and output, the set of possible interactions between the processing data procedures and the informational items, the technological adjacency matrix or frame matrix, the set of the acceptable procedure execution orders, the characteristics of components and technical tools are defined [7].

The solutions for defined problems provide an opportunity to design a flowchart of the multi-version, modular structure of software on the system level, which provide the minimum interface between software modules. It should be noted, that duplication of multi-version items (multi-versions) in the modules of software provides a decrease in value of the criterion of informational connection between these modules.

II. The Mathematical Problem Definition of the Designing of the Multi-Version, Modular Structure of Software

Consider the problem definition of the designing of multi-version and modular structure of software, which provides minimization of the inter-module interface.

The two or more multi-versions are added to the module being designed and the following variables must be taken into the account [8].

R_{ij}– the reliability of the j-th version of the i-th module;

X_{ij} – a bool variable, if the j-th version is used in the i-th module, then the variable is true or else the variable is false;

n – the amount of modules;

m_i – the amount of versions of the i-th module;

R_i– the reliability of the i-th module;

$$R_i = 1 - \prod_{j=1}^{m_i} (1 - R_{ij})^{X_{ij}}, \quad i = \overline{1, n}. \tag{1}$$

It is important to mention that the reliability of the j-th version of the i-th module in this article is not a stochastic variable but a constant which is defined in the technical specification. The stochastic estimation of that variable should be done by the developers of software using their experience in the area of the software development. The methods to determine the significance level are described in "The data analysis methods" by professor A.I. Rouban.

Programs are created and orders, which define which modules and which of their execution orders are used:

K – the amount of programs;

S_k – the set of modules which match the k-th program.

Then, additional variables are presented using the limitations for system reliability and inter-module interface.

F_k – the usage of the k-th program;

R – system reliability.

R maximization can be presented as,

$$\max \quad R = \sum_{k=1}^{K} F_k \prod_{i \in S_k} R_i. \tag{2}$$

Inter-module interface is the amount of the same multi-versions of modules which are used by different programs [9].

The following variable is presented in order to formalize the problem definition which provides inter-module minimization.

W_{ik} – a bool variable, if the i-th module is used by the k-th program, then the variable equals true, otherwise it equals false.

$$Y_{kj} = \begin{cases} 1, if \ \sum_{i=1}^{n} W_{ik} X_{ij} \geq 1 \\ 0, if \ \sum_{i=1}^{n} W_{ik} X_{ij} = 0 \end{cases}. \tag{3}$$

The Variable Y_{kj} is used to formalize the relationship between the system and single multi-versions of modules.

The next equation as a criterion is used to minimize the inter-module interface on the level of single versions.

$$\min \sum_{j=1}^{J} \sum_{k=1}^{K-1} \sum_{k'=k+1}^{K} Y_{kj} Y_{k'j}. \tag{4}$$

This problem definition provides an opportunity to create a multi-version set of module components which are used in the predefined sequence. The Obtained set of multi-versions will meet the requirements, providing maximum software reliability and the minimum value of the characteristics of the inter-module interface.

References

Basic format for books:

[1] L. Lazic, “Software Quality & Testing Metrics”, Dubrovnik, Croatia: 7th European Computing Conference “Recent advances in information science”, 2013, pp 16-17

[2] [1] R. Wason, P. Ahmed, M. Qasim Rafiq, “A Tool for Runtime Reliability Estimation and Control Using Automata Based Software Reliability Model”, Brasov, Romania: 4th International Conference on Mathematical Models for Engineering Science (MMES '13), 2013, pp 51-57.

[3] A. Ghazarian “Dimension-Oriented Software Engineering (DOSE): Rule-Based Software Development through Traceability Patterns”, Paris, France: 3rd European Conference of Computer Science (ECCS '12), 2013, pp 10-11.

[4] I.V. Kovalev, D. V. Kapulin, R.U. Tsarev, “The structure reliability of software for the informational-control systems,” Krasnoyarsk, Russia: Krasnoyarsk State Agricultural University, 2011, pp 170-193.

[5] I.V. Kovalev, “System of multi-version development of spacecrafts control software,” Verlag Sinzheim, Germany: Pro Universiate Verlag Sinzheim, 2001, pp 70-87.

[6] A. G. Mamikonov, V. V. Kulba, “The designing of optimal, modular IPS,” Moscow, Russia: Nauka, 1986, pp 30-43.

[7] I. V. Kovalev, R. V. Younoussov, “Fault-tolerant software architecture creation model based on reiabilityevalutation,” Advances in Modeling and Analysis, vol. 7, no. 3, pp. 31–44, Dec. 2002.

[8] I. A. Kapchinskii, A. S. Kunetsov, A. V. Shtentsel, “The principles of the multi-version software designing,” Vestnik SibGAU, vol. 1, no. 18, pp. 18–23, Jan. 2008.

[9] A. V. Novoy, A. V. Shtentsel, “The estimation of reliability of the structure of multi-version software for the processing and control information systems,” Vestnik SibGAU, vol. 3, no. 20, pp. 73–79, Mar. 2008.