Improving Simulators Quality Using Model and Data Validation Techniques

Industrial Project Examples

Vesselin Gueorguiev
Technical University Sofia,
Sofia, Bulgaria
e-mail: veg@tu-sofia.bg

Abstract - **This paper presents ideas and approaches to design, implementation and validation of simulators using the program generation approach. This approach is well presented in today's industrial control systems configurators. It has now extensive use also in embedded systems design and implementation. Unfortunately, the validation of the generated control systems usually is not part of the generating tool. Results of simulator implementation using the program generation approach and their validation are presented. Analyses of problems found and the steps for problem fixing are presented in the context of increasing the system model reliability. This paper also presents the validation of the program generation approach for design and implementation of object simulators and how the quality of the simulator can be increased using validation techniques.**

Keywords – simulation validation; program generation; re-design after validation; model reliability.

I. INTRODUCTION

The area of Validation and Verification (V&V) of system design and system modelling and simulation has been a hot topic for the last more than three decades [1][2]. A huge number of papers have been written on it. A large number of tools for V&V of different complexity has been designed, implemented and used. In this paper some validation aspects will be discussed. Ways to improve the model of the simulated system will also be discussed.

It is clear that there is no perfect solution for any validation problem. This is based on the generic fact that human activities of any kind are not only formal (scientific), but informal (art) too. The titles of two basic (very formal) too often cited books help to prove this – *The Art of Computer Programming* by Donald Knuth [22] and *The Art of Simulation* by K.D. Tocher [23]. Nobody can eliminate all problems and errors in system models and their implementation; here, an approach will be presented how it will be possible to increase models' reliability and to decrease the risk of design and implementation failures. The approach 'I was smart, I thought' is always valid; but, for the engineering practice, is needed something much more stable. Elements of these approaches will be presented hereafter.

Simulator-based studies of theories, algorithms and their implementation validity today are old and extensively exploited topics. These include vehicle, flight, nuclear power plant, robots and many other simulators [6][7][8][9]. Computer-based simulators of many different complex objects are a reality today. First, they appeared in military, aircraft and nuclear power plant applications; now they are available everywhere. Simulation of various systems has a lot of advantages compared to experimenting and using the actual systems. Such an advantage is the possibility to train the personnel to operate various types of machines, or to use such simulations in preparation and optimization of control algorithms, or to repeat and analyse specific situations.

In the area of control systems design, there are many implementation approaches – from ad-hoc design and programming to fully automated code generation [10]. One specific approach to control systems implementation is the program generation. Using the same approach for simulator implementation is very promising, too. This approach reduces the amount of investments and risks in the design and implementation phases of the control system design. Simulators of that type are not used only for control system tests and improvements. Personnel training and abnormal situations analyses are other areas of their use.

In this paper we discuss not only validation of generated control systems, but validation of the program generation approach as a validated basis for control systems and simulators generation.

The paper is structured as follows: Section II presents validation techniques for simulators. Section III presents the program generator used for simulator generation and its formal model. Section IV presents validation of the implemented simulators; Section V is the conclusion.

II. VALIDATION TECHNIQUES FOR SIMULATORS

To check that a programmed system implements just what is designed, an analyst needs to verify that system. Many methods have been developed over time to achieve this [2][11]. Additionally, the analyst has to check that the implementation is free of errors. But one question has not been answered yet, namely whether the design is correct and if the conceptual model is an accurate representation of the system it has to represent.

Let us start with the presumption that the validation is a compulsory element of the activities and procedures for improving the quality of the generated software as well as that of the whole system. Validation can be a "process of evaluating a system or its components in the process of its development or at its end to determine whether they satisfy specified requirements" [12] or a "process of collection of information showing that both software and associated products satisfy the system requirements at the end of each

 ISBN: 978-1-61208-371-1

cycle of its life, as well as satisfy some user needs in a specific case".

Simulators, as software products, can be created using every one of the four basic models for development: (i) development of a new product; (ii) modification; (iii) configuration / reconfiguration, and (iv) use of ready-for-use software applications (through the "Pipes & Filters" architecture). In case of simulators produced by program generation, the very good results gives the method for validation of product type "configuration / reconfiguration". On this basis, a modification of the classical methodology for validation of this class of applications was implemented. The main characteristics of this new approach are the following:

Validation when creating a new software product using the "Configure/Reconfigure" approach

It is performed in two main ways:

- Validation of the basic software;
- Validation of the configuration data.

The validation of the basic software is done by using the existing software (by applying one of the two options).

- For each subsystem and for each module responsible for any of the functions of the new SW;
- For each operation of the basic SW which results in a change of configuration/reconfiguration of the SW.

Validation of the configuration data is similar to a new program validation.

Data validation

Validation of data is divided into two activities:

- Validation of data values;
- Validation of techniques and methods for data storage, modification and use.

In both cases, the validation has to determine whether the data are used correctly and whether they allow to develop SW that will function correctly.

Validation of data values

The idea is to determine the correctness of every data at every moment of the software operation.

There exist different forms, some of which are entirely manually implemented, and other are fully automated.

- Check if the input signals are received in the right places;
- Check whether the input signals comply with the criteria for quality and correctness;
- Check for correctness of the reactions generated by the input signals.

It is a good practice first to perform validation using specially developed test environment, and only after that perform it in the actual environment.

Validation of the data management system

These are validation techniques and methods for data storage, modification and use. They are applicable in all cases, whether there are automated tools for database management or not.

The selection of validation techniques depends on the type of stored information and its usage. For the purpose of validation of program generated simulators, only items that affect the validated operation are checked.

Validation as part of a system

The final validation results from the successful validation of the following elements:

- Evaluation as a part of the system;
- Validation as part of a technological process;
- Product validation.

This is very important in case of a "partial" or "semi-natural" simulator.

The features of validation of a software application as part of an integral system are:

- In this case, in addition to the validation of software, a number of activities to validate the integral system are performed;
- Evaluation of the relation 'Requirements to the requirements − requirements to the software';
- In some cases of application development (using pre-programed software pieces, configuration/ reconfiguration) validation on system level can be performed directly.

III. THE PROGRAM GENERATOR PRGEN

The PrGen program generator is designed to create distributed real-time control systems. It is implemented in many different versions [13][14][15]. It is based on an extended Moore machine implementing specific actions in each node of the state machine. Specific elements of its design are reflecting possibilities to generate both stand-alone and distributed systems. A graph representation of the control algorithms is chosen. The system can be described by its activities. Each activity is a separate thread. After some model and library extensions, the PrGEn started to be used also for simulator generation.

The PrGen program generator is designed to answer the following requirements [14][15]:

- hybrid object/process configuration specification
- an object model facilitating the implementation of open and reconfigurable systems;
- a process model capturing both the reactive and the transformational aspects of system behavior in the context of various types of control systems;
- predictable scheduling of process execution and communication in the context of local and remote subsystems interactions;
- support for modern software engineering techniques such as program generation, formal verification of process and process interaction, etc.

As has been pointed out by Angelov and Ivanov [14] the specification of system reactions and signal transformation is very important for simulator quality and model validation. They can be formally presented as follows:

 ISBN: 978-1-61208-371-1

Specification of system reactions

System reactions can be defined as functions, specifying the output signals, generated in response to certain activating events. $R = \{ r_i \}$, where :

$$r_i : \quad A \times E \times C \rightarrow Y$$

It can be shown that the function r_i is a composite function, i.e., $r_i = o_i \circ p_i$, *where* p_i is a state transition function and o_i is an output function. These are defined below as follows:

- p_i is a state transition function that can be generally defined as: $p_i : A \times E \times C \rightarrow A$

Consequently, $\forall a_m \in A$ we can specify a subset of successor states:

$$Fa_m = \{a_{n_1}, a_{n_2}, ..., a_{n_r}\},$$

and accordingly – a subset of alternative transitions :

$$a_m(t\_) \,\&\, e^m_{n_1}(t) \,\&\, c^m_{n_1}(x(t)) \rightarrow a_{n_1}(t);$$
$$a_m(t\_) \,\&\, e^m_{n_2}(t) \,\&\, c^m_{n_2}(x(t)) \rightarrow a_{n_2}(t);$$
$$\ldots$$
$$a_m(t\_) \,\&\, e^m_{n_r}(t) \,\&\, c^m_{n_r}(x(t)) \rightarrow a_{n_r}(t).$$

This will ultimately result in the construction of the state transition graph of process *P*.

- o_i is an output function, i.e: $o_i: A \rightarrow Y$

Consequently, the reaction function r_i can be specified as a composition of the above two functions:

$$r_i : A \times E \times C \rightarrow Y \Leftrightarrow A \times E \times C \xrightarrow{p_i} A \xrightarrow{o_i} Y$$

Specifically, the above transformation implies that $\forall\, r_i \in R : r_i \leftrightarrow y_k = f(a(t))$, which follows the well known definition of the Moore machine.

Specification of sygnal transformations

Signal transformation functions specify how the output signals are generated within the corresponding system reactions and the associated process states. Specifically,

$$\forall\, y_k \in Y \leftrightarrow s_k \in S \text{ and;}$$
$$s_k : X_k(t) \mapsto y_k(t), \; X_k \subseteq X$$

In one specific case y_k = const. This case is common for a class of discrete controllers, i.e., the so-called state-logic controllers, whereby the controller generates predefined combinations of discrete *on/off* control signals, within various operators (reactions) associated with the corresponding controller states.

In the general case, y_k may be a complex function that can be represented as a composition of simple functions, i.e.:

$$y_k = w^k_l \circ w^k_{l-1} \circ w^k_{l-2} ..... \circ w^k_1 ,$$

where w^i_j are basic application functions that are usually implemented as a library of standard function modules (FM's). The above composition corresponds to a sequence of computations, which can be described by means of two types of models:

(1) Conventional control flow model, such as flowcharts, computation graphs, etc.
(2) Data flow models, e.g., (quasi) analog signal flow diagrams.

The presented mathematical model meets all the formal requirements to it. Formally, an extended Moore machine implementing activities (actions) of analogue, discrete and communication type at any state node can represent most of the industrial systems and objects available today. It is not limited either by numerical transformations or by logical constructions.

IV. VALIDATION OF IMPLEMENTED SIMULATORS

Below, simulators of different objects and validation of their conceptual models and implementations will be presented.

Harbour crane

The simulated harbour crane is of the type shown in Figure 1. It has to be modelled for the following actions:

- A1: Crane with no container, turning to container and lowering hook.
- A2: Crane with container, lifting the container and turning to the ship.
- A3: Lowering the load and positioning the container into the ship.
- A4: Empty crane, from initial position (turned to the port). Turning to the ship, positioning above container, lifting load and unloading it into port.

Each action consists of several different movements in vertical and horizontal directions. Additional limits (dimensions) for horizontal and vertical movements are included.

The motivation to design and implement this simulator is that experiments with the real crane are dangerous and expensive and that the exploitation personnel need a full-time and functional training environment.

Figure 1. Harbor Crane

The crane has four different mechanical movements: turn around vertical axe, grapple lifting/lowering, grapple control and grapple horizontal shifting. Additionally the crane can move along its railway but it is out of scope of the presented simulator. The mechanical construction is independent for each of these movements. They can be started simultaneously but interviewing operators it was found that they run every movement separately. The explanation was that in the other case the crane can lose its stability. Because of this, limit for simultaneous movements, limits for horizontal and vertical speed, etc. have been incorporated in the simulator. A presentation of the MATLAB® models of one of the crane's movements is shown in Figure 2a and Figure 2b.

The simulator includes models for all movements, as well as simulation for operator's interface devices, predictor of the absolute (3D) grapple position and a load simulator.

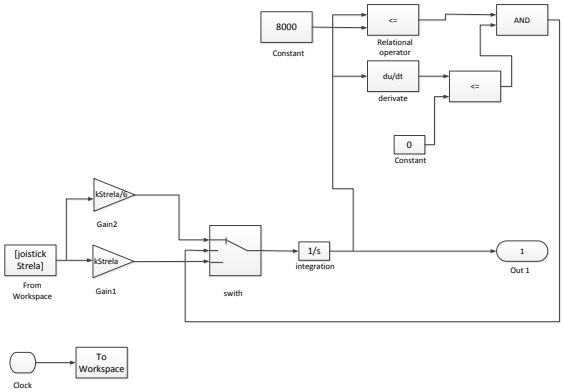

Figure 2a. Harbor Crane grapple non-linear model

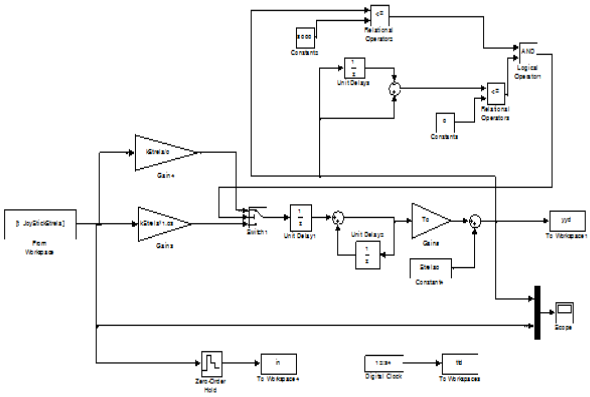

Figure 2b. Harbor Crane grapple discrete model

The motor control is implemented by drive controllers connected to the main controller via Profibus. To make simulation easier Profibus was substituted with MODBUS. The main controller supported both protocols. The simulation of Profibus connections by MODBUS is shown in Figure 3.

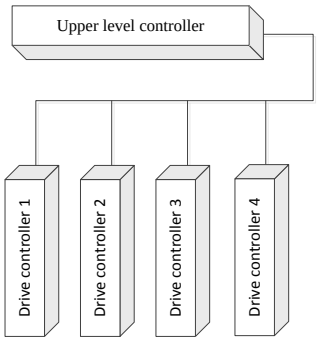

Figure 3. Harbor Crane Profibus/MODBUS model

Validation of the designed simulator envisaged that in the MATLAB® environment after structural and parametric identification the mathematical model and real data are acceptably close. After that a real simulator was implemented. It was developed using the PrGen program generator and its modules library. Real-time execution is done on single-board ARM computer equipped with peripheral devices and communication modules corresponding to the real object. Mathematical transformations are identical to these implemented in MATLAB. Surprisingly, the results from the simulator were very different from the MATLAB results. Detailed data validation was implemented. Finally it was found that the main difference came from the different accuracy in floating-point calculations. After passing incorrect data through non-linearities we received results different by 20% from the original ones. This problem was solved using calculations with extended accuracy. Sensitivity of some non-linear elements was changed as well. Results after simulator tuning for actions A1 to A4 are shown in Table 1.

TABLE 1. COMPARISON BETWEEN CRANE REAL DATA AND SIMULATOR OUTPUTS

N.	Measure-ments Crane (samples)	measure-ments Simulator (samples)	Min. Movement Acc. %	Max. Movement Acc. %	Total Acc. %
A 1	680,000	890,000	95	98	97
A 2	1, 280,000	1,460.000	94	97	95
A 3	1,800,000	1,960,000	95	98	97
A 4	3,640,000	4,000,000	93	97	96

Data validation using plotting is presented in Figure 4 and Figure 5. They show the similarity between real data and simulator outputs. They demonstrate the very high quality of the simulation. This made possible development and fine tuning of the crane control system and it also reduced development costs.

 ISBN: 978-1-61208-371-1

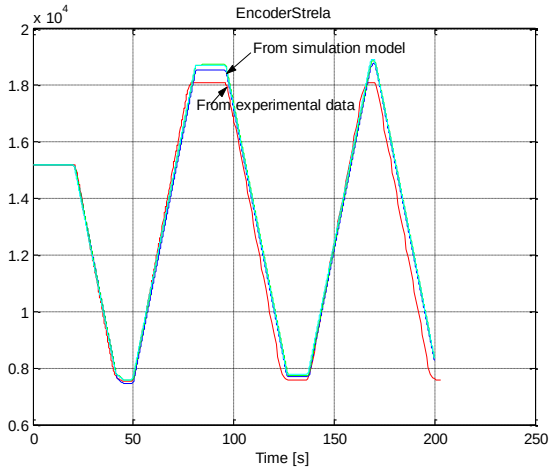

Figure 4. Harbor Crane simulator and experimental data comparison: simulation of grapple horizontal movement;

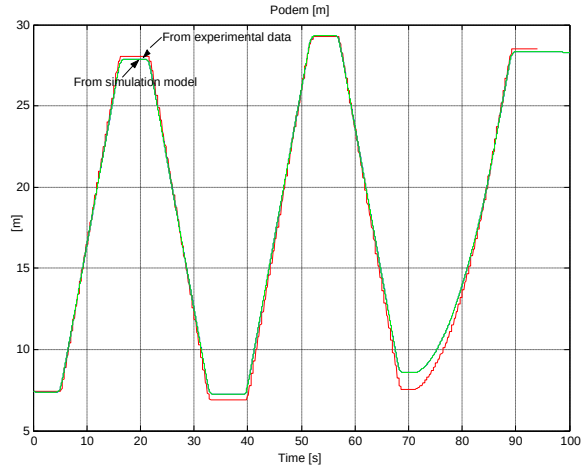

Figure 5. Harbor Crane simulator and experimental data comparison: simulation of grapple vertical movement

Large-scale textile plotter

Similar approach for validation of the quality of simulation was used for a large-scale textile plotter. The motivation to invest time and money in this was the requirements to combine maximal performance and geometrical accuracy.

Printing drawings with a length of 5 or more (up to 20) meters is one of the available solutions to prepare the fabric cutting sketch for real cutting. Several different solutions for this process are known; but one of the best is to draw this sketch on an unbreakable paper sheet with the required length. This is done by a special kind of plotters. Normally, they are of flat-bed type. The simulator of a plotter of that type and its validation will be presented here.

The scheme of the plotter is presented in Figure 6 and Figure 7.

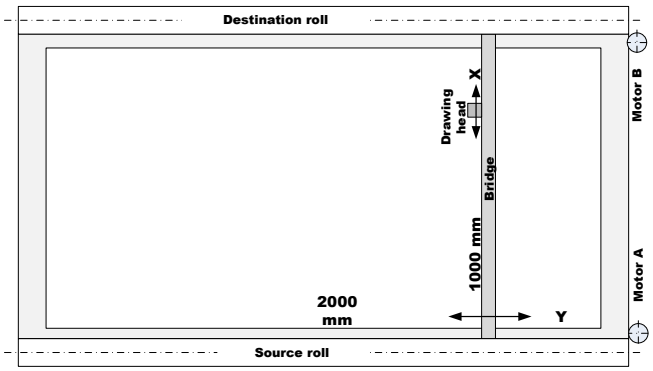

Figure 6. Large-scale textile plotter - top view

Kinematics and wiring schemes of the plotter are shown in Figure 8 and Figure 9. The closed loop kinematics, shown in Figure 8 and Figure 9, generates dynamics problems based on the fact that movements on axis X or Y need both motors to work but movement in direction with $tg\ \alpha = \pm 1$ engages only one of the motors which has to move both the bridge and the head.

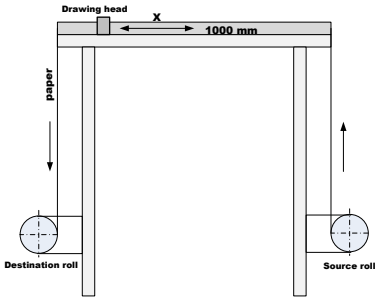

Figure 7. Large-scale textile plotter - side view

Simulation of such a dynamics is rather complicated. The weight of the object to be moved varies 20 times – it is only the head or both head and bridge. According to the angle of movement, every motor has a different load. The load varies all the time when the head has to draw some curve. Identification and modelling of that plotter are out of the scope of this paper. Here, we will discuss the simulator and its validation.

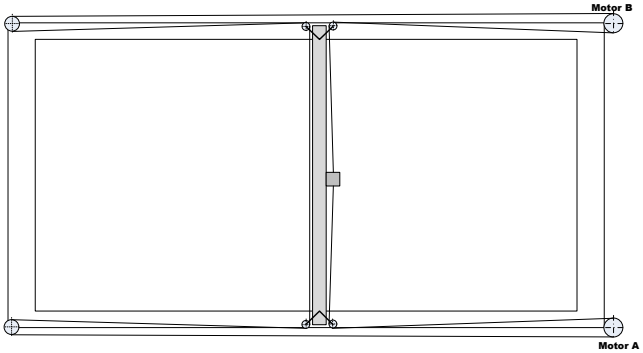

Figure 8. Kinematics scheme

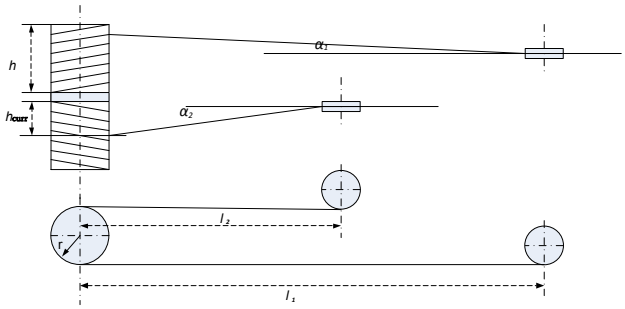

Figure 9. Wiring scheme

The simulation procedure is implemented in two different types of complexity. The first one is a simple computer simulation. The second is a parallel run of the simulator and the real machine. The possibility to run both types of simulation is the reason to avoid usage of MATLAB and SIMULINK for a real-time test-bed. The simulator is implemented on a 32-bit computer with a math co-processor running a real-time operating system and the real-time interpreter of the generated by the PrGen system. For testing purposes, it is equipped with a display having appropriate dimensions and resolution. The communication between the plotter controller and the simulator meets the requirements for Real-Time protocols. It is capable of transferring all the information for simulation and for the system log in a single control time interval. Visualisation and log modules of the simulator draw and save required and actual trajectories, internal parameters, etc.

Validation of the implemented simulator is done using model and data validation. Model validation is done off-line comparing real data and mathematical model outputs. After tuning, the mathematical model is implemented in the simulator. The simulator computational model is tuned again using the same approach. Real data are compared to the simulator outputs. Main differences are found in calculations accuracy. It follows from both calculation orders and numerical accuracy. After re-ordering calculation sequences in the simulator in the same way as they are done in MATLAB and increasing floating point accuracy both MATLAB and real simulator calculations become practically identical.

The implemented simulator was used for two different purposes: 1) to develop and implement new control algorithms to increase plotting accuracy and speed; 2) to analyse problems in drawn sketches, frame stitching, etc.

Using this simulator in simulated time (fast time) made possible to run long-lasting experiments (taking tens of minutes in normal speed) in minutes and thus enabled extensive experiments including full drawing of large sketches.

The simulator validity was proven using data validation on several levels. First, there was model and object output comparison implementing one and the same input sequences. Second, there was geometry accuracy validation. Drawn on paper and calculated by the simulator sizes were compared. All differences had the acceptable tolerance of ±0.25 mm, which is the initial requirement for mechanical accuracy.

V. CONCLUSION

This paper presented the validation of the program generation approach for design and implementation of object simulators and how the quality of the simulator can be increased using validation techniques. The presented results confirm that program generation is a promising way to implement object simulators with hardware and software structures representing the modelled object very closely. Once the program generator internal model is validated it becomes a useful tool for implementation of both simulators and control systems. The validation of the tool and generated by it systems are separated.

Validation of the implemented simulators follows both approaches – model validation and data validation. By implementing sequentially these validation techniques, the designed simulator can achieve required accuracy and functionality.

Using one and the same environment and tool for both controller and simulator building has the advantage to use only one tool for everything. This makes possible to distribute all changes in every building element immediately both in the controller and the simulator.

ACKNOWLEDGMENT

This work is funded partially by the Bulgarian NSF under DRNF02/3 project.

REFERENCES

[1] S. Robinson, Simulation model verification and validation: increasing the users' reliability, Proceedings of the 1997 Winter Simulation Conference ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson.

[2] J.P.C. Kleijenen, Verification and Validation of simulation models, European Journal of Operational Research, vol. 82, 1995, pp. 145-162.

[3] G.W. Silvaggio, Emulated Digital Control System Validation in Nuclear Power Plant Training Simulators, http://scs.org/upload/documents/conferences/powerplantsim/2011/presentations/Session%207/Westinghouse/silvaggioFullPaper.pdf, [last accessed: 08.08.2014].

[4] Weilin Li, X. Zhang, and H. Li, Co-simulation platforms for co-design of networked control systems: An overview, Control Engineering Practice, vol. 23, 2014, pp. 44–56.

[5] O. Balci, 1997. "Verification, Validation and Accreditation of Simulation Models." In Proceedings of the 1997 Winter Simulation Conference, pp. 135-141.

[6] The Rapid Automotive Performance Simulator (RAPTOR), http://www.swri.org/4org/d03/vehsys/advveh/raptor/default.htm [last accessed: 08.08.2014].

[7] M. Pasquier, M. Duoba, and A. Rousseau, Validating Simulation Tools for Vehicle System Studies Using Advanced Control and Testing Procedure, http: //www.autonomie.net/docs/6 - papers/validation/validating_simulation_tools.pdf [last accessed: 08.08.2014].

[8] IAEA, Use of control room simulators for training of nuclear power plant personnel, Vienna, 2004, IAEA-TECDOC-1411, ISBN 92–0–110604–1.

[9] Johnstone M., D. Creighton, and S. Nahavandi, Enabling Industrial Scale Simulation / Emulation Models, In Proceedings of the 2007 Winter Simulation Conference, 2007, pp. 1028-1034.

 ISBN: 978-1-61208-371-1

[10] MathWorks, Generate and verify embedded code for prototyping or production, http://www.mathworks.com/embedded-code-generation/, [last accessed: 08.08.2014].

[11] F. G. Gonzalez, Real-Time Simulation and Control of Large Scale Distributed Discrete Event Systems, Conference on Systems Engineering Research, 2013, Procedia Computer Science 16 (2013), pp. 177 – 186.

[12] IEEE Standard Glossary of Software Engineering Terminology, New York, USA, 1990, ISBN 1-55937-067-X.

[13] N. Baldzhiev, V. Bodurski, and V. Gueorguiev, I. E. Ivanov, Implementation of Objects Simulators and Validators using Program Generation Approach, DESE 2011, Dubai, UAE, December 2011.

[14] C. K. Angelov and I. E. Ivanov, "Formal Specification of Distributed Computer Control Systems (DCCS). Specification of DCCS Subsystems and Subsystem Interactions". Proc. of the International Conference "Automation & Informatics'2001", May 30 - June 2, 2001, Sofia, Bulgaria, vol. 1, pp. 41-48.

[15] C. K. Angelov, I. E. Ivanov, K. L. Perev, and V. E. Georgiev. Design for Open and Predictable Automation Systems. Proc. of the 45th International Scientific Colloquium of the Technical University of Ilmenau, Oct. 2000, Ilmenau, Germany.

[16] I. E. Ivanov and V. Georgiev, "Formal models for system design", Proc. of IEEE spring seminar 27th ISSE, Annual School Lectures, Bulgaria, 2004,vol. 24, pp. 564-568.

[17] D. Harel, "Statecharts: A visual formalism for complex systems" Science of Computer Programming 8 (1987) 231-274.

[18] I. E. Ivanov , "Control Programs Generation Based on Component Specifications", PhD thesis, 2005, Sofia, (in Bulgarian).

[19] C. K. Angelov, I. E. Ivanov, and A. A. Bozhilov. Transparent Real-Time Communication in Distributed Computer Control Systems. Proc. of the International Conference "Automation & Informatics'2000", Oct. 2000, Sofia, Bulgaria, vol. 1, pp. 1-4.

[20] A. Dimov, I. E. Ivanov, and K. Milenkov, "Component-based Approach for Distributed hard Real-time Systems", Information Technologies and Control, 2, 2005.

[21] A. Dimov and I. E. Ivanov, Towards development of adaptive embedded software systems, Proceedings of TU Sofia, vol. 62, book.1, 2012, pp.. 133-140.

[22] D. Knuth, The Art of Computer Programming I, Addison-Wesley,1968

[23] K. D. Tocher, The Art of Simulation, The English Universities Press Ltd., 1963

 ISBN: 978-1-61208-371-1