Journal of Engineering and Applied Sciences 13 (3): 601-606, 2018
ISSN: 1816-949X

Visual Verification Tool for Real-Time Software

Andrey Tyugashev
Department of Applied Mathematics, Informatics and Information Systems,
Samara State Transport University, Samara, Russian Federation

Abstract: For space exploration, we need the appropriate software. Because its failure can lead to large financial, environmental and human losses. Thus, it becomes necessary to create software that will control the design and development of spacecraft. The study is devoted to verification problem of real-time software. The visual checking is a way that allows getting a fast 'rough' solution in many cases. To achieve this goal, we have used a range of complementary methods, among which are the methods: analysis and information modeling. The study also summarizes the experience of domestic and Foreign researchers on the subject. Special software toolset has been developed to support. As a result of computer simulation, a software was developed GRAFKONT which checks the programs in real time. Programs that were screened are intended for developing and designing on-board control software in spacecraft. For visualization of the systems in the spacecraft, GRAFKONT uses a graphical model which are presented in the form of screenshots in this study.

Key words: Software verification, flight control software, visual checking, graph of spacecrafts, domestic, designing

INTRODUCTION

Nowadays, computers are widely used not only for solving traditional computational problems but also in many other areas for instance as a main component of automated control systems that control technical objects, technological processes, communication equipment (Bulata *et al*., 2016; Stukalenko *et al*., 2016). Thus, realization of control algorithms is carried out by special software. Such kind of software is critical. An error in the control program can lead to a catastrophe huge material losses and human victims (Kim *et al*., 2015; Shang *et al*., 2015; Tate, 2016). It is no wonder that such close attention is paid to providing control software correctly. The situation is even more complicated because control software as a rule is real-time software. It means that not only the sequence of control signals is important but also time when they were carried out.

Specification and verification of the real-time software should be carried out with accurate consideration of its singularities (Khurshid *et al*., 2012; Roy *et al*., 2016). Let us note that the real-time systems can be divided into two big classes: event-driven and time-driven. We can say that in the event-driven systems the moment the event is processed is not valuable (i.e., event-driven systems have 'asynchronous' nature in some sense) while in time-driven systems, a concrete moment of time is 'the trigger mechanism' of desired actions. In this case, the main requirement to the control algorithm is implementation of control plan with a correct binding of processes in time. We often face such a situation when it is necessary to provide implementation of the coordinated control of technical complex containing a big number of devices, sensors, subsystems that should co-operate for execution of target task onboard spacecraft (Jacobs and Wortman, 2016; Savitha *et al*., 2014).

It is important to consider the following issue. Very often, a critical path on project network graph of spacecrafts and the whole space launch systems building is a path of control system development (Liu *et al*., 2014; Mehrabian and Khorasani, 2015; Zhao and Jia, 2016). In control system development, in turn, the most complex, long and labor-consuming process is creating and debugging control software. Thus, hundreds of people are involved into different activities connected to this process, including payload specialists, onboard equipment specialists, mathematicians, programmers, etc. All of these specialists need to have clear mutual understanding that is why it is necessary to use clear and adequate models of software in all stages of its lifecycle.

To provide necessary level of quality and reliability of control software, it goes throw multiple stage process of testing and debugging. At first, unit testing is being carried out using special automatic test tools that allow specifying test scripts by special language. Each test script is to be used for one path (branch) in the control program control flow graph. Then, integration testing is to be completed. At this stage, issues connected to correct interaction between the units should be resolved. At last, in the stage of complex testing, correctness of the interaction between the software and spacecraft onboard

equipment is to be checked. It is possible to use the real devices and aggregates of the spacecraft at this stage as well as specifically developed program simulators of the equipment (Tyugashev, 2006).

At the same time as it was formulated by classics of software engineering, program testing does not guarantee total absence of errors; it only allows finding some of them in certain cases (Tyugashev, 2006). It makes especially effective the development of other methods of verification and validation of control algorithms and real-time software. Some of them are program text inspections, symbolic execution of the program and the various methods of formal verification based on logical inference.

MATERIALS AND METHODS

The theoretical basis of the study was the set of complementary methods relevant goals, among them are: system information analysis as an instantiation of the system approach; information modeling which is the concretization of the scientific method modeling.

Moreover, during the work was studied and summarized the experience of scientists considered this issue. We used general scientific methods of analysis and synthesis and system-analytical, semantic and logical methods.

Visual verification of real-time control software: In development of programming languages, the following issue was noted by specialists: the question about borders of applicability of formal language tool and whether it should be connected to the human nature and structure of human memory. Thus, the one of the most natural forms of information representation for the human beings is a graphical image picture, diagram, scheme, etc. The typical words for the graphical representation are 'friendly', 'easy', 'clear', etc. There are several important pre-conditions for the usage of graphical tools in development of the real-time spacecraft onboard programs. The main reasons are the following: complexity of the problem to formulate a consistent (non-conflicting) specification. Necessity to find a good understanding between onboard system specialists, algorithm specialists, programmers and other participants of the program lifecycle. Necessity to keep accurate documentation of control programs. Necessity to keep documentation updated to current version of program. Possibility of usage of different perspective computer onboard platforms (Tyugahsev, 2009).

For the real-time control systems, the correctness of the synchronization of concurrent processes becomes very important. The key relationships in this sphere are 'begin-begin' (coincidence at begin, CB), end-begin (immediate following, →), end-end (coincidence at end), etc. The suitable clear representation in this case is a cyclogram (or Gantt diagram) as it is presented on Fig. 1. If the process is repeated several times, it can be shown by the several line segments at the corresponding 'lane' intended for this functional process.

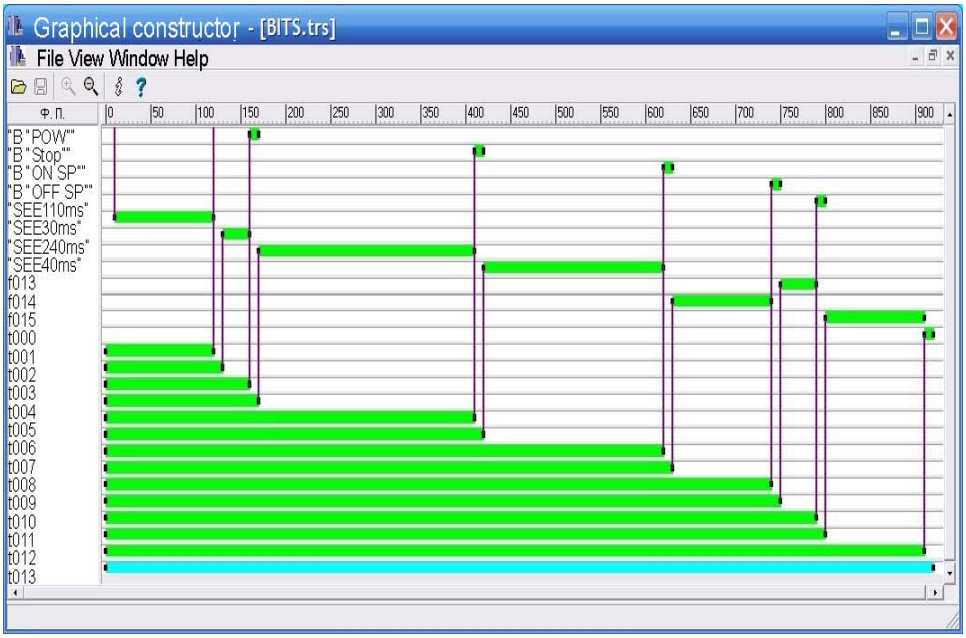

Fig. 1: Gantt diagram of concurrent processes

Unfortunately, the widely used visual methodologies, for example, UML did not support reflecting of such important issues; only in UML2.0, some steps towards these possibilities had been made.

Author proposed the semantic model of the real-time control programs that correspond to representation as a cyclogram (Gantt diagram). The semantic model is constructed on the basis of functional processes (functional tasks). Each functional task is to be defined by a tuple of four objects task name, the moment it begins, task runtime and a logical vector (set of the logical conditions that cause the execution of the task). Thus, the tuple describes one execution of the functional task at the specified moment of time with the certain set of logical conditions and duration (Kalentuev and Tyugahsev, 2006; Tyugahsev, 2005; Tyugashev, 2006). There are no evident specifications of the control's transition from one functional task to another in the proposed semantic model. Therefore, we can implement the same semantic model by different programs with the different control flow graph (more exactly with the different time-logic schemes real-time analogs of traditional control flow graphs (Tyugahsev, 2009).

This allows carrying out optimizing transformations of real-time control programs. The situation is the same as the situation with the calculating programs when it is possible to write several different programs that perform the same transformation of input data input output data (realize the same function).

As it was noted above, during specification of the real-time control programs, the one of the key issues is correct synchronization of the concurrent processes as well as concordance from the logical point of view. For example, the programmer can be interested in answers to such questions as "Is it true that execution of the functional task f1 ends before the start of the execution of functional task f1?", "does functional tasks f1 and f5 start at the same moment of time?", "is it possible that such combination of logical conditions allows imposition of execution of f2 and f7?", "does functional task f3 end by moment t = 500?", "how many concurrent functional tasks are executed at the moment t = 120, if the specified set of logical conditions is true?" and so on. Similar questions are to be answered during verification of the most important features of control programs. Visual representation allows performing quick check of implementing of desired features by the program and if it is necessary, using special toolset for further clarification.

In the special visual toolset named GRAFKONT (Tyugashev, 2006) developed by the researchers, we use the special problem-oriented language (Tyugahsev, 2005, 2009) which allows describing of coordination of execution of functional tasks in time as well as in logical space. For example, f1>f2 means that functional task f2 begins at a moment when f1 ends, f1→f2 means that two functional tasks should begin at the same moment of time, (~a3)=>f7 means that functional task f7 executes only if condition a3 is false.

There is a special simple algorithm implemented in GRAFKONT visual toolset which performs the reverse engineering, i.e., construction of semantic model (in the sense specified above), from the existing control program. This algorithm is based on the following. The internal logical scheme of the control program is reflected by special GRAFKONT system data structures. Because each typical action branching (by checking value of the flag in the program memory or the value formed by sensor), functional subroutine call, time delay, etc. is implemented by the standard macros (sequence of commands), we can use existing program code: the first step is to reconstruct the logical scheme of the program, i.e., form special internal data structure; further, fixing the variant of execution, i.e., set of values of logical conditions, construct the cyclogram reflecting the name and duration of the executed functional tasks. And then we can check justice of demanded specification on the constructed cyclogram as it is described below. The time-logical scheme in GRAFKONT also has two visual representations 'time diagram' (Fig. 2) and program flowchart (Fig. 3), it is very natural for the visual toolset. The algorithm of reverse engineering 'runs' all actions of control program, counting the "current time" from the zero value corresponding to the first "enter" of the control program and then adding the discovered time delays to the time value. As a result, we have the beginning time and using the user provided process duration information, end time of all functional tasks. Another parameter which is "collected" by the algorithm is a 'current' logical vector constructed from the conditions we are choosing on the branches.

This feature allows, in turn, performing automated verification of the demanded time constraints and properties of the program connected with synchronization of the processes (begin-begin, end-begin, end-end, etc.) using corresponding tools for visualization as it is shown in Fig. 4 and 5.

Therefore, besides the clear picture that is used by humans to quick check demanded requirements on the qualitative level (Fig. 2 and 3), it is possible to automatically form the set of the control program features in the form of discovered formulas. This is possible because the key relationships between the functional tasks in GRAFKONT system have a natural physical interpretation. For example, "begin-begin" relationship means tf1 = tf2, "end-end" relationship means tf1+tf1 =

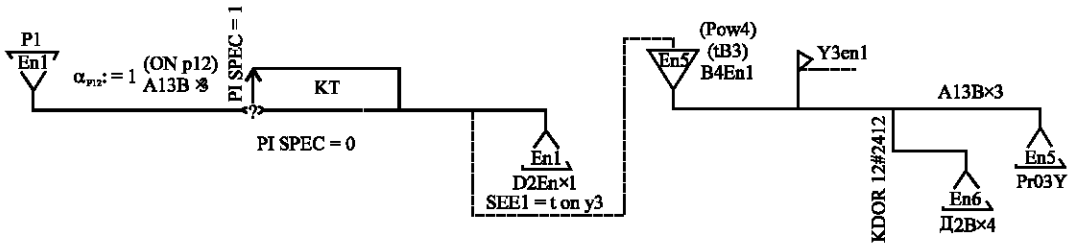

Fig. 2: Time diagram

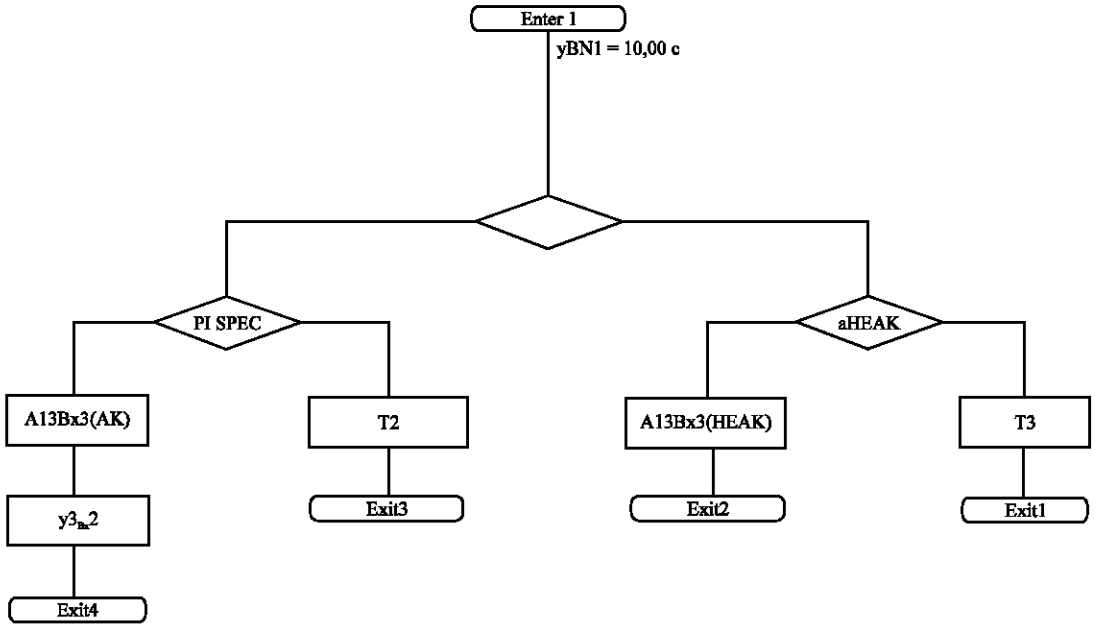

Fig. 3: Program flowchart

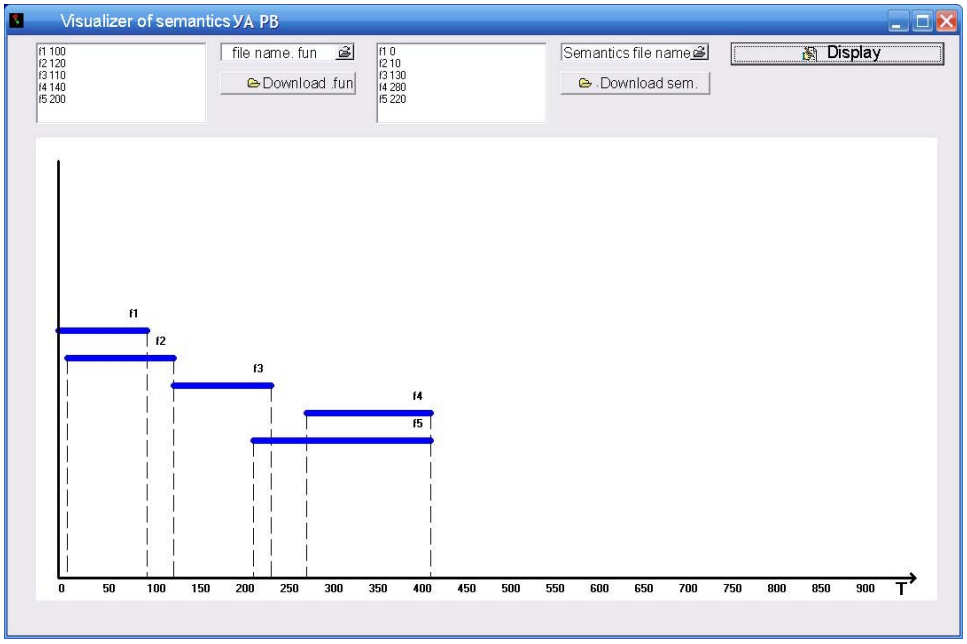

Fig. 4: Semantics visualization

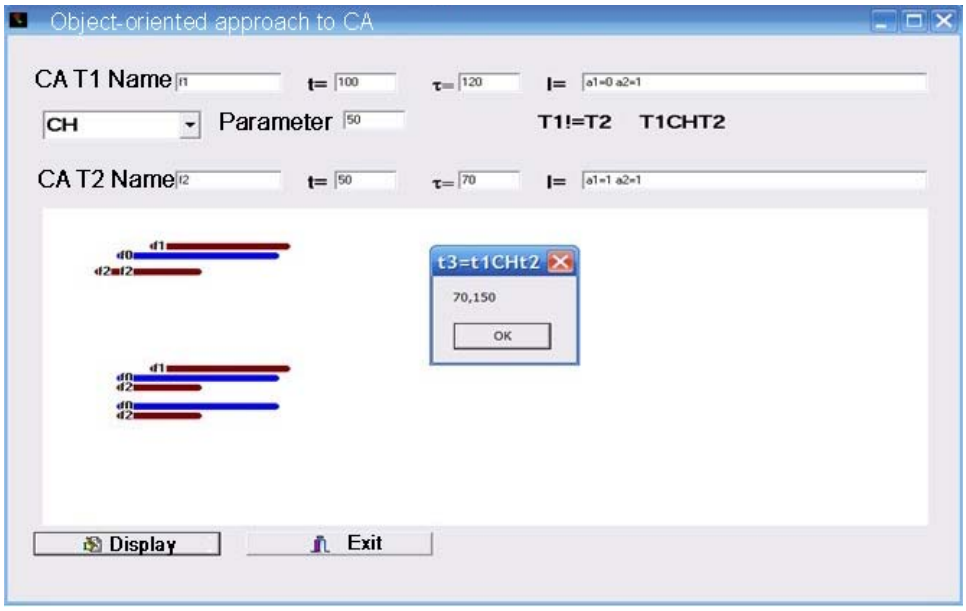

Fig. 5: Tools for visualization

tf2+tf2, →relationship means tf1+tf1 = tf2 where tf1 moment of f1 functional task begin, tf1 –duration of f1, tf2 and tf2 begin time and duration of f2). The simple algorithm allows discovering all relationships between the functional tasks. Proposed semantic model allows making model checking ease (Fig. 4).

The formal theory of the real-time control program also allows performing, in fact, some analog of a quick and clear 'model checking' verification method. Note that real-time program calculus described in, for example, Tyugahsev (2005, 2007) allows also formal inference of properties of the real-time control programs, i.e., just a formal verification of the specifications.

CONCLUSION

The special toolset for visualization of real-time onboard control program semantics has been reviewed. The semantic model uses the concept of 'schedule' that can be naturally represented in visual (graphical) form as a cyclogram (Gantt diagram). All illustrations in the study are screenshots of the different modules of GRAFKONT visual toolset developed by the researchers for the Russian space centers and intended for support of processes of design, development, documenting and verification of the real-time onboard control spacecraft's software.

The visual toolset provides opportunities for checking required properties of the real-time programs connected to synchronization of executing functional tasks. Usage of GRAFKONT allows lowering time and expenditure of labor expenses in development of onboard control spacecraft programs, raising reliability and quality of the software and lowering level of dependency on unique skills and knowledge of the certain person (programmer, onboard systems specialist, algorithm development specialist, etc.) by achievement of the best mutual understanding in the group of developers.

Thus, the proposed program is able to significantly improve safety in the field of space technology as well as significantly reduce the financial costs of preparation of space flights.

REFERENCES

Bulata, P.V., K.N. Volkovb and T.Y. Ilyinaa, 2016. Interaction of a shock wave with a cloud of particles. IEJME. Math. Educ., 11: 2949-2962.

Jacobs, S. and K.A. Wortman, 2016. Solar probe plus spacecraft flight software requirements verification test framework. Proceedings of the IEEE Conference on Aerospace, March 5-12, 2016, IEEE, Big Sky, Montana, USA., ISBN:978-1-4673-7676-1, pp: 1-8.

Kalentuev, A. and A. Tyugahsev, 2006. CALS Technologies in Lifecycle of Complex Control Programs. Russian Academy of Sciences, Samara, Russia,.

Khurshid, A., W. Zhou, M. Caesar and P. Godfrey, 2012. Veriflow: Verifying network-wide invariants in real time. ACM. SIGCOMM. Comput. Commun. Rev., 42: 467-472.

Kim, J.H., K.H. Lee, S.K. Hong and H.D. Kim, 2015. Development trend of cold gas propulsion system of a simulator for maneuvering and attitude control design verification of spacecraft. J. Korean Soc. Propul. Eng., 19: 87-97.

Liu, X., C. Gan and P. Lu, 2014. Distributed attitude coordinated tracking control for multi-group spacecrafts based on input normalized adaptive neural network. Proceedings of the Conference on American Control (ACC'14), June 4-6, 2014, IEEE, Portland, Oregon, USA., ISBN:978-1-4799-3272-6, pp: 2741-2746.

Mehrabian, A. and K. Khorasani, 2015. Distributed and cooperative quaternion-based attitude synchronization and tracking control for a network of heterogeneous spacecraft formation flying mission. J. Franklin Inst., 352: 3885-3913.

Roy, S., J. Misra and I. Saha, 2016. A simplification of a real-time verification problem. Software Test. Verif. Reliab., 26: 548-571.

Savitha, A., M. Ravindra, N.P. Kumar, R.R. Chetwani and K.M. Baradwaj, 2014. Automated verification of spacecraft telemetry data. Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC'14), December 18-20, 2014, IEEE, Coimbatore, India, ISBN:978-1-4799-3974-9, pp: 1-4.

Shang, Y., J. Wang, Z. Gong, X. Li and Y. Pei *et al*., 2015. Night vision imaging system design, integration and verification in spacecraft vacuum thermal test. Proceedings of the International Conference on Optical Instruments and Technology, April 11-12, 2015, SPIE, Bellingham, Washington, USA., pp: 962201-962208.

Stukalenko, N.M., B.B. Zhakhina, A.K. Kukubaeva, N.K. Smagulova and G.K. Kazhibaeva, 2016. Studying innovation technologies in modern education. Intl. J. Environ. Sci. Educ., 11: 7297-7308.

Tate, J.P., 2016. Handbook for RF ionization breakdown prevention in spacecraft components. Proceedings of the IEEE International Conference on Plasma Science (ICOPS'16), June 19-23, 2016, IEEE, Banff, Alberta, Canada, ISBN:978-1-4673-9602-8, pp: 1-1.

Tyugahsev, A., 2005. Algebraic models of real-time spacecraft's control algorithms and programs. Samara Tech. Univ. Bull., 1: 19-25.

Tyugahsev, A., 2009. Graphical Programming Languages and its Applications in Real-Time Control Systems. Samara Scientific Center of the Russian Academy of Sciences, Samara, Russia,.

Tyugashev, A.A., 2006. Integrated environment for designing real-time control algorithms. J. Comput. Syst. Sci. Intl., 45: 287-300.

Zhao, L. and Y. Jia, 2016. Neural network-based distributed adaptive attitude synchronization control of spacecraft formation under modified fast terminal sliding mode. Neurocomputing, 171: 230-241.