Formation of optimal composition of the modules of single-function multiversion software for automated control system of the satellite communication system

V I Kudymov¹, V V Brezitskaya², P V Zelenkov², I N Kartsan³,⁶ and Yu N Malanina⁵

¹ JSC “Academician M F Reshetnev Information satellite systems”, 52 Lenin street, Zheleznogorsk, Krasnoyarsk region, 662972, Russia
² Siberian federal university, 79 Svobodny avenue, Krasnoyarsk, 660041, Russia
³ Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russia
⁴ Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations, 61 Uritskogo street, Krasnoyarsk, 660049, Russia
⁵ Management Department, Irkutsk State Transport University, Irkutsk, Russia
⁶ E-mail: kartsan2003@mail.ru

Abstract. When creating the automated control systems (ACS) for satellite systems, the main task is to ensure high quality and reliability of all hardware and software complex components. It concerns not only spacecrafts but also the ground segments of system. In this direction there is an active work. In this article we want to focus on the development of software for satellite communication systems that can be implemented in both space and ground segments. One of the promising directions in this area is the introduction of multiversion technology. The proposed model significantly develops the approach of development of highly reliable software, as it deals with multiversion software for ACS of satellite communication systems, consisting of one or more programs, and each program consists of a number of modules, the serial execution of which corresponds to the successful implementation of the function. Below is a model that is applicable to various structures of multiversion software for ACS of satellite communication systems (from a very simple structure to a more complex) and gives the designer the flexibility in choosing the appropriate modules for the designed software of the system.

1. Introduction
We briefly show the principle of software development using the multiversion programming approach. Classical multiversion or N-variant programming is a technique in which all versions are designed according to identical basic requirements, and the solution on the correctness of the conclusion is based on a comparison of all the conclusions (fig.1). In multiversion software, all versions of the same module are executed simultaneously (in parallel), and, accordingly, the input of the decision-making unit, that implements the selection algorithm, receives data from all versions.
All versions have different data processing algorithms, the versions are developed according to the same specification and are functionally equivalent.

The solution selection algorithm (usually a voting algorithm) for choosing the correct conclusion is universal and does not depend on a particular application. Since all versions are built according to identical requirements, the use of multiversion software requires certain development costs. But the complexity (the difficulty of development) will not necessarily be much greater than the complexity of forming a single version.

We also describe the basic assumptions of the methodology of multiversion software design, concerning, first, the use of modular programming methods; secondly, the independence of the development of modules versions and the feasibility of assessing their reliability and cost using the modules COTS support (such modules have been produced and tested independently, so their reliability can be evaluated using any of the available models, and their actual cost is the cost of procurement and support); third, there is a limit on the cost of multiversion software for ACS of satellite communication system.

As part of our problem, we consider the structure of models and present the formulation of the task and the methods of solution for each of them by introducing the following symbols:

- $K$ – number of functions of the multiversion software for ACS of satellite communication system, which are to be performed;
- $F_k$ – the frequency of use of software k-function, $k = 1, 2, ..., K$;
- $n$ – the number of multi-version modules of software for ACS of satellite communication system;
- $m_i$ – number of versions of $i$-module, $i = 1, ..., n$;
- $R$ – the probability of failure-free operation of multiversion software for ACS of satellite communication system;
- $R_i$ – the probability of failure-free operation of $i$-module;
- $R_{ij}$ – the probability of failure-free operation of $j$-version of $i$-module;
- $Z_{ij}$ – Boolean variable equal to 1, if $j$-version for $i$-module has been chosen, otherwise 0;
- $C_{ij}$ – cost of procurement and support of $j$-version for $i$-module;
- $B$ – a limitation on the cost of multiversion software of ACS for satellite communication system.

Next we consider the proposed approach.

2. The proposed method of formation of the optimal composition of the modules of single-function software

In this SNR model the optimal composition of modules of single-function multiversion software for ACS of satellite system is formed without the software redundancy. Multiversion software consists of a single program that performs one "main" function. The program is composed of a set of modules of
sequential execution. More than one version of each module is available, but due to strict cost
limitations (and such a case should not be excluded from consideration) and/or the non-
critical nature of part or all of the software system, it is undesirable or impossible to keep multiple versions of
modules. The model proposed for this situation allows to form the optimal composition of a set of
modules for a single program, maximizing reliability, with the existing cost limitations, due to which
the total cost of development remains acceptable. The target function is the following:

\[
\max R = \max \left\{ \prod_{i=1}^{n} \sum_{j=1}^{m} Z_{ij} R_{ij} \right\},
\]

\[
\sum_{j=1}^{m} Z_{ij} = 1, \quad i = 1, \ldots, n;
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{m} Z_{ij} C_{ij} \leq B;
\]

\[
Z_{ij} = 0, 1; \quad i = 1, \ldots, n; \quad j = 1, \ldots, m.
\]

The target function of the SNR model reflects the sequential execution of modules. The set of
limitations (2.2) ensures that one version is necessarily selected for each module. The
limitations (2.3) ensure that the total cost does not exceed B. This task corresponds to the nonlinear integer
programming task. A simple numerical example for this model, considered below, demonstrates the
applicability of the branch and bound algorithms to the task solution.

3. The example of realisation of the proposed method
First we enter the initial data. Then assume that \( n = 3, m_1 = 3, m_2 = 3, \) and \( m_3 = 2 \). Reliability and
cost characteristics of the modules are represented by the following pairs:

\[
[R_{11} = 0.90; C_{11} = 3], \quad [R_{12} = 0.80; C_{12} = 1], \quad [R_{13} = 0.85; C_{13} = 2],
\]

\[
[R_{21} = 0.95; C_{21} = 3], \quad [R_{22} = 0.80; C_{22} = 2], \quad [R_{23} = 0.70; C_{23} = 1],
\]

\[
[R_{31} = 0.98; C_{31} = 3], \quad [R_{32} = 0.94; C_{32} = 2].
\]

Given \( B = 6 \), the task can be formulated as follows:

\[
\max (0.9Z_{11} + 0.8Z_{2} + 0.85Z_{13}) \cdot (0.95Z_{21} + 0.8Z_{22} + 0.7Z_{23}) \cdot (0.98Z_{31} + 0.94Z_{32}),
\]

under limitations:

\[
Z_{11} + Z_{12} + Z_{13} = 1, \quad Z_{21} + Z_{22} + Z_{13} = 1, \quad Z_{31} + Z_{32} = 1,
\]

\[
3Z_{11} + 2Z_{12} + 2Z_{13} + 3Z_{21} + 2Z_{22} + Z_{23} + 3Z_{31} + 2Z_{32} \leq 6,
\]

where \( Z_{11}, Z_{12}, Z_{13}, Z_{21}, \ldots, Z_{32} = [0, 1] \).

The optimal solution found by the proposed method is \((Z_{12}, Z_{21}, Z_{32})\) with the value of the target
function 0.714 and the cost of 6.

It can be noted that the task is feasible if

\[
\sum_{i=1}^{n} (\min C_{ij}) \leq B.
\]

Let \( Z_{i} \) correspond to the choice of the module for the task \( i, (i = 1, \ldots, n) \), and let \( Z_{i}^{o}, i = 1, \ldots, n \) be
the optimal solution of the target functional when the limits (3) are ignored. Obviously, if \((Z_{1}^{o}, \ldots, Z_{n}^{o})\)
the optimal solution is achieved, it should also be the optimal solution for the values of the target function

\[
\prod_{j=1}^{m} (\max R_{ij}).
\]
Without loss of generality we can assume that \((Z_1, ..., Z_k)\) for \(k < n\) is a particular solution of our task. The upper limit on this particular solution can be given as

\[
\left( \prod_{i=1}^{k} R_i \right) \left( \prod_{i=k+1}^{n} \max R_i \right). \tag{4}
\]

The left part \((4)\) reflects the actual contribution of a particular solution to the value of the target function, while the right part indicates the optimal choice, ignoring the cost limitations (i.e. choice \((Z_{k+1}^1, ..., Z_n^1))\). It is obvious that \((Z_1, ..., Z_k)\) cannot provide any feasible executable solution, if

\[
\sum_{i=1}^{k} C_i^o + \sum_{i=k+1}^{n} (\min C_i) > B. \tag{5}
\]

The upper limit \((4)\) is achieved, if

\[
\sum_{i=1}^{k} C_i^o + \sum_{i=k+1}^{n} C_i^o \leq B, \tag{6}
\]

where \(C_i^o\) is the cost corresponding to \(Z_i^o\) for \(i = k+1, ..., n\).

The proposed branch and bound algorithms begins by selecting a known executable solution for the task that can be a lower limit (LL). For example, one of the possible executable solutions, if it exists, has been given by the choice of the cheapest versions for each module. At each i-level of the decision tree, we choose one of the versions for the i-module \(1 \leq I \leq n\). For each particular solution we define three values:

1) \(a\) — the value of the upper limit according to \((4)\);
2) \(b\) — possible lowest cost for any solution that includes a particular solution specified by the value to the left of inequality sign \((5)\);
3) \(c\) — the solution cost \((Z_1, Z_2, ..., Z_k, Z_k^{k+1}, ..., Z_n)\), given by the value to the left of inequality sign \((6)\).

The particular solution \(Z_1, Z_2, ..., Z_k\) is accepted if simultaneously either \(b > B\), or \((3)\ a < LL\). A new solution is considered to be found whenever \(C \leq B\) (solution \((Z_1, Z_2, ..., Z_k, Z_k^{k+1}, ..., Z_n)\)). A new feasible solution becomes a new mandatory solution if \(a > LL\). The method begins with the root node branching at level 0 to \(m_1\) branches, each of which corresponds to the choice of \(Z_i^o\) for \(j = 1, ..., m_i\). The next node where the branching will be performed, is selected according to the maximum upper limit.

4. Conclusion
This article solves one of the problems of developing the highly reliable software for ACS of satellite systems. The method of forming the optimal composition of modules of single-functional multiversion software without redundancy is presented. The example with calculation and computation of the optimal composition of the test sample is given. The limitations and their essence are described for use in the presented method.

Thus, it should be noted that the implementation of the proposed method will significantly save money for the development of highly reliable multiversion software.

Acknowledgements
This research was supported by the Ministry of Education and Science of the Russian Federation (agreement from 26.09.2017 № 14.577.21.0246; unique ID project RFMEFI57717X0246).

References
[1] Kartsan I N, Zelenkov P V, Tyapkin V N, Dmitriev D D and Goncharov A E 2015 IOP Conf. Ser.: Mater. Sci. Eng. 94 (1) doi: 10.1088/1757-899X/94/1/012010
[2] Fateev Yu L, Dmitriev D D, Tyapkin V N, Kartsan I N and Goncharov A E 2015 IOP Conf. Ser.: Mater. Sci. Eng. 94(1) 012022 doi: 10.1088/1757-899X/94/1/012022
[3] Kartsan I N, Zelenkov P V, Tyapkin V N, Dmitriev D D and Goncharov A E 2016 IOP Conf. Ser.: Mater. Sci. Eng. 155(1) 012010 doi: 10.1088/1757-899X/155/1/012010
[4] Tyapkin V N, Fateev Y L, Dmitriev D D, Kartsan I N, Zelenkov P V, Goncharov A E, Nasyrov I R 2016 IOP Conf. Ser.: Mater. Sci. Eng. 22(1) doi: 10.1088/1757-899X/22/1/012035
[5] Kartsan I N, Tyapkin V N, Dmitriev D D, Goncharov A E and Kovalev I V 2017 IOP Conf. Ser.: Mater. Sci. Eng. 255(1) doi: 10.1088/1757-899X/255/1/012009
[6] Kartsan I N, Dmitriev D D, Sokolovskii A V, Ratushnyak V N, Gladyshev A B and Kovalev I V 2017 IOP Conf. Ser.: Mater. Sci. Eng. 255(1) doi: 10.1088/1757-899X/255/1/012010
[7] Kartsan I N, Goncharov A E, Dmitriev D D, Ratuschnyak V N and Kovalev I V 2017 IOP Conf. Ser.: Mater. Sci. Eng. 255(1) doi: 10.1088/1757-899X/255/1/012011
[8] Tyapkin V N, Kartsan I N, Dmitriev D D and Efremova S V 2017 Algorithms for adaptive processing of signals in a flat phased antenna array International Siberian Conference on Control and Communications SIBCON 2017 (Astana, Kazakhstan 29 - 30 June 2017) doi: 10.1109/SIBCON.2017.7998452
[9] Gruzenkin D V, Chernigovskiy A S and Tsarev R Y 2017 N-version software module requirements to grant the software execution fault-tolerance Advances in Intelligent Systems and Computing 661 293
[10] Gruzenkin D V, Tsarev R Y and Pupkov A N 2016 Technique of selecting multiversion software system structure with minimum simultaneous module version usage Advances in Intelligent Systems and Computing 465 375-86
[11] Tsarev R Y, Kovalev I V, Soloviev E V, Prokopenko A V and Knyazkov A N 2016 Ant colony optimization for multiversion software synthesis International Journal of Soft Computing 11 136
[12] Engel E A and Kovalev I V 2017 Solar irradiance forecasting based on the multi-agent adaptive fuzzy neuronet Lecture Notes in Computer Science 10386 135-40
[13] Kovalev I, Zelenkov P, Karaseva M and Kovalev D 2015 Ant algorithm modification for multiversion software building Lecture Notes in Computer Science 9140 222-8
[14] Yronen Y P, Yronen E A, Ivanov V V, Kovalev I V and Zelenkov P V 2015 IOP Conf. Ser.: Mater. Sci. Eng. 94(1) 012023
[15] Kovalev I, Loginov Yu and Zelenkov P 2015 An integrated system of training engineers for aerospace industry in siberia using innovative technology of the student project-and-team work Procedia - Social and Behavioral Sciences 174 537