VERIFICATION OF INDUSTRIAL CONTROL ALGORITHMS IN VIRTUAL LABORATORY STANDS

Tatyana Liakh[(a)], Vladimir Zyubin[(b)]

[(a),(b)] Novosibirsk State University,
Institute of Automation and Electrometry SB RAS

[(a)]antsys_nsu@mail.ru, [(b)]zyubin@iae.nsk.su

ABSTRACT
The complexity of industrial control systems increases every year. The imposed requirements should provide the reliability of the developed algorithm. But today the common practice of industrial automation is characterized by the following: generally testing of control algorithms starts only when you run the software on a new facility.
In the article the control algorithms development method was put forward. The method uses virtual laboratory stands for control algorithm creation and its verification. Virtual laboratory stand consists of the five components: graphical representation of the automated object (GRAO), control algorithm module (CA), virtual plant module (VP), scripts control module (SCM) and verification module (VM). All modules are described as a hyper-process with the Reflex language. SCM imitates different situations on the object: environmental changes and breakdowns. VM watches over the CA during the simulation and automatically checks whether the algorithm meets the given specification under the certain script.
Keywords: control algorithms, industrial automation, process-oriented programming, virtual plants, language Reflex, verification

1. INTRODUCTION

Testing and verification problem of industrial control algorithms is one of the key problems in industrial automation area. Industrial control algorithms are logically complicated because of the great number of dependences between different parts of an algorithm. But the cost of errors in such algorithms is extremely high: errors in control algorithm leads to malfunctions, abnormal situations and accidents on the technological plant.

Today many approaches and mathematical models for the control algorithms development exists. Domain-specific languages (DSL) are perspective instrument for this issue. Reflex language is the DSL for the control algorithms development in industrial automation and robotics. The Reflex language is based on the hyper-process model. Hyper-process is an extension of the classical state-machine model (Zyubin 2007). The basic concept of the Reflex language – process. The program on the Reflex language is a set of parallel executable processes. They can trigger each other, stop, and monitor states. This allows to process signals from the technological plant in parallel. Reflex also allows operations with time intervals and offers means for the interaction with sensors and control devices. Ease of use of the Reflex language and its adequacy to industrial automation tasks was confirmed in automation projects for complex technical objects.

But testing and verification of the algorithm is still a serious issue. It is impossible to test the algorithm without plant. In most cases control algorithms testing starts only when you run the software on a new plant. As a result, the testing of the algorithm is postponed until the start-and-adjustment works begin. Such practice leads to high risks, emergency situations or even to accidents at the facility.

The contribution is structured as follows. First, we discuss specificity of control algorithms for complex technical objects. Second, we describe shortly the hyper-process mathematical model and the Reflex language. After that, we demonstrate the approach of the control algorithms verification via virtual plants conception in virtual laboratory stands.

2. THE SPECIFICITY OF INDUSTRIAL CONTROL ALGORITHMS

Due to the specificity of the automation tasks, especially in the case of complex technical objects, it is extremely hard to develop control algorithms for such issues. The reason is that industrial algorithms have a number of properties, unique to the field of industrial automation (Zyubin 2005, Kof 2003):

1. Presence of an external environment to interact with.
2. Cyclic and event-driven functioning.
3. Synchronism – control algorithm implies synchronization of its functioning with physical processes in the external environment.
4. Mass logical parallelism – it reflects existence of a large set of concurrent (or to be precise – independent and weakly connected) physical processes in the controlled objects. As the events appear in an arbitrary consequence, any

attempt to describe the system reaction within a monolithic block leads to a combinatorial task with exponential increase of complexity, so called the combinatorial explosion of complexity

5. Hierarchical structure. Any complex control algorithm has to have hierarchical structure that reflects artificial nature of the external environment, the designer plan that is implemented in form of the facilities (Zyubin 2004). Because of the logical parallelism, the hierarchical structure consists of chains independently executed in parallel. It means that the divergence and convergence of control flow are a significant part of control algorithm.

3. HYPER-AUTOMATON MODEL OF CONTROL ALGORITHMS

The variety of different formalisms for specifying control and reactive systems were introduced (Hoare 1985, Harel 1987). One of these models - hyper-automaton model, which is introduced as a useful formalism for control algorithms reasoning.

The hyper-automaton model is an extension if the the finite state automata (FSA). A hyper-automaton is an ordered set of processes, which are cyclically stirred to activity with a period of activation TH. A hyperautomaton is a triplet:

$$H = <T_H, P, p_1> \quad (1)$$

- T_H is a period of activation.
- P is a finite nonempty and ordered set of processes.
- ($P = \{p_1, p_2, \dots, p_M\}$), where M is the number of processes.
- p_1 is the first marked process, $p1 \in P$, which is the only non-passive when hyper-automaton starts.

A process is a state machine where the states are functions. Mathematically, i-th process is a quintuple:

$$p_i = <F_i, f_{1i}, fcur_i, T_i>, \text{ where} \quad (2)$$

- F_i is a set of mutually exclusive functions.
- f_{1i} is the first function, ($f_{1i} \in F_i$).
- $fcur_i$ is the current function, ($fcur_i \in F_i$).
- T_i is the current time.

The F_i and f_{1i} elements characterize static features of the process. The $fcur_i$ and T_i elements give us means for reasoning about a process dynamics.

A function of a process is a set of events and reactions to the events. Mathematically, j-th functions of i-th process is a twain:

$$f_{ji} = <X_{ji}, Y_{ji}>, \text{ where} \quad (3)$$

- X_{ji} is a set of events ($X_{ji} = \{x_{ji1}, x_{ji2}, \dots, x_{jiL}\}$).
- Yji is a set of reactions ($Y_{ji} = \{y_{ji1}, y_{ji2}, \dots, y_{jiL}\}$).

Hyper-automation model can be implemented by means of general-purpose programming languages, but in case of control algorythms one more often choose domain-specific languages (DSL), designed for industrial automation issues. One oh these languages is the Rexlex language. The Reflex language was used in a number of automation protects for complex technical plants (Liakh 2016) and shows its adequacy to industrial automation tasks.

4. INDUSTRIAL CONTROL ALGORITHMS TESTING WITH VIRTUAL PLANTS

But the description problem of control algorithms is no the only problem in industrial automation, Verification of the algorithm is also a serious challenge for the developers.

The concept of Virtual Plant (VP) was put forward to solve the control algorithms testing issues (Zyubin 2007). This concept was offered in the Institute of Automation and Electrometry. VP – is the program imitator of the automated technological process. VP code and control algorithm (CA) code executes separately (Fig. 1). Unified data ex-change between VP and CA ensures the connections to be saved when the CA is changed. This approach allows us to use the iterative development model and debug the algorithm code before start-and-adjustment works begin.

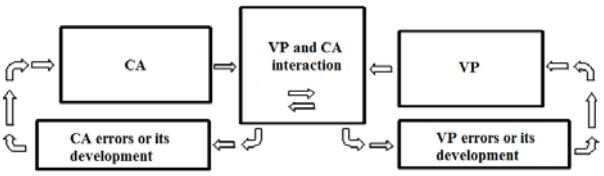


Figure 1: Control Algorithm Iterative Development Model via the Virtual Plant Conception

For correct simulation of the plant one need to describe signal processing and parallel operations on the object. That's why CA and VP were both created on Reflex language. VP operates in multiple modes: correct operation mode, or malfunction-imitation modes. The testing software was created with means of LabVIEW package. This software allows running VP and CA simultaneously, testing VP and CA and imitating data exchange and external events (Liakh and Zyubin 2014). The CA of the Large Solar Vacuum Telescope (LSVT, Baikal Astrophysical Observatory) was developed via this software (Liakh and Zyubin 2016).

But some disadvantages of such approach were noticed. Many operations had to be made manually by the operator. Operator controlled VP modes, imitated GUI commands and environmental changes. Also the operator had to check the correctness of algorithm

reactions manually. All this disadvantages led to the long testing process. Also the possibility of missed errors and unexplored behavior grows.

5. METHOD EXTENTION WITH THE SCRIPTS CONTROL MODULE AND VERIFICATION MODULE

To increase the reliability of the algorithm and to speed up the testing process, one offer the modification of the method. One notice that in the method mentioned above operator has multiple issues. First, he has to manage the sequence of different events for the CA and the VP. Also he has to check whether the CA meets a given specification under a certain script.

That's why one offer to supplement the existing scheme with two modules: scripts control module (SCM) and verification module (VM). The interaction between all modules is described on the Fig. 2.

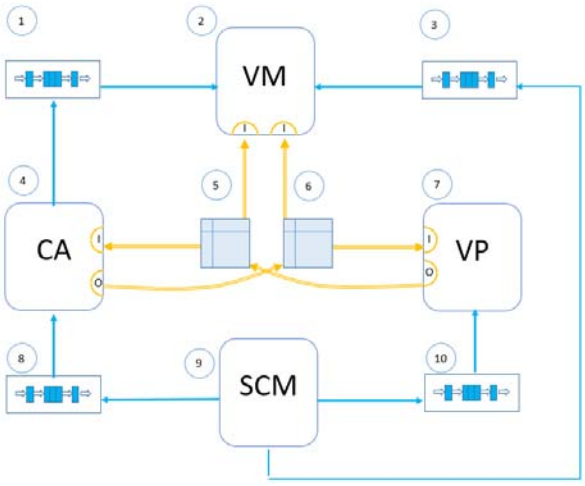
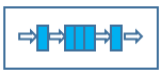


Figure 2: Control Algorithm Iterative Development Model via the Verification Module and Scripts Control Module

Table 1: Conventional Notations

	Event-Driven Algorithmic Module
	
	Digital Ports (Input/Output)
	
	Message Queue
	Digital Ports Values Cache
	Digital Data
	Messages

Four interacted algorithmic modules (hyper-processes) are used to control algorithm verification. They run in the determined order, form message queues, digital ports values and factual parameters values.

1. Control algorithm (CA, 4) – CA module implements the control algorithm logic.
2. Virtual plant (VP, 7) – VP module implements the plant operation logic. VP imitates data flow to CA from ADC and data flow from digital input devices.
3. Verification module (VM, 2) – VM automatically checks whether the CA meets a given specification under a certain script. VM alternately checks a set of temporal requirements and reports about malfunctions. VB verifies only CA block.
4. Script Control Module (SCM, 9) – SCM manages the sequence of different events for CA, VM and VP, such as: operator actions, environment states and malfunctions of devices. SCM sends commands through the message queues. SCM imitates operator commands for CA block. For VP SCM sets operation modes: it forces VP to imitate different accidents and breakdowns – or operate in correct mode. Also SCM sends environmental data for the VP. SCM reports VB which of scripts is actual now. According these information, VB choses properties to verify.

Algorithmic blocks interact according to the scheme:

1. SCM launch. SCM forms three message queues: for CA (8), VP (10) and VB (3). For the CA SCM sends messages, thereby imitating operator commands. SCM informs VP about the current operation mode (regular operation mode, equipment failure or an accident). Also SC informs VP about environmental state (e.g. temperature or pressure) – but it is for VP to decide, how process these information – or ignore it. In the VB message queue SCM puts messages to identify current script: this information allows

ISBN 978-88-97999-85-0; Affenzeller, Bruzzone, Jiménez, Longo and Piera Eds.

VP to determine, how CA should operate in case of correct operation.

2. CA launch. In the module the control code of the technological control system executes. CA takes input digital data from the ports cache (5). Also CA handles messages from SCM queue (8). At the end of the cycle CA puts its digital output to the CA output ports cache (6). Output messages CA puts into the VM input message queue (1).
3. VM launch. VM analyses all data flow from the CA module: digital input and output values from port cashes (5, 6), and its output messages (1). With this data VM is able to verify whether the CA meets all imposed requirements. From the SCM message queue VM takes information about current script – in order to determine the correctness of the algorithm reactions. VM sends testing results to the operator GUI.
4. VP launch. VP takes information about current mode and environmental changes from the SCM message queue. VP takes input digital data from the CA output ports cache (6). This imitates the data exchange between the CA and a real plant. At the end of the cycle VP puts its digital output to the CA input ports cache (5).

Development of industrial control algorithms occurs iteratively:

1. A functionally separate part of the CA code is created in Reflex language, i.e. a set of processes responsible for processing a specific task.
2. A functionally separate part of the VP code in Reflex language is created. This code simulates the operation of those elements of the control plant that are controlled as described in the algorithmic module.
3. The SCM block is described. SCM sends three types of messages for other algorithmic blocks:
   - Messages for the CA block. These messages simulate the actions of the operator.
   - Messages for the VP block. These messages determine the scenarios set for all parts of the plant already described and for the whole virtual plant.
   - Messages for the VM - these messages notify the VM about the current verification mode. According to these messages, the VM determines which scenario is being executed at the moment and what reaction to expect from the CA.
4. The VM block is described. In the VM block, the response to the messages from the SCM, described in step №3, is added. Also, the requirements imposed on the created algorithmic control unit are described. Since the development happens iteratively, these requirements can also verify the operation of the algorithm parts created in the previous steps – in this case, the joint interaction of various parts of the algorithm is verified.

The Reflex language was chosen as a programming language for the all event-driven modules. The language Reflex based on the hyper-process model. Hyper-process model reflects openness, event-driven nature, cyclicity, synchronism, and mass logical parallelism of a control algorithm. The temporal dependences of the hyper-processes allows do describe the temporal requirements.

3. CONCLUSION

In the contribution the scheme of the iterative CA development was put forward. The scheme for the automated verification of the developed algorithm was created. Such approach to the CA testing and development allows to give the strict answer if the algorithm satisfies all necessary requirements. Also virtual stands are helpful for the engineering students training.

ACKNOWLEDGMENTS
The research has been supported by Russian Foundation for Basic Research (grant 17-07-01600).

REFERENCES

Zyubin V. E., Hyper-automaton: a Model of Control Algorithms // Proceedings of IEEE International Siberian Conference on Control and Communications, SIBCON-07. Russia, Tomsk April 20-21, 2007, PP.51-57

Zyubin, V. E., Multicore Processors and Programming. // Open Systems J., №7-8, 2005, PP.12-19

Kof, L., Schätz, B., Combining Aspects of Reactive Systems. // Proc. of Andrei Ershov Fifth Int. Conf. Perspectives of System Informatics. Novosibirsk, 2003, PP.239-243

Zyubin, V. E., Text and Graphics: What Language Does Programmer Need? // Open Systems J., №1, 2004, PP. 54–58

Hoare, C. A. R., Communicating Sequential Processes. // PrenticeHall Int., 1985

Harel, D. Statecharts: a Visual Formalism for Complex Systems. // Science of Computer Programming 8. Elsevier Science Publishers B.V., North-Holland,1987, PP. 231–274

Liakh T. V., Zyubin V. E., Application of the virtual plant for industrial automation issues // Proc. Ershov informatics conference-14, satellite "Workshop on Science Intensive Applied Software", Saint Petersburg, Russia, 2014, 43-48.

Liakh T. V., Zyubin V. E. The Reflex Language Usage to Automate the Large Solar Vacuum Telescope // 17th International Conference of Young

Specialists on Micro/Nanotechnologies and Electron Devices (EDM). (Erlagol, Altai Republic, Russia, June 30 2016-July 4, 2016). pp. 137-139.

AUTHORS BIOGRAPHY

Zyubin V. E. Lead researcher of the Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences. In 1992 he graduated from the Novosibirsk State Technical University, majoring in "Automation and Remote Control". The degree of Doctor of Technical Sciences was awarded in 2014. Associate Professor in the specialty.

He has authored over 90 scientific publications.

Interests – Languages of technological programming, programming psychology, human factors, complex control algorithms, the theory of finite automata, hyper-automata model, event polymorphism, semiotics and pragmatics, the massive parallelism of logic, Reflex language, languages of IEC 61131-3 standard

Liakh T. V. is a postgraduate student of the Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences. In 2013 Tatiana graduated with honors from the Novosibirsk State University (Physical Faculty, Automation of physical and technological researches department).

From 2009 to 2011 Tatiana was engaged in the researches in the field of computer simulation and artificial intelligence in the SoftLab-NSK company.

Since 2011 Tatiana develops industrial automation systems in the Institute of Automation and Electrometry. She has participated in many projects of automation, such as the development of the automated control system of the Large Solar Vacuum Telescope – the biggest Eurasian telescope and main instrument of the Baikal Astrophysical Observatory.

Interests – Complex control algorithms, methods of industrial algorithms development, programming languages for industrial automation and industrial algorithms verification.