35

COMPONENT DESIGN AND FORMAL VALIDATION OF SFA SYSTEMS: A CASE STUDY

Valeriy Vyatkin, Hans-Michael Hanisch
Martin Luther University of Halle-Wittenberg
Valeriy.Vyatkin@iw.uni-halle.de
Hans-Michael.Hanisch@iw.uni-halle.de

In this paper we present a case study of component based automation system design using IEC61499 accompanied by subsequent application of the formal modeling methods and the corresponding verification tools.
Our approach to validation is based on the results of formerly conducted research and development works on formal modeling of distributed control systems. The validation tool VEDA (Verification Environment for Distributed Applications) is intended on integration with IEC61499 engineering tools by means of using standardized source-code syntax and XML-based document types.

1. INTRODUCTION

The new developing international standard IEC61499 [1,2] provides an architectural framework for development and deployment of scalable flexible automation systems powered by distributed intelligence.

In this paper we attempt to illustrate the component based automation system design using IEC61499. The *component* is understood as a container that encapsulates heterogeneous properties of real industrial objects, such as dynamic models, the intelligence needed to control the underlying equipment in order to solve the predestined tasks, and the interfaces to process and to other objects constituting industrial systems. In this sense, the concept of component is somewhat complementary to the mechatronic approach e.g. [3], which serves as a framework to combine mechanical and electronic circuitry elements and properties of the equipment.

The goal of the component-based design is to facilitate development, deployment and, particularly re-engineering of automation systems. The key goal is to optimize testing of the modified configurations by application of automated validation methods. Testing of the resultant system, if it is done according to state-of-the-art simulation methods, could nullify such gains of the component-oriented design as fast re-configuration.

V. Mařík et al. (eds.), *Knowledge and Technology Integration in Production and Services*

However, along with new challenges, IEC61499 has created new opportunities of formal methods application in order to improve reliability of flexible automation systems. Thus, the formal verification tools can be easier applied for automated check of the validity of a pre-given set of safety properties for the modified system architectures, finding erroneous situations arising from the integration of different objects. In this paper we present a case study of component based automation system design using IEC61499 accompanied by subsequent application of the formal modeling methods and the corresponding verification tools.

2. COMPONENTS

We will illustrate the component structure on the drilling station represented in the following Figure 1 with the functionality as follows:

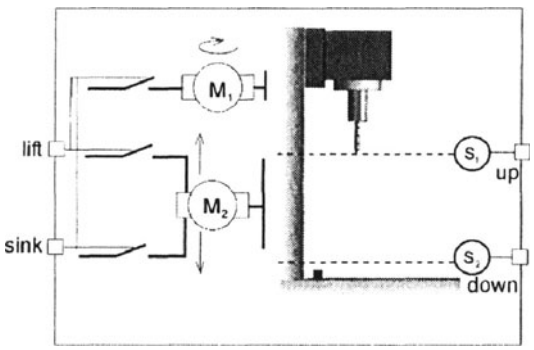

Figure 1. A drilling Station.

The spin motor M_1 rotates the bore of the drill. The step motor M_2 moves the head of the drill in vertical direction. The motor is controlled by two Boolean level signals: lift and sink. These signals are connected in parallel to the spin motor: thus the drill rotates always when the step motor moves the head. Position of the head is detected by two logic sensors: up and down

The corresponding *component* is presented in Figure 2. It serves to encapsulate heterogeneous control-related properties and functions of the object. The interface of the component is unified with the interface of IEC61499 function blocks. There are pre-defined classes of inputs/outputs, e.g. Input Commands, uniting the pulse signals, representing the commands from operator, or from the other components; or State Information, representing the parameters of the object or of the environment.

The cornerstone of the component is one or several sub-applications, defining as the structure, as well functionality of the object. The sub-application is an architectural unit defined in IEC61499 for distributable compositions of function blocks. Several sub-applications may be necessary for different configurations such as a pure run-time configuration, or a configuration with real object substituted by

its simulation model, or combination of those allowing comparison of the outputs of the real process with the simulated ones.

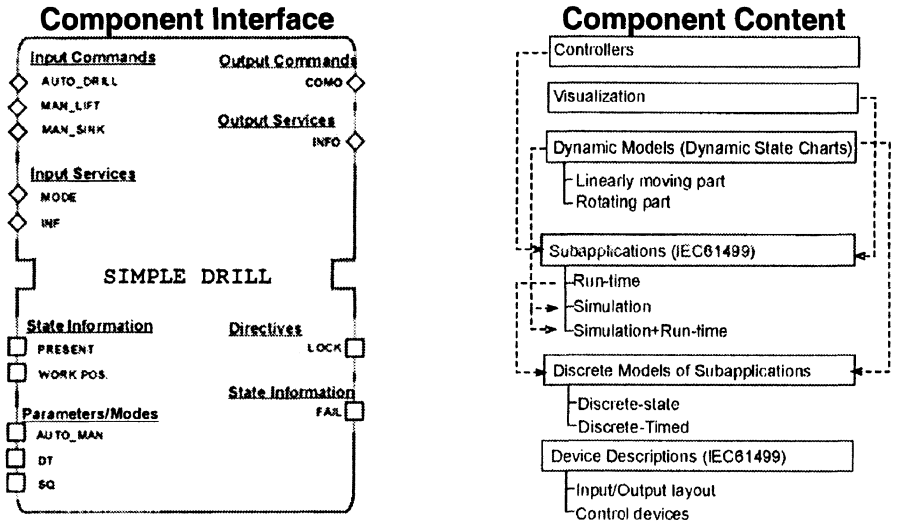

Figure 2. Interface and Constituent Elements of the Component "SIMPLE DRILL".

All these configurations may share common control and visualization functions. The corresponding simulation function blocks may be generated from the formal models of the object's behavior. The models and repositories cannot be encapsulated directly in the framework of IEC61499, so the component serves as a container for all these loosely connected elements. The common XML-based presentation will facilitate integration of elements with the component.

The formal models of the whole sub-applications contained in the component are intended for the use in the automated validation of applications, generated as a result of several components interconnection. These models can be generated from the sub-application descriptions. For the blocks, representing the models of objects within the sub-applications the corresponding "verification-oriented" models can be generated more efficiently with the help of the formal models of objects.

The contained sub-applications are built according to the MVC (Model/View/Control) methodology suggested in [4]. The Figure 3 shows the hierarchical structure sub-application combining the simulation and interaction with the real object. The upper level of the hierarchy is represented by the DM_MVC sub-application. The interface of the component is mapped onto the interface of the contained sub-applications.

Model/View/Controller

The sub-application 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 (Drill with Motor Model and View) represents the functionality of the equipment, while the block CONTROLLER stands for 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.

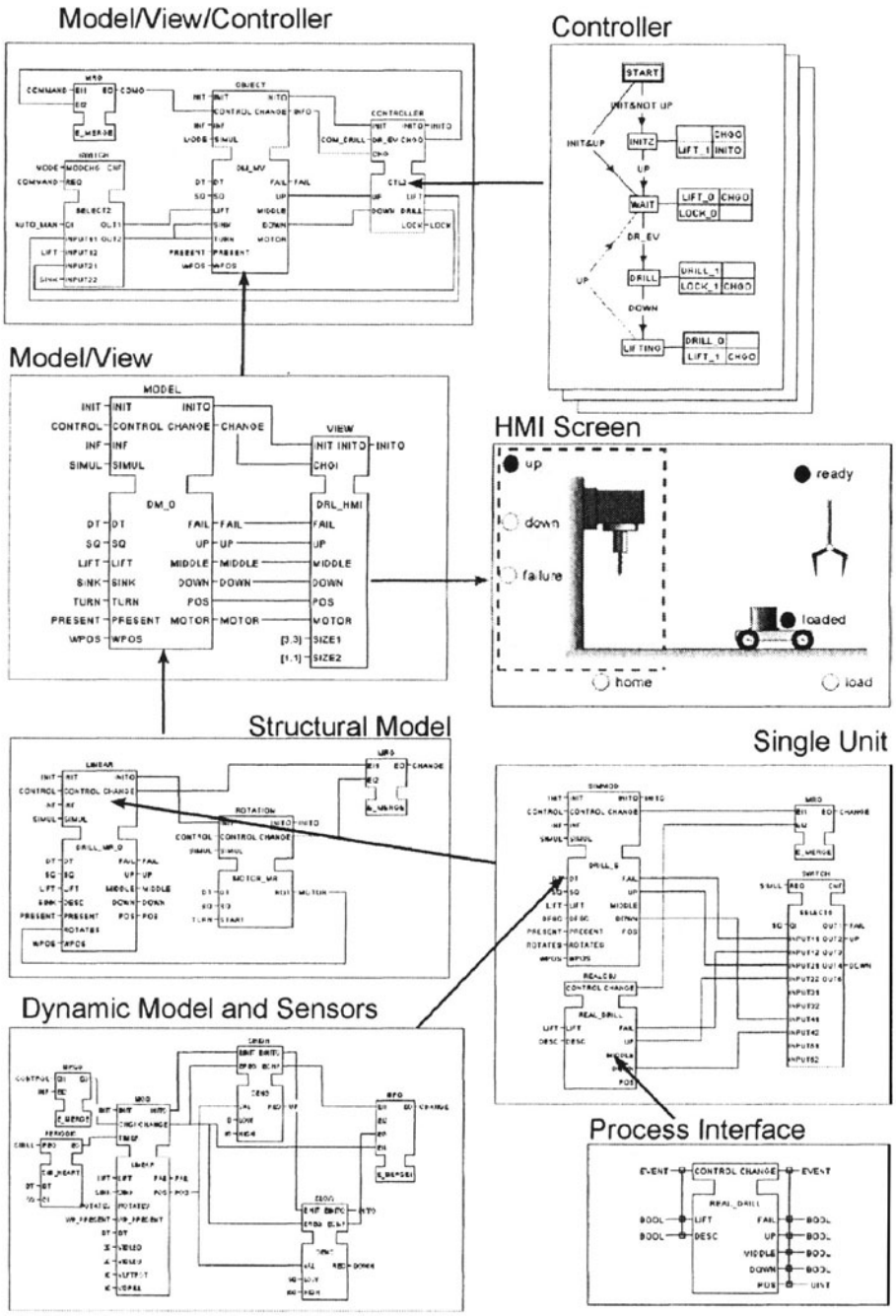

Figure 3. Hierarchical Structure of the MVC Sub-Application.

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 HMI display as shown in the Figure 3, is refreshed given the values obtained from the OBJECT block. Location of the drill's head is displayed according to the coordinate POS.

The block OBJECT of type DM (stands for Drill with Motor) 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 DM.

Controller

The sequential control of the drill is defined in the form of sequential function chart. A repository of controllers may be necessary to implement several behavior scenarios of the object.

Structural Model

This level represents the structure of the object. Thus, the DM block represents the model of drill composed from two components: a model of the head as a vertically moving object, and a model of spindle's rotation.

The model reflects the fact of relative independence of the components: axis position of the head has no influence on its rotation, however the rotation status of the motor 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 DRILL_MR_0) and ROTATION 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.

Model of a single unit

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 can be substituted by direct interfaces to real devices.

The model and the interface to the real process are combined within one function block DRILL_MR (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.

Dynamic model

The block MOD (of type LINEAR) encapsulates discrete implementation of the dynamic model of the vertically moving head driven by the motor M_2. 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 vertical position of the head. Blocks SHIGH, SLOW of type SENS represent the discrete sensors, that indicate correspondingly up and the low positions of the head. The LOW and HIGH parameters of the SENS block represent the interval in which must fall the numeric input value VAL in order to the logic output RES to be produced.

3. BUILDING SYSTEMS

3.1 Integration

We illustrate the component-based system design on the following prototype of an automated manufacturing cell, consisting of 3 units as shown in the Figure 4.

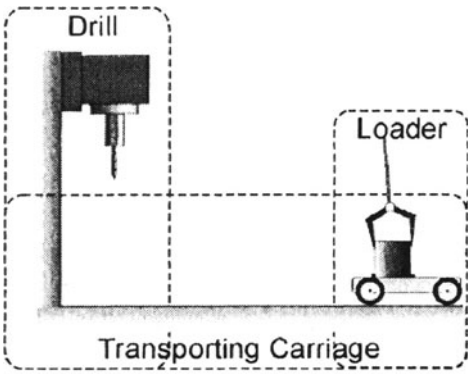

Figure 4. A Prototype of a Modular Manufacturing Cell.

These are the boring machine (*drill*), *carriage*, which delivers workpieces to the home position of the drill, and the *loader* that loads/unloads the carriage in the *loading* position that is opposite to the home position.

The appearance of the workpiece on the carriage is detected by an embedded sensor, so no particular communication between carriage and loader is necessary. Arrival of a new workpiece may serve as a signal to the carriage to approach the drill and request the processing service from it.

This object can be built using constituents from different vendors, having diverse dynamic characteristics, sizes, layouts of sensors/actuators, and other differences. In our case-study we have considered 3 models of drills, and 2 models of loaders and carriages. All the differences between the equipment units are encapsulated within the component descriptions.

Visual tools can facilitate the design process, reducing it to interconnection of components as shown in the following Figure 5. The resultant application is built automatically from the corresponding sub-applications contained in the given components.

The application is appended by the HMI panel, that can be designed with the help of a visual editor and placed into repository is as a component.

In our example, the panel has one button to control switching the simulation mode, and two LEDs indicating failures in drill and in the carriage.

The panel is selected from the repository and added to the Design Screen, that implies the appearance of the corresponding component in the Application window.

This description is enough to generate the application. In this stage the application is independent from the architecture of hardware, where it will be executed.

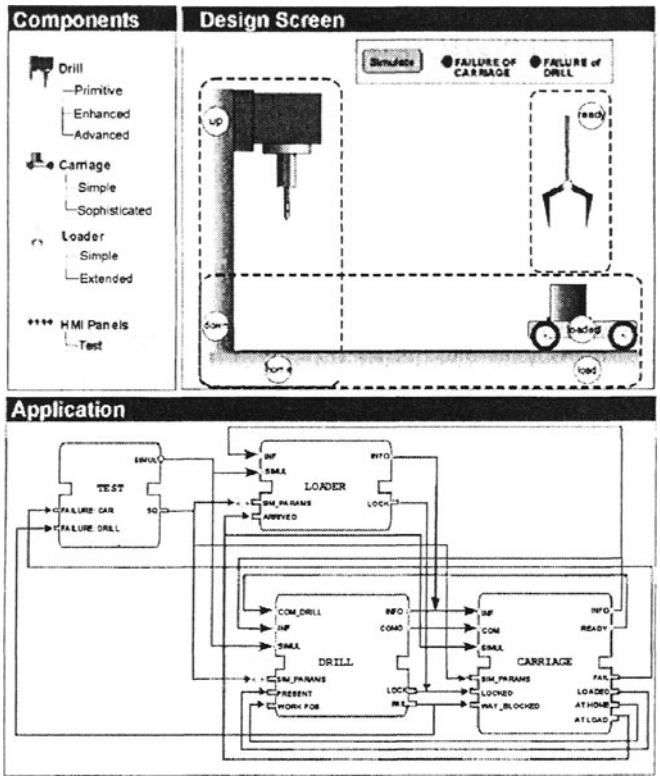

Figure 5. Application Design in a Visual Tool by Plug-And-Play of Components.

3.2 Distribution

Next step of system's implementation is the planning of the architecture. Some possible architectures can be considered with respect to our example. In the architecture, presented in Figure 6 the control is distributed over the constituent parts of the system, while the simulation is conducted on the PC-based station.

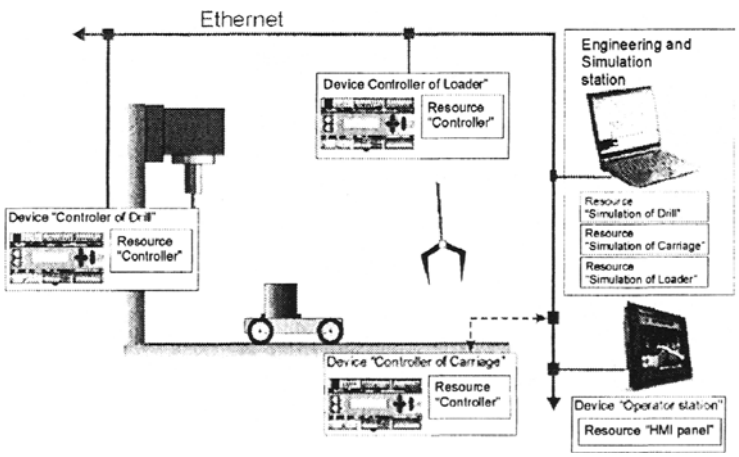

Figure 6. Distributed Control System Architecture.

System configuration in IEC61499 consists of the description of the set of container devices and their resources, and of the mapping of the application's parts onto the containers.

4. MODELING AND VALIDATION

Testing of scalable, flexible automation (SFA) systems having distributed architecture of control is complicated by the following reasons:

- Asynchronous event-driven logic of execution, statically unpredictable combinations of concurrent processes in plant and dynamic scheduling of algorithms in controllers;
- Communication phenomena: use of different protocols, influence of delays, etc.
- Reconfiguration phenomena: the same application could be executed on different architectures;

Dynamic models of controlled equipment are indispensable for simulation, verification, and for interpretation of the verification results by simulation. It is important to unify the modeling process by using a standard self-explanatory problem-oriented visual modeling language.

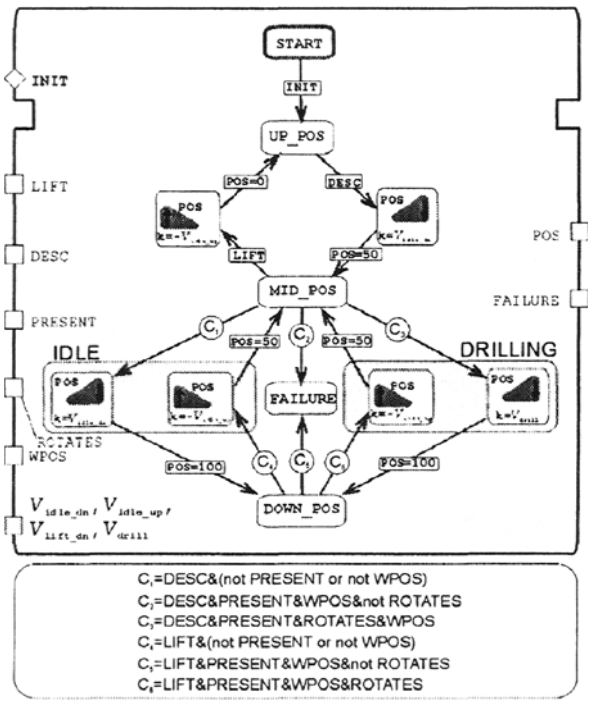

Figure 7. Modular Dynamic State Chart Model of the Linearly Moving Part of the Drill.

We apply for this purpose a customized form of State Charts (the same as used in UML). The customizations concern the set of state shapes, corresponding to

particular dynamic properties of parameters, and modular interface, compatible with interface of IEC 61499 function blocks.

The modular interface is unified also with the interface of hybrid and discrete state modeling formalisms, such as: Condition/ Event Automata [5] and Net Condition/Event Systems [6], that simplifies the transformation of state-chart models to these formalisms. The dynamic state chart is built from states (rectangular shapes) and state transitions (arcs) marked with Boolean conditions.

In the chart in the Figure 7 there are two types of states: fixed position states UP_POS, MID_POS, DOWN_POS and dynamic states with linear change of parameter POS as $POS=POS_{old}+kdt$, where the coefficient k is the speed of moving, dt – time increment.

The model describes the following behavior. The head moves free in the upper part of the axis, no matter present the workpiece or not. When the middle position is reached and the control signal DESC remains ON, the head 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. Thus the presented model defines *uncontrolled behavior* of the drill (its vertically moving part).

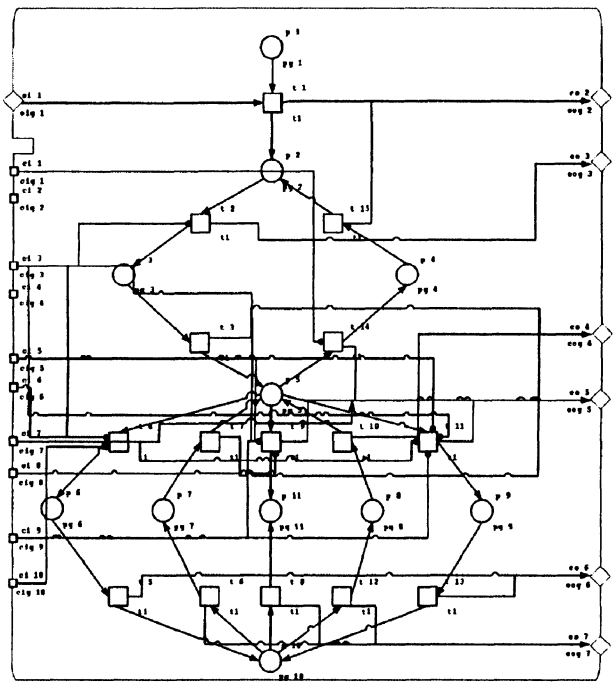

Figure 8. Net Condition/Event Model of the Linearly Moving Part of the Drill.

Note, that the presented model generates the numerical value POS and can be re-used as a core of models for several types of drills with different number of logic position sensors.

The State Chart model can be used to generate discrete state model in Net Condition/Event Systems as presented in Figure 8. The latter is required for

conducting the formal validation procedure using the tool VEDA, as it is described in [7].

The module modeling the drill is incorporated into the modularly built NCES model of the application, substituting the block DRILL_M in Figure 3.

The tool VEDA inputs the applications, generates the NCES models for the remaining blocks (not explicitly presented in NCES) and verifies the compliance of the models behavior with the set of pre-given specifications of permitted/forbidden behavior.

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. M. Bonfe, C. Fantuzzi: *Mechatronic Objects encapsulation in IEC 1131-3 Norm,* Intl. Conf. on Control Applications, Anchorage, 2002
4. J.H. Christensen: *Design patterns for system engineering with IEC 61459.* Proc. Of Conference "Verteile Automatisierung" (Distributed Automation), pages 63--71, Magdeburg, Germany, 2000
5. S.Kowalewski, P.Herrmann, S.Engell, R.Huuk, H.Krumm, Y.Lakhnech, B.Lukoschus, and H.Treseler: *Approaches to the formal verification of hybrid systems.* Automatisierungstechnik, 2:66--73, 2001.
6. Rausch, M., H.-M. Hanisch. *Net condition/event systems with multiple condition outputs.* In: Symposium on Emerging Technologies and Factory Automation, 1995. Vol.1., INRIA/IEEE. Paris, France. pp.592--600.
7. 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 as No.1, 2003*