Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 1019–1026

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Intelligent Method of Computer-Aided Design and Checking of Real-Time Spacecraft Control Algorithms

Andrey Tyugashev[a]*, Sergei Orlov[b]

[a]*Samara State Transport University, 2V Svobody Str., Samara, 443066, Russia*
[b]*Samara State Technical University,244Bldg, Molodogvardeyskaya Str.,4443100, Russia*

Abstract

The paper is devoted to description of the method for computer aided design of consistent real-time spacecraft onboard control algorithms. This kind of control algorithms should organize synchronized cooperation of different onboard apparatus and functional software for successful completion of space exploration mission. We introduce the mathematical models for onboard resources, equipment and apparatus, and for real-time control algorithm to be implemented by onboard real-time software. The important feature of the proposed model is considering that each onboard apparatus consumes some onboard resources, thus consistent control algorithm must not violate limits of these resources. There is discussion devoted to construction of effective and consistent real-time control algorithm which guarantees completion of the mission in case of limited accessible onboard energy, computer power, etc. We also present the specially developed CAD/CASE tool allowing automated checking of existing algorithms whether they violate the resource constraints as well as design of new algorithms from scratch.

Keywords: Real-Time Control Algorithm, Onboard Resources, Computer-Aided Design; Consistent Control Logic;Spacecraft Onboard Control System;

* Corresponding author. Tel.: +7-917-1040493; fax: +7-846-262-30-76.
*E-mail address:*a.tyugashev@samgups.ru

1877-0509
10.1016/j.procs.2019.09.269

1. Introduction

The modern spacecrafts are equipped with wide spectrum of onboard systems (such as Motion Control System, Telemetry System, Onboard Control System, Power Supply System, Thermal Control System, etc.). The Onboard Systems, in turn, include various apparatus, devices, sensors, actuators, engines, etc [1-2].

All of named onboard apparatus should work in cooperative manner to complete the space mission. To do this, we need to accomplish the timed set of operations involving various onboard equipment. All operations must be logically coordinated and correctly synchronized with taking in account the parameters of spacecraft motion and speed of onboard physical processes. This set of operations can be reviewed as a special form of 'active schedule' and has the name 'cyclogram' in Aerospace Industry. Each operation to be executed consumes some kind of onboard resources such as electric power, and the 'consistent' control algorithms must consider the onboard limits of available resources. Furthermore, we know that in accordance with Ashby's Law of Requisite Variety [3], the variety and complexity of control system should reflect the complexity of controlled object, in this case complexity level of onboard control means should corresponds to complexity of spacecraft. Thus, it is not a surprise that the Onboard Control System is one of the most complex onboard systems. Nowadays, it usually based on set of onboard computers forming the onboard computer network. These computers use the Onboard Real-Time Operating System and onboard flight control software [4-5].

The flight control software implements the real-time control algorithms which, in turn, represent the required cyclogram.

One can note that at pre-mission tests and space operations we can face with apparatus failures and with fails of flight control software running on main onboard computer or as embedded software of particular onboard system. Frequently, such situations are not expected at design phase and not predicted in operation manuals. Consequently, it is very hard to parry these situations at operational phase. Fortunately, in some cases, special redundancy of software can compensate failures [6-7]. For example, some fails during the space mission could be inspired by violation of the limits for available onboard resources.

The growing complexity of tasks executed onboard lead to increase of number of onboard system, and correspondingly to increase of control logic's complexity. In fact, the complexity of flight onboard software has been increased dramatically during last decades [6].

The limits of available onboard resources are very important issue to be considered at design phase of control logic if we want to avoid many emergency situations. Each onboard device consumes onboard resources, for example, electric energy. We can also consider other limited onboard resources such as free computer memory, central processor's time slots, etc [8-9].

Particular onboard device has a set of functioning modes connected with various levels of consuming of onboard resources. The minimum number of modes associated with the device is two (switched on and off). Some kind of equipment has a wide spectrum of modes with varying combinations of needed resources. Over time and depending on current situation, the modes of functioning could be switched slowly or quickly. But each kind of onboard resource has a 'red line' (maximal available level). For instance, spacecraft can be equipped with solar panels providing certain level of electric power. The control logic must not violate these red lines during whole flight.

Switching from mode to mode frequently implemented by so-named "apparatus control commands" (coded sequences of electric impulses) issued by real-time control software. Control command could be named as "SWITCH ON Device1 in System1" or "CHANGE MODE OF GyroDyne1 to Mode2". So, we need the approach allowing design of robust control logic with warranty of absence of violations of available resource levels and improved probability of success of spacecraft mission.

Usually, the control logic of unmanned spacecrafts is implemented by special sort of software - control algorithms [11]. The particular control algorithm should be implemented by particular module. The very important issue is that program modules are running under control of multi-thread real-time onboard computer's operating system, so we deal with parallel execution of many control algorithms. As a rule, the module contains many kinds of operators, but we are especially interested in following actions: 1) call of other software module implementing other control algorithm. 2) dispatching of control command to particular device; 3) setting or resetting of some flag (logical condition). The value of the flag is quite important for execution of control algorithms. For example, the condition can mean: "second solar panel is fully operational" or "level of charge of battery is low". The set of all

conditions forms the so-named 'spacecraft state vector'. If we fix the certain values of the flag, we should get the concrete scenario of execution of the control algorithm or branches of control logic. Each action in control algorithm is associated with certain moment of time.

Of course, various scenarios of control algorithm lead to execution of various sequences of actions with corresponding time stamps depending on validity of logical conditions (flags). So, we can speak about particular cyclograms for each scenario. One can form the logical 'state' vector or vector consisting of these 'flags' representing the current state of spacecraft.

The additional complexity of control algorithm's design in case of limited resources is caused by the following circumstance. As it was mentioned above, the several control algorithms can be running in parallel executing different sequences of actions. And we must consider the chain (in fact, tree) of calls CA1→CA2→CA3… One control algorithm can activate dozens of other control algorithms. It is a quite difficult problem for a human to consider all branches 'execution tree' with the checking of absence of violation of resource limits.

We propose a computer-aided design approach to resolve of this problem.

The second section of the paper is devoted to formal problem statement, analysis of control algorithm's design stages input and output information, and discussion about ways of creation of computer aided design support tool with utilization of opportunities provided by modern constraint programming and Satisfiability Modulo Theories approaches.

2. Problem statement and discussion

In this section we will provide some mathematical formalism can be utilized for modeling of spacecraft onboard apparatus and control algorithms and then discuss the ways of implementation of computer aided design tool.

The onboard reality specified above 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;

$Lim\ (AP_j) = \{\ (LimAP_1, LimAP_2, ... LimAP_M)\ \}$ is a set of limits of each kind of resource;

$REA\ (BA_i) = \{\ (\ RM_{i1}, RM_{i2} ... RM_{iN}\ )\ \}$ is a set of modes of functioning corresponding to onboard devices and systems;

$CL\ (RM_{ik}) = \{\ (\ CP_{i1}, CP_{i2} ... CP_{iN}\ )\ \}$ is a vector of levels of consumption of each kind of resource in certain device's mode;

The starting point in control algorithm design process is a set of tasks to be executed by onboard equipment. The important issue is that we must implement action at exact time moment corresponding to moving of spacecraft or speed of other physical process. We must consider:

$Z = \{\ PT_j, t_j\ \}$ - is a set of tasks to be executed by spacecraft's equipment.

Finally, the onboard complex of control algorithms can be represented as

$UA = \{\ CA_m\ \}$, $m=1...U$.

Then we need accomplish the following steps:

1. Determine mapping

$$f_1: Z \rightarrow EA \tag{1}$$

i.e., define which device is used in processes of certain task's execution, and which modes of functioning of this device we need.

2. Determine mapping:

$$f_2: EA \rightarrow UA \tag{2}$$

i.e. mapping between elements of onboard apparatus and algorithms which control them.

3. Set time parameters:

$$f_3: EA \rightarrow T \text{ and } UA \rightarrow T \tag{3}$$

Based on this, we can build the cyclogram of functioning of various onboard devices and then calculate the level of consumption of all kinds of resources as the function of time.

The last mapping allows specifying the following tuples:

$$\{\ BA_i\ ,\ CA_{ij}\ ,\ t_i,\ l_i\ \} \tag{4}$$

For particular onboard device, each tuple defines the control algorithm CA_i and scenario of its execution (branch) j starting at time moment t_i in situation described by logical condition vector l_i.

The logical vector can be represented, for example, as

$$l=(\alpha_1=TRUE,\ \alpha_2=FALSE,\ldots\ \alpha_q=H)$$

where α_i is a particular condition. Some conditions could be insignificant for particular scenario of execution; in this case we use third '*H*' value of truth, in this aspect our approach can be considered as variant of three values logic.

The set of these tuples allows building the ordered sequence of 'sections' of spacecraft flight and to determine the different scenarios of execution of control algorithm. On the other hand, this sequence can be used for checking of overlays of various devices' modes and functioning of software modules. After that, linking the execution of control algorithms with functioning of onboard apparatus (onboard systems and particular devices) we can get the time diagram (cyclogram) of onboard equipment's work and the consumption schedules for particular onboard resource.

The representation of cyclogram by the special tool can be found in Figure 1.

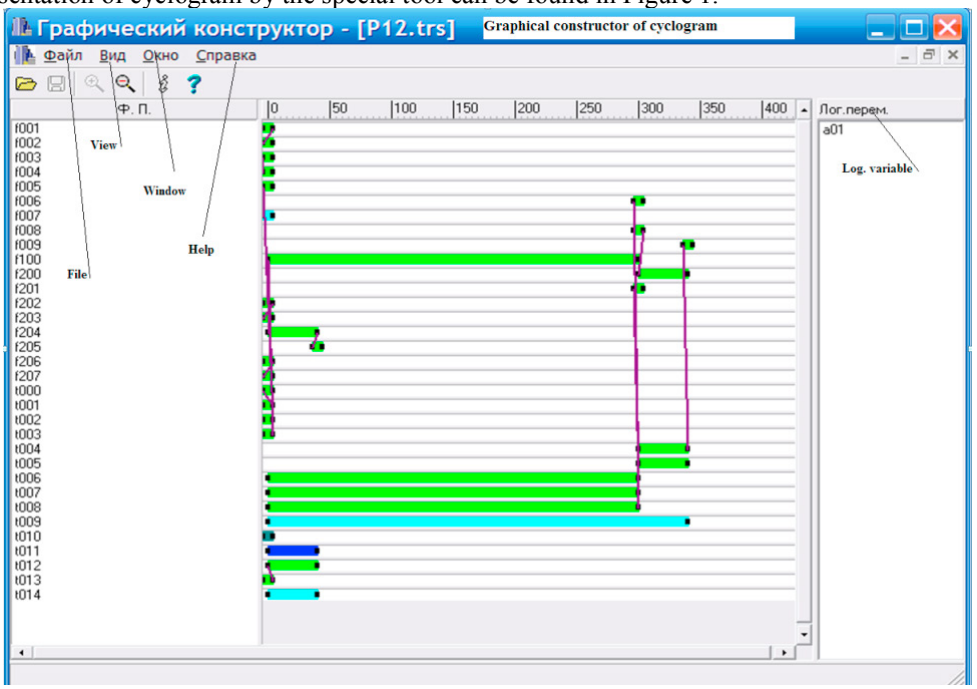

Fig. 1. Cyclogram of onboard operations at screenshot of special software tool.

Alternate approach which can be applied [11-14] is based on following tuples:

$$CA_m=\{<f_i,\ t_i,\ \tau_i,\ l_i>\},f_i = Call\ (CA_k)\ V\ KU(BA_i)\ V\ Set(\alpha_l), \quad (5)$$

where f_i is an action to be executed at time moment t_i with duration τ_i and associated with logical vector l_i. Particular f_i might be call of other control algorithms, issue of particular control command to onboard device, or setting or clearing of some 'flag', i.e. element of spacecraft's state vector.

2.1. Stages of design of control algorithm

To provide some control algorithm design automation means we need to accomplish step-by-step analysis of design stages and the information used at each stage.

The process of design of control logic supported and implemented by onboard apparatus and flight control software must allow harmonizing functioning of different onboard systems and units in physical and logical senses [14]. In physical sense, for example, we need consider the maximum available levels of onboard resources. At this

stage, the base cyclograms of onboard apparatus and units work should be built. Correspondingly, we need to build the certain logical procedures both for control of 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 complex operation of spacecraft [11, 14] at the next stage. At this stage, the following information must be considered:

- Materials on the control logic of systems and units during execution of functional tasks
- Requirements to time of functional tasks' execution including requirements on possibility or restriction of simultaneous execution (mutual overlay) of various functional tasks
- Requirements related to the sequence of execution of different sections of control algorithm.

As a result, we will get the following stuff:

- Input data for development of control algorithms for complex operation
- Timing diagrams showing the operation of systems and units, indicating the modes of operation of the equipment and algorithms for different scenarios of spacecraft mission realization
- Estimation of required energy and other kind of onboard resources consumed by spacecraft equipment during operations
- Materials on mutual overlay of algorithms and modes of onboard device functioning.

2.2. Mathematical models for input and output data of design stages

To build the formal representation of models for input and output data of different stages of design of spacecraft's control logic [16] we need first consider the structure of the model which will allow to describe the named entities related to different kinds of onboard systems, devices and aggregates with the specifying of work modes. On the other hand, the model should describe the set of control algorithms divided into the 'sections' and with the different scenarios of execution (branches). The 'core' part of onboard flight control software is so-names 'control algorithms for complex operations'. The model of such kind of algorithms can be reviewed as a 'base' model.

We propose the use of the following additional sets:

$Alts = \{ CA^{ij} \}$ is a set of alternatives of control algorithm where the i is ID of the algorithm, and j is a number of possible scenario (branch or path of execution). As a rule, each control algorithm has several (K_i) scenarios of execution depending on truth of certain logical conditions. So, CA^{ij} is a control algorithm with ID i implementing scenario j.

$\Omega = \{ \omega_d , < \}$ is a ordered (in time) set of sections of spacecraft operation.

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

$$CA^{ij} = \{ T^{j}_{work}, A^{ij}_{on}. A^{ij}_{off}, BA_{ij}, KU_{ij}, FP_{ij} \}, i=1..N, j=1..K. \quad (6)$$

where $T^{j}_{work} = \{ (t_w{}^{ij}{}_1 , l_w{}^{ij}{}_1) , (t_w{}^{ij}{}_2, l_w{}^{ij}{}_2) , \dots (t_w{}^{ij}{}_{Ki}, l_w{}^{ij}{}_{Ki}) \}$.

$t_w{}^{ij}{}_1$ is a start time of execution of control algorithm scenario in case of truth of logical condition $l_w{}^{ij}$. The number of pairs ($t_w{}^{ij}{}_{Ki} , l_w{}^{ij}{}_{Ki}$) defines the number of calls of scenario j of control algorithms depending on l_{wi}. The full set of logical conditions $l_w{}^{ij}$ defines the 'state vector' [6, 7].

A^{ij}_{on} is a set of other algorithms called by scenario j of CA^{ij}.

$A^{ij}_{on} = \{(CA^{ij}{}_1, t_w{}^{ij}{}_1, l_w{}^{ij}{}_1), (CA^{ij}{}_2. t_w{}^{ij}{}_2, l_w{}^{ij}{}_2), \dots (CA^{ij}{}_K, t_w{}^{ij}{}_K, l_w{}^{ij}{}_K) \}$.

where tuple (CA_{ijK}, t_{wijK}, l_{wijK}) means that our control algorithm calls algorithm $CA^{ij}{}_K$ at time $t_w{}^{ij}$ if $l_w{}^{ij}{}_K$ is true.

Similarly, A^{ij}_{off} is a set of other algorithms which are to be shut down by scenario j of CA^{ij}.

$A^{ij}_{off} = \{(CA^{ij}{}_1, t_w{}^{ij}{}_1, l_w{}^{ij}{}_1), (CA^{ij}{}_2. t_w{}^{ij}{}_2, l_{wij2}), \dots (CA^{ij}{}_K, t_w{}^{ij}{}_K, l_w{}^{ij}{}_K) \}$.

where tuple ($CA^{ij}{}_K , t_w{}^{ij}{}_K , l_w{}^{ij}{}_K$) means that our control algorithm shuts down algorithm $CA^{ij}{}_K$ at time $t_w{}^{ij}$ if $l_w{}^{ij}{}_K$ is true.

Similarly, BA_{ij} is onboard equipment controlled by this algorithm.

$BA_{ij} = \{ (Nam_{ij}, R_{ij}, P_{ij}) \}$

where Nam_{ij} is a name (ID) of particular onboard unit or device, R_{ij} is a mode of work of this device, and P_{ij} is a vector of consumption of onboard resources in mode R_{ij}.

KU_{ij} is a set of apparatus' commands issued by control algorithm to particular onboard device or unit.

$KU_{ij} = \{ (NK^{ij}_1, t_w^{ij}_1, l_w^{ij}_1), (NK^{ij}_2, t_w^{ij}_2, l_w^{ij}_2), \ldots (NK^{ij}_K, t_w^{ij}_K, l_w^{ij}_K) \}$

where NK^{ij} is a name (ID) of control command, t_w^{ij} is a time stamp of issuing of a command in case of truth of $l_w^{ij}_K$.

$FP^{ij} = \{(PI^{ij}_1, ts^{ij}_1, ls^{ij}_1), (PI^{ij}_2, ts^{ij}_2, ls^{ij}_2), \ldots (PI^{ij}_k, ts^{ij}_k, ls^{ij}_k) \}$ is a set of flags (predicates) formed by scenario j of algorithm CA^{ij} execution at time moments ts^{ij}_k in case of truth of condition ls^{ij}_k.

The presented models contain enough information required to determine the sequence of sections of spacecraft control and to form diagrams showing spacecraft functioning in different scenarios from various points of view. Also we can determine overlays between algorithms and functioning of certain onboard equipment. These different points of view are shown in Fig. 2.

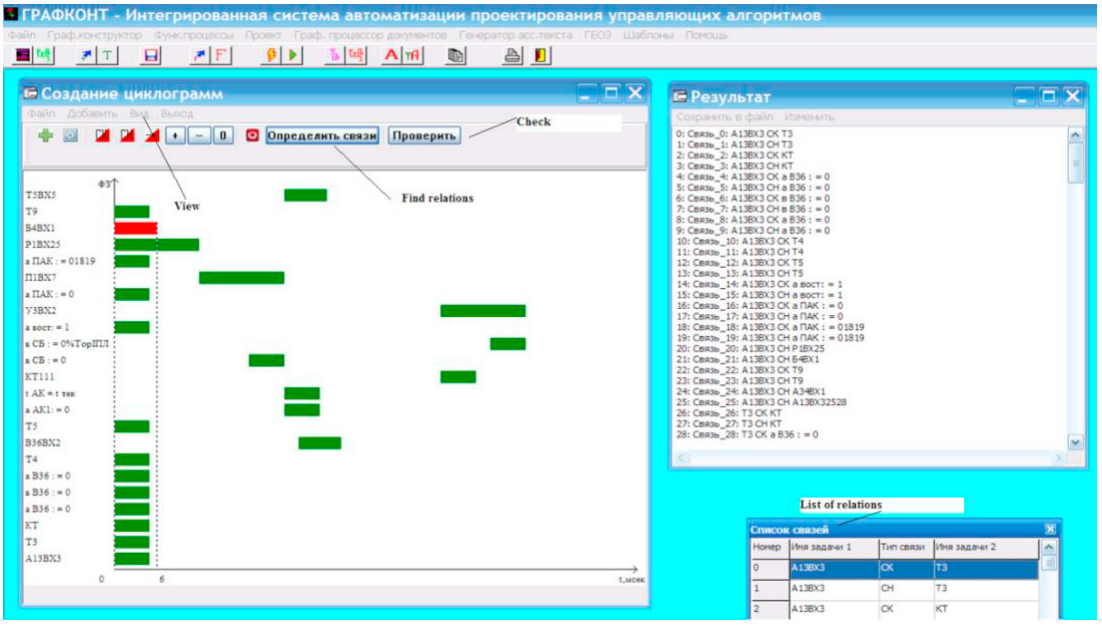

Fig.2. The screenshot of another software tool developed by authors.

2.3. Use of Satisfiability Modulo Theories solvers and Constraint Programming

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

Boolean satisfiability problem can be reviewed as partially similar problem to existence of consistent control algorithm which executes the required timed sequence of actions in case of limited onboard resources. The SMT approach looking even more suitable and we can consider use of ABSolver, Alt-Ergo, Barcelogic, MathSAT, CVC, OpenSMT, Simplify, STeP, Yices, Z3, and other available tools to be applied. In [18-19] we discussed ways 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, see Fig. 3.

Constraint Programming can be reviewed as approach well corresponding to problems of control in case of limited available resources by its nature. Constraints are the sort of limits. So, this circumstance opens the door to apply the constraint programming in our problem domain.

First, we 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. And after that we can specify existing limits for onboard resources and check whether the limits being violated.

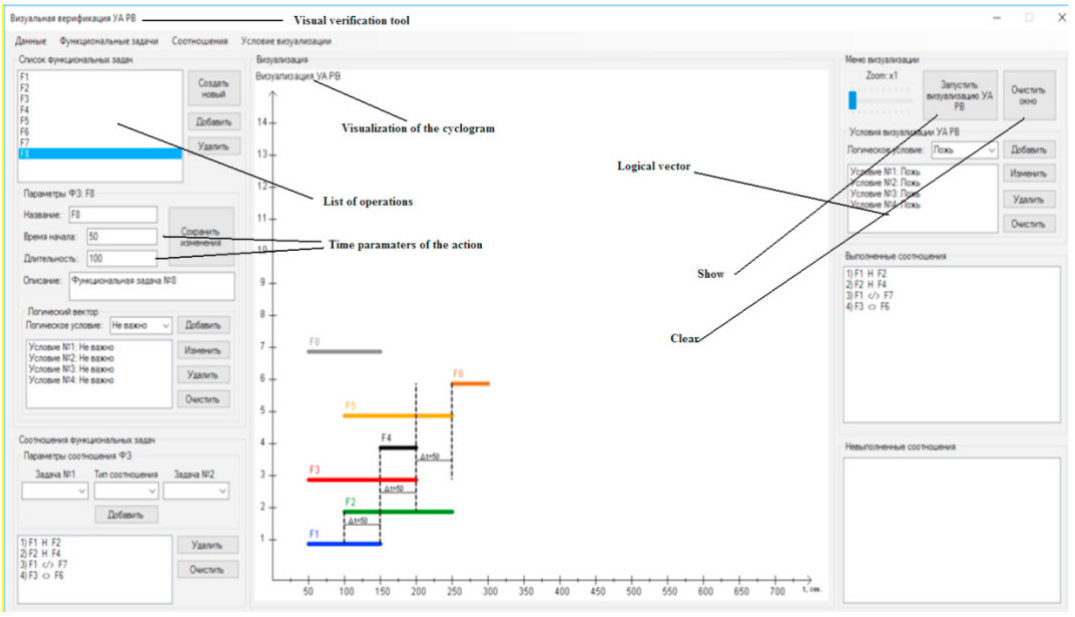

Fig. 3. The SMT based version of verifier's prototype.

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 a pre-processing steps which, among many other functionalities, 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 determine 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 below. 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. So automatic derivation of implied constraints is an enabling analysis for many other analyses.

3. Conclusion and Future Work

The analysis preformed for the processes of spacecraft control algorithm's design has shown the existence of opportunity of formal representation of data related to different phases of design process. This formal representation has been specified as mathematical models which to be converted into electronic models used in special computer-aided design tools. Now the authors work on implementation of these software 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 support of spacecraft control algorithms specification, visualization and checking, in case of restricted onboard resources. We plan to provide designer 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 period of spacecraft mission. The very important feature is building and visualization of cyclograms showing execution of tasks, calls of program modules, functioning of certain equipment.

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

References

[1] Kozlov, Dmitry, Anshakov, Gennadyi, and Yakov Mostovoy. (1998) *Control of Earth Observation Satellites: Computer Technologies*. Moscow, Mashinostroenie (In Russian).

[2]. Akhmetov, Ravil, Valentin Makarov, and Anatoly Sollogub. (2012) "Principles of the Earth Observation Satellites Control in Contingencies" *Information and Control Systems* **1**:16-22. (In Russian).

[3] Ashby, W. Ross. (1956) *Introduction to Cybernetics.* New York, Chapman & Hall

[4] Tomayko, James. (1994) *Computers in Space: Journeys with NASA*, Indianapolis, Alpha Books

[5] Eickhoff, Jens.(2012) *Onboard Computers, Onboard Software and Satellite Operations. An Introduction,* Heidelberg, Springer-Verlag

[6] Tyugashev, Andrey, Iliya Ermakov, and Iliya Ilyin. (2012) "Ways to Get More Reliable and Safe Software in Aerospace Industry", in *Proc. Program Semantics, Specification and Verification: Theory and Applications (PSSV 2012)*, July 1-2, in Nizhni Novgorod, Russia, 121-926

[7] Khartov, Victor (2006) "Autonomous Control of Communication and Navigation Spacecraft" *Aviakosmicheskoe priborostroenie (Aerospace Instrument-Making)* **6:** 12-23, (in Russian).

[8] Tiugashev, Andrei, and Alexander V. Belozubov (2016) "Toolset for construction and verification of rules for spacecraft's autonomous decision making" *Procedia computer science* **96**:811-818.

[9] Kalentyev, Anatoly, and Sygurov, Yuriy.(2010)"Development of information support for the spacecraft control algorithm design process" *Vestnik SGAU* **21(1)**:58–62. (in Russian).

[10] Tyugashev, Andrey. (2006) "Integrated environment for designing real-time control algorithms" *Journal of Computer and Systems Sciences International* **45(2)**: 287–300.

[11] Tyugashev, Andrey. (2016) "Language and Toolset for Visual Construction of Programs for Intelligent Autonomous Spacecraft Control" *IFAC – PapersOnLine* **49(5)**:120-5.

[12] Kalentyev, Anatoly, and Tyugashev, Andrey.(2006) *CALS technology in lifecycle of complex control programs*, Samara, Scientific Center of Russian Academy of Sciences. (in Russian).

[12]Tyugashev, Andrey, Dmitrii Zheleznov, and Sergey Nikishchenkov (2017) "A technology and software toolset for design and verification of real-time control algorithms"*Russian Electrical Engineering* **88**(3):154–158.

[14] Filatov, Artem, Tkachenko, Ivan, Tyugashev, Andrey, and Sopchenko, Elena. (2015) "Structure and algorithms of motion control system's software of the small spacecraft".in *Proc.International Conference Information Technology and Nanotechnology (ITNT 2015)*, in Samara, Russia, 246-251.

[15] Bogatov Artem, and Tyugashev, Andrey (2013) "The logical calculus of control algorithms".in Proc International Symposium for Reliability and Quality –. Vol.1. – P. 307–308. in Penza, Russia, (in Russian).

[16] Garcia, Maria de la Banda, Stuckey, Peter J., Van Hentenryck, Pascal, and Mark Wallace. (2014) "The Future of Optimization Technology" *Constraints* **19:**126–138.

[17] Marriott, Kim, Nethercote, Nicholas, Rafeh, Reza, Stuckey, Peter J., Garcia, Maria de la Banda, and Mark Wallace.(2008) "The design of the Zinc modelling language. "*Constraints* **13(3)**:229–267.

[18] Tyugashev, Andrey. (2018). "Application of SMT solvers for evaluation of Real-Time control logic of spacecraft."*Journal of Physics: Conference Series* **1096(1)***:012156*

[19] Tiugashev, Andrei. (2017)"Build and evaluation of real-time control algorithms in case of incomplete information about functional processes' parameters", in *Proc.of XX IEEE International Conference on Soft Computing and Measurements (SCM 2017)*, May 2017, Saint Petersburg, Russia,179-185.