Visual presentation of life cycle control model of the cross-platform onboard software

I V Kovalev, V V Losev, M V Saramud, D I Kovalev and M N Chuvashova

1 Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations, Krasnoyarsk, Russia
2 Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
3 Siberian Federal University, 78 Svobodny ave., Krasnoyarsk, Russia

E-mail: basilos@mail.ru

Abstract. The visual presentation of the software life cycle control model is considered in this scientific paper. A realization of the control model is connected with identifying the semantics, properties and attributes of formation stages, implementation and maintenance of software. For research the methodology IDEFO is applied. The software lifecycle control model is obtained in a graphical notation. Functional modeling of complex processes, associated not only with development, but also with testing and long-term maintenance of onboard software, is described and presented. It is shown that visual modeling tools are able to be as an effective mean of making control decisions.

The life cycle of cross-platform onboard software (OBS) is a complex of stages, which accompanies a number of activities, during which the software for control systems of unmanned aerial objects is formed, implemented and accompanied. The formation of this process is associated primarily with the receipt of a system aimed at solving a deterministic set of tasks. Research of this process, the identification of the properties of individual stages and the accompanying measures requires modeling the life cycle of cross-platform OBS.

Modeling the life cycle of a cross-platform OBS, first of all, it is connected with the formalization of its processes and stages. Also application of the methodology, which permits to describe the semantics of the processes and stages of the formation, implementation and maintenance of software.

For visual modeling of the life cycle of a cross-platform OBS, let us turn to the IDEFO methodology [1,2], namely the IDEFO methodology, with the goal of building a functional model of the life cycle and describing the rules of this model (logical relationships between works).

The IDEFO functional model is a set of blocks, each of which is a “black box” with inputs and outputs, controls and mechanisms that are detailed (decomposed) to the required level.

Let us consider a functional model of the first level (figure 1). As the main block, we consider the function "Development of onboard software”. The introductory arrows interpret certain tasks aimed at ensuring the principle of cross-platform. For example, the “control system” based on the real-time operating system FreeRTOS [3,4,5]; selection of the hardware platform to which FreeRTOS is ported; a set of subsystems that provides the main and “useful” functional of the managed object [6,7]. The control arrows represent the control mechanisms, in particular, the terms of reference describing all
aspects of the design, testing and maintenance of the OBS; the methodology that reflects the approaches and principles of implementation of the functional blocks, presented in figure 2. They are stages of the life cycle of cross-platform OBS; standards according to which long-term maintenance of OBS is carried out, including standards of the operating organization.

The developer of OBS, tester and software in the complex involved at each stage are considered as mechanisms.

![Diagram](attachment:image.png)

**Figure 1.** Description of the top-level functional model.

Decomposing the functional model of the upper level, we obtain four main steps:

- Development of the architectural design of OBS.
- Development of the detailed design of OBS.
- Testing the component of OBS.
- Servicing of OBS.

Describing parameters of the upper level functional model (input, output, control and mechanisms) are also present in the second level functional model. However, there require attention binders which are the output and at the same time input parameters of the selected steps and their functional blocks.

The development stage of the OBS [8] architectural design is associated with the system description in terms of its constituent modules (components). Decision strategy requires resolving issues that are related to all architectural levels of the developed system. The quality of an architectural design is extremely important for the long-term success of the system. Accurate architectural design permits to create the supported systems, i.e. systems that are able to understand, maintain and scale (expand).

Without these qualities, the intrinsic complexity of the software is out of control. Therefore, it is extremely important that as a result of architectural design an adaptive system emerges that would be saved in the programming step and carefully maintained after delivery of the system, i.e. at the step of
maintenance and operation. The result, and at the same time, the input parameter of the new step is the component structure (architecture) of the OBS [9].

![Diagram](image.png)

**Figure 2.** Description of the second level functional model.

In the following step the detailed design function is implemented, i.e. descriptions of the internal operation of each component of the onboard software. Its result is detailed algorithms and data structures for each component. In the end, the components are unwrapped on the nodes of the underlying implementation platform. Accordingly, algorithms and data structures should take into account constraints (both contributing to and impeding the operation of the system) imposed on the underlying platform. The result, and at the same time, the input parameter of the new phase is the algorithms and data structures of the projected components.

The reliability of complex software is primarily determined by the quality of their components - modules and functional groups of programs. Many of them don’t directly use the information of the external environment and real-time values, as a result of which concepts and criteria that characterize the reliability of programs don’t apply such components. For these components, the main quality criteria it is advisable to use deterministic or stochastic correctness. However, the quality of such components that make a complex real-time software can determine their reliability and safety. Therefore, when creating reliable software systems, it is important to use correct and high-quality software components. Their high quality is achieved by regulated, systematic testing and debugging. The relative simplicity and visibility of most components permit to use sufficiently formalized methods and techniques for the automated planning and effective implementation of testing [10].

The methods and tools are used by this stage to ensure the automation of component testing, including repetitive, significantly reduce the cost of maintaining the components of cross-platform OBS. It is achieved both by shortening the component testing time itself and by automating the testing process. The result, and at the same time, the input parameter of the new stage is the assembly of the components under the stated hardware platform.
The final stage, considered in this cycle of life is the maintenance of OBS [11]. The long-term maintenance is the management of the configuration of OBS. It permits to get all the hierarchical relationship of documents and components that constitute a project configuration and development archives management. Thus, the result of the stage and the output product of the considered model is a documented cross-platform OBS.

The used IDEF0 methodology permits to describe and present the functional modeling of complex processes associated not only with development, but also with testing and a long-term maintenance of the onboard software. It is a visual modeling instrumentation and an effective means of making management decisions.

**Acknowledgments**

This work was supported by Ministry of Science and Higher Education of Russian Federation within limits of state contract № 2.2867.2017/4.6.

**References**

[1] Information technology - Modeling Languages - Part 1: Syntax and Semantics for IDEF0 ISO/IEC/IEEE 31320-1:2012 (Adoption of IEEE Std 1320.1-1998) 2012 pp 1-120

[2] Fu M, Wang D, Wang J and Li M 2018 Modeling Method of Operational Task Combined with IDEF and UML IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)

[3] Kovalev I V, Losev V V, Saramud M V, Kovalev D I and Petrosysan M O 2017 To the question of implementation of multi-version execution environment software of onboard autonomous pilotless objects by means of real-time operating system Vestnik SibGAU 1(1) 58–61

[4] Hsu H and Hsueh C-W 2016 FreeRTOS Porting on x86 Platform 2016 International Computer Symposium (ICS) pp 120-3

[5] Saramud M V, Kovalev I V, Losev V V and Petrosysan M O 2018 Software interfaces and decision block for the execution environment of multi-version software in real-time operating systems International Journal on Information Technologies and Security 1(10) 25-34

[6] Saramud M V, Kovalev I V, Losev V V and Voroshilova A A 2019 A tool for developing and verifying components of cross-platform on-board software International Journal on Information Technologies and Security 1(11) 13-20

[7] Saramud M V, Kovalev I V, Losev V V and Petrosysan M O 2018 Application of FreeRTOS for Implementation of the Execution Environment of Real-time Multi-version Software International Journal on Information Technologies and Security 3(10) 75-82

[8] Tang Yu R and Li Y 2011 The software architecture of a reconfigurable real-time onboard control system for a small UAV helicopter 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) pp 228-33

[9] Agarwal R, Chetwani R R, Ravindra M and Bharadwaj K M 2014 Novel methodology for requirements to design traceability of onboard software International Conference on Advances in Electronics Computers and Communications pp 1-6

[10] Mouzakitis A, Muniyappa N, Parker R and Puthiyapurayil S 2010 Advanced automated onboard vehicle diagnostics testing UKACC International Conference on Control pp 1-6

[11] Liang K, Guo J and Wang J 2014 On-board software maintenance for manned spacecraft IEEE 5th International Conference on Software Engineering and Service Science pp 235-9