IEC61499 as an Architectural Framework for Integration of Formal Models and Methods in Practical Control Engineering

Valeriy Vyatkin
Valeriy.Vyatkin@iw.uni-halle.de

Hans-Michael Hanisch
Hans-Michael.Hanisch@iw.uni-halle.de

Sirko Karras
Sirko.Karras@iw.uni-halle.de

Xiujun Cai
Cai.Xiujun@iw.uni-halle.de

Martin Luther University Halle-Wittenberg, Department of Engineering Sciences, Automation Technology Lab., 06110, Halle,

1. Introduction

Modeling is extensively used in research work on intelligent manufacturing to solve such problems as optimal dynamic reconfiguration of production facilities, or optimization of material flow and throughput. Also, models are used for avoidance of possible deadlocks, or for prevention of malfunctions caused by incorrectness in control algorithms. As the components of automation systems are getting more autonomous, the more embedded and reusable must be the models.

Models may give opportunities to simulate the system's behavior, to analyze system's structure (such as connectivity of certain elements), or to analyze properties of system's dynamic behavior even without conducting the simulation. A production plan may impose a branching structure of the control with multiple scenarios of automation system's behavior. However, even a purely sequential control may lead to such behavior due to malfunctions in some hardware elements. In both cases the formal analysis might be the only way to reveal potentially dangerous situations. This sort of analysis requires modeling of closed-loop plant/controller systems, where the model encapsulates properties of the plant, the controller as well as the structure of the system.

Modeling can be especially beneficial for flexible, re-configurable automation systems. In traditional hierarchical control systems, re-configuration requires to re-design controllers, sometimes in a very sophisticated way. Then the new functionality must be tested to ensure the desired behavior in the closed loop, and eventually the new functionality must be encoded in a language used for controller programming. In the same manner process monitoring, diagnosis, and supervision by the operator have to be re-designed with similar levels of difficulty. All these tasks have to be accomplished quickly to minimize lost production. A faster system integration, however, could be achieved if the testing routines were substituted by formal validation of newly arisen system configurations.

Unfortunately, modeling still is not a part of regular activity in engineering of industrial automation systems. Development and application of models are rarely done in a systematic way. Usually the models are developed for a particular modeling task, e.g. parameter estimation, productivity optimization, prediction of interaction results, formal validation of controllers, etc. The occasional application, however, reduces the prospective benefits and increases the costs of model development.

One of the reasons discouraging control engineers from embedding modeling in their projects is the absence of a unifying architectural framework, which would integrate the control, service and modeling software in a single and consistent way, and would be equally good as for efficient implementation, as well as for the analysis. In this paper we attempt to show the integration of modeling into an open, vendor- and network-independent architecture based on strict compliance with the new upcoming global standard IEC 61499 [1,2] for the use of function blocks in distributed industrial control and automation systems. The IEC61499 represents the missing consistent architectural framework to integrate run-time control and diagnosis applications, simulation and formal analysis of agile distributed automation systems.

This paper provides an example of using IEC61499 to combine systematic system engineering of distributed automation systems with modeling and formal verification of such systems.

2. Encapsulation of Modeling in Component Architecture

Many of modern production systems are built as a composition of automatically controlled machines produced by different vendors. Integration of single machines into production systems requires corresponding integration of control software modules into decentralized controllers for the whole system.

Attempts to cope with the arising problems have led to the concept of component software design where the proprietary software of constituent machines is encapsulated into so called “software components”. An example of practical development in this direction is the project *PROFInet* of the Profibus User Organization (PNO) [3] which targets the integration of existing compliant control hardware (Programmable Logic Controllers) and software (following the standard IEC61131-3) of different vendors into a single system, providing data exchange between the components.

Further requirements to the component-based system engineering may include the means of hardware-independent design of the components and encapsulation of versatile functionalities, such as interface abstractions, controllers, simulation models, visualization functions, and formal models for verification. These requirements can be fulfilled by means of compliance with IEC61499.

The IEC61499 architecture creates the following prerequisites for embedded modeling:

1. Event-driven function blocks guarantee the independence of execution semantics from the particular implementation devices where the blocks are executed. This, in turn, creates necessary prerequisites for practical application of the MVCDA architecture for encapsulating versatile functionalities of control systems into software components.
2. Formal modeling elements are incorporated to the description of components: basic function blocks have execution control charts, service interface blocks may have their behavior documented by service primitives following ISO TR 8509.

3. The IEC61499 formally defines the critical algorithms of implementation such as input event processing, or algorithms dispatching, etc.

The ideas of system engineering in IEC61499 will be illustrated on the small example of “DRILLING STATION” described as follows.

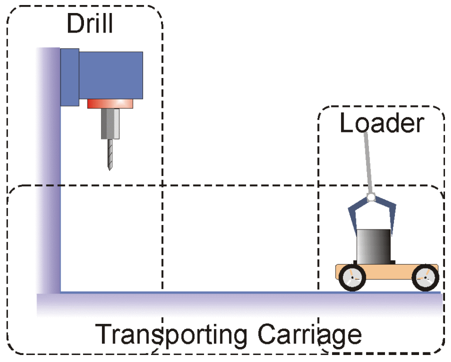

Figure 1. Structure of the production cell: a processing unit (drill), a transportation unit (carriage), and a logistics unit (loader).

The station consists of a boring machine (*drill*) and a *carriage*, which delivers workpieces to the home position of the drill. The loading/unloading of the carriage is performed by the *loader* in the *loading* position, that is opposite to the home position. It is assumed that all the constituent units show a big deal of autonomous behavior that allows their integration virtually without any “master controller”.

Despite its fairly primitive nature, this sample object allows illustration of various phenomena arising in component-based industrial systems, e.g. concurrent operations in different components, or impacts of reconfigurations, such as substitution of a component by an almost functionally equivalent one, having slight differences in interfaces, dynamic properties, etc.

Component-based system engineering assumes that the hardware tools are accompanied by software components, which encapsulate the relevant functionalities. The software engineering then follows the engineering of the system in the sense that each tool in the real system is represented by the corresponding component in the software, while the information interconnection of the components generally follows the relations of the parts in the real system.

In terms of IEC61499, the component may be represented as a composite function block and the functioning of the integral system as an application formed from the function blocks interconnected via events and data.

With respect to the sample drilling station it may result in the structure shown in Figure 2. The block diagram shows the data and message connections of the components. The application supports both automatic and manual modes of control.

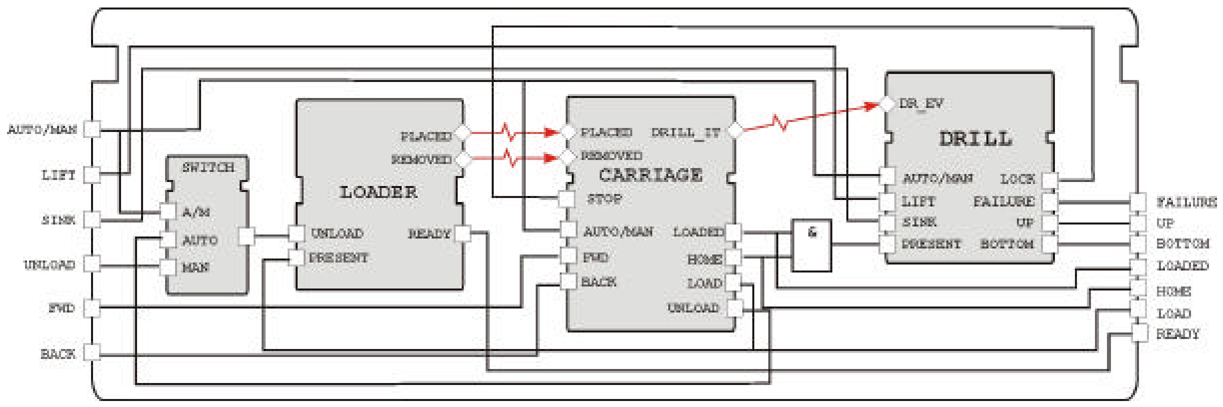

Figure 2. Component representation of the functionality of “Drilling station”.

Further implementation steps may include:

- Definition of the architecture of the control system as a set of network-connected control devices and resources such as intelligent sensors/actuators, computational resources, fieldbus segments, or visualization panels. IEC61499 provides the necessary architectural means for definition of these types.

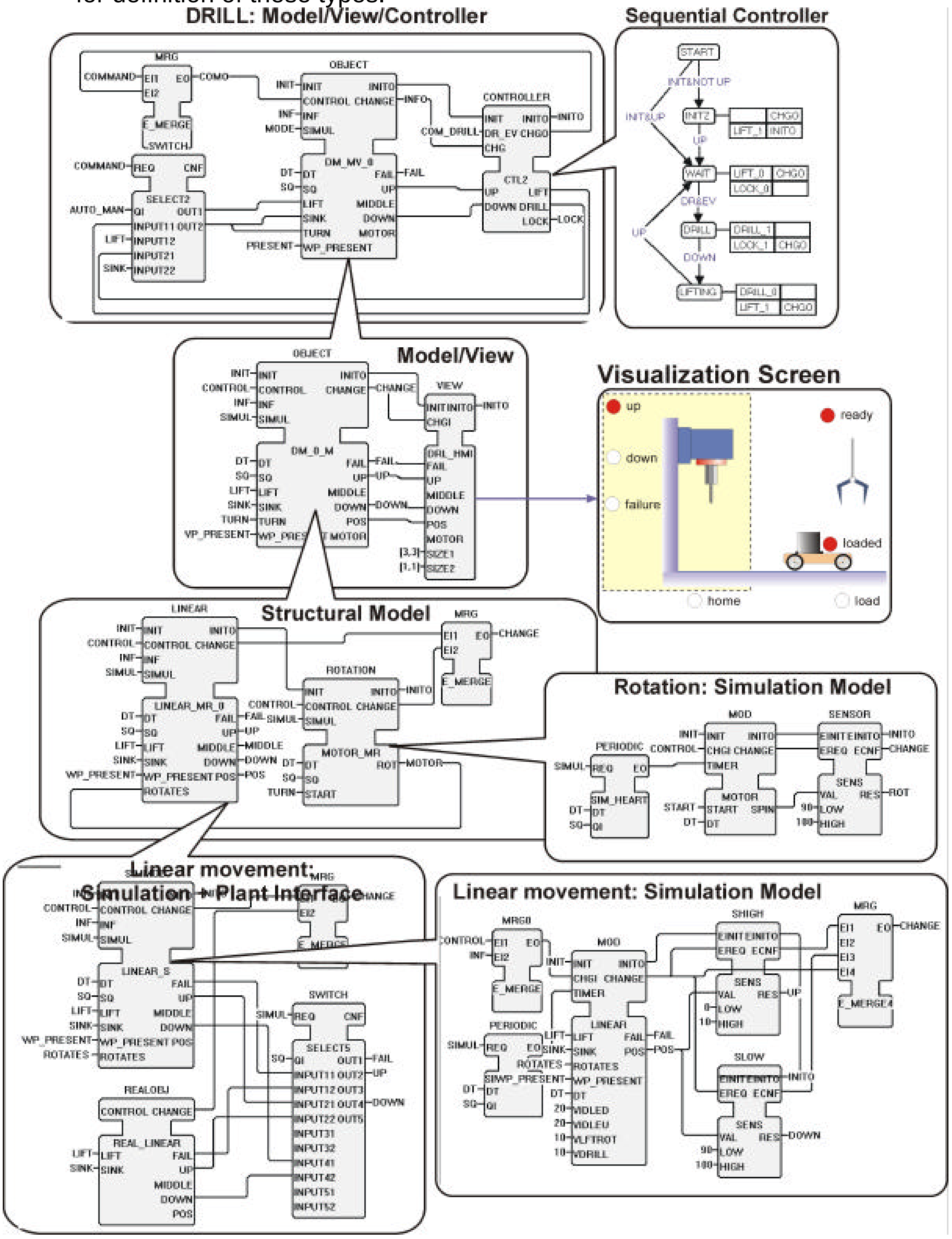
DRILL: Model/View/Controller
Sequential Controller
Model/View
Visualization Screen
Structural Model
Rotation: Simulation Model
Linear movement: Simulation + Plant Interface
Linear movement: Simulation Model
up
down
failure
ready
loaded
home
load
START
WAIT
DRILL
LIFTING

Figure 3. Hierarchical structure of the "Drill" component in MVC design pattern.

- Mapping of the application onto the architecture, i.e. assign the constituent function blocks to the resources, where they will be executed. Thus, the executable *system configuration* is obtained that can be uploaded to the physical devices and executed. At this step the communication functions shall be added to the distributed parts of application in order to provide seamless data and event flow exchange.

Thanks to the opportunities opened by IEC61499, the control application can be designed following the object-oriented Model/View/Controller (MVC) pattern [4,5] that logically integrates closed-loop plant-controller models into system design. Thus, modeling can be embedded into software components within the same formal framework. Application of the MVC pattern is illustrated in Figure 3 on example of the “DRILL” component, whose hierarchical structure is further discussed.

Level 1: Model/View/Controller

The component DRILL is constituted from the blocks OBJECT and CONTROLLER interconnected in closed loop to each other, and also connected to the inputs and outputs of the component.

The block OBJECT of type DM_MV_0 (Drill with Motor Model and View, Type 0) represents the functionality of the equipment, while the block CONTROLLER encapsulates the control logic. The execution modes include the MANUAL and the AUTOMATIC mode, determined by the AUTO_MAN qualifier. If the qualifier is TRUE (automatic mode) then the OBJECT block receives the control commands (LIFT, SINK, TURN) from the CONTROLLER. Otherwise these signals are taken from the inputs of the block itself, which can be connected to manual control buttons.

Level 2: Model/View

The block VIEW is responsible for displaying of the image of drill on the operator station screen. At every event CHGI the outlined part of the display as shown in Figure 3, is refreshed given the values received from the OBJECT block. The location of the drill’s head is displayed according to the coordinate POS. The block OBJECT of type DM_0 (stands for Drill with Motor, Type 0) represents the drill itself. Its interface almost copies the interface of DM_MV: all commands and data coming from the controller are directly transferred to it.

Level 2:Controller

The sequential control of the drill is defined in the form of sequential function chart and encapsulated in the block CONTROLLER of type CTL2.

Level 3:Structural Model

This level represents the structure of the drill composed from two sub-models: a model of the drill’s head as a vertically moving object, and a model of spindle’s rotation. The model reflects the fact of relative independence of the components: the axis position of the head has no influence on its rotation. The rotation status of the motor, however, influences the results of drilling. For this reason the ROT (rotation) output of the ROTATION (Motor) block is connected to the ROTATES input of the model of the head. The blocks LINEAR (of type LINEAR_S_0) and ROTATION (of type MOTOR_S) encapsulate the functionality of real component units of the object: they receive control inputs and generate the output parameters such as axis position of the head and turning speed of the spin of the motor. They also produce the values of Boolean and analog sensors, e.g. the position sensors UP and DOWN.

Level 4:Model of a Single Unit

The next level of the component hierarchy is represented by single functional units of the equipment, such as vertically moving head and rotation motor of the drill. These components can be either further defined by means of dynamic models and models of the corresponding sensors, or they can be substituted by direct interfaces to real devices.

The model and the interface to the real process are combined within one function block LINEAR_MR_0 (Model + Real object). The event input SIMUL with qualifier SQ controls the way of the outputs assignment: if SQ=TRUE then the SWITCH relays outputs of the simulation model (SIMMOD). Otherwise, if SQ=FALSE, the outputs are taken from the block REALOBJ serving as an interface to the actual DRILL.

Level 5:Simulation Model

The central part of the simulation model is block MOD (of type LINEAR) that encapsulates a discrete implementation of the dynamic model of the vertically moving drill's head. Internals of this block are discussed in the next section. The state of the model is re-evaluated at every event TIMER. These events are generated by the block PERIODIC with frequency defined by the time discretization parameter DT as long as the simulation qualifier SQ is TRUE. The model produces the numeric parameter POS (in the interval from 0 to 100) indicating the vertical position of the head. Blocks SHIGH, SLOW of type SENS represent the discrete sensors indicating correspondingly up and low positions of the head. The LOW and HIGH parameters of the SENS block represent the threshold interval in which the numeric input value VAL must fall in order to the logic output RES to be produced.

3. Modeling the Dynamic and Logic of Processes

As shown in Figure 4, the drill's head moves vertically within coordinate variation limits 0 and 100. The higher edge of the workpiece is assumed to have vertical coordinate 50.

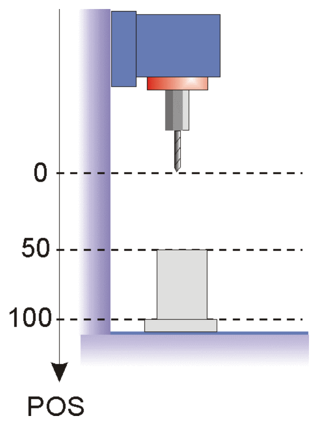

Figure 4. Drill's vertical movement axis.

Even primitive dynamic processes such as linear movement of drill's head cannot be efficiently described by pure mathematical equations in presence of logic control signals. The model has to be hybrid, i.e. include both mathematical definition of the coordinate change, along with the logic model of state switching. Direct implementation of such modeling is possible by means of Execution Control Charts of IEC61499 basic function blocks. The block's interface reflects the fact, that usually the control actions are transmitted to the plant by Boolean signals (LIFT, SINK in this model). The model also needs some information about the external environment: the condition PRESENT stands for the workpiece status, and ROTATES informs the model about the spinning status of the spindle.

Depending on the values of these two conditions, the linear moving may have different speed in the lower part of the moving interval, e.g.: the drill cannot move down if the spindle does not rotate, but the workpiece is present. On the other hand, if no workpiece is present, rotation of the spindle does not influence vertical movement.

The model delivers two output values. The numeric output POS represents the vertical coordinate of the drill's head, and the logic value FAILURE is an integral condition representing all sorts of incorrect or failure situations.

The state chart (ECC) model of the linear progress of the drill is shown in Figure 5. The state chart is built from states (rectangular shapes) and state transitions (arcs) marked with Boolean conditions. In the chart are two types of states, namely: fixed position states UP_POS (POS=0), MID_POS (POS=50), DOWN_POS (POS=100) and dynamic states (MOVE_UP, MOVE_DOWN, etc.) with linear change of parameter POS as $POS=POS_{old}+kdt$, where the coefficient *k* is the speed of moving, *dt* is the time increment and POS_{old} is the previously calculated value of the parameter.

The model describes the uncontrolled behavior as follows. The spindle moves freely in the upper part of the axis, no matter whether the workpiece present or not. When the middle position is reached and the control signal SINK remains ON, the spindle continues its moving downwards. Should the workpiece be in the home position, and the bore spins, then normal drilling goes on. If the drill does not rotate, then it just hits the blank workpiece, and a failure occurs. If no workpiece is present, then the drill moves down idle, with the speed higher than that of drilling. The same applies to the moving upwards.

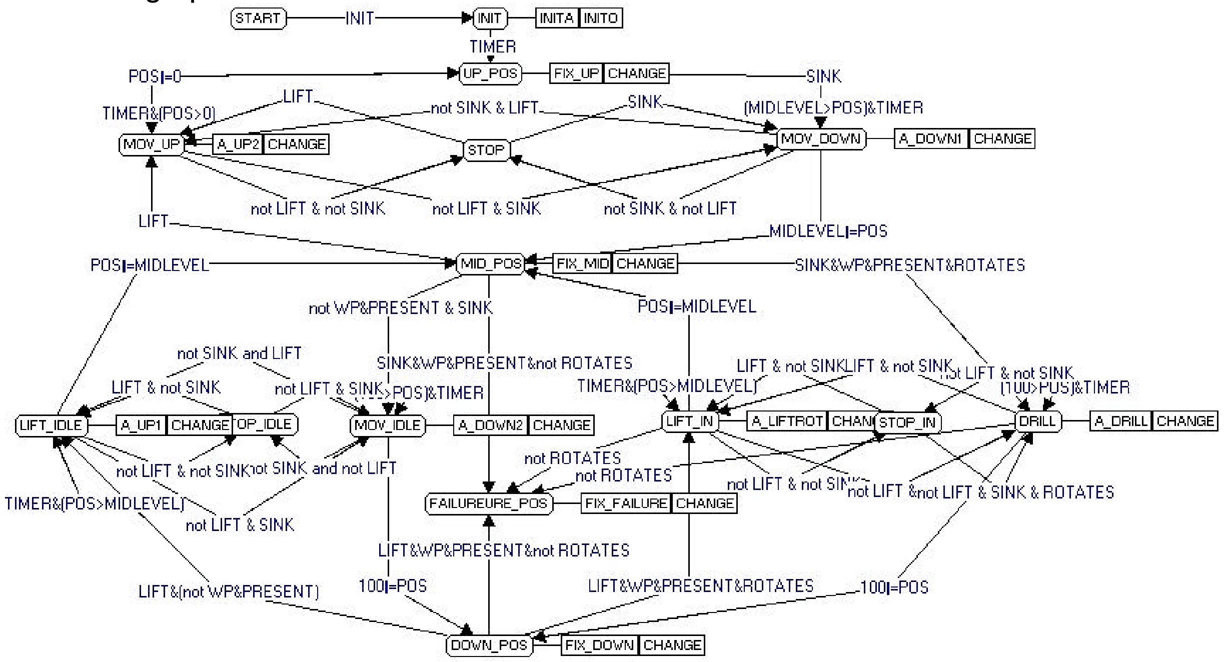

Figure 5. Execution Control Chart implementation of the state chart model.

4. Model Analysis

The new system configuration can be analyzed by the prototype of the verification tool VEDA (Verification Environment for Distributed Applications) [8,9]. The tool is integrated with the engineering environment FBDK and the run-time platform FBRT and allows formal verification of IEC61499 system configurations. VEDA checks whether the system configuration complies with a pre-given set of conditions, describing forbidden or desired behavior of the control system.

To conduct the formal validation procedure, VEDA transforms the function block model into the corresponding model in Net Condition/Event Systems [6,7]. This formalism was especially adjusted to cope with the complexity of formal verification for systems with distributed states, in particular IEC61499 function blocks, containing asynchronous parts (models of equipment) along with synchronous models of controllers.

An example of the NCES model corresponding to the “Linear model of DRILL with sensors” is shown in Figure 6. Thanks to the ECC structural representation, the NCES model can be generated from the ECC automatically.

The tool VEDA inputs the applications, generates the NCES models for the remaining (controller) blocks presented in languages IEC61499 and verifies the compliance of the models behavior with the set of pre-given specifications of permitted/forbidden behavior. If a trajectory is found, which does not comply with the specifications, it can be analyzed in detail by means of simulation (e.g. using the simulation function block LINEAR with the corresponding input parameters).

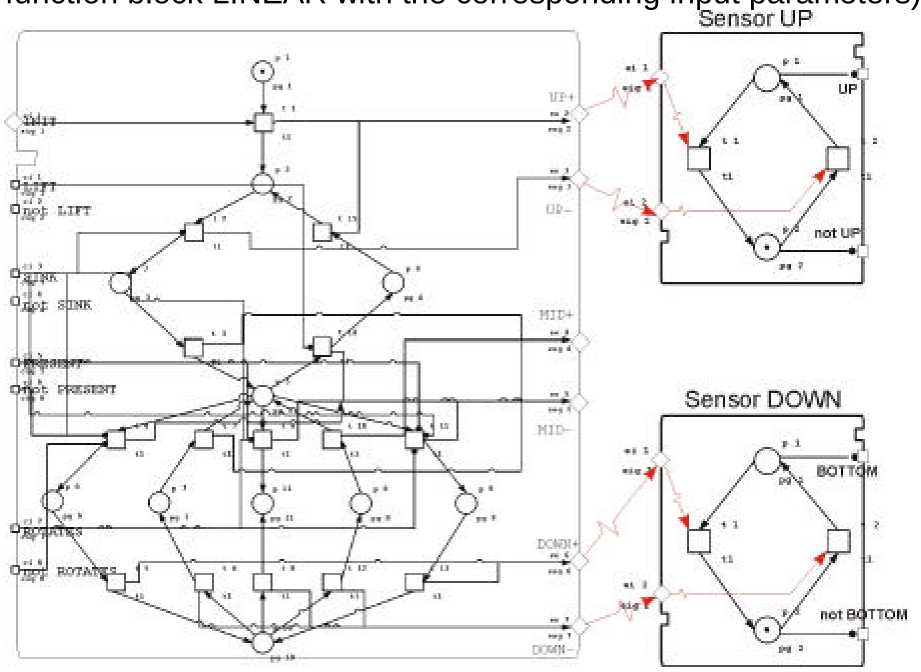

Figure 6. Net Condition/Event discrete state model of the linearly moving part of the drill with two logic position sensors.

5. Conclusion

This paper illustrates the modeling approach that targets distributed automation systems, enables both simulation, formal verification and their combinations, as well as combines the following features:

? *Heterogeneous* – combining the features of discrete and hybrid formalisms, that enables mutual positive impacts and benefits, e.g. interpretation of verification results by simulation;

? *Scalable* to the desired precision of modeling by abstraction of unnecessary details;

? *Embeddable* – encapsulated into component's description. This way the model can follow industrial objects throughout their life-cycle.

Currently the work is under way on implementation and formal analysis of a realistic-scale distributed control testbed (Figure 7) based on full compliance with IEC61499 architecture [10].

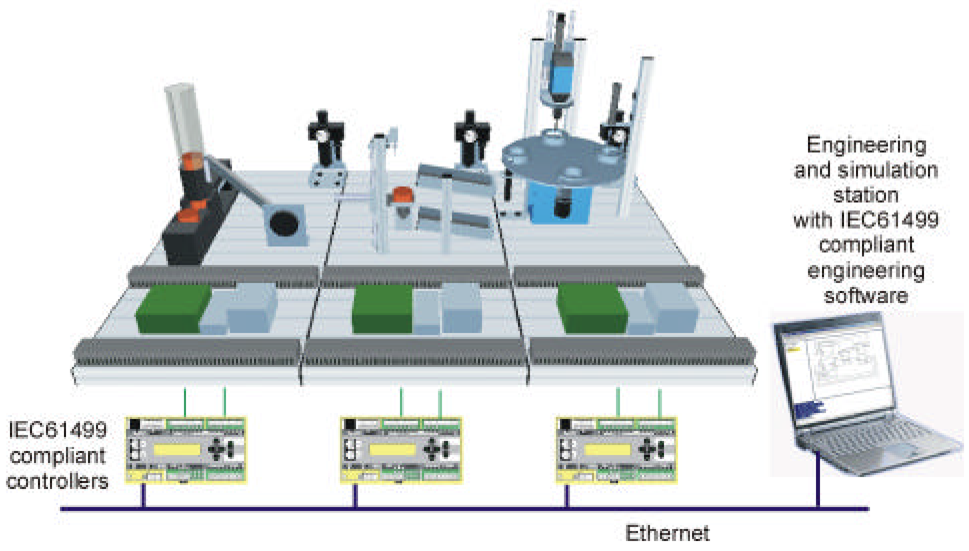
Engineering and simulation station with IEC61499 compliant engineering software
IEC61499 compliant controllers
Ethernet

Figure 7. Distributed automation testbed.

6. References

1. *Function Blocks for Industrial Process Measurement and Control Systems*. Publicly Available Specification, International Electrotechnical Commission, Tech. Comm. 65, Working group 6, Geneva, 1998.
2. R. Lewis: Modeling Control Systems using IEC 61499, IEE, London, 2001
3. PROFInet – project of Profibus User Organization, URL: http://www.profibus.com
4. J.H. Christensen: *Design patterns for system engineering with IEC 61499*. Proc. Of Conference "Verteilet Automatisierung" (Distributed Automation), pages 63--71, Magdeburg, Germany, 2000
5. J.H.Christensen: IEC 61499 ARCHITECTURE, ENGINEERING, METHODOLOGIES AND SOFTWARE TOOLS, 5th IFIP International Conference on Information Technology for BALANCED AUTOMATION SYSTEMS In Manufacturing and Services, to appear in Proceedings, Cancun, Mexico, September, 2002
6. H.-M. Hanisch und A. Luder: Modular Modeling of Closed-Loop Systems. Colloquium on Petri Net Technologies for Modeling Communication Based Systems, Berlin, Germany, October 21-22, 1999, Proceedings, pp. 103-126
7. P. Starke, S. Roch, K. Schmidt, H.-M. Hanisch, A. Luder: Analysing signal-event systems, Technical report**,** *Humboldt Universitat zu Berlin*, Institut fur Informatik, http://www.informatik.hu-berlin.de/lehrstuehle/automaten/tools/, July, 1999

8. Vyatkin V., Hanisch H.-M. *Bringing the model-based verification of distributed control systems to the engineering practice*, in book Intelligent Manufacturing Systems 2001, Elsevier Science, pp.152-157, November 2001
9. Vyatkin V., Hanisch H.-M.: Verification of Distributed Control Systems in Intelligent Manufacturing, Journal of Intelligent Manufacturing, special issue on Internet Based modeling in Intelligent Manufacturing, to appear in No.1, 2003
10. Distributed Automation Testbed at http://at.iw.uni-halle.de/~testbeds