N-version Design of Fault-Tolerant Control Software for Communications Satellite System

Vitaly A. Kulyagin, Roman Yu. Tsarev,
Alexander V. Prokopenko, Alexander Yu. Nikiforov
Department of Informatics
Siberian Federal University
Krasnoyarsk, Russia
Email: vitalyas86@rambler.ru, tsarev.sfu@mail.ru

Igor V. Kovalev
Systems Analysis and Operations Research
Siberian State Aerospace University
Krasnoyarsk, Russia
Email: kovalev.fsu@mail.ru

***Abstract*—The article considers the problem of fault-tolerant satellite control software design. To provide the control software fault-tolerance the N-version programming based on program redundancy is used. In the paper the peculiarities of the design of control software for communications satellite system are presented. The article suggests the model for the choice of optimal structure of N-version software under resource requirement constraints. To solve the optimization problem it is proposed an algorithm. Numerical example illustrates how fault-tolerant control software for communications satellite system can be designed in order to increase software reliability using N-version programming.**

***Keywords*—N-version programming; N-version software; fault-tolerance; optimization; control; communications satellite system**

I. INTRODUCTION

Main objectives of space technology developers today are the following: increasing effectiveness of existing spacecraft control software use and optimizing development and design process for new ones. With the modern power and reliability of hardware, software reliability has become the major issue in the realization of highly dependable communications satellite system.

Nowadays, satellite development and implementation industry use the serial methods along with the individual manufacturing, characterized by a high unification level of the base software, engineering methods and software design approaches on the one hand, and by the use of custom-made on the other hand.

Orbital exploitation and functioning of a satellite is regulated by the technology control cycles (TCC), which determine time and informational interrelations between separate satellite control circuits [1]. That is characterized by periodicity, planned duration and ordering in solving tasks of control and information processing. As a consequence of potential engineering malfunction or a modification of the objective action software, the change of satellite parameters and functioning modes is possible during the exploitation. It can affect both structure and algorithms of the control software. So, in the process of recovering a satellite from abnormal modes (accordingly to the abnormal TCC), a range of the control software characteristics, as well as the characteristics of a control process (reliability, fault-tolerance, processing speed), determines a probability for a successful completing of this operation together with extent of damage.

At present, such the requirements as speed and cost efficiency of development, external support tools of execution, testing and analysis, as well as availability of methods and software tools for a wide range of the specialists responsible for communications satellite control systems and technological control cycles in general, have the higher importance among all the requirements of software quality, software design methods and methodologies [2, 3].

The primary control software requirements and the standard structure are determined by the specific character of communications satellite control system functioning under real conditions and by introducing of the satellite control software as the software product. Among those requirements first of all the following ones are marked out: the need for the effective memory use and the productivity; the ability for mass multiplication, the prolonged and sustained functioning; the high-level documentation; maintenance effectiveness, modernizing ability, reliability; portability and accompaniment ability. All of the enumerated requirements should apply at the same time with the primary functional tasks of communications satellite system.

For the wide range of different satellite types control tasks the structure of software realizing them can be represented by the typical scheme that includes the following software groups: communication with external customers, computational process organization, computational control and providing of reliable functioning, functional tasks solving according to satellite TCCs. Each of the groups includes the programs realizing complete functions.

The enumerated programs always present at the real software of satellite control processes and their capacity depends on the complexity, structure and peculiarities of the solving problems accordingly to the formed TCC. The objective determines the compound of program group that provides the functional tasks solving.

The satellite control processes are characterized by the periodicity of tasks solving and information processing. The

This research was supported by the Mikhail Prokhorov Foundation through the Mobility for Academics program.

same kind tasks repeating time and admissible lagging of control actions generation are determined by the relation between the accuracy of primary parameters of controlled object estimation and their variation rapidity. Thus, TCC regulating the process of basic functional tasks solved by the software complex and the time of response to entering information are extremely significant factors in the process of the given type of software creation. They determine the software structure and programming quality, the methods of communication, methods of computational process control and other characteristics.

Under the conditions of extreme charge of communications satellite system the characteristics of control software reliability gets the specific significance. The term of errors in software is associated with the software characteristics. The results of different software error analysis showed that the complex software packages could not exist in absolutely-tuned state. Some percentage of undetected errors remains in software in the process of its whole life cycle [4, 5].

The accordance of TCC software to the enumerated above characteristics provides the ability of applying of N-version programming approach to the given class of software. N-version programming as a methodology for designing fault-tolerant and high-reliability software enables to solve the mentioned problems subject to integration with the software engineering design methods and to the adequate description of spacecraft control technology [6-9].

The idea of multiversion or N-version programming (NVP) has been introduced by A. Avizienis in 1977 as *an independent generation* of $N \geq 2$ functionally equivalent software modules from the same *initial specification* [10]. The numerous theoretical and practical investigations of this methodology defined three composing elements of the NVP which play the base role in designing fault-tolerant software [10-14]:

1. the initial specification and multi-version programming process guaranteeing independence and a functional equivalence of N custom programming processes;

2. the result (N-version software or NVS) of the NVP process for which the tools of the concurrent execution are available together with the cross-check points and cross-check vectors;

3. the external tools of software versions maintenance (N-version executive or NVX) which provide the execution of the NVS and supply it with the decision algorithms in the cross-check points.

Thus, the NVP has the goal of an independent and concurrent generating of $N \geq 2$ functionally equivalent software modules (versions), which are to be executed and a decision about a software functioning accuracy is to be made. The satellite control software should be enhanced by redundant versions of the software modules as it is made in multiple (redundant) computation channels that act in the critical conditions with the requirements of fault-tolerance.

It should be noted that N-version programming guarantees that the imperfections of one software version cannot cause the faults in the whole safe-critical control process. And at the maintenance phase under the software developing and modifying, the most of the attention is to be paid to the specification, which allows getting the control system specialists to take part in the N-version components building and to fulfill the software version reliability requirements.

II. Optimization of N-version control software for communications satellite system

In designing N-version control software for communications satellite system, the goal is to assign program modules for the execution of critical control tasks within limited cost and time margins. One important requirement concerns the application process duration T. With regard to this requirement, the program modules are assigned according to the following rules (which will be explained in the subsequent sections): realizability of control process; correction of the so-called evolution time vector as a result of realizability conditions violation; test of restrictions for the number of the read or write channels for intermediate results storage. We present a formulation of the reliability model under side conditions, e. g. resource requirements and cost and timing constraints, and propose the methods of solving this optimization problem.

Assuming that N-version control software for communications satellite system has to solve a number of I tasks we have to assign to each of the tasks a certain program module. Some of the tasks might be structurally identical, equivalent or similar so that they can be carried out by the same program module (e.g. in the case of cyclic (event-triggered or time-triggered) reuse of the module in a control software or use of the module under varying parameterization). Let J be the number of the used program modules. Thus, with regard to the required modules we can divide the I tasks into J classes. Let n_j, $j = 1, \ldots, J$ denote the number of tasks in class j, and A_j be the set of indexes of the tasks in this task class:

$$\mathbf{card}\, A_j = n_j,\ j = 1,\ldots,J\ .$$

Let us assume that when using N-version programming, we have K_j versions of the module j which is to be used for j-th task class ($j = 1, \ldots, J$), i.e. the degree of redundancy can be varied, depending on the criticality of the module or the existing potential of versions (Fig. 1). To formally describe the application of a version to a problem, we introduce the following Boolean variable: $X_{ij}^{kj} = 1$ if for solving problem task i of class j, the k_j-th version of module j ($k_j = 1, \ldots, K_j$) is used, and $X_{ij}^{kj} = 0$ – otherwise.

The vector $X = (X_{ij}^{kj})$, $i = 1, \ldots, I$; $j = 1, \ldots, J$; $k_j = 1, \ldots, K_j$ is called the system configuration vector [15], representing the potential assignment of module versions to tasks of the system.

Let us introduce the following notations for a reliability model: R_{kj} – estimated reliability of version k_j of module j: $R_{kj} = 1 - P_{kj}$, where P_{kj} is the corresponding fault distribution function; R_j – estimated reliability of the fault-tolerant

implementation of module j as a system of K_j independent versions; R – estimated reliability of the entire software.

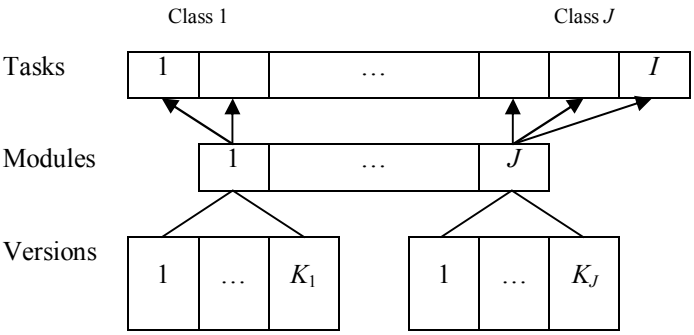

Fig. 1. Structure of N-version control software for communications satellite system of a figure caption.

The problem of maximizing reliability of N-version control software for communications satellite system by choosing an optimal set of modules can be formulated as follows:

$$\max R = \prod_{j=1}^{J} R_j ,$$

where

$$R_j = 1 - \prod_{k_j=1}^{K_j} (1 - R_{kj})^{X_{ij}^{kj}}$$

subject to

$$\sum_{i_j \in A_j} X_{ij}^{kj} \geq n_j, j = 1,...,J .$$

The set of cost constraints which guarantees that total expenditures will not exceed an upper bound B are given as follows:

$$\sum_{j} \sum_{i_j \in A_j} \sum_{k_j=1}^{K_j} [C_{kj}^1 + C_{kj}^2] X_{ij}^{kj} \leq B ,$$

where C_{kj}^1 and C_{kj}^2 are the cost of producing and running the k-th version of program module j.

The problem of optimization under resource requirement constraints needs a more complicated formulation, because the model description of the application process has to be taken into account. The application of the N-version control software for communications satellite system could be solved in one of the task sequences. To describe the progress of execution of the application process, let us introduce the so-called *evolution time vector* (ETV) which is given by:

$$t = \{t_1,...,t_i,...,t_I\},$$

where t_i is the time instant when a component of the software system is activated to execute the i-th (i = 1, …, I) task. The components of the ETV apparently satisfy the condition

$$t_i^l \leq t_i \leq t_i^u, \forall i = 1,...,I,$$

where t_i^l is the earliest time instant when the execution of task i can be started, and t_i^u is the latest time instant when the execution of task i might start, both of them depend on the execution time of preceding tasks. Obviously, the timing criterion can be improved by the variation of t_i in the range between t_i^l and t_i^u.

Let us introduce a second vector comprising I components, the *realization vector* (RV). The component h_{ij}^{kj} of the RV represents the time needed for the execution of task i by means of the k_j-th version of program module j.

The program modules are assigned according to the following time constraints rules:

1. Realizability of the application process.

2. Correction of ETV components as a reaction to violations of the realizability conditions.

In addition to cost and time constraints, choosing an optimal structure of fault-tolerant N-version control software for communications satellite system is also characterized by resource requirements. In the general case it is assumed that a task needs all its resources throughout its execution. We consider two kinds of resources: so-called *active resources* as well as *passive* ones: A resource is active, by definition [7], if it has processing power (for example, a CPU and I/O devices); otherwise, it is passive (for example, files and data structures). A task will request at least one active resource and zero or more passive resources. Thus, a passive resource must always be used with some active resource. Some resources can be (simultaneously) shared by multiple tasks while others, such as read or write channels, have to be assigned exclusively to one task.

Let us now consider the restrictions caused by the needed number of the read or write channels, respectively, for intermediate results storage. These numbers are defined by the schedule of tasks and ETV / RV components.

These restrictions for the needed number Ψ^R / Ψ^W of read or write channels, respectively, for intermediate results storage have the following form:

$$\Psi^R \leq \Psi_{\max}^R, \Psi^W \leq \Psi_{\max}^W , \qquad (1)$$

where

$$\Psi^R = \max_{\mu} d(t,\mu) ,$$

$$t_i + h_{ij}^{kj} = \mu ,$$

$$t_m - t_i > h_{ij}^{kj}, (\forall i < m; i,m \in I) .$$

Here, the time μ is the end time instant of the execution of task i by version k_j of module j; $d(t, \mu)$ is the number of ETV components t which are equal to the time μ. These components represent a subset of tasks which during the time μ all might need read or write channels.

Considering the maximum of $d(t, \mu)$ in (1) just states that we have to observe that the maximum number of tasks potentially needing read channels at the same time does not exceed the number of available ones.

Analogously we can formulate the restrictions with regard to the write channels.

In addition for our case we will have the following constraints to the maximal ETV time duration:

$$\max_i(t_i + h_{ij}^{kj}) - \min_i t_i \le T\ ,\ i = 1,...,I. \qquad (2)$$

This relation simply states that for all tasks i, the completion of the task does not exceed the required upper bound T for the execution time of the entire control process.

Based on the discussion above, one of the possible algorithms of module assignment can be formulated as follows:

Step 1. Let us assign the k_j-th program module for solving task i=1 of class j.

Step 2. Test of realizability conditions the control process (2):

- if the conditions are fulfilled, then go to step 4;

- if the conditions are not fulfilled, then go to step 3.

Step 3. Correction of the ETV component values:

- if the conditions are fulfilled, then go on;

- if the conditions are not fulfilled, then go to step 1.

Step 4. Test of the restrictions for the number of the read or write channels according to (1):

- if restrictions fulfilled, then go to step 5;

- if not fulfilled, then go to 1.

Step 5. Increment i: ($i = i + 1$) and let us assign k_j-th program module for executing task i of class j:

- if i+1 ≤ I, then go to step 2;

- if i+1 > I, then go on to step 6.

Step 6. Calculation of the application process duration under the derived module assignment.

III. Results and Discussion

In the following example, we briefly summarize N-version design of fault-tolerant control software for communications satellite system. This N-version software comprises control functions of communications satellite system. The information was collected on timing and resource requirement constraints of tasks, as well as data on cost and reliability of used program modules of the control software. The entire system comprised 64 modules, 9 of which were identified to correspond to a safety-critical task class.

Table 1 presents the values of the cost and reliability of safety-critical modules of the N-version control software for communications satellite system for the optimal solution computed by means of the optimization model and algorithm proposed in this paper.

TABLE I. Cost and Reliability of N-version Modules

Module	Cost		Reliability	
	Basic software	*N-version software*	*Basic software*	*N-version software*
4	14	22	0,9689	0,99842 (+2,95%)
5	15	29	0,9869	0,99998 (+1,31%)
16	15	29	0,9963	0,99997 (+3,67%)
17	8	14	0,9227	0,99394 (+7,12%)
18	23	51	0,9853	0,99999 (+1,47%)
19	16	23	0,9867	0,99956 (+1,29%)
48	39	72	0,9692	0,99851 (+ 2,93%)
51	62	170	0,9862	0,99998 (+1,38%)
52	66	131	0,9964	0,99995 (+ 3,55%)

The term “basic software” corresponds to the program modules of software without redundancy; for them, always that version among the candidates was considered which had the largest reliability while at the same time fulfilling all required side conditions. Thus, that the set of cost constraints for N-version software is met.

Thus, presented results show how control software for communications satellite system can be designed in order to increase its reliability. The module parameters depend on the model description parameters for the control process and the number of restrictions. Several cases of different spacecraft systems have been considered in [16] to show how the number of safety-critical components and parameters of other control processes influences the reliability of control software.

IV. Conclusion

The actual problem has been solved keeping up with basic tendencies in the evolution of the engineering software design theory and practice for the up-to-date communications satellite system. Important issue in technological control cycles design is the N-version programming software development for satellites. In the work, the problem of optimal N-version software structure was considered. The developed optimization model allows constructing fault-tolerant N-version software under certain constraints. These constraints are defined by technological control cycles of communications satellite system. The solution of the optimization problem of fault-tolerant N-version software for communications satellite control system can be achieved using the algorithm proposed in the article.

The results of the N-version software design for real communications satellite control system confirms the efficiency of N-version programming application. The results show improvement in reliability with the given constraints. The implementation of proposed approach gives an opportunity to solve the problems which cannot be solved using traditional methods or such solution is not effective.

Acknowledgment

The authors would like to thank the organizers of the International Siberian Conference on Control and Communications (Sibcon-2015) for the opportunity to take part at the conference. The authors gratefully acknowledge financial support from the Mikhail Prokhorov Foundation.

References

[1] I. Kovalev, "Software engineering of spacecraft control technological cycles," In Proc. of AMSE International Conference on Modelling, Measurement & Control, B, vol. 56(3), pp. 45-49, 1994.

[2] C.A. Lewis, R.W. Smith, and A. Beaulieu, "A model driven framework for N-version programming," In Proc. of IEEE International Systems Conference, SysCon 2011, pp. 59-65, April 2011.

[3] J.-H. Lü, S.-L. Ma, X.-J. Li, and S.-W. Gao, "Formal semantics model for automatic test of safety critical systems," Ruan Jian Xue Bao/Journal of Software, vol. 25(3), pp. 489-505, 2014.

[4] S. Lee, X. Bao, and T. Zhao, "A Safety-critical software development strategy based on theory of diverse design," In Proc. of 9th International Conference on Reliability, Maintainability and Safety, ICRMS'2011, pp. 694-699, June 2011.

[5] I.V. Kovalev, N.N. Dgioeva, and M.Ju. Slobodin, "The mathematical system model for the problem of multi-version software design," In Proc. of AMSE International Conference on Modelling and Simulation, MS'2004, pp. 7-12, 2004.

[6] S. Singh, "Software quality by design in aerospace systems," In Proc. of International Conference on Software Engineering: Software Quality: The Road Ahead, CONSEG 2011, pp. 174-178, February 2011.

[7] I.V. Kovalev, and K.-E. Grosspietsch, "Deriving the optimal structure of N-version software under resource requirements and cost/timing constraints," In Proc. of the EUROMICRO, vol. 2, pp. 200-207, 2000.

[8] D. Simon, C. Hourtolle, H. Biondi, J. Bernelas, P. Duverneuil, S. Gallet, P. Vielcanet, S. De Viguerie, F. Gsell, and J.N. Chelotti, "A software fault tolerance experiment for space applications," In Digest of Papers - FTCS (Fault-Tolerant Computing Symposium), pp. 28-35, June 1990.

[9] A. Avizienis, M.R. Lyu, and W. Schutz, "In search of effective diversity: A six-language study of fault-tolerant flight control software," In Digest of Papers - FTCS (Fault-Tolerant Computing Symposium), pp. 15-22, 1988.

[10] A. Avizienis, and L. Chen, "On the implementation of N-version programming for software fault-tolerance during program execution," In Proc. IEEE Comput Soc Int Comput Software & Appl Conf, COMPSAC '77, pp. 149-155, November 1977.

[11] M. Xie, C. Xiong, and S.-H. Ng, "A study of N-version programming and its impact on software availability," International Journal of Systems Science, vol. 5(10), pp. 2145-2157, 2014.

[12] I.M. Golubev, R.Ju. Tsarev, and T.I. Semenko, "N-version software systems design," In Proc. of the 11th International Scientific and Practical Conference of Students, Postgraduates and Young Scientists; "Modern Techniques and Technologies", MTT 2005, pp. 147-149, March 2005.

[13] G. Latif-Shabgahi, J.M. Bass, and S. Bennett, "A taxonomy for software voting algorithms used in safety-critical systems," IEEE Transactions on Reliability, vol. 53(3), pp. 319-328, 2004.

[14] A. Karimi, F. Zarafshan, S.A.R. Al-Haddad, and A.R. Ramli, "A novel n-input voting algorithm for x-by-wire fault-tolerant systems," The Scientific World Journal, vol. 2014, 2014.

[15] I.V. Kovalev, "Optimization problems when realizing the spacecrafts control," Advances in Modeling and Analysis C, vol. 52(2), pp. 63-70, 1998.

[16] I.V. Kovalev, "System of multi-version development of spacecraft control software," Pro Iniversitate Verlag, Sinzheim, 2010.