

# ON INTEGRATION OF MODEL-BASED VALIDATION IN SOFTWARE ENGINEERING OF INDUSTRIAL AUTOMATION SYSTEMS

Valeriy Vyatkin

Department of Electrical and Computer Engineering  
The University of Auckland  
Auckland, New Zealand  
v.vyatkin@auckland.ac.nz

Sirko Karras, Thomas Pfeiffer and Hans-Michael Hanisch

Martin Luther University of Halle-Wittenberg,  
Dept. of Engineering Sciences  
D-06099 Halle, Germany  
{Sirko.Karras, Thomas.Pfeiffer, Hans-Michael.Hanisch}@iw.uni-halle.de

## ABSTRACT

This paper introduces idea of a system architecture for industrial automation software systems which integrates the formal background of modular place transition models with the ideas of the upcoming IEC61499 standard for component based distributed measurement and control systems. Goal of the architecture is to support simulation and formal verification as a natural part of the engineering process in industrial automation.

## KEY WORDS

Industrial automation, modeling, formal verification, architecture, intelligent control

## 1 Introduction

The automation technology is an engineering discipline that covers subjects like manufacturing and process systems, electrical and computer engineering, and computer science. 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. Although automation technology is deeply influenced by information technology in that 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.

Formal verification is an important means for validation of software intensive flexible automation systems ([1-4]). To do the verification one needs formal model of the system under study. As the automation systems consist of machinery and of their controllers, both the object dynamics and the control logic have to be modeled in the closed loop. Therefore, any scientific methodology for design and verification of control systems must take into consideration the behavior of the controlled object, along with its distributed and hierarchical structure. Having only a model of the controller is in general not sufficient to prove the correctness of the specifications.

Special focus of our past research work has been on precise modeling of behavior of automation systems and their validation through simulation and formal verification (e.g. [1-2,5-11]). A number of methods and tools have been developed that are united by the approach to modeling that includes closed-loop representation of the control system and modular hierarchical organization of both control and object parts of the model. These tools and methods form the framework for formal methods application graphically represented in Figure 1. The figure shows several possible scenarios, for instance: model-based system design with subsequent code generation, or system analysis that starts with the ready code and generates its model, and is conducted using simulation or model checking, etc. The figure indicates presence of several entry points and several scenarios that is possible to conduct within the framework.

From these experiences we have realized the importance of systematic approach to modeling that should produce models of systems as an integral part of engineering process. Otherwise, if the user is required to develop the

models from scratch, it diminishes drastically the benefits of the approach.

However, the current practice of control engineering is not much aligned with this idea. Re-use of the models is not easy, since there is no systematic way to compose models of complex systems from simpler ones is provided. As a result the modeling every time has to be conducted from scratch that makes questionable the feasibility of practical application of the closed-loop modeling for systems' validation.

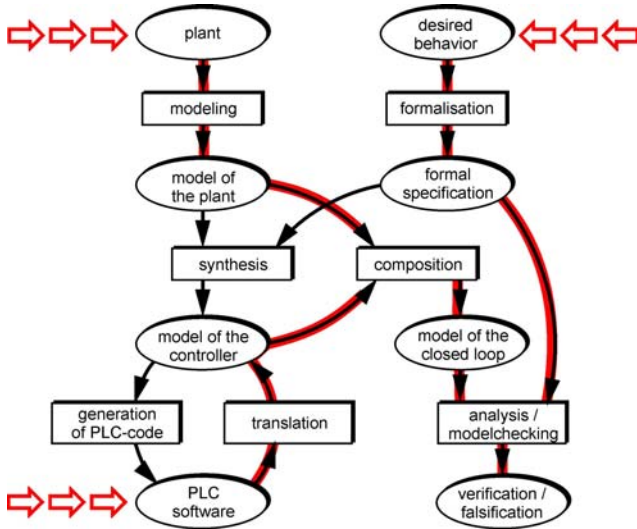


Figure 1. The framework for the formal methods application in industrial automation.

The goal of the recently started research project VAIAS - Validatable Architectures for Industrial Automation Systems (funded by German Ministry for Education and Research and by the industrial partners) is the development of an architecture that attempts to combine:

- an approach to software (and system) engineering following the structure of the original system (object-

based engineering), and

- inherited validatability, that means facilitated application of formal analysis methods through the architecture.

The validatability of the architecture is seen to be reached by:

- the use of formal methods for specification of behavioral and structural properties of the architecture, and
- in particular, by providing the room for the behavioral models of single equipment units, prototype models of controllers and other relevant software components tightly connected to the software components within a single integral architecture.

This way VAIAS intends to meet the new challenges of the automation world providing new software architecture that could better fit to the decentralized reconfigurable nature of automation systems of new generation, will have a higher inherited level of robustness, and will be "friendlier" to formal analysis and synthesis.

There is a number of works on mechatronic architectures and mechatronic ontologies appeared recently. What the majority of these works have in common is the understanding of heterogeneous nature of mechatronic knowledge. Thus, our idea to develop a successful approach to validation of automation systems is to rely on the existing mechatronic architecture, e.g. [12-14], (and tools supporting it) and extend it as needed. This approach could save efforts on handling geometrical, hydraulic, electric, and other kinds of information, and focus on the issues of behavior modeling, embedded control, execution run-time and simulation.

There are dozen of research groups worldwide which work in the direction of object-oriented system engineering in industrial automation, using UML, function blocks, Java as basic ideas ([15-19]). There is also extensive literature on formal methods used in industrial automation, in particular of formal verification. However, less attention has been paid to the combination of both these techniques.

The paper is structured as follows. Section 2 presents a short overview of past relevant works. Section 3 contains informal discussion of some of VAIAS ideas, and Section 4 illustrates these on an example. The issues of formal verification and implementation are not touched, we discussing only the descriptive features of the being developed architecture.

## 2 Architecture building blocks

The combination of simulation, verification and execution is considered in VAIAS as the

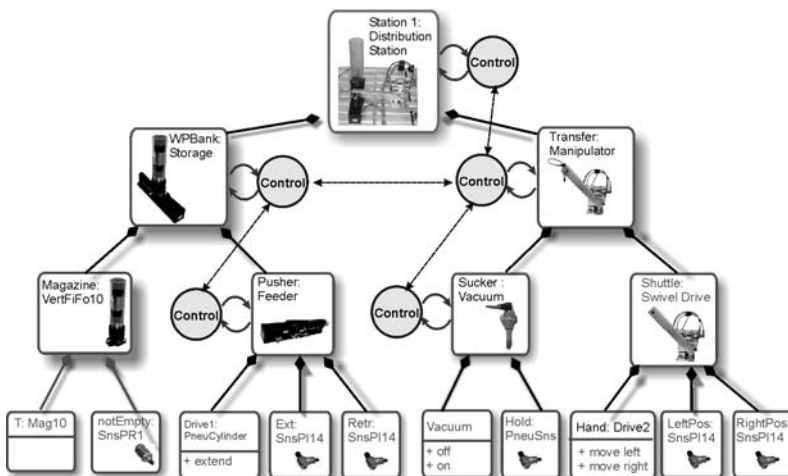


Figure 2. Structural diagram representing hierarchy of automation object components and interaction of their embedded controllers.

key facilitator of control system engineering, eventually leading to faster development of better quality systems. This combination is enabled by the idea of embedded modeling as well as the idea of a single source for models.

The basic building blocks of VAIAS are Automation Objects (AO), which can be *basic* or *composite*. Description of a basic block type does not include references to other AO types. The architecture will be based on the standard IEC61499 architecture at the execution level, and will employ some ideas of UML at the design level.

The VAIAS architecture will consider the model development as an integral part reference component-based architecture for industrial automation systems standardized by the IEC611499 standard. The latter allows definition of device-independent distributed applications and their mapping onto different topologies of hardware devices having different low-level run-time platforms. of system engineering. The models will allow simulation with adjustable precision, formal verification and code development or generation.

VAIAS intends to occupy an intermediate place between the domain-specific architectures, like the mechatronic architectures described in [12], and the general purpose executable distributed component architecture of IEC61499. The engineering process in VAIAS is seen as a gradual refinement of informal requirements until they take the formal shape of executable code that satisfies specifications. Similar to the classic UML-based system engineering, graphic diagrams are used in VAIAS to structure, formalize and specify the requirements.

This process may include the use of model and code generators, as well as of formal validation tools. However, the tool framework of VAIAS will allow manual interaction at each stage of the process. In particular, the following descriptive means will be used:

- Basic automation objects will be described in a multi-layer way, where a layer may correspond to a particular functional characteristic of the object. It is assumed that each automation object may have models of its internal dynamics and models (or program implementations) of the corresponding functionality (e.g. control, visualization, communication, diagnostics, etc.).
- Models of internal dynamics of basic objects can be originally specified as hybrid state charts; The other modeling forms can be derived from this form either manually or automatically.
- Complex automation objects could be described also in multi-layer form, where each layer is specified by means of block

diagrams (Modular View). The overall structure of the object (its structural skeleton) can be presented as an object diagram or as a class diagram. Interactions between the constituent objects can be specified by means of sequence diagrams which can be a source of modular view diagrams.

- The geometrical layout of a composite automation object requires the corresponding data structure, maybe similar to the suggested in [12]. The layout data can be further used as a source data for the components implementing visualization.

Quite natural understanding of the object-oriented system engineering in industrial automation consists in designing software structures whose structure follows the structure of the mechanical units.

In this section this idea will be illustrated on example that is a simple automation system that includes two intelligent mechatronic actors: an automated storage of workpieces, and a manipulator that extracts the workpieces from the storage and passes them to other processing units. The word “intelligent” here means that the corresponding objects have some pre-programmed functionality.

Figure 2 shows the hierarchical structure of this system. Each of the mechatronic actors, in fact, is also complex. Thus, the storage of workpieces consists of a workpiece magazine and of a feeder that shifts forward a workpiece from the lower position of the magazine to the position from where it can be picked by the manipulator. After the feeder returns to its initial position the pile of workpieces in the magazine falls down and thus the lower place again becomes occupied.

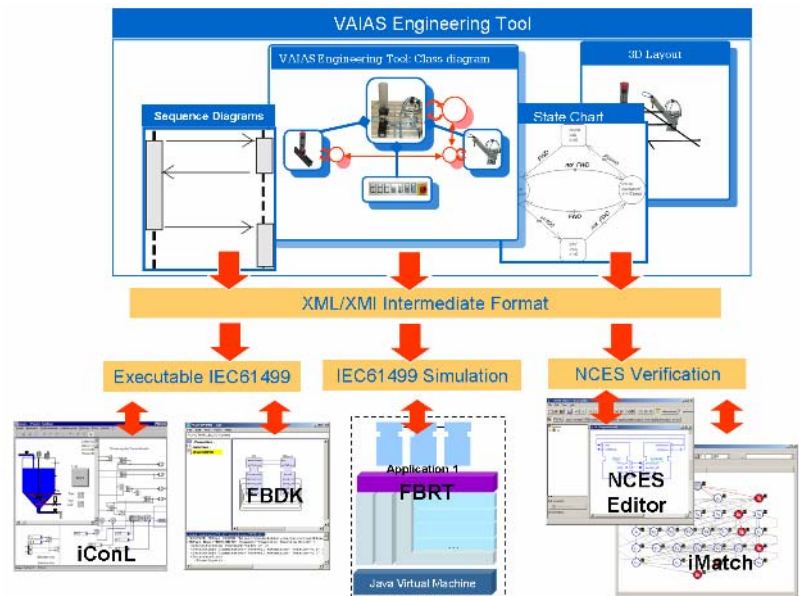


Figure 3. Three application scenarios of the VAIAS framework.

The diagrams like the one in Figure 2 represent the structure of *complex* automation objects. The complex AOs are composed from other complex or *basic* objects. The basic AOs are those located in the leaves of the tree. For example, a pneumatic cylinder is a basic object as it does not include any others.

The concept of the Automation Object is not yet formally defined. There is ongoing work by International Electrotechnical Commission [21] which follows the standards [22, 23, 24]. Here we assume that an AO is a capsule to embody the data and software components relevant to a mechatronic unit (e.g. those implementing its different functions Operation Control, Safety control, Diagnostics, Visualization, etc.). In particular we are considering a layered structure of both complex and basic AOs, where a layer is specified by its functionality (e.g. Control, Interface, Model, Layout, View, etc.). A layer may contain elements of different types, for instance, the modelling layer may contain a generic model in form of hybrid state chart, a simplified discrete model in form of Petri nets, and program implementation of the state chart model in form of source code, say complying with IEC61499 standard [23-25].

The control of each mechatronic actor interacts with the controllers of the included components as it is shown in Figure 2. In turn, it provides some interface for the higher levels of control.

### 3 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 VAIAS architecture are supported by the tool framework as presented in Figure 3.

At first the system is being composed from the constituent “automated mechatronic objects”. Engineering tools allow description of the system’s structure, geometrical positioning of the components, and their interaction with

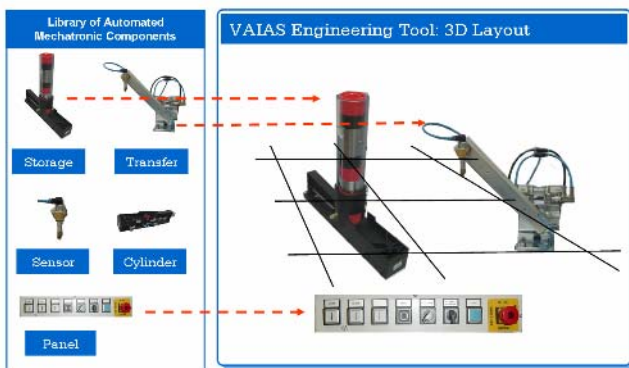


Figure 4. VAIAS engineering tool that allows description of the system by picking components and placing them together in the CAD-like tool.

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 seems to be XML/XMI.

This data will form the input of the three subsequent scenarios:

**Executable system configuration.** The intermediate XMI format is converted into function block executable specification following IEC61499 standard. This form is precise enough but still independent from a particular hardware architecture. The structure of the executable spec follows the MVA/CDA approach (model-view-adapters)/(control-diagnostics-adapters). Further translation to machine-executable form is performed by the corresponding tools, such as the function block development kit (FBDK) or new generation of the iConL tool that will be able of importing the XML function block specifications.

**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 as statistics gathering. In Figure 3 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** good 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.

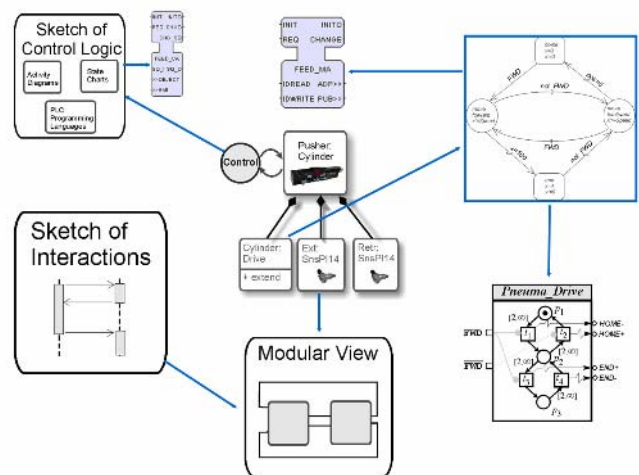


Figure 5. Data representation in VAIAS.



The internal organization of a composite automation object is illustrated in Figure 5. Control of the object can be represented as a *sketch* in a number of supported model forms (e.g. state charts) or more precisely in a programming language out of those traditionally used in automation. The latter form can be derived automatically from the sketch using three model-based code generation software tools.

Figure 6 shows the structure that is good for both executable and simulation software configurations for this system. The corresponding function blocks, selected from the respective Automation Object repositories **STORAGE** and **MANIPULATOR**, are placed together and connected to each other according to the required signal exchange between their respective mechatronic actors. Note, that the Figure does not use the traditional mnemonic of IEC61499 function blocks, instead normal block diagram mnemonic is used.

The configuration implements control, visualization and human-machine interface of the system. The connection between function blocks are simplified by the use of adapter interface mechanism of IEC61499 that allows representation of multiple bi-directional connections by just one line, like in a multi-wire cable. The main advantage of the presented solution is the simplicity of system integration/reconfiguration as the software structure follows the structure of the mechatronic part. What is left to system engineer is just wire/rewire the pre-programmed software blocks.

The idea of smooth integration of independent software components is well visible for visualization part of the system. As shown in Figure 6 the function block "Arm View" is responsible for drawing of only a part of the visualization screen (manipulator), while the magazine's image is rendered by the block "Magazine View". The direct interface components, such as "Magazine", "Arm" and "HMI Panel" can be substituted by simulation model function block that will convert the control application to the simulation application.

The example illustrates the engineering using the Automation Object. The software of the composed system was created by wiring of pre-programmed function blocks supplied for each of the mechatronic actors. The reconfiguration of the mechatronic part of the system (say substitution of the manipulator by a similar one of another vendor) would result in substituting some of the function blocks by their counterparts taken from another "Automation Object" and by rewiring them with connection arcs.

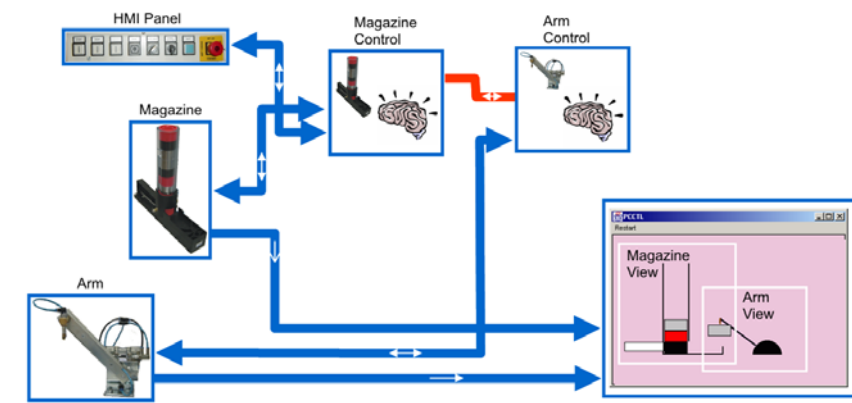


Figure 6. Block diagram of the executable system configuration.

## 4 Acknowledgements

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

## 5 References

1. H.-M. Hanisch: *Closed-Loop Modeling and Related Problems of Embedded Control Systems in Engineering*, 2004, in Abstract State Machines 2004, LNCS 3052
2. 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
3. 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, vol.14, N.1, 2003, pp.123-136
4. Bani Younis, M.; Frey, G.: *Formalization of Existing PLC Programs: A Survey*. Proceedings of CESA 2003, Lille (France), Paper No. S2-R-00-0239, July 2003.
5. 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
6. V. Vyatkin, H.-M. Hanisch, G. Bouzon: *Open Object-oriented validation framework for modular industrial automation systems*, INCOM'2004, Proceedings, Salvador, Brazil, April, 2004
7. Vyatkin V., Hanisch H.-M., Pfeiffer T.: *Modular typed formalism for systematic modeling of automation systems*, 1<sup>st</sup> IEEE Conference on

- Industrial Informatics (INDIN'03), Proceedings, Banff, Canada, August 2003
8. 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, November, 2004
  9. Hanisch, H.-M. and A. Lüder: *Modular modeling of closed-loop systems*, Colloquium on Petri Net Technologies for Modeling Communication Based Systems, Proceedings, pp.103—126, Berlin, Germany, 2000
  10. Vyatkin V.: *Intelligent Mechatronic Components: Control System Engineering using an Open Distributed Architecture*, IEEE Conference on Emerging Technologies in Factory Automation (ETFA'03), Proceedings, Lisbon, Portugal, September 2003
  11. V. Vyatkin, C. Peniche: *How the IEC61499 architecture fits to the requirements of intelligent automation systems?*, 2<sup>nd</sup> IEEE Conference INDIN'2004, Berlin
  12. Jose L. Martinez Lastra, *Reference Mechatronic Architectures for Actor-based Assembly systems*, Thesis for degree of Doctor of Technology, Tampere University of Technology, 2004, ISBN 952-15-1210-5
  13. K. Feldmann, W. Wolf, M. Weber: Development of an Open, Event-based and Platform Independent Architecture for Distributed and Intelligent Control Systems, INDIN'04, Proceedings, pp. 560-566
  14. Vyatkin V., J. LM Lastra: *Architectural Foundations for Reconfigurable Manufacturing Systems*, 3<sup>rd</sup> International Symposium on Open Control Systems SoftSympo'03, Helsinki, September, 2003
  15. M. Bonfe, C. Fantuzzi: *Design and Verification of Mechatronic Object-Oriented Models for Industrial Control Systems*, IEEE Conference ETFA'2003, Lisbon, 2003, Proceedings, vol. II, pp.253-260
  16. K. Thramboulidis, Development of Distributed Industrial Control Applications: The CORFU Framework, 4th IEEE International Workshop on Factory Communication Systems, Sweden, 2002, Proceedings.
  17. Thramboulidis K.S. Using UML in Control and Automation: A Model Driven Approach, 2<sup>nd</sup> international Conference on Industrial Informatics INDIN'04, 24-26 June 2004, Berlin, Germany
  18. W. Zhang, Ch. Diedrich. Relations between Function Block and object-oriented automation application design, Private communication, 2003
  19. Bonfe M., Fantuzzi C. An Application of Object-Oriented Modeling Tools to Design the Logic Control System of a Packaging Machine, 2<sup>nd</sup> international Conference on Industrial Informatics INDIN'04, 24-26 June 2004, Berlin, Germany
  20. W. Zhang, Ch. Diedrich. Comparison between FB-oriented and Object-oriented designs in control, Private communication, 2003
  21. Automation Objects for industrial-process measurement and control systems - IEC SB3/TC 65, Working draft, 2002
  22. International Standard IEC 1131-3, Programmable Controllers - Part 3, International Electrotechnical Commission, 1993, Geneva, Switzerland
  23. Function blocks for industrial-process measurement and control systems - Part 1: Architecture, International Electrotechnical Commission, Geneva, 2005.
  24. Function blocks for industrial-process measurement and control systems - Part 2: Software tools requirements, International Electrotechnical Commission, Geneva, 2001
  25. 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, Proceedings, Cancun, Mexico, September, 2002
  26. Function Block Development Kit, downloadable from [www.holobloc.com](http://www.holobloc.com)