Reuse of Components in Formal Modeling and Verification of Distributed Control Systems

Valeriy Vyatkin
The University of Auckland
Dept. of Electrical and Computer Engineering
Auckland, New Zealand
V.Vyatkin@auckland.ac.nz

Hans-Michael Hanisch
Martin Luther University of Halle-Wittenberg,
Dept. of Engineering Sciences
D-06099 Halle, Germany
Hans-Michael.Hanisch@iw.uni-halle.de

Abstract

This paper describes formal modeling and verification of automation systems from the system engineering point of view. Reuse of model components is the key issue in order to bring the scientific modeling methodology into engineering practice.

The reuse is achieved by the combination of modular modeling of automation systems with object-oriented description of models in UML style. This allows to benefit from advantages of both worlds: efficiently manage highly hierarchical complex models with UML tools and end up with efficiently executable models with distributed states that are compatible also with IEC61499 function block specifications. The approach is supported by the tool framework that is described in the contribution.

1. Introduction

Reuse of components in system design is the cornerstone of engineering. In the information technology, attempts to reuse the intellectual property, in particular represented in form of software components, gave birth to object-oriented methods of programming and UML-based modeling and design of software systems. It gradually finds its way also to the area of automation technology, for example as the Function Block paradigm of the IEC61499 standard, a number of works on Java applications in automation and UML applications in automation.

Automation technology, however, is different from software engineering, as well as from mechanical engineering. It is an engineering discipline that applies knowledge from such subjects as manufacturing and process systems, electrical, computer and control engineering, as well as from software engineering and computer science.

Thus, although automation technology is deeply influenced by information technology in the sense that the control system itself is an information processing system, automation technology is not identical with information technology. The major point of concern in automation technology is the object that is controlled, and the control system serves only as a means to reach the goals coming from the controlled object.

The major responsibility of an automation engineer is to design and implement a control system that interacts with the object of control in a closed loop and that ensures that the controlled object behaves safely and efficiently.

Therefore, any keen and scientific methodology for design and verification of control systems must take the behavior of the controlled object, along with its distributed and hierarchical structure, into consideration. Having only a model of the controller is in general not sufficient to prove the correctness of the specifications.

In this contribution we continue the discussion of a systematic modeling and verification methodology that started in [HaVy05]. Our approach includes models of the controller as well as models of the controlled object and is intended to be integrated into routine engineering process. For that reason we need to address the reuse issue in more detail.

The goal of the recently started research project VAIAS - Validatable Architectures for Industrial Automation Systems- that is funded by German Ministry for Education and Research and by the industrial partners is the development of an architecture that combines two issues:

1. an object-based approach to software and system engineering following the structure of the original systems, and
2. inherited validatability that allows application of formal analysis methods.

The validatability of the architecture shall be reached by the use of formal methods for specification of behavioral and structural properties of the system. VAIAS intends to meet the new challenges of the automation world by providing a new software architecture that could better fit to the decentralized reconfigurable nature of new automation systems. A higher inherited level of robustness will be reached by means of formal analysis and synthesis.

VAIAS defines three mutually interconnected scenarios in system engineering:

1. development of the automation software in an object-oriented way,
2. simulation of such systems and
3. their formal verification.

It was found, that many features of models required for all three scenarios are very similar and thus can be shared. In particular, the structure of all three models can be almost identical if proper description means are used.

We use function blocks of IEC61499 standard for description of the executable component-based model of the automation software. This executable specification form can be also used to define the simulation model of the automation system. For formal analysis, we use the formalism known as Net Condition/ Event Systems. We pursue the following features:

- reduced complexity, time and effort in the design will be achieved through the use of pre-designed elements and standard design templates;
- improved assurance of correctness will be achieved by the use of simulation and formal verification.

This contribution will give an outline of an appropriate methodology for modeling and verification of distributed systems. It is organized as follows. In Section 2 we attempt to identify and classify the problems arising from decentralized control organization combined with component structure of the controlled plant. In Section 3 we present a brief survey of the results reached so far on application of object-oriented design methodologies in control engineering. Then, in Section 4 we present the main ideas of the modeling and verification framework

being developed. Section 5 extends it by several application scenarios. The paper is concluded with the future work plans and acknowledgements.

2. Problems of Decentralized Control

When manufacturing systems are composed from standard building blocks, the natural desire of any control engineer is to reuse some parts of the control design. This justifies the idea of modular control design. But, the modularity itself is not sufficient for really complex systems. Object-oriented methods are therefore being applied to this problem.

As we have mentioned, control design and validation must take into account the closed-loop behavior of plant and controller. Complex systems with distributed control may have several closed loops and hierarchical compositions of them. This will be illustrated on the following example.

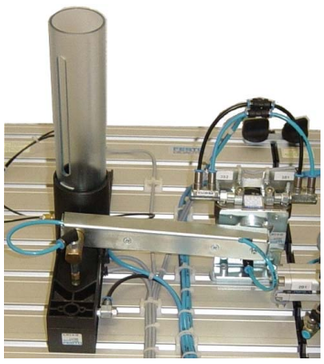

Figure 2: Example of a manufacturing system

Figure 1 shows an example of a part of a manufacturing system in laboratory scale. This primitive (but still representative!) example of a manufacturing system with decentralized control consists of two independent mechatronic actors. These are a storage unit and a transfer unit. The storage unit stores a pile of workpieces. The feder, which is a part of the storage unit, can push one of the workpieces from the pile to the output position. From there it can be picked up by the transfer unit. The transfer unit grasps the workpiece using its vacuum sucker, and carries it to the subsequent unloading position located on the right side (not shown in the Figure). Thus, we distinguish two end positions of the transfer unit: the left or loading position and the right, unloading position.

It is assumed that both of these constituent parts have some pre-programmed autonomous functionality. Their controllers interact with each other and with the superior levels of control.

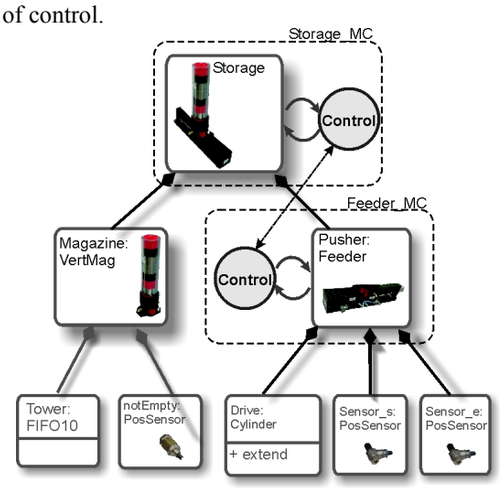

Figure 3. Structure of the Storage unit.

Each of the actors is composed from mechatronic elements potentially having some control logic. For example, Figure 3 shows the structure of the storage unit. The transfer unit has approximately the same complexity.

The storage is composed from two parts, one of which is the magazine for ten workpieces with the presence sensor, and the other is the pneumatic cylinder with two end position sensors.

The diagram in Figure 3 is obviously inspired by the class diagrams of UML. It shows that certain components (like pusher) may have their own embedded control logic. Thus, the whole system can be regarded as a hierarchical composition of closed-loop plant/controller systems.

Integration of such multi-level decentralized controllers is complex even for such a primitive example. It may, however, pave the way torwards methodologies that reuse components instead of starting from scratch over and over again.

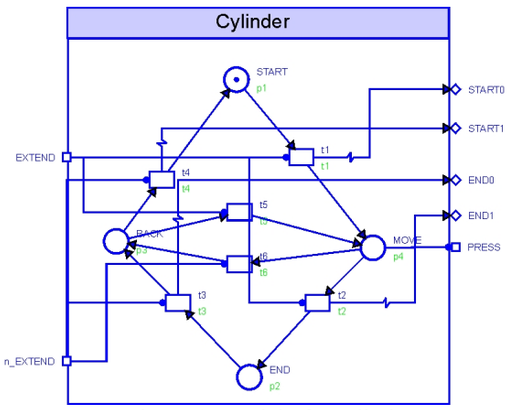

Figure 1. Model of a cylinder

For example, the system in Figure 3 contains two identical sensors. If the sensor requires specific control or modeling, there is no need to develop it twice. It could be done once and reused as many times as needed.

3. Object-oriented approaches to control engineering

There is growing recognition of object-based design of software for automation systems. In particular, a number of researches touch the use of the Unified Modeling Language (UML) [Gom00, Doug99] for the development of industrial automation systems. Thus, UML was used in a number of works in the industrial automation context, with most attention paid to the issues of modeling for specification and design of controllers, modeling as a part of system engineering, and modeling for formal verification.

In particular, a methodological framework for the application of formal design and verification techniques is presented in [Bonfe03, Bonfe04]. The object-oriented modeling approach is based on an extended subset of UML and adopts several of its diagram types. The system is decomposed in a top-down approach into mechatronic objects whose signal-based interfaces are specified by mechatronic classes. Then Class Diagrams – representing the structural model – are used to describe the system's hierarchical architecture by means of composition links between those mechatronic classes. UML Collaboration Diagrams – stereotyped as Mechatronic Data Flow Diagrams – specify object interaction and Statecharts specify the behavior of a mechatronic object. Additionally it is also possible to describe the behavior of the uncontrolled plant by Statecharts that are associated with classes stereotyped as hardware.

Commercial CASE tools may be used to create and edit the various diagram types, and the symbolic model checking tool SMV allows for the verification of desired

properties which have to be expressed in the temporal logic CTL.

That approach permits a modular, reusable and implementation platform independent description of control system design models. However, to ease its practical application, tools are required for the translation of the model into the input language of SMV and for generating PLC code.

4. Modeling and Verification Methodology

Modeling the uncontrolled plant behavior is definitely the most critical issue in all model-based technologies since any result is only as good as the model is. It is also a critical issue with respect to time and effort that have to be spent on obtainig the model.

We prefer a way of modeling that is rather an engineering process of the model than a design process. This means that we suppose that there is a library of models for subcomponents available that serves as a basis for model engineering.

There are some specifics in the control area regarding the models that are used. Requirements to the modeling formalism origin from two aspects:

- How to build the model?
- How to use the model?

First, we have to realize that the human factor in engineering the model is very essential. The human beings who are involved in the process have to trust the model. It is therefore very useful to have models which are graphical and executable to support validation.

Since almost any engineer in the area of control system design is familiar with some kind of block-diagrams, we have chosen a way of model-engineering that is based on blocks and interconnection of blocks by means of signals.

Models must be modular with a clear interface and hidden dynamic behavior. Models should also support an incremental way of modeling. They have to be extendable to ensure that new or more precise information can be included in the model during the lifecycle of the system that is modeled.

Last, but not least, they have to have capabilities for parametrization to adapt models to similar, but not identical components.

Another very important issue is how the models support specifications. Specifications describe forbidden behavior as well as desired behavior. Forbidden behavior describes situations that cause danger for the controlled object, human operators, or the environment. Such behavior is specified by means of forbidden states. It is mostly some kind of local specification that includes only states of a small number of subsystems.

Due to the distributed nature of the models, specification of local properties is much easier than specification of such properties in a huge composed model with thousands or millions of global states.

Desired behavior is specified as sequences of states or state transitions, that are in most cases also local.

The price one has to pay for all these nice (and required!) features regarding the use of the models is the fact that the pure “standard”-models that are used in the scientific community for formal verification do not support such a way of model engineering. This has some consequences.

The first one is that the theoretical results and appropriate software tools cannot directly be applied to the models.

One solution is to define and implement transformation rules from the engineering models and the specifications to the models and the specifications that are used for formal verification. This, however, is only half of the truth since a retransformation from the results of the formal verification into the original models and specification has to be provided as well.

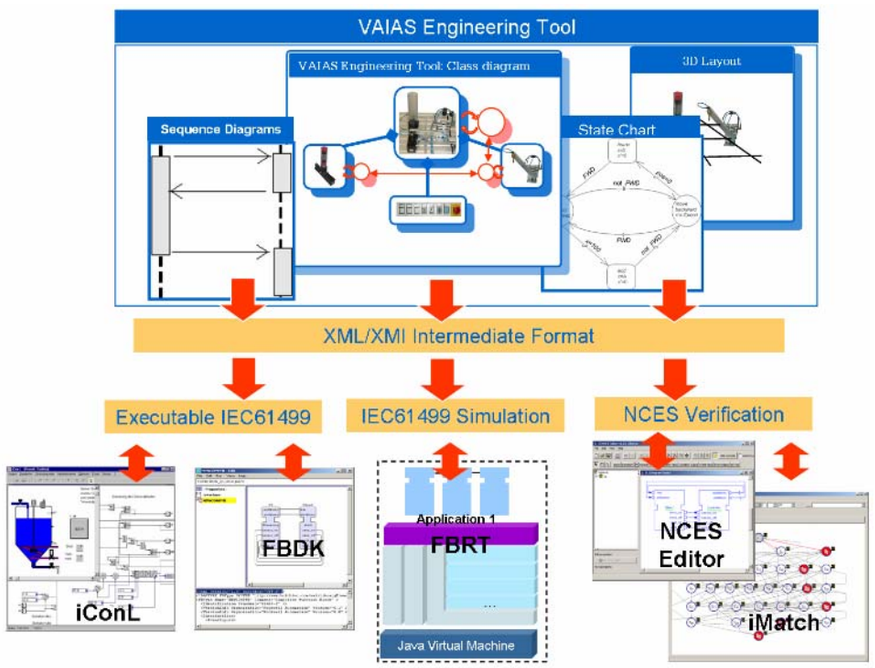

Figure 4. VAIAS engineering tool framework.

This can be a rather demanding and complicated task.

Another solution is to develop formal verification methods and software tools for the original models. This avoids the need for retransformation of the verification results and has been done for our approach as will be described later on.

Our experience gained within preceding applications [Hanisch et al 01, Lobov et al. 03, Lobov et al. 04] shows that it is highly desirable to use models that may constitute a one-to-one mapping of the structure of the manufacturing process onto the structure of the corresponding model. Coming up with large monolithic models is almost infeasible due to the size and complexity of the original system.

We use a modeling formalism of Net Condition/Event Systems (that were called NCES or SNS in the past [HaLu99, HaLu00, Thi02]).

They provide means for modular, graphical modeling in a way that is intuitively understandable by any engineer. The basic patterns are modules or blocks that encapsulate some kind of dynamic behavior model as exemplified in Figure 3 for a model of cylinder. Modules can be interconnected by signals. We have two types of signals, namely condition signals that represent state information and event signals that provide state transition information. The syntax and semantics as well as composition rules of the models are exactly defined, but these formalisms go beyond the scope of this contribution.

If we compare the modeling formalism with the IEC 61499 specification of Function Blocks, we see some strong similarities:

- dynamic behavior is encapsulated in the blocks and hidden to their environment
- blocks resp. modules have a signal interface that clearly distinguishes between data (resp. conditions) and events.
- The structure of the system is given by the interconnection of blocks by means of signals.
- The dynamic behavior of the system is given by the behavior of the modules and the structure, i.e. the interconnection of blocks or modules.

It is therefore very natural to map Function Blocks to formal modules and to interconnect them exactly in the same way as the Function Blocks are interconnected in the application. Due to these strong similarities, a mapping of a Function Block application to a formal model with identical structure could be defined and implemented. That has been done and integrated in the tool that is described in the following section.

The formalism of Net Condition/Event Systems has undergone some "modifications" as described in [VHP03, VHB04] in order to be compatible with the IEC61499 executable specification and its supporting tools.

In particular, we have developed an open XML format for the models. It allows for interoperability of independently developed tools, that already had its positive impact – the NCES models can be created, opened and configured with the editor developed in our group in Halle, as well as by the Function Block Development Kit FBDK and by a tool developed at Tampere University of Technology [Lobov et al, 04] and by iMATCh.

V. VAIAS APPLICATION SCENARIOS

Achieving better "validability" of automation software requires performing of simulation and formal verification procedures through the engineering cycle, probably in a repetitive manner.

The application scenarios provided by the VAIAS architecture are supported by the tool framework as presented in Figure 4. At first the system is being composed from the constituent "automated mechatronic objects". Engineering tools allow the description of the system's structure, geometrical positioning of the

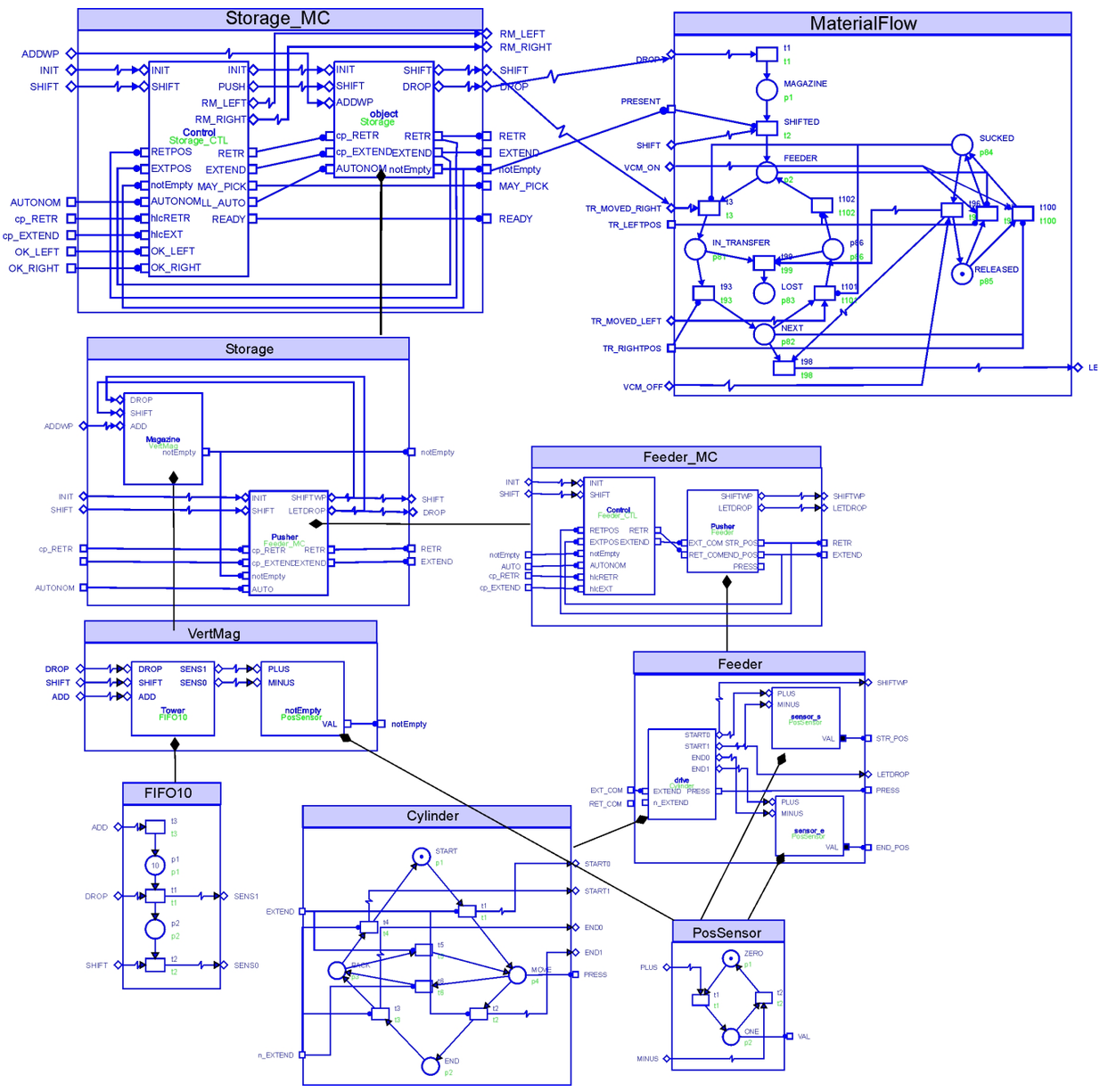

Figure 5. NCES model of the whole storage unit

components, and their interaction with each other. As a result, the engineering tool produces a number of data files containing the described dynamic and static properties of the system. The most suitable data format for the result of the engineering process are XML/XMI. This data will form the input of the three subsequent scenarios.

Executable system configuration.

The intermediate XMI format is converted into the executable specification of Function Blocks following the IEC61499 standard. This form is precise enough but still independent from particular hardware architecture. The structure of the executable specification follows the MVA/CDA approach (model-view-adapters)/(control-diagnostics-adapters). Further translation to a machine-executable form is performed by the corresponding tools, such as the Function Block Development Kit (FBDK).

System configuration with simulated plant.

This configuration is also described by means of Function Blocks of IEC61499 standard and is different from the former one only in the block standing for the mechatronic components. The simulation can be conducted using the same distributed function block run-time platform as the former configuration, with some add-ons, such us statistics gathering. In Figure 4 the execution is illustrated by means of FBRT – Function Block Run-Time of Rockwell Automation [FBDK], that is a distributed Java-based platform.

System configuration with discrete state model of the plant ready for formal verification.

This is a modular object-oriented model of the executable system, where the modules corresponding to the object are inserted from the corresponding automation object repository, and the models of the software components (such as controller, diagnostics, supervisor, etc.) are generated using the corresponding model-generators.

Graphical editors provide full graphical authoring and editing of the models. The editor uses an open XML-based data format for basic and composite NCES models. The data format of composite model blocks intentionally was made identical with that of IEC61499 Function Blocks, supported by the tool FBDK [Hlbl].

iMATCh is an integrated tool that contains a model builder (assembler), a translator to the flat format for subsequent model-checking, interfaces to several model-checkers, and the means for analysis of scenarios (e.g. their visualization in form of state/time diagrams), or even system simulation along the selected scenarios. iMATCh inputs the model type files given in XML and is capable of:

- Assembling a composite, hierarchically organized model from modules contained in different libraries. The component model types are instantiated into NCES modules. This is illustrated in **Figure 5**. For example, the model type PosSensor is instantiated three times: twice in the definition of the model type Feeder (instances **sensor_s** and **sensor_e**) and in the model type VertMag (instance **NotEmpty**). The model of the storage connected in the closed loop to the model of its autonomous controller is created from the model types stored in the library of model types. The complete model of the object from Figure 2 would contain also the model of the transfer unit connected to the model of material flow and to the model of the storage.
- Translating the model into a "flat" NCES with the through numbering of places and transitions. The inter-module connections are converted into event and condition arcs between places and transitions. Thus, the module boundaries are removed and the model-checking tools can be applied. In particular, the translator generates files in the input format of SESA model checker.
- The model checker **SESA** allows for efficient model-checking of fairly complex systems (millions of discrete states).
- The application methodologies are represented as libraries of standard model elements and by the web-based documentation;

The model of the plant is engineered by predefined components that are stored in a repository of models. The modularity of the modeling approach is helpful for this. Therefore, the modeling process does not need to be started from scratch. Predefined and validated component models are encapsulated in modules and can be used over and over again without dealing with their internal details.

The validation of automation systems modeled by NCES can be performed by simulation and formal verification via model-checking.

Simulation usually follows a limited number of scenarios in the system's behavior. In contrast, the model-checking studies multiple scenarios caused for example by some unpredictable factors, such as variable durations of some operations, communication delays, malfunctions, etc.

The results of the model-checking, such as a reachability space (full, or generated until an example/counterexample is found) can be visualized as state/time diagrams of relevant values (e.g. represented as marking of certain places, or firing of certain transitions).

The verification consists in proving specifications with respect to the dynamic behavior of the model. The specifications can be given either in form of second order predicates or in form of temporal logic expressions. Terms of the expressions can be formed by referencing inputs, outputs and internal variables of the controller or variables of the model of plant. The latter have to be eventually expressed via marking of places in the model.

It is worth mentioning that the closed-loop approach to the modeling enables expression of the specifications directly in terms of the machine behavior (not only input and output signals of the controller).

In particular, the following properties of automation systems could be scrutinized applying the formal validation techniques:

1. Robustness of the system in case of malfunctions of some sensors;
2. The control programs in some programming languages (e.g. Structured Text and Sequential Function Charts) have branching structure. Formal verification may help to prove that the response time is never exceeded in any feasible IO combination.
3. Quality assurance: sometimes it is important to ensure that the plant never runs in undesirable situations, when, for example, inexact synchronization of processes in the plant occurs as a result of wrong synchronization of control programs;

5. Conclusion and Future Work

In this paper we presented the state of the art of the development of the modeling and verification scenario in the VAIAS research project. The work is still in progress and is currently in the stage of accomplishing pilot applications done as "proof of the concept".

This step will be followed by the tool development and integration that is supposed to make the presented vision real.

6. Acknowledgements

This work was supported in part by the cooperative project VAIAS funded by the German Ministry for Education and Research (BMBF).

References

[HaVy05] H.-M. Hanisch and V. Vyatkin: *Modeling and Verification of Distributed Control Systems*, International conference "Design, Analysis, and Simulation of Distributed Systems (DADS'2005)", Proceedings, San Diego, April, 2005

[BoFa03] M. Bonfe, C. Fantuzzi: Design and verification of mechatronic object-oriented models for industrial control systems, IEEE Conference on Emerging Technologies and Factory Automation (ETFA'03), Proceedings, vol. II, pp.253--260, Lisbon, September, 2003

[Hanisch et al 01] H.-M. Hanisch, T. Pannier, D. Peter, S. Roch and P. Starke: Modeling and verification of a modular lever crossing controller design. Automatisierungstechnik, 2000

[Ha04] H.-M. Hanisch: Closed-Loop Modeling and Related Problems of Embedded Control Systems in Engineering, 2004, in Abstract State Machines 2004, LNCS 3052

[HaLu99] H.-M. Hanisch and A. Lüder: *Modular Modelling of Closed-Loop Systems*, Colloquium on Petri Net Technologies for Modelling Communication Based Systems, Berlin, Germany, October 21-22, 1999, Proceedings, pp. 103-126

[HaLu00] H.-M. Hanisch und A. Lüder: A Signal Extension for Petri Nets and its Use in Controller Design, Fundamenta Informaticae. Nr.4, March 2000, pp. 415 – 431.

[HaVy04] H.-M. Hanisch and V. Vyatkin: Achieving Reconfigurability of Automation Systems by Using the New International Standard IEC 61499: A Developer's View, The Industrial Information Technology Handbook, CRC Press, October 2004

[Hlbl] http://www.holobloc.com – web site devoted to IEC61499

[IEC1131] International Standard IEC 1131-3, Programmable Controllers - Part 3, International Electrotechnical Commission, 1993, Geneva, Switzerland

[IEC1499] Function Blocks for Industrial Process Measurement and Control Systems, Publicly Available Specification, International Electrotechnical Commission, Part 1: Architecture, Tech. Comm. 65, Working group 6, Geneva, 2002

[Lew01] R. Lewis: Modeling Distributed Control Systems using IEC 61499, Institution of Electrical Engineers; London, 2001

[Lobov et al. 04] A. Lobov, J. LM Lastra, R. Tuokko, V. Vyatkin: *Modelling and Verification of PLC-based Systems Programmed with Ladder Diagrams,* INCOM'2004, Proc., Salvador, Brazil, April, 2004

[Lobov et al. 03] A. Lobov, J. L.M. Lastra, R. Tuokko, V.Vyatkin, "Methodology for Modeling Visual Flowchart Control Programs using Net Condition/Event Systems Formalism in Distributed Environments", IEEE Conference on Emerging Technologies and Factory Automation (ETFA'03), Proc., Lisbon, Portugal, September 2003

[StaGu03] M. Stanica, H. Guéguen, "A Timed Automata Model of IEC 61499 Basic Function Blocks Semantic", ECRTS'03 Euromicro European Conference on Real-Time Systems, Porto, Portugal, July 2003

[Thi02] J. Thieme: *Symbolische Erreichbarskeitanalyse und automatische Implementierung struktuirter, zeitbewerter Steuerungsmodelle*, Dissertation zur Erlagung des Grades Dr.-Ing., Berlin: Logos Verl., 2002

[VHP03] V. Vyatkin, H.-M., Hanisch, T.Pfeiffer, "Modular typed formalism for systematic modeling of automation systems", 1st IEEE Conference on Industrial Informatics (INDIN'03), Proceedings, Banff, Canada, August 2003, ISBN 0780382005

[VHB04] V. Vyatkin, H.-M. Hanisch, G. Bouzon: *Open Object-Oriented Validation Framework For Modular Industrial Automation Systems*, INCOM'2004, Proceedings, Salvador, Brazil, April, 2004

[VyHa03] V. Vyatkin, H.-M. Hanisch: Verification of Distributed Control Systems in Intelligent Manufacturing, Journal of Intelligent Manufacturing, vol.14, N.1, 2003, pp.123-136

[Wurmus00] H. Wurmus, B. Wagner,: «IEC 61499 konforme Beschreibung verteilter Steuerungen mit Petri-Netzen»; Conference Distributed Automation 2000 (Verteilte Automatisierung), Institut für Automation und Kommunikation e.V., Magdeburg 2000