BRINGING THE MODEL-BASED VERIFICATION OF DISTRIBUTED CONTROL SYSTEMS INTO THE ENGINEERING PRACTICE

V.Vyatkin, H.-M. Hanisch

Martin Luther University of Halle-Wittenberg,
Dept. of Engineering Science,
D-06099 Halle, Germany
phone: +49-(345)-552-5972; fax:+49-(345)-552-7304
Valeriy.Vyatkin @ iw.uni-halle.de

Abstract: In this paper we present the software tool VEDA for modelling and verification of distributed control systems. The tool provides an integrated environment for formal, model-based verification of the execution control of function blocks following the new international standard IEC61499. The modelling is performed in closed-loop way using manually developed models of plants and automatically generated models of controllers.

Keywords: Verification, Distributed Computer Control Systems, Modelling

1. INTRODUCTION

Current practice of testing industrial control systems is far from being satisfactory. Validation which relies on a finite number of tests guarantees only that the developed software correctly works for this finite number of input tests. In reality the software may encounter with input combinations not covered by the tests, thus generating unpredictable outputs. This may lead to the erroneous behavior of the production equipment with the corresponding consequences. Methods of formal model-based verification of control systems developed during the last decade target a solution of this problem. Representative description of the latter can be found in (Clarke *et al.*, 1986; Ostroff, 1989; Kowalewski *et al.*, 2001). However the verification is still far away from the everyday practice due to a number of reasons such as follows:

(1) The control design companies do not want to cast doubt on their practice of software development, insisting on that the settled routine (based on certain norms, rules, and software engineering concepts) ensures the quality of the final product.
(2) Verification requires essential changes to the software engineering practice. In particular, formal modelling of the plants to be controlled is different from the modelling used for simulation and testing.
(3) The variety of developed formalisms makes standardization of modelling difficult.
(4) The verification trials conducted in academia deal with very much simplified examples of control systems. Comprehensive verification of control applications, taking into account not only control logic, but also system issues, is computationally complex and therefore unfeasible.
(5) Despite the number of theoretical and practical works in this direction, there are very few software packages available for smooth integration of verification into the control engineering practice.

Meantime, modern trends in control engineering make problems of testing and validation substantially harder than earlier. State-of-the-art in the control technology is realized by wide application of distributed architectures. If compared with local control systems, the distributed ones consist of controllers interconnected to the environment and to each other by means of networks. Such

design provides an exceptional flexibility of the production equipment, and higher cost-efficiency than provided by traditional methodologies.

The standard IEC61499 (*Function Blocks for Industrial Process Measurement and Control Systems*, 1998) is an attempt to provide a comprehensive software engineering concept for distributed measurement and control applications. In particular it aims at providing a uniform programming paradigm of Programmable Logic Controllers (PLC) and Distributed Control Systems (DCS).

According to the standard an application is built as a net of function blocks interconnected via event and data signals. Control and data flows are clearly separated. Execution control of a single function block is implemented as a state machine called the Execution Control Chart (ECC). It deals only with event and logic inputs and outputs and performs only basic tasks: based on the input events it changes an active state, invokes the algorithms associated to that state, and sets corresponding output events. A block may include a number of algorithms implementing more complex data processing, e.g. functions of continuous control, etc.

In this paper a software tool VEDA (Verification Environment for Distributed Applications) is presented, which is destined to support the verification as a natural part of control engineering. VEDA and some other tools were developed in the framework of research project "Modeling and Verification of Function Blocks following IEC61499". The tool is based on the methodologies of model-based formal verification, developed during the last decade. A short discussion of the latter forestalls the description of VEDA's functionality.

2. AN EXAMPLE OF THE DISTRIBUTED CONTROL APPLICATION FOLLOWING IEC61499

Let us illustrate the impact of the distributed design with the help of a simple plant "BORING STATION" as presented in Figure 1. It consists of a boring machine (drill) and a transfer stage, which delivers workpieces to the home position of the drill. The loading/unloading of the transfer stage is performed in the position detected by sensor *load.pos.*, opposite to the home position. The drill has to start drilling when the workpiece comes to the home position. When drilling is over, the workpiece is moved away. The presence of the workpiece on the tray is reported by the sensor *loaded*.

Control of the plant is performed by two independent controllers, one for the drill, and the other for the transfer stage. Sensors and actuators are connected to their respective controllers by segments of networks. The controllers also communicate to each other via network. Access to the data is

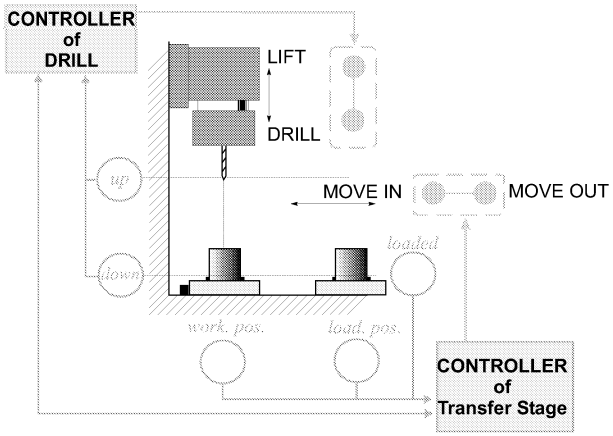

Fig. 1. Structure of the distributed control system.

limited for each controller by the data available in its own network segments, plus the data explicitly provided by the other controller. According to IEC61499, a function block consists of head (the upper part) and body (the lower part). The head is connected to the event inputs and outputs and is responsible for the execution logic. The body is connected to the data input/outputs and contains the data processing algorithms, which are called by the execution control. A block diagram (following IEC61499) of the distributed control system of the plant is presented in Figure 2. Function block TS_CTL implements the control logic of the transfer stage, block DR_CTL controls the drill. In our diagram block MOD represents either plant or its model, and the block $INIT$ models initialization of the controllers (for example in case of a start-up of the control device).

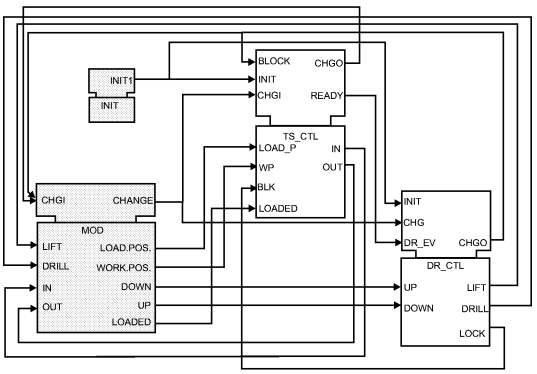

Fig. 2. Block diagram of the control system following IEC61499.

The internal control logic of the blocks TS_CTL and DR_CTL is implemented by means of Execution Control Charts (ECC) presented in Figure 4. The ECCs are state-machines whose syntax is a simplified version of the Sequential Function Chart (SFC), which is a programming language defined by the standard IEC61131-3 (*International Standard IEC 1131-3, Programmable Controllers - Part 3*, 1993). For simplicity we extended syntax of ECC, allowing simple algorithms, such as an assignment of a variable, to be presented directly at the places, reserved for the algorithm calls. The control logic of drill and

transfer stage is quite simple: once loaded with a workpiece the transfer stage delivers it to the working position, and then reports to the controller of the drill about that. The latter blocks moving of the transfer stage during the drilling, by issuing the corresponding signal LOCK. As a result of the design step, the source code of the application (network of interconnected function blocks) is obtained. In general it includes the component algorithms, and is presented in Extended Structured Text format.

3. MODELLING ESSENTIALS

For the formal model-checking the interconnected system has to be substituted by its finite-state model. We use for this purpose A signal-Net Systems (SNS) that is a formalism, proposed in (Rausch and Hanisch., 1995) (originally called Net Condition/Event Systems) as a derivative of Petri nets (Petri, 1980), and Condition/Event systems (R.S.Sreenivas and B.H.Krogh, 1991*a*; R.S.Sreenivas and B.H.Krogh, 1991*b*). It has been developed and successfully applied for modelling, verification and synthesis of controllers and control systems of various types (Hanisch and Lüder, 1999; Hanisch *et al.*, 1997; Vyatkin and Hanisch, 1999; H.-M.Hanisch *et al.*, 2000).

Signal-Net system is a marked place transition net, having in addition to the token-flow arcs, two additional types of *signal* arcs. Condition arcs from places to transitions provide additional enableness condition. Event arcs from transitions to transitions provide one-sided synchronization: firing of the source transition forces firing of the recipient, if the latter is enabled by marking and conditions.

Thanks to the signal arcs the nets can be organized into modules with condition and event inputs/outputs, and the modules also can be interconnected by condition and event arcs. The interconnected modular model can be transformed into *autonomous* net without inputs and outputs. The original structure of the plant/controller system is preserved in the SNS model.

An example of the SNS model for the drilling station is given in Figure 3. Every component of the system is modeled by an SNS module. Interconnections between component blocks of the system are mapped to condition and event arcs connecting modules of the model.

Modelling by means of SNS proves to be beneficial as opposed to other similar means of modelling (e.g. finite automata) particularly in the following aspects:

(1) Modularity. SNS includes means of organizing modular models: event and condition arcs. These new types of arcs help to combine strong features of Petri nets with structural clarity of Condition/Event systems (R.S.Sreenivas and B.H.Krogh, 1991*a*).

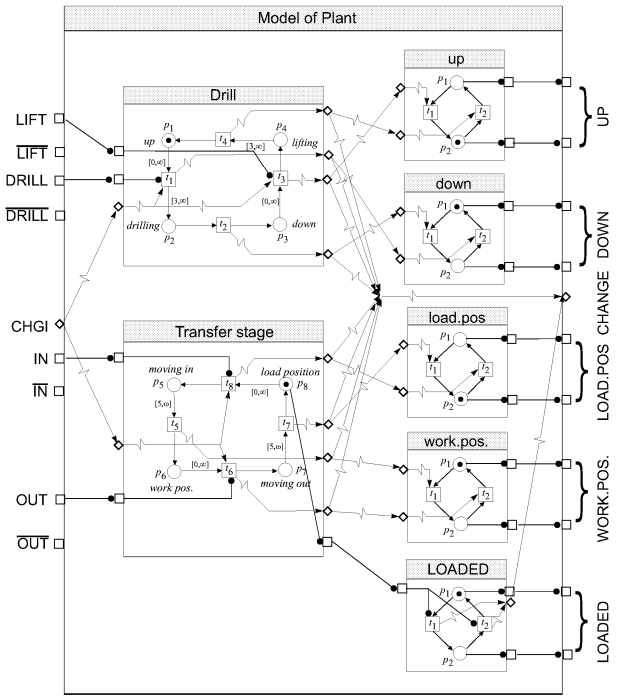

Fig. 3. Modular SNS model of the plant.

(2) Clarity. The semantics extension induced by the signal arcs is simpler and has less impact on the complexity of the model-checking than that of other modular formalisms, such as Input/Output automata.

(3) Brevity. State of the SNS model, defined by marking of places, as well as state transition, defined by a combination of net transitions are distributed. This allows to build more compact models than by using other similar implementations of C/E systems, such as Condition/Event Automata (H.Chen and H.M.Hanisch, 2000). It is especially sensible when modelling of controller's internals is concerned.

(4) Adequacy. Modelling is always a trade-off between the complexity and adequacy of representation the dynamics of the modeled system. In this sense the SNS demonstrate quite satisfactory performance. In particular it well copes with representation of both synchronous (controller) and asynchronous (plant) components in the same model.

(5) Implementability. Discrete state formalisms obviously have limited expressive power than the hybrid ones, such as presented in (H.-M.Hanisch *et al.*, 2001; Kowalewski *et al.*, 2001). This is true also for SNS. However, currently existing implementations of SNS model-checking, such as SESA (*SESA - Signal-Net System Analyzer*, n.d.), allow the model-checking of realistic-scale applications, having millions of reachable states, that is hardly to expect from the hybrid formalisms.

The models of the control function blocks are generated automatically by VEDA given their source code. The methodology of the modelling

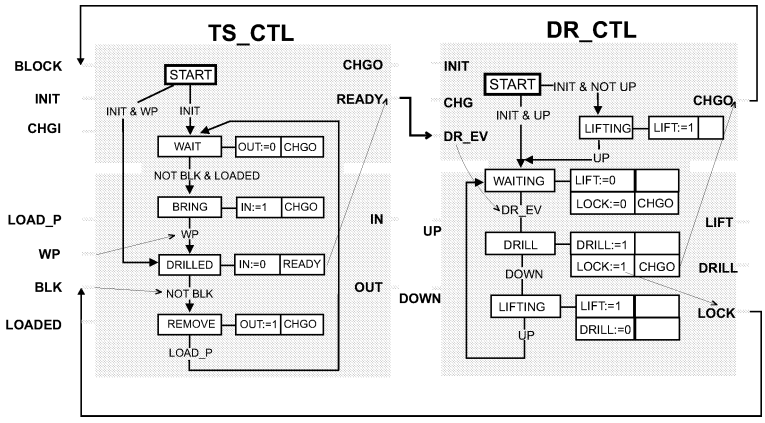

Fig. 4. Execution Control Charts of function blocks TS_CTL and DR_CTL.

is discussed in our prior works (Vyatkin and Hanisch, 2000*b*; Vyatkin and Hanisch, 2000*a*). Taking into account the distributed nature of the systems being studied we assume that an application may consist of several components interconnected by data and event signals. Each component can be responsible for control of a particular unit of the plant.

4. VERIFICATION TOOLS

The overall correctness of control applications following IEC61499 depends not only on the control logic and data processing within the algorithms, but also on the scheduling of the algorithms, communication between devices, particular architecture of the system and mapping components of the application to particular architecture.

Full verification of an IEC61499 application requires application of hybrid modelling formalisms, and nevertheless (as it is shown, for example, in (Enste and U.Epple, 2001)) it seems to be unfeasible for the realistic size designs. It does not mean, however, that the verification is unable to contribute to the improvement of the testing of function blocks.

Due to the complexity-related reasons we are not able at this stage of the work to perform the complete verification of industrial-scale function blocks. For this reason we constrain ourselves to verification of the execution control only. We rely on the following argumentation:

(1) Verification of the execution control sets a corner stone, which can be further upgraded to the full-scale verification.
(2) Even the (quite limited) verification of the execution control is able to extend the borders of testing and debugging, and contribute essentially to the improvement of the reliability of controllers.

Below we present the integrated software tool VEDA - Verification Environment for Distributed Applications. Given a controller, coded according to IEC61499 the following steps are assumed as the preceding to VEDA application:

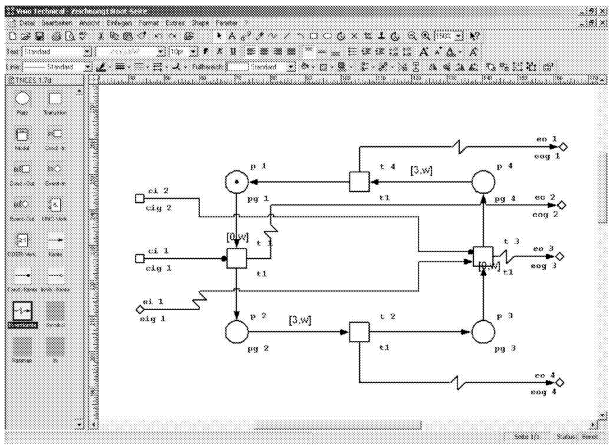

Fig. 5. Graphical editor of SNS models.

(1) Develop the SNS model of plant and integrate it with the control application, prepared following the IEC61499. The model design is supported by means of the graphical editor as presented in Figure 5. The formal model can be appended by a visualization model, which assigns visualization actions to certain markings of the formal model. For this purpose we developed an object-oriented description language SML (Simple Modelling Language). The resulting composite model can be presented as a function block with interface following IEC61499. Taking into account the distributive nature of the plant, the model can be also a net of component function blocks. Thus, the block MOD in Figure 2 can be substituted by the model from Figure 3. The arcs linking the block MOD to other function blocks are substituted by condition and event arcs connecting the model of MOD with the models of respective blocks.
(2) Present the to-be-verified control application in the closed-loop manner, where the function blocks containing controllers are interconnected with the function blocks containing the models of plant's components (as in Figure 2).
(3) Formalize the specifications of correct or incorrect behavior of the control system. These

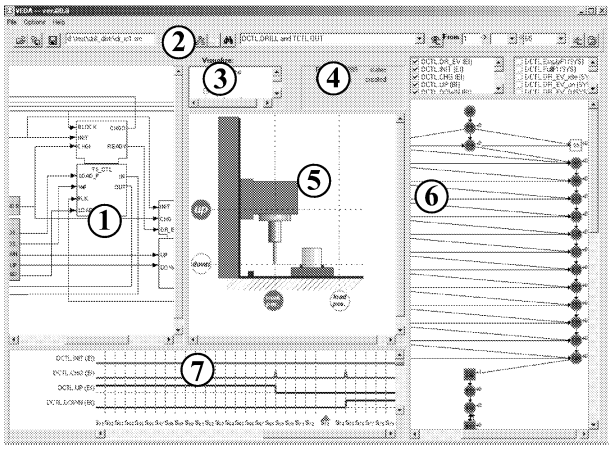

Fig. 6. VEDA's display.

can be done using second order predicates or temporal logic formulae over the variables declared in the controllers or over parameters of the model of plant.

The main VEDA screen is shown in Figure 6. Its structure is as follows:

(1) **Application/model source view.** Displays the application being verified in one of the available graphical or textual formats. For example, the IEC61499 overall structure can be presented as a net of interconnected component function blocks, as well as equivalent ASCII text presentation. The same applies to every component function block. Syntax of function blocks is extended to allow a block to contain a formal SNS model (such blocks are gray shaded on the screen).

(2) **Modelling/model-checking controls** implement the following functions: create the reachability space of the model, search for the states satisfying certain logic conditions, or more sophisticated navigation functions in the reachability space, e.g. search for a trajectory satisfying to a number of conditions, expressed either as a sequence of (second-order) predicates, or as a temporal logic formula.

(3) **Tree view** shows hierarchical structure of the application and the variety of presentation options, which can be applied to each component. E.g. a block containing the formal model of plant can be seen as its interface (inputs and outputs), as well as the textual or graphical form of the SNS model (in window 1). A "pure" IEC61499 basic function block is visible as its interface, structured text of its content, or graphical view of its execution control chart. The graphical views can display information about particular state of the model (marking of places, etc), or state of the execution control.

(4) **Model/modelchecking status.** Shows the size of the generated model, size of the reachability space, and results of search for a particular state.

(5) **Process visualization display** shows the modeled plant in the state selected in the reachability space. For the states having non-zero time duration, process animation can be displayed.

(6) Behavior of the interconnected model is represented by means of its **reachability graph**, where nodes correspond to the states, and arcs represent transitions between the states. Thus, a path (a trajectory) in the reachability graph corresponds to a particular scenario in the behavior of the model. The specified path is highlighted in the graph.

(7) For detailed analysis of the system's behavior along the highlighted trajectory it can be represented by means of **asynchronous timing diagrams**, i.e. values of the requested inputs, outputs or internal variables can be shown graphically with respect to the states of the trajectory. In addition, each state can be visualized in animation window (5), and in the source window (1).

5. EXAMPLE OF VERIFICATION SESSION

Let us illustrate application of VEDA to the control system of the drilling station. Formulation of specifications is certainly a human activity, but usually technical documentation provides a lot of raw material for the specifications. Specification of correct behavior of the plant includes freedom from dangerous situations. One of the dangerous situations can be described as "Attempt to move the transfer stage during drilling". This condition can be represented as a predicate in terms of input/output variables of the blocks as follows: "DCTL.LIFT and TCTL.OUT". VEDA checked the validity of the predicate in the reachability space of the closed-loop model (289 states) and found the states where the condition holds. To analyze reasons of the incorrect behavior, it is possible to visualize the trajectories leading to this state with signal diagrams of the variables, and provide the view of the animated process visualization display along it. The reason becomes clear at the look at the signal diagram in Figure 7 and to the corresponding visualization of ECC (see Figure 4) and process view.

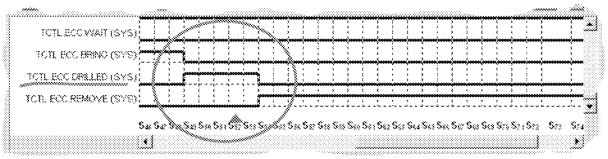

Fig. 7. Timing diagram of the state activity variables: state DRILLED of the transfer stage is transient.

Once the transfer stage arrives to the working position, its controller has to come to the state DRILLED and send a corresponding message READY to the controller of drill. The latter issues blocking condition BLK, which does not allow the transfer stage to move away before the drilling is over. In fact, the state DRILLED in the controller of the transfer stage is unstable: the condition

on the transition to the next state REMOVE is immediately TRUE, due to the logic of ECC execution: the outgoing transition conditions are evaluated even before the output events, related to a particular state are issued. This implies that the state of TCTL is changed to the REMOVE even before the signal READY to DCTL has been issued. To fix this error the transition condition NOT BLK has to be fortified as: BLOCK AND NOT BLK, where BLOCK is an event issued by DCTL after setting the BLK to 1. After the modification, the erroneous states disappear from the reachability space.

The repetitive application of such a procedure with various specifications helps to improve the controller's correctness.

6. ACKNOWLEDGMENTS

The work is supported by the Deutsche Forschungsgemeinschaft under reference Ha 1886/10-1.

We express our gratitude to Dmitri Zagraevski for valuable comments and fruitful discussions of this paper.

7. REFERENCES

Clarke, E., E.A. Emerson and A.P. Sista (1986). Automatic verification of finite state concurrent systems using temporal logic. *ACM Trans. on Programming Languages and Systems* (8), 244–263.

Enste, U. and U.Epple (2001). Technical application of hybrid modeling methods to specify function block systems. *Automatisierungstechnik* **2**, 52–59.

Function Blocks for Industrial Process Measurement and Control Systems (1998). International Electrotechnical Commission, Tech.Comm. 65, Working group 6. Geneva.

H.-M.Hanisch, K.Lautenbach, C.Simon and J.Thieme (2001). Modeling and validation of hybrid systems using extended timestamp nets. *Automatisierungstechnik* **2**, 60–65.

H.-M.Hanisch, T.Pannier, D.Peter, S.Roch and P.Starke (2000). Modeling and verification of a modular lever crossing controller design. *Automatisierungstechnik*.

Hanisch, H.-M. and A. Lüder (1999). Modular modeling of closed-loop systems. In: *Proc. of Colloquium on Petri Net Technologies for Modeling Communication Based Systems*. Berlin, Germany. pp. 103–126.

Hanisch, H.-M., J. Thieme, A. Lüder and O. Wienhold (1997). Modeling of PLC behavior by means of timed net condition/event systems. In: *International conference on Emerging Technologies and Factory Automation (ETFA'97)*. Los Angeles, USA.

H.Chen and H.M.Hanisch (2000). Model aggregation for hierarchical control synthesis of discrete event systems. In: *Proc. Of Conference on Decision and Control (CDC)*.

International Standard IEC 1131-3, Programmable Controllers - Part 3 (1993). International Electrotechnical Commission. Geneva, Suisse.

Ostroff, J.S. (1989). *Temporal Logic for real-time systems*. Wiley. London.

Petri, C.A. (1980). Concurrency, net and applications. *LNCS* **84**, 251–260.

Rausch, M. and H.-M. Hanisch. (1995). Net condition/event systems with multiple condition outputs. In: *Symposium on Emerging Technologies and Factory Automation*. Vol. 1. INRIA/IEEE. Paris, France. pp. 592–600.

R.S.Sreenivas and B.H.Krogh (1991*a*). On condition/event systems with discrete state realizations. *Discrete Event Dynamic Systems: Theory and Applications* **2**(1), 209–236.

R.S.Sreenivas and B.H.Krogh (1991*b*). Petri net based models for condition/event systems. In: *Proc. of American Control Conference*. Vol. 3. Boston, USA. pp. 2899–2904.

SESA - Signal-Net System Analyzer (n.d.). http://www.informatik.hu-berlin.de/lehrstuehle/automaten/tools/.

S.Kowalewski, P.Herrmann, S.Engell, R.Huuk, H.Krumm, Y.Lakhnech, B.Lukoschus and H.Treseler (2001). Approaches to the formal verification of hybrid systems. *Automatisierungstechnik* **2**, 66–73.

Vyatkin, V. and H.-M. Hanisch (1999). A modeling approach for verification of IEC1499 function blocks using net condition/event systems. In: *Proceedings of the ETFA'99 Workshop*. Barcelona, Spain. pp. 261–270.

Vyatkin, V. and H.-M. Hanisch (2000*a*). Development of adequate formalisms for verification of IEC 1499 distributed applications. In: *39th conference of Society of Instrument and Control Engineers (SICE) of Japan*. Iizuka, Japan.

Vyatkin, V. and H.-M. Hanisch (2000*b*). Modeling of IEC 61499 function blocks as a clue to their verification. In: *XI Workshop on Supervising and Diagnostics of Machining Systems*. Karpacz, Poland.