Method for modeling of Spacecraft onboard apparatus and building of consistent control logic with limited onboard resources

A A Tyugashev¹, Yu M Sygurov²

¹Samara State Transport University, Svobody str. 2V, Samara, Russia, 443066
²Space Rocket Centre ‘Progress’, Zemetsa str. 18, Samara, Russia, 443009

e-mail: tau797@mail.ru

Abstract. The paper is devoted to the method for building consistent spacecraft onboard control logic in case of limited available resources. The consistent onboard control logic is extremely important for the success of a space mission. The special aspect of the complexity of control logic related to limited available onboard resources – electric energy, computer power, etc. The paper introduces the mathematical models for onboard resources, equipment and devices, and their working modes. Also, we present some prototypes of software tool allowing automated checking of control logic whether they violate the resource constraints as well as the design of new control logic from scratch.

1. Introduction
The modern spacecraft have various onboard systems (Motion Control System, Telemetry System, Onboard Control System, Power Supply System, Thermal Control System, etc.). In turn, each onboard system includes various devices, sensors, engines, etc [1-2].

All of these devices should cooperate in the space mission. To do this, spacecraft needs to accomplish the timed set of operations performed by onboard sensors, aggregates, etc. All operations should be coordinated and right synchronized with taking into account the parameters of spacecraft motion and speed of onboard physical processes. One can name this set of operations as an ‘active schedule’, but a special name ‘cyclogram’ or ‘cyclogramme’ is used in the aerospace industry. The operations to be executed consume onboard resources such as electric power, processor time and memory of the onboard computer, etc. Thus, the consistent control logic should consider the existing limits of the available resources. We can underline that the onboard flight control software performs ‘system forming’ role for the onboard equipment, i.e. integrates various onboard apparatus into a holistic entity. Of course, the complexity level of onboard control means should correspond to the complexity of the spacecraft apparatus. It is not a wonder that Onboard Control System is one of the most complex spacecraft systems. Moreover, today the spacecraft have several onboard computers integrated into the onboard network. These computers are being controlled by real-time operating systems [3-4]. The growing complexity of tasks executed onboard leads to an increase in control logic’s complexity. It is not a wonder that the complexity of flight onboard software has been increased dramatically during the last decades [5].
During the real mission, we face failures of the onboard equipment and fails of software running on the main onboard computer or as embedded software of the particular onboard system. Other problems could be inspired by the violation of the limits of available onboard resources. Some of the abnormal situations could not be predicted at the design phase, so these situations are not described in operation manuals and it is very hard to parry these situations. Fortunately, the special redundancy of software can help to deal with these problems [5-6]. But it is much better to exclude dangerous situations at the design stage by considering the limits of available onboard resources [8-9].

The onboard devices have a set of working modes with different levels of consumption of onboard resources. The minimum modes associated with the device are: ‘ON’ and ‘OFF’. Some apparatus has a wide spectrum of working modes with varying combinations of needed resources. Depending on the operation mode and situation, the modes might be switched slowly or quickly. Usually, each onboard resource has a maximal available level, i.e. ‘red line’. For example, the spacecraft is equipped with solar panels providing a restricted level of electric power. The consistent control logic should execute all required tasks without violation of these ‘red lines’ during the whole flight.

Switching from mode to mode is implemented by ‘onboard apparatus control commands’, i.e. codes issued through an onboard control interface bus. For instance, we can use the command «SWITCH ON Device1 in System1» or «CHANGE MODE OF GyroDyn1 to Mode2». So, we need the approach allowing the design of consistent control logic with the warranty of absence of violations of available resource levels. These measures can help to improve the probability of a space mission’s success.

Usually, the complex control logic involving devices belong to various onboard systems is implemented by a special sort of software named as real-time control algorithms [10]. A certain control algorithm should be implemented by the particular program module. These modules are running as processes of the multi-program real-time onboard operating system. In other words, we have a parallel execution of many control algorithms onboard. As a rule, the control program can include many kinds of operators, but we especially interested in followings: 1) call of other program module implementing another control algorithm; 2) issue of an apparatus control command to a particular device; 3) setting or resetting of some logical value (flag). The values of these flags have an impact on the execution of onboard actions. For example, some flag can means «level of charge of battery 1 is extremely low», and other means «second solar panel is fully operational». The full set of logical values forms the so-named ‘spacecraft state vector’, or ‘logical vector’. Some subsets of this vector could be reviewed as ‘particular state vector’, for example, state vector of a certain onboard system [11]. If we fix the values of conditions, we get the concrete scenario of execution of the control algorithm, i.e. branch of control logic.

Various scenarios of execution lead to the execution of various timed sequences of operations. In other words, we should form a particular cyclogram for each possible scenario.

The additional level complexity of control logic in case of limited available resources is caused by the following circumstances. As it was mentioned above, the several control algorithms run in a parallel manner, and each of them executes different sequences of actions. So, we must consider the chain (in fact, tree) of calls CA1→CA2→CA3...

In fact, one control algorithm can activate dozens of other algorithms with corresponding cyclograms of actions. It is a very difficult problem for a designer to consider all branches of ‘execution tree’ with full checking of the absence of violation of available resources’ limits.

The main goal of this work is to construct the adequate mathematical models for all named entities because we didn’t found such models for the problem domain of control of automatic spacecraft, in literature; these models must be appropriate to be used in practice at the Samara Space Center. On this base, further, we formulate the method of how to automate the evaluation of the feasibility of implementation of wanted control logic by the existing set of onboard devices and control algorithms.

The second section of the paper is devoted to the formal problem statement, analysis of input and output data of each stage of onboard control logic design, and to the discussion about ways of development of special decision making automation tools with the utilization of opportunities provided by constraint programming and SMT solvers.
2. Method and Discussion
In this section, we will provide the mathematical formalisms for modeling of spacecraft onboard equipment and flight control program modules, and discuss the ways of implementation of automation tools.

The spacecraft onboard situation can be represented by the following sets:

- \( EA = \{ BA_i \}, \ i=1...N \) is a set of onboard devices;
- \( AP = \{ AP_j \}, \ j=1...M \) is a set of onboard resources;
- \( Lim(AP) = \{LimAP_i, LimAP_2, ..., LimAP_M\} \) is a set of limits of available resources;
- \( REA(BA_i) = \{RM_{i1}, RM_{i2}, ..., RM_{iM}\} \) is a set of modes of onboard devices;
- \( CL(RM_{ij}) = \{CP_{ij}, CP_{i2}, ..., CP_{iN}\} \) is a vector of levels of consumption of each kind of resources in corresponding mode;

The starting point in the design process of decision making is a set of tasks to be executed by onboard equipment. The important issue is that we must implement action at a specified time moment corresponding to speeds of onboard physical processes. We use the following set:

\[ Z = \{PT_i,t_i\} \] - set of tasks to be executed by spacecraft as it presented in Fig. 1.

The onboard complex of control algorithms can be represented as

\[ UA = \{ CA_m \}, \ m=1...U. \]

Then the designer must accomplish the following steps:

1. Determine the correspondence:

\[ f_1: Z \rightarrow EA \]

i.e., define which devices to be involved in the process of execution of each task, and which modes of functioning of this device should be used.

2. Determine the correspondence:

\[ f_2: EA \rightarrow UA \]

i.e. the mapping between elements of onboard equipment and program modules control them.

3. Set time parameters:

\[ f_3: EA \rightarrow T \text{ and } UA \rightarrow T \]

Using these mappings we can build the cyclogram of the functioning of various onboard devices and calculate the total level of consumption of particular onboard resource as a function of time.

The last mapping allows us to form the following set of tuples:

\[ \{ BA_i, CA_{ij}, t_i, l_i \} \]

Each tuple defines the scenario of execution of \( CA_i \), starting at moment \( t_i \) in the situation described by logical vector \( l_i \).

The logical (state) vector can be represented, for example, as

\[ l=(a_i=TRUE, \ a_2=FALSE, ..., a_q=H), \]

where \( a_i \) is a particular condition (flag). Some conditions could be insignificant for this scenario of execution, in this case, we use additional ‘H’ value of truth, in this aspect our approach can be considered as a variant of three-values logic.

The set of these tuples represents the sequence of ‘sections’ of space mission and to determine the different scenarios of the implementation of control logic. On the other hand, one can apply this sequence to check of overlays of devices’ modes and functioning of particular software modules. After that, we link the execution of program modules and work of onboard equipment. In other words, we form the schedule of onboard equipment’s functioning and the corresponding schedule of consumption of onboard resources.

The representation of cyclogram in the screen form of a special software tool can be found in Figure 1. One can recognize the time scale at the top part of the window devoted to show of concrete cyclogram, the horizontal bars represent the tasks (with the specified duration) to be executed by onboard apparatus, and different colors mention different conditions caused execution of this task.

We must mention the existence of an alternate approach described in publications [10-14]. A similar but different model is based on the following tuples:

\[ CAM=\{ \langle f_i, t_i, \tau_i, l_i \rangle, f_i = Call(CA_{ik}) \ V KU(BA_i) \ V Set(a_i) \} \]

where \( f_i \) is an action to be executed at time moment \( t_i \) with duration \( \tau_i \) and associated with logical vector \( l_i \). In this model, \( f_i \) might be a call of other control algorithms, issue of the apparatus control command, or setting or clearing of logical value.
2.1. Input and Output Data at Stages of design of decision making
To provide ‘smart’ decision-making design support tool we need to analyze design process stages from the input and output data point of view.

The process of design of ‘smart’ spacecraft operations implemented by onboard devices and software modules must harmonize the cooperative work of all units in physical and logical senses [14]. In a physical sense, we need consider the maximum available levels of onboard resources. At the first stage, the set of cyclograms of onboard apparatus and units work should be built for various possible situations reflected by spacecraft state control vector. The important issue is that we need to design the control logic both for control of the particular onboard device (control ‘in small’) and for control of spacecraft as a whole (control ‘in big’). These procedures should be implemented by control algorithms for the complex operation of spacecraft [10] at the next stage. We should consider the following information:

- Materials on the control logic of systems and units during the execution of particular tasks
- Requirements to time characteristics of execution of the tasks, including restrictions of simultaneous execution of various functional tasks due to the physical sense of the tasks
- Requirements are related to the sequence of execution of different sections of the control process.

After that we get the following results which are the input data for the next stages:

- Input data for the design of control algorithms for complex spacecraft operations
- Time diagrams showing the operation of systems and units, indicating the modes of operation of the equipment and algorithms for different scenarios of control
- Estimation of required levels of onboard resources consumed by spacecraft equipment during operations
- Materials on a mutual overlay of execution of control algorithms and modes of onboard device functioning.

2.2. Mathematical models for input and output data of design stages
To build the formal mathematical models for input and output data of mentioned design stages [15] we need consider the structure of the model which allows describing the named in previous sections entities related to various onboard systems, devices, and actuators with their work modes. On the other hand, the model should reflect the set of control algorithms divided into the ‘sections’, and adequately
reflect various scenarios of execution. The model of control algorithms for complex operations can be reviewed as a ‘base’ model at this stage.

Correspondingly, we propose the following additional sets:

Alts = \{ CA^0 j \} is a set of branches of control algorithm where i is ID of the algorithm, and j is several of possible scenarios (branch of execution). As a rule, each control algorithm has several (K_j) scenarios of execution depending on the values of certain flags. In these terms, CA^0 j is a control algorithm with ID i implementing scenario j.

\[ \Omega = \{ \omega_d \} \) is a ordered in time set of sections of spacecraft mission.

Using this approach we can represent the particular control algorithm by the following structure:

\[ \text{CA}^0 = \{ T^0_{\text{work}}, A^0_{\text{on}}, A^0_{\text{off}} \} \]

where T^0_{\text{work}} = \{ (t^0_w^1, l^0_w^1), (t^0_w^2, l^0_w^2), ... (t^0_w^K, l^0_w^K) \}.

\[ t^0_w^j \] is a start time of execution of the control algorithm scenario in case of validity of logical condition l^0_w^j. The number of pairs (t^0_w^j, l^0_w^j) defines the number of calls of scenario j of control algorithms depending on l^0_w. The full set of logical conditions t^0_w defines the ‘state vector’ in the sense of [6, 7].

A^0_{\text{on}} is a set of algorithms called by scenario j of CA^0 j.

\[ A^0_{\text{on}} = \{ (CA^0_{1ij}, t^0_{wij}, l^0_{wij}), (CA^0_{2ij}, t^0_{wij}, l^0_{wij}), ... (CA^0_{Kij}, t^0_{wij}, l^0_{wij}) \} \]

where presence of (CA^0_{ijK}, t^0_{wij}, l^0_{wij}) means that this control algorithm calls algorithm CA^0_{ij} at time t^0_{wij} if l^0_{wij} is true.

Similarly, A^0_{\text{off}} is a set of algorithms to be shut down by scenario j of CA^0 j.

\[ A^0_{\text{off}} = \{ (CA^0_{1ij}, t^0_{wij}, l^0_{wij}), (CA^0_{2ij}, t^0_{wij}, l^0_{wij}), ... (CA^0_{Kij}, t^0_{wij}, l^0_{wij}) \} \]

where tuple (CA^0_{ij}, t^0_{wij}, l^0_{wij}) means that our control algorithm shuts down algorithm CA^0_{ij} at time t^0_{wij} if l^0_{wij} is true.

Further, BA^0_j is onboard apparatus controlled by the control algorithm:

\[ \text{BA}^0_j = \{ (\text{Nam}_j, R^0_j, P^0_j) \} \]

where Nam^0_j is an ID of the particular onboard device, R^0_j is a mode of functioning of this device, and P^0_j is a vector which components are the levels of consumption of corresponding onboard resources in mode R^0_j.

KU^0_j is a set of commands issued by the control algorithm to onboard devices.

\[ \text{KU}^0_j = \{ (NK^0_{ij}, t^0_{wij}, l^0_{wij}), (NK^0_{ij}, t^0_{wij}, l^0_{wij}), ... (NK^0_{ij}, t^0_{wij}, l^0_{wij}) \} \]

where NK^0_j is a name (ID) of control command, t^0_{wij} is a time moment of issuing of command in case of truth of l^0_{wij}.

FP^0_j = \{ (PF^0_{ij}, t^0_{sij}, l^0_{sij}), (PF^0_{ij}, t^0_{sij}, l^0_{sij}), ... (PF^0_{ij}, t^0_{sij}, l^0_{sij}) \} is a set of flags (logical values) formed by scenario j of algorithm CA^0 j execution at time moments t^0_{sij} in case of truth of l^0_{sij}. These models contain enough information required to determine the sequence of sections of spacecraft control and to form schedules reflecting its functioning in different scenarios, from various points of view. Using that, we can determine the overlays of algorithms and work of certain onboard devices. Some other points of view are shown in Fig. 2.

2.3. Application of Satisfiability Modulo Theories solvers and Constraint Programming

The very prospective modern methods are connected with the application of such approaches as constraint programming (CP), boolean satisfiability (SAT) solvers, mixed-integer programming (MIP), mixed integer non-linear programming (MINLP), local search (LS), and dynamic programming (DP). Many of the named approaches are supported by software tools based on advanced methods and algorithms [16]. In fact, these tools can be reviewed as ‘universal solvers’ accessible using API, special problem statement languages or even free and online. Some of the solvers propose efficient, highly parallel mechanisms to be applied in various domains [17].

Boolean satisfiability problem can be reviewed as a problem partially similar to the existence of a control algorithm with ‘smart’ decision making with a warranty of correct completion or the space mission in case of limited onboard resources. Moreover, probably the approach based on the use of SMT solvers is even more suitable. We can consider applying of ABSolver, Alt-Ergo, Barcelogic, MathSAT, CVC, OpenSMT, Simplify, STEP, Yices, Z3, and other available tools to be applied. In [18-19] we described the case study of utilization of opportunities provided by Z3 solver and described special software tool allowing checking of satisfiability of important synchronization
requirements to spacecraft control logic. Furthermore, by its nature, constraint programming also can be reviewed as an approach well corresponding to the problem of control in case of limited available resources because constraints are the sort of limits. This circumstance opens the door to apply the constraint programming in our problem domain.

![Figure 2. The screenshot of other software prototype tool.](image)

To do this we first need to represent the mathematical models described in previous sections by the means and input languages of CP tool. Then we can formulate the real-time control algorithm in these terms. After that, we should specify existing limits for onboard resources and check whether the limits being violated.

But in this way, we can face the following problems [16]. A modeling language should aim to convey problem structure to the solver as explicitly as possible. However, users may not always use the appropriate abstractions or may not recognize them in their model specifications based on input language. Thus, some solvers frequently use pre-processing steps which, among other actions, tries to infer some implicit structure from the model (e.g., recognizing knapsack cuts by combining constraints). An appropriate modeling system must reach a similar goal but for a much richer modeling language and a larger set of underlying solving technologies. A focus should be on automatically determining global substructure in the form of implied global constraints.

Adding these global constraints to the model (with or without removing the constraints that imply them) can dramatically improve solving abilities. More importantly, it may make life considerably easier for other model analyses and transformations listed in the paper. For instance, current methods for automatic derivation of search procedures strongly make use of global constraints [16-17]. A better understanding of the global substructure can also help to transform a model written for one solver technology for another. In this manner, the automatic derivation of implied constraints is an enabling analysis for many other analyses.

3. Conclusion and Future Work

The analysis performed for the design of ‘smart’ spacecraft onboard decision making has shown the opportunity of formal representation of data related to different phases of the design process. This formal representation has been specified as mathematical models which to be converted into electronic models used in special software design support tools. Now the authors work on the implementation of these tools. The very promising approach is connected with constraint programming as it was discussed in subsection 2.3.

The goal of this computer-aided design system is the support of spacecraft control algorithms specification, visualization, and checking, in case of restricted onboard resources. We plan to provide
designers with the tools allowing automatic calculation of estimated levels of consumption of onboard resources based on formal models of control algorithms. These estimations should be determined for all suggested time of the spacecraft mission. The very important feature is building and visualization of cyclograms showing the execution of tasks, calls of program modules, the functioning of certain equipment.

In addition to previous publications [8,10-15], we have presented a mathematical model for restricted onboard resources and state the problem from the feasibility of the target tasks of spacecraft in terms of resources consumed by onboard equipment in various working modes. In contrast to [12], in subsection 2.2 we proposed the prospective approach of how to reach the practical solution of the problem using Constraint Programming and Satisfiability Modulo Theories Solvers.

Supposed introduction of these tools into the spacecraft control logic design process at Space Rocket Centre ‘Progress’ will help to support decision making, allow to optimize the cyclograms of the functioning of the onboard apparatus, reduce labor costs, increase the quality of design documentation. These aspects are quite actual and prospective in modern conditions when we face the growing importance of information models and the complexity of software used in the space industry. We hope this future work will have a significant practical impact.

4. References
[1] Kozlov D I, Anshakov G P and Mostovoy Ya A 1998 Control of Earth Observation Satellites: Computer Technologies (Moscow: Mashinostroenie) (In Russian)
[2] Akhmetov R, Makarov V and Sologub A 2012 Principles of the Earth Observation Satellites Control in Contingencies Information and Control Systems 1 16-22
[3] Tomayko J 1994 Computers in Space: Journeys with NASA (Indianapolis: Alpha Books)
[4] Eickhoff J 2012 Onboard Computers, Onboard Software and Satellite Operations. An Introduction (Heidelberg: Springer)
[5] Tyugashev A A, Ermakov I E and Ilyin I V 2012 Ways to Get More Reliable and Safe Software in Aerospace Industry Proc. Program Semantics, Specification and Verification: Theory and Applications (PSS) 121-926
[6] Krasner S and Bernard D 1997 Integrating Autonomy Technologies into an Embedded Spacecraft System-Flight Software System Engineering for New Millennium Proc. IEEE Aerospace Conference (IEEE Press)
[7] Khartov V 2006 Autonomous Control of Communication and Navigation Spacecraft Aerospace Instrument-Making 6 12-23
[8] Tyugashev A A and Belozubov A V 2016 Toolset for construction and verification of rules for spacecraft's autonomous decision making Proc. 20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, Procedia computer science 96 811-818
[9] Kalentyev A A and Sygurov Yu M 2010 Development of information support for the spacecraft control algorithm design process Vestnik SGAU 21(1) 58-62
[10] Tyugashev A A 2006 Integrated environment for designing real-time control algorithms Journal of Computer and Systems Sciences Intl. 45(2) 287-300
[11] Tyugashev A A 2016 Language and Toolset for Visual Construction of Programs for Intelligent Autonomous Spacecraft Control IFAC – PapersOnLine 49(5) 120-125
[12] Kalentyev A A and Tyugashev A A 2006 CALS technology in lifecycle of complex control programs (Samara: Scientific Center of Russian Academy of Sciences)
[13] Tyugashev A A, Zheleznov D V and Nikishchenkov S A 2017 A technology and software toolset for design and verification of real-time control algorithms Russian Electrical Engineering 88(3) 154-158
[14] Filatov A V, Tkachenko I S, Tyugashev A A and Sopchenko E V 2015 Structure and algorithms of motion control system software of the small spacecraft Proc. Int. Conference Information Technology and Nanotechnology (ITNT) 246-251
[15] Bogatov A A and Tyugashev A A 2013 The logical calculus of control algorithms Proc. Int. Symposium for Reliability and Quality 1 307-308
[16] Garcia M, Stuckey P, Pascal V H and Wallace M 2014 The Future of Optimization Technology Constraints 19 126-138
[17] Marriott K, Nethercote N, Rafeh R, Stuckey P, Garcia M and Wallace M 2008 The design of the Zinc modelling language Constraints 12(3) 229-267
[18] Tyugashev A A 2018 Application of SMT solvers for evaluation of Real-Time control logic of spacecraft J. Physics: Conference Series 1096 012156
[19] Tyugashev A A 2017 Build and evaluation of real-time control algorithms in case of incomplete information about functional processes’ parameters Proc. XXth IEEE International Conference on Soft Computing and Measurements (SCM) 179-185