Model-Driven Development of Control Software for Distributed Automation: A Survey and an Approach

Chia-Han Yang, Valeriy Vyatkin, and Cheng Pang, *Member, IEEE*

***Abstract*—This paper presents a survey on model-driven design and validation approaches for distributed automation and control systems with essentially decentralized logic. Driven by the goals of flexibility and performance improvement, researchers have explored several approaches to distributed systems design, including multiagent systems, middleware, and distributed component architectures. This also results in several international standards and reference architectures, such as IEC 61499, OpenRTM, IEC 61804, etc. Verification and validation of distributed systems is another grand challenge. This survey presents methods of using traditional and novel modeling and simulation tools in the context of distributed systems. In particular, this paper then focuses on the developments related to IEC 61499 standard, which displays a range of research directions that aim to fill the gaps in the distributed systems modeling, implementation, and validation.**

***Index Terms*—Distributed systems, function blocks, IEC 61499, modelling, simulation.**

I. Introduction

THE EVER increasing demand for flexibility and reconfigurability of control system in manufacturing and process industries is undisputable, as indicated in many publications, for example [1]–[3]. The requirement to react on the ever changing market demands by producing small quantities of many customized products rather than mass production of a single product [2], [4], and [5] implies modularity and reconfigurability of production machinery and the corresponding modularity and distribution of automation hardware and software.

These trends have been seen in the past but on a limited scale. For instance, the concept of distributed control systems (DCS) has been known in process industries for a few decades. It was influenced by the spatial distribution of the plants. This approach requires the use of field area networks (i.e., fieldbuses) to connect sensors, actuators, and local regulators with a centralized control unit implementing a control algorithm. However, while increasing the flexibility of hardware maintenance, this traditional DCS approach has little to do with flexibility of production.

Manuscript received December 6, 2011; revised June 12, 2012 and March 31, 2013; accepted May 13, 2013. Date of publication November 13, 2013; date of current version February 12, 2014. This paper was recommended by Associate Editor T. I. Strasser.

C.-H. Yang is with the Centre for Autonomous System (CAS), University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: yang.ch.john@gmail.com).

V. Vyatkin and C. Pang are with the Luleå University of Technology, Luleå SE-971 87, Sweden.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCC.2013.2266914

In the meantime, the discrete manufacturing industries are facing similar market challenges. There was a substantial body of research on the use of so called "intelligent mechatronic components (IMC)" in order to improve the flexibility of the production systems [6]–[12]. Such components, possessing individual functionality or services and equipped with embedded control devices, can be aggregated into automated machines and systems, arguably easier than traditional mechanical and mechatronic components, also enabling easier reconfiguration of systems composed thereof. The use of such intelligent components promises essential benefits for design and reconfiguration of automated production systems thanks to encapsulation and reuse of a good deal of intellectual property relevant to a particular mechanical component, machine or system.

Combination of the decentralized control logic with its distributed deployment results in a new approach to automation that is often referred to as distributed intelligence (DI). In the Webster's dictionary, "intelligence" is defined as "the ability to learn or understand or to deal with new or trying situations." However, in the industrial automation and control context (e.g., in [13]) this word is often used to describe any system with decentralized logic, as opposed to another kind of distributed architecture, where logic is executed on one computer device, but sources of data are distributed (e.g., a programmable controller with remote I/Os connected via a fieldbus). The DI approach relies on decentralized control hardware architecture with multiple controllers in charge of individual mechatronic devices or assemblies thereof. These controllers may communicate and collaborate with each other through common communication channels such as Ethernet and field area networks.

The existing design paradigms have shown their severe limitations when it comes to implementation of the DI concept in industry. The limitations pertain to all phases of system engineering, from requirements formalization, software construction, verification and validation (V&V), dependable execution, and maintenance.

This paper provides a summary of various design and validation concepts that are related to distributed control or can help in achieving it. To this end, it also surveys modeling frameworks that are used for V&V of complex automation and control systems (ACS).

The rest of this paper is structured as follows. Section II summarized the various sources of DI in automation systems. This discussion is followed by surveys in three major streams

related to distributed systems (Section III), model-driven software engineering (Section IV), and model-driven V&V (Section V). Section VI examines how the trends from all these streams have been addressed in the international standard IEC 61499. Section VII presented the elements of IMC architecture that used IEC 61499 as an enabling technology and attempted to combine best practices from the surveyed engineering and control validation streams.

II. Sources of Distributed Intelligence and Model-Driven Design and Validation

Even traditional centralized automation systems may include at least four data processing device types communicating via networks: engineering station with simulation, database and programming software, human–machine interface (HMI) device with visualization and human interface software, programmable logic controller (PLC) with control software and intelligent sensors or actuators, e.g., motor drives implementing motion control functionality. Ensuring correct operation of these components requires addressing the issues of correct coordination, synchronization, and access to shared resources. The situation aggravates when several controllers need to be integrated. There is vast theory of distributed computing which addresses these issues. To mask the complexity of distributed systems development [14], many distributed programming languages [15] have been developed by computer scientists; however none of these is implemented on the existing PLC platforms. Methods interconnecting PLCs using middleware, for example [16], or DCOM based PROFInet-CBA [17] lack system-level view that is needed to validate properties of entire systems rather than parts thereof. The fact that control hardware and software in a distributed system may come from different vendors implies portability and interoperability requirements. These and other reasons motivate the need of higher level software models similar to those used in general purpose software development, e.g., unified modeling language (UML) [18] and its derivative, SysML [19].

With the growing complexity of PLC controlled systems, the software complexity is growing as well. To keep up with the timing requirements, it is often needed to distribute software across several concurrently running PLCs. It would make sense to design automation applications as logically distributed modular software systems with components coordinating their actions only via message passing. The next step would be to convert transparently modular organization of automation software to virtually and physically distributed configurations. However, the existing PLC programming architecture (standardized in the form of IEC 61131-3 standard [20]) does not provide such mechanisms.

The International Electrotechnical Commission (IEC) has addressed these issues in the IEC 61499 standard [21] by defining a reference architecture for DCS design using event-driven modules called function blocks (FBs). This architecture enables application-centric and vendor-independent design while achieving flexibility in terms of both software and hardware. According to [22], there is a misconception about IEC 61499 that it lacks higher level constructs convenient for distributed logic design, such as synchronous and asynchronous messages of UML, but these can be easily implemented as design-time structures in the corresponding tools.

Along with the software construction challenges, another main challenge of distributed systems design is their V&V. When controllers are independent of each other and distributed across the system network, the communication intensity between the controllers would certainly increase. This complicates their testing. Even though the control design is more manageable through the software module concept, it is still challenging to grasp the overall behavior of the distributed system without computer-aided verification process, especially when each controller in the system network is designed by a different developer. The software design environment with advanced validation and verification capabilities that can tackle challenges of distributed systems design is essential for the successful implementation of systems with DI.

Closed-loop modeling and simulation is standard in control engineering. The plant model describes the behavior of the physical system. The controller sets control inputs of the plant based on the control algorithms and readings of the plant's sensors. According to [23], the benefit of V&V using closed-loop models is as follows: "The validation of controller design by itself has no meaning and does not guarantee the correct behavior of the systems. This simple truth has often (and is still) misunderstood or even neglected. In fact, from verification perspective, for example, no liveness property can be proven by open-loop model."

Formal V&V are techniques complementary to manual debugging and simulation based verification. The idea of formal verification is to prove rigorously (with the help of software tools) that certain properties hold in the execution of a control system. In several recent works [24]–[29], the closed-loop concept has been also brought into the formal verification.

Another major challenge comes from the system engineering side. In order to be used in industry, the perceived switching cost to the new distributed architectures needs to be less than the perceived benefit. The costs of the change can be very substantial especially in restructuring and retraining to familiarize with the new design approach and new design tools [30]. This problem also leads to an idea of linking existing tools and languages with the new ones.

One can conclude that there are three sources of knowledge used to address inherited complexity of distributed automation systems design. These include theory and practice of distributed systems design, model-driven software engineering and traditional control-engineering approaches that imply software and hardware "in the loop" validation and block-diagram model and code organization. The developments related to these streams will be surveyed in the three subsequent sections.

III. Distributed Control

A. Challenges of Distributed Implementations

The idea of the truly DCSs originates from the control implementation in process industry. This is where each physical element such as heater, motor, pump, or valve is directly

connected in closed-loop to its own automatic control unit with possible start/stop activation from a central controller. There is a belief that the role of central controller can be minimized and distributed control nodes can achieve the same control goal communicating in a peer-to-peer manner, making the entire system more flexible and reliable. In [1], the reconfigurability was considered as the motivation of distributed control.

Although various applications have been deployed based on the distributed logic concept and the approach is confirmed to be useful, the widespread adoption of this concept by industry is still low. The following challenges have been identified as the main reasons for that:

1) the risks that accompany every new technology that has not been proven in large scale industrial applications [31];
2) lack of mature enough design and development tools for industrial development [31];
3) paradigm misunderstanding due to the small number of successful industrial applications [32];
4) increased complexity of the software structure. This is due to the fact that distributed architecture requires each distributed node to have accurate status of the environment and corresponding mechanisms have to be introduced to guarantee the correct functionality and reliability of the system [33];
5) lack of industrial recognition of the potentials of the new technology, due to insufficient publicity of successful industrial projects [21].

According to [34], early implementations of distributed automation systems involved splitting the control program into separate pieces while keeping essentially the same code structure. This code can be distributed physically throughout the hardware linked in a communication network. Custom joining code would then be written to knit the smaller components into a complete system. While this approach allowed certain degree of physical distribution, there was no notion of logical separation of system's functionality. Thus this would give the same result as a tightly-coupled computer cluster [35] that runs a single process.

The complexity of distributed systems design made the researchers looking for self-organization mechanisms, that has given rise to the agent-based control concept [36] discussed in the following section.

B. Multiagent Systems (MAS)

Therefore more modern techniques have been investigated, describing the notion of an intelligent, autonomous and goal-oriented agent in a MAS [37] and how this agent-based concept is used in production or manufacturing systems [38]–[40]. An agent is a concept that represents an autonomous combination of software and hardware interacting with the environment. A MAS is composed of multiple agents communicating and working together in order to achieve a common goal. An agent-based architecture provides robustness and flexibility and is proven to be specifically appropriate for dynamic distributed systems [41]. An agent-based system may include both local and global controller agents that collaboratively process the information obtained from sensors and generate control reactions. As an example in process industry, this approach is proven to be useful when dealing with a system of networks of interconnected continuous stirred tank reactors (CSTRs) [36]. A framework for modeling agent-based control of service-enabled manufacturing systems is presented in [42].

In a MAS, each single agent exchanges information with others in order to achieve its own objectives. The functionality of agents is usually distinguished as high level control or low level control such as controlling subjacent physical machines. There are some successfully adopted applications based on MAS in various industries, for example, steel rod bar mill of BHP Billiton in Melbourne [43], distributed control of ship equipment in U.S. Navy shipboard systems [44] and production control of semiconductor wafer fabrication facilities called FABMAS [45]. The low level of those systems comprises device-specific implementations with IEC 61131-3 compliant PLCs communicating via technologies such as distributed component object model (DCOM) and Ethernet. Works [32] and [46] present examples of modeling distributed agents communicating through a network for simulation purposes.

Another research stream has been dealing with making MAS more intelligent based on biological principles, for example, the holonic manufacturing concept [41], [47]. The work [41] presents some case studies of MAS applications in process industry, including an intelligent search system to provide a knowledge management platform and a system to provide concurrent process design to ease communication between system engineers.

C. Summary of Findings

In summary, one can conclude that existing PLC-based architecture is not sufficient to achieve distributed automation as it is insufficiently addresses various design transparencies and introduces too much overhead in execution and design. The attempts to address the coordination and self-organization issues via multiagent architectures are promising, but require the corresponding support in the lower level architecture, i.e., an agent-ready next generation PLC.

IV. Software Models and Architectures

A. Model-Driven Software Development

Model-driven design (MDD) is a dominating technology in general software engineering. In control and automation, the specific of this approach is that software objects are often associated with some components of the controlled plant.

Bonfe and Fantuzzi [48] have introduced the use of UML in automation and Thramboulidis [49] in particular in the IEC 61499 context. The latter work proposed generation of FB from UML diagrams, while Dubinin *et al.* [50] proposed the UML-FB architecture supporting round-trip engineering of UML diagrams generation from FB designs and vice versa.

One characteristic concept in the mentioned works is encapsulation of hardware and software models of an ACS into a single design artifact, which can be further reused by composition to more complex artifacts. In particular, the

software model can be structured following the mechanical and functional structure of the hardware model.

For example, the concept of automation object was proposed in [7] as an abstraction unifying a mechatronic component, an embedded control device, and a software component. When designing a new ACS, the automation objects modeling the components are selected from a repository and then hierarchically composed following the desired physical structure. The resultant automation object becomes the central knowledge base for subsequent MDD phases. The automation object notion has been subsequently extended and referred to as IMC in [23], where the closed-loop modeling methodology and corresponding model transformation approaches are introduced. Each IMC is internally organized following the model-view-controller (MVC) design pattern [51]. Different languages and formalisms are employed to develop the domain specific models. For instance, MATLAB/Simulink can be used to create the hybrid model capturing both continuous and discrete dynamics of the IMC while IEC 61499 is used to model the executable behavior, which can be deployed to the controllers directly. The net condition/event system formalism [52] was adopted to define the IMC's formal model for the purposes of formal verification.

Sünder *et al.* [53] presented a similar idea, where the automation component concept as a variant of automation object is proposed. The concept includes a universal component interface and unified hierarchical architecture allowing flexible reconfiguration of ACSs. This model was later adopted and extended in the MEDEIA project [54]–[56], which developed a meta-design architecture allowing domain specific knowledge to be expressed in its native form and then unified into the generic automation component model. Thus, model transformation is reduced to the bidirectional transformation between the domain specific models and the automation component model.

Thramboulidis [57] introduced the model-integrated mechatronics paradigm for the model-driven concurrent engineering of ACSs. The core of this paradigm is the mechatronic component construct, which is the composition of a mechanical part, an electronic part, and a software part. The MDD process following this paradigm starts with the modeling of mechatronic components in the mechatronic layer, which is vertically projected to the application layer for modeling the control software; the resource layer for modeling the control system infrastructure; and finally the mechanical process layer for modeling the mechanical composition. The UML profile introduced in [58], [59] is used to support this MDD process to generate the control application developed in IEC 61499 FB.

A similar MDD idea has also been implemented in the AUKOTON research project [60]–[62]. In AUKOTON, the proposed UML automation profile is used to unify the domain knowledge into a platform independent functional model, from which PLCopen control code is generated.

SysML is a UML derivative for engineering applications that is getting increasingly popular. In particular, SysML supports such design phases as requirements capturing and formalization of specifications. Hirsch and Hanisch [63] and

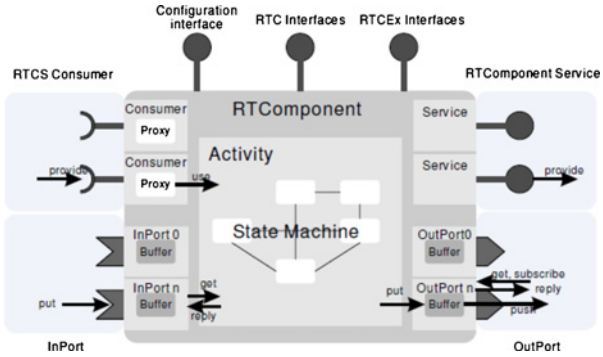


Fig. 1. Proposed architecture of RT-component model [67].

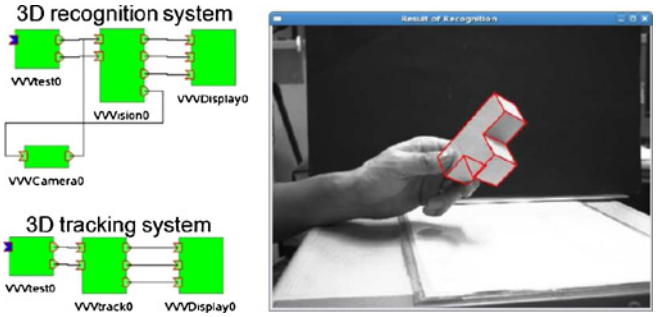


Fig. 2. 3-D recognition and tracking using the proposed OpenRTM development tools [67].

Hirsch [64] provide a pathway for linking FB technology with SysML. Thoma *et al.* [65] use SysML to model fault centric system for reliability testing.

B. Block-Diagram Design Languages

The block-diagram way of thinking is traditional in control engineering, and there are a number of languages and tools that support it. For example, OpenRTM (Robotic Technology Middleware) is proposed for component-based robot system integration [66]. The component's model is shown in Fig. 1

The functionality of the RT component (RTC) is as follows:

1) component metadata for dynamic component assembly;
2) component action and execution context for business logic execution;
3) data ports for data exchange between RTCs.

There are already various industrial application built based on this framework, including 3-D recognition, tracking, dynamic simulation, learning systems, etc. Fig. 2 shows the 3-D recognition and tracking implementation example based on OpenRTM. However, despite a number of applications, this concept has not been formally standardized and is so far only applied in the robotics related research projects.

The IEC 61804 standard draft [68], describes the specification and requirements of distributed process control systems based on FB [69]. The electronic device description language (EDDL) is the language that is stated in part 2 of IEC 61804 specification and describes the properties of automation system component [68], such as vendor information, version of firmware/hardware, data format, etc. Through this language, all the information will be carried between

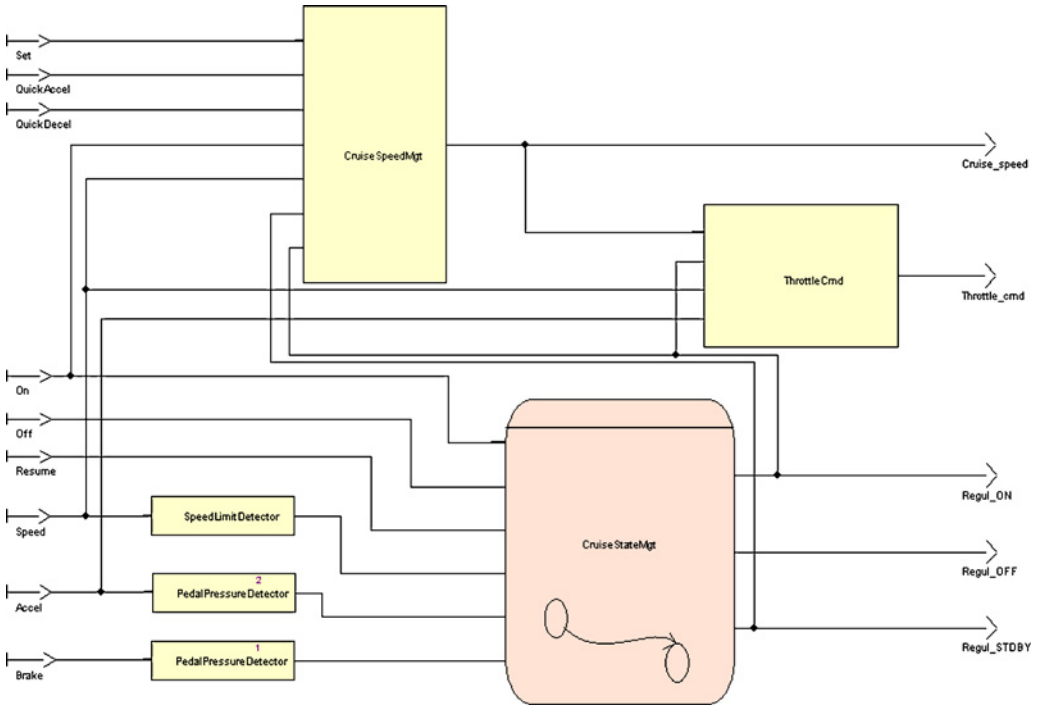


Fig. 3. Block-diagram design in SCADE.

the devices (controllers, sensors, actuators and engineering stations, etc.) by a fieldbus. This language fills in the gap between the FB specification and product implementation by allowing manufacturers to use the same description method for devices of different technologies and platforms. The FB design of a process control system for example, is only an abstract representation which may be implemented differently with different device types [69], such as field devices (FD), PLCs, visualization stations, and device description (DD).

There are also other popular modeling and simulation tools such as LabView [70] and SCADE . These tools all follow the block-diagram based modeling approach. For example, SCADE is a model based environment mainly used for development of safety critical embedded software, a product of Esterel technologies. With native integration of the synchronous Scade language and its unified formal notation, SCADE combines unique features for integrated design of safety critical applications. It covers requirements management, MDD, simulation, verification, certified code generation, and provides interoperability with other development tools and platforms. SCADE combines a number of models, such as hierarchical state charts and block diagrams as seen in Fig. 3.

However, despite strong marketing efforts of LabView and Esterel technologies, their presence in the industrial automation domain is marginal, partially due to the lack of support for automation legacy ways of design.

C. IEC 61499 Function Blocks

The IEC established the international standard IEC 61499 [21] to provide a reference software and system architecture for truly distributed control design. This architecture enhances the interoperability and reusability of components, aiming at increasing flexibility and reconfigurability of the control systems.

The IEC 61499 standard introduces the concept of event-driven modules, known as FB, to address the increasing demand of flexibility, reusability, reconfigurability, and distributed control applications [72]. The standard provides a graphical method for control flow design, and allows reuse of algorithms written in the legacy PLC languages. With such distributable modules, it enables application-driven design, i.e., full specification of behavior to be completed before considering hardware layout. This is very different from the traditional device-driven design approach with PLC where hardware layout needs to be considered prior to the control software implementation.

A number of works explored implementation of multiagent control with IEC 61499 distributed architecture. The fully distributed approach to baggage handling systems (BHS) automation was demonstrated in [73] and [74]. A hierarchical multiagent architecture based on IEC 61499 which enables elements of self-configuration in manufacturing systems was developed in [75], and [76] investigates the use of IEC 61499 to implement multiagent control in material handling systems. The work [77] discusses the architectural solutions for joining IEC 61499 lower-level agents into upper multiagent manufacturing platform. In [78] multiagent control for SmartGrid automation was reported.

V. Concepts and Environments for Modeling and Simulation

A. Concepts

This section presents some concepts for modeling of distributed ACSs. This discussion is followed by a survey of several modeling tools.

In general, many automation systems cannot be classified as purely discrete or purely continuous. For example, in process industry, it may contain discrete operations such as sequences of valves openings along a pipeline, as well as continuous control of flows or of some chemical reactions. Such systems are also known as hybrid [79]. In process industries, a batch processing is one example of such a system. The process in the batch process reactor, for example, can be described by both continuous variables (e.g., temperature) and discrete variables (e.g., switches). Batch processing was introduced first in the production of high value, low volume products, such as pharmaceutical, cosmetics, and perfume products, and spread gradually to the food processing and other industries. Modeling, verification, and validation of systems with hybrid, i.e., continuous and discrete dynamics, executing intelligent control algorithms in decentralized nodes, are highly sophisticated. A big challenge will be incurred when introducing truly distributed control approach into such process control systems.

Another emerging modeling and design concept is called cyber physical system (CPS) [80]. A CPS is a class of systems with a tight coupling and coordination between the physical and computational elements. Thus the physical processes of the system are monitored and controlled by their corresponding computational processes. The abstractions available in modern computing and software engineering require significant advancement before they allow for full description of a CPS. One example is lack of capabilities of modeling adequately concurrent physical processes.

One more trend in modeling complex systems is represented by the system of systems (SoS) concept. SoS are differentiated from large, complex, but monolithic systems in several properties, which were first introduced in [81] and [82], and they are stated as follows:

1) operational independence of the constituent systems;
2) managerial independence of the constituent systems;
3) geographic distribution;
4) emergent behaviors;
5) evolutionary and adaptive development.

Zhou *et al.* [82] introduced the SoS concept to the domain of industrial automation, having demonstrated its value for system-level parameters estimation.

B. Modeling Tools

MATLAB is a numerical modeling environment developed by MathWorks [83]. It allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, and Fortran. Although MATLAB is intended primarily for numerical modeling, there are toolboxes available, allowing accesses to symbolic computing capabilities. An additional package, Simulink, adds graphical multidomain simulation and MDD for dynamic and embedded systems. Stateflow is a package provided within Simulink, providing customizable block described in a form of finite state machine (FSM).

MATLAB provides a well-developed environment for validation and verification of models. For example, *CHECKMATE* is a MATLAB-based tool for simulation and verification of hybrid systems with nonlinear continuous dynamics in Simulink environment [84].

The hybrid system modeling language (HSML) is created specifically for modeling hybrid systems using state/event and discrete time modules in MATLAB and SIMULINK [85], [86]. In [86], it is indicated that HSML is particularly useful for time and state-event handling. There are also some other software tools for simulation and verification of different types of hybrid systems, such as HyTech [87] which uses symbolic model checking techniques in continuous state space to verify systems modeled with linear hybrid automata (LHA), and VERDICT [88] that provides an environment for modular modeling and simulation for timed and hybrid systems.

The Ptolemy II modeling environment also aims at modeling and simulation of hybrid systems [89]. Here, a hybrid system can be modeled by a FSM with continuous time (CT) models. Ptolemy II is very similar to Simulink as it is also a graphical tool and can build system models by using a network of block-diagram representation. Another claim of Ptolemy II is its ability to model cyber-physical systems.

Modelica is an object-oriented, equation based language for modeling physical systems (e.g., mechanical, electrical, electronics, thermal, etc.) [90]. The free Modelica Standard Library, developed by The Modelica Association, contains approximately 1280 generic model components and 910 functions in various domains (Version 3.2). This language is used in various modeling and simulation environments and software tools, such as OpenModelica [91], MapleSim [92], MathModelica [93], etc. In these modeling environments, visual component modeling is also supported.

C. Simulation Environments for Distributed Systems

Santos *et al.* [94] present an industrial case study of a distributed continuous process simulation of a beet sugar factory. This simulation work is done by using DCOM components written with a modeling language called EcosimPro.

Another example is distributed simulation (DS) toolbox for MATLAB [95]. The DS Toolbox for Simulink and Stateflow enables the realization and simulation of distributed Simulink or Stateflow models. It provides blocks with the same structuring functionality but with additional features for parallel and DS: subsystems are handled as black-boxes in the master model and are implemented and simulated in separate Simulink instances (slave models) on the same or even on distant computers. The user can create their models in the common way and distribute these on several computers which are interconnected via a standard network. During the simulation all connected models on all computers run truly in parallel (co-simulation).

Mahalik and Kim [96] address specification of requirement, design, and development of a hardware-in-the-loop simulation (HILS)-based tool for the configuration, validation, and management of DCSs. The tool supports modularity, flexibility, user-friendliness, and multiuser capability. The utility of the developed tool is tested through case studies with two exemplar platforms such as a printed circuit board drilling machine and semiautonomous mobile robotic systems. In [97],

this paper is extended by support of virtual DCSs capturing specification, methodology, and prototype design and capable of providing services to the proposed management layer that integrates simulation platform, a top-level ware, by using which distributed control network design can be achieved.

D. Common Problems and Challenges

According to what has been described above, there is still no single validation tool that for both simulation and formal verification of DCSs. It would be beneficial if a software package contained preset models specifically for process control (i.e., models for pipe-line, valve, etc.). The tools handling hybrid systems only work with their own modeling language and do not support code generation for implementation on distributed hardware. A possible way of achieving this is suggested in [23].

Most of the tools described above can handle hybrid systems, but they are not intended to deal with systems with decentralized intelligence. The FB architecture of the IEC 61499 standard, discussed in the next section, can possibly fill this gap, as it was established specifically to handle decentralized design with direct deployment functionality.

VI. IEC 61499 Distributed FB Architecture

A. Requirements

As a conclusion from the previous review, one can state that development frameworks for distributed systems have to support their efficient design and validation by providing the following transparencies:

1) description, coordination, and deployment of distributed processes;
2) easy conversion of component organization to distributed organization (i.e., from virtually distributed to physically distributed systems);
3) support of model-driven software engineering;
4) support of legacy design approaches;
5) ability to implement various functionalities of automation systems;
6) description of plant-controller closed loop systems required for implementation of Hardware in the Loop (HiL) or Software in the Loop (SiL) simulation configurations.

In the following parts of this section it will be shown how these requirements are addressed in the IEC 61499 architecture as compared to the traditional PLC architecture.

B. Addressing Requirements and Comparison with State of the Art

IEC 61499 is an open standard aimed at supplementing IEC 61131-3 [20] standard by adding modern design features and hardware abstraction.

It is possible to build distributed systems with IEC 61131-3 PLCs, but with much increased performance and design penalties and overheads [98]. These include longer engineering time to work out the ownership of the output modules and heavily increased communication overhead time between PLCs for data exchange over field bus, etc.

Other advantages of IEC 61499 over PLCs include the following.

1) Transparent mapping of IEC 61499 FB applications to different devices is enhanced by event-driven control flow definition and supported by tools which insert the required communication code as required. Therefore hardware configuration of controllers may be selected at the end of development process. In the case of typical PLC development, knowing the exact layout of PLCs and connected I/Os is mandatory at the beginning of development.
2) The existence of global variables in PLC programming languages hinders reuse of components.
3) The order of components invocation within a PLC scan is implicit. There are mechanisms to explicitly define the order, but these require manual change of the order in case if a new component is inserted or deleted, increasing the risk of introducing errors.
4) HMI elements can be encapsulated into software components along with control logic, making the generation of the entire system HMI simpler. This opportunity has been demonstrated in the nxtStudio tool [99].

C. Pilot Applications of IEC 61499

The benefits of the FB architecture are being explored by researchers both in discrete manufacturing systems and in the process industry [100]. For example in [101], specific distributed process control programming tools for FB description were developed, and the problems occurring when introducing this new standard into the process control domain were investigated. As a starting point, researchers integrated models to a lab-scale model of batch process such as the FESTO mini pulp process (MPP) model. From the results, the authors of [101] are seeking a migration path to this recently developed standard for the distributed batch process industry and are attempting to exploit the IEC 61499 model in the batch process. Here, a hybrid approach of integration IEC 61499 with UML is explored to address the current trends in software engineering such as component based and model driven development [102]. This approach aims to transform and reduce switching cost from the ISA SP88 [103], an industrially accepted family of standards in batch control, to IEC 61499.

Also, the work [30] has specifically exploited the possible migration path to IEC 61499 standard for the distributed process industry by considering switching cost. It stated that the adoption of this new standard is only possible if the perceived switching cost is less than the perceived benefits. From their previous experiments with professionals, the switching cost is very high due to the bewildering range of design decisions. Therefore direct adoption in the context of IEC 61499 cannot be applied successfully. Their proposed solution is to use SP88 standard as a specification and set up formal rules or general guidelines to construct corresponding IEC 61499 blocks [30], 104]. It is also suggested the component based approach for the batch process industry presented in [105] may ease the

adoption. Even though switching cost is highly reduced as a result, this approach introduces retraining cost. Therefore improving the industrial acceptance of IEC 61499 in industry still remains to be a challenge.

D. Tools

This section describes some design environments and design approaches developed for IEC 61499 FB. Over the years, several development tools and systems complying with IEC 61499 were already presented to and even introduced into the market. These tools include:

1) FB Development Kit (FBDK) [106];
2) Engineering Support System CORFU [107];
3) 4DIAC IDE [108];
4) ISaGRAF workbench [109];
5) nxtStudio [99];
6) Synchronous Compiler [110];
7) Cyclic run-time [111].

FBDK is one of the earliest well-developed tools for FB development. Even though it is considered mainly as a research tool, it is capable for demonstrating various benefits of FB in practice, such as system-level modeling of distributed systems and code deployment. The tool is Java-based and relies on a Java-based run-time called FB run time (FBRT) in execution.

CORFU is an Engineering Support System that extends the IEC 61499 model to cover requirements specifications using UML. Thus CORFU adopts a hybrid model-based approach for the development of automation control systems that integrates UML with the FB concept.

4DIAC is another pioneering open-source tool in FB development that compiles FB applications for execution on a C-based run-time called FORTE. Both FBRT and FORTE can be used in this tool for code deployment.

ISaGRAF [109] is tool based on a combination of IEC 61131-3 and IEC 61499 standards. Due to this fact, the specification of the FB execution model is based on cyclic execution, similar to that the IEC 61131-3 PLC programming environments.

The NxtStudio tool is developed by NxtControl [99]. The tool uses a customized FORTE run-time. The tool introduces a novel composite automation type (CAT) that is a FB including visualization functionality for visualized simulation purposes, by directly following the MVC concept.

The Synchronous Compiler compiles FB applications into C code. This compiler is based on the synchronous execution model [110] and is proven to be very efficient in terms of the target code performance [112]. It can be very useful in the context of distributed control where the model can sometimes be big in size and resource consuming in simulation. Such models can be transformed to the open IEC 61499 form to be run efficiently on distributed or centralized platforms, as proposed by Yang and Vyatkin [113].

E. Gaps

Great expectations for the IEC 61499 technology have been partially cooled down by its slow adoption in industry and a number of technical issues discovered through pilot implementation and related research.

The slow adoption can be explained by such factors, as relatively small share of systems requiring essentially decentralized logic, as compared to traditionally centralized control. Changing the design paradigm from centralized to distributed is a way to handle the design complexity, but requires overcoming quite steep educational curve. Tools supporting the new standard have to be mature enough to compete with PLC tools that have been refined through decades. This creates a classic "chicken and egg" situation; there is insufficient investment to polish the tools, which is possible only in real-life large scale development activities.

The implementation of the interoperability promise of IEC 61499 tools depends on the availability of libraries of communication FB for particular protocols. The tools existing on the market support communication via TCP/IP protocol, along with Profibus, EtherCAT, Modbus, and CIP. These libraries are usually platform dependent, but can have uniform interfaces across different platforms.

The portability of IEC 61499 applications is secured through a number of measures, such as XML based representation format, as well as run-time environments with standardized services and interfaces which mask platform dependencies like virtual machines. Nevertheless, 100% portability between existing tools has not been achieved yet, mainly due to the fact that real need for portability has not yet been demanded by the customers due to low market penetration.

There have been certain semantic gaps in the first edition of IEC 61499 standard as a consequence of the way of finding compromise between several industrial players. For example, the message passing communication in distributed systems creates a fundamental determinism challenge. It is modeled in IEC 61499 by using the event abstraction. However, its implementation at the low level can be done in different ways. In the second edition of IEC 61499 the semantic gaps have been substantially reduced.

Detailed discussion of the gaps is clearly outside the scope of this survey. They are being discussed in the research publications [114]–[116] and solutions get implemented in the newer generation of tools [117]. Therefore, in the authors' opinion, the IEC 61499 architecture represents a sound base for bringing the MDD to the distributed automation practice.

VII. Synergy of Model-driven Concepts

As it can be concluded from the previous discussion, there are attempts to address design complexity of automation systems both in terms of model-driven software design, and in terms of their model-driven V&V. In this section, an attempt of synergy of these activities is presented that results in the concept of IMC architecture. An IMC is composed according to the MVC pattern described in the following subsection.

A. MVC Design Pattern

MVC design pattern [118] is adapted by Christensen in [51] to the domain of industrial automation and integrated with the IEC 61499 standard architecture. According to the MVC pattern as indicated in Fig. 4, software is organized from two

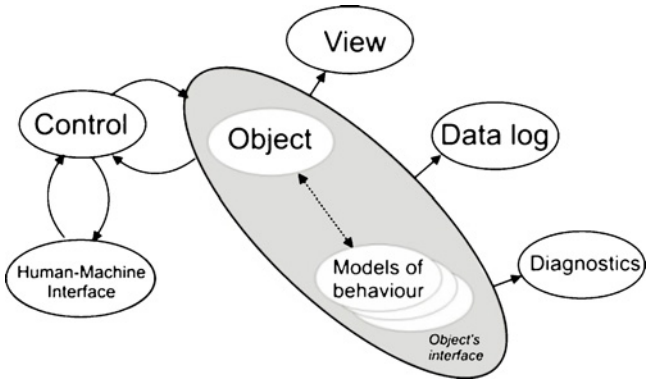


Fig. 4. MVC design pattern architecture [23].

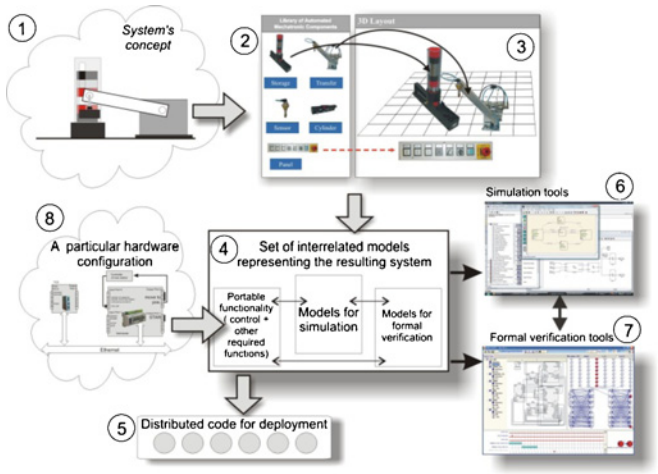


Fig. 5. Sketch of the IMC-based engineering framework supporting the integrated validation [23].

core interconnected components and several auxiliary ones. The core components are:

1) *Autonomous* (*low-level*) *Controller*: implements a set of operations, published as services to be used directly or by higher level controller-coordinator, and
2) *Object*: provides an interface to the input/output signals of the IMC, or to one of the behavioral models included in its IP repository.

The behavioral models, provided in a repository, can be used for verification of the IMC's behavior, or as building blocks for creating the behavioral model of the composite system. Identical interface makes the model component interchangeable with the object component, thus providing an easy pathway from simulation to deployment. The combination of these two functions enables simulation of the system in closed-loop with the actual, ready for deployment control code. Moreover, the simulation model is created with a high degree of components' reuse.

Additionally, the view component supports interactive simulation by rendering the system's status based on the parameters provided by the model. It also can be reused in different deployment scenarios. Being connected to the real object instead of the model, the view component will render the object's status in real time. Other functional components, such

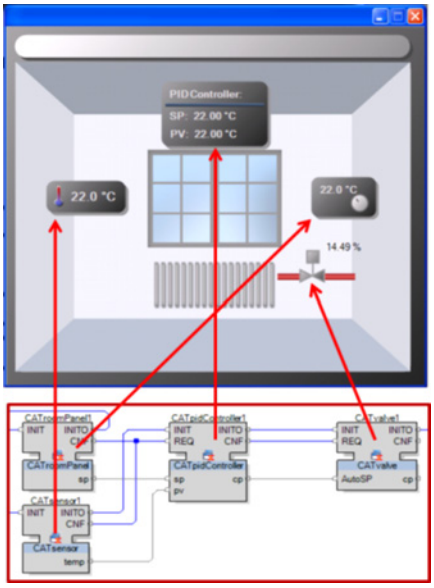


Fig. 6. Visualization of distributed control implemented with nxtControl [99].

as diagnostics and database logger are also fed by the data from the object or the model. In contrast, the HMI component is connected in the closed-loop with the controller.

B. Intelligent Mechatronic Components Architecture

The intelligent mechatronics component (IMC) architecture was proposed in [23] as an attempt to address both design and validation challenges of distributed systems. The idea is to allow the developer thinking in terms of machines or their autonomous parts thereof by increasing the level of abstraction. IMCs are structured according to the MVC pattern. As a result, distributed controller and simulation model can be automatically generated for a system composed of independent modules in a drag-and-drop design environment.

The MVC pattern allows precise closed-loop simulation and formal verification of complex mechatronic systems by reusing models of their constituent parts. Vendors of devices, machines, or components following the IMC architecture will provide not just the controller program code but also the model of the components. The component models will be capable of communicating and exchanging data with one another. This allows end users to immediately establish a model of the system built from the collection of such components, for validation and verification purpose. An overview of the design flow can be seen in Fig. 5.

As a result of using this framework, the model of the system's behavior can be designed with high degree reuse of the component models.

C. Use Case Examples

Distributed controls of BHS have been implemented in the authors' research group. This include a simulation for real-time tracking based on time prediction [73] and also using ISaGRAF tool with an OPC server (see Fig. 7) [74]. Both examples shown here are the representations of small airport BHSs.

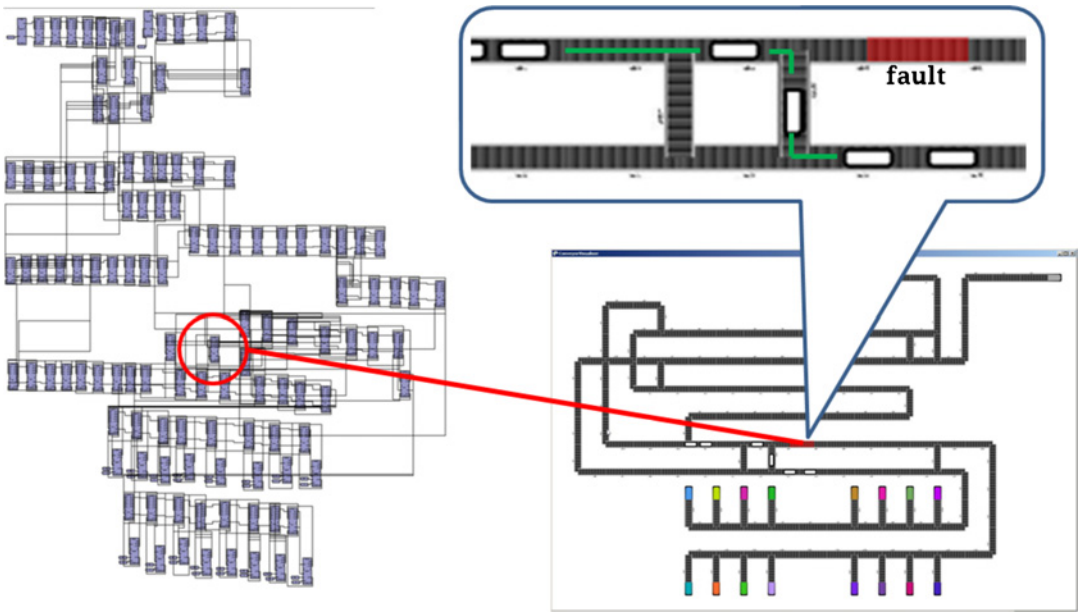


Fig. 7. BHS visualization communicating via OPC server with ISaGRAF distributed control [74].

The nxtControl company has also demonstrated some visualized simulation examples with the development of their IEC 61499 compliant development tool called nxtStudio [99]. They implement the view component by using the CAT concept where a FB model is linked with a visualized element. Fig. 6 shows one of the examples from the building management systems sector.

Here a set of controls and displays in a room is represented by FB connected to a FB implementing PID controller. The network of FB is encapsulated to a composite FB "Room," which can be assembled to floors and buildings. A remarkable feature of the CAT concept is ability to combine control logic with HMI elements and faceplates. This eliminates the need in third party SCADA design tools. NxtStudio automatically splits these functionalities and deploys them onto HMI panels versus embedded control devices.

D. Verification and Validation

V&V are software quality control procedures. Verification is ensuring that the product has been built according to the requirements and design specifications. Validation ensures that the product meets the user's needs, and that the specifications were correct in the first place. There are numerous works on V&V of IEC 61499 based systems, which can be classified in three categories: simulation, formal verification, and specification compliance. The specification compliance at an abstract level before actual design and implementation can be done by bridging FB with UML or SysML, for example [8], [50], [51], [64], [102], and [115].

Simulation and model-checking are both verification techniques used in industrial automation, e.g., as exemplified in [119]. The use of simulation in a FB integrated development environment was demonstrated in [51]. A simulation run helps to check system's correctness for one specific behavior scenario (i.e., with a fixed set of input parameters). Formal verification via model-checking [120]–[122] proves the system's correctness for all scenarios. The use of model-checking in industrial automation was exemplified in [119]. The formal verification of FB is done through their modeling in various formal languages, such as NCES, Timed automata, State charts, etc., which can be verified using model-checkers such as SESA, ViVe, and SMV [123]–[127].

Both model-checking and simulation can take an important role in the design flow of distributed systems, especially the complex ones. Particularly with the tool chain and the design flow suggested in [23], because of the benefits of modular design (the software reusability with encapsulated models and programming codes for validation and deployment purpose), simulation, and validation can be implemented in parallel with the system design to check the system's correctness. If there is any issue of violating specifications, a change made to the system validation model will also correspond to a change in the code for deployment. Once the validation is completed, the software program is ready to be deployed into the real hardware.

One idea, proposed in [113], is to link FB design environments with popular tools such as MATLAB where proper validation and verification can already be performed. This can immediately show benefits by saving time and cost for redeveloping the model for validation and verification purpose.

The problem for wider adoption of the closed-loop verification in the broader area of industrial automation is the lack of model design methods, which can transform this activity from a form of art to a systematic routine well integrated into usual activities of control engineers. It has been demonstrated in a number of publications [51], [72], and [73], that using FB as a modeling language is feasible and beneficial. Another possible use of models is implementation of model-predictive control in a FB-compliant embedded controller.

TABLE I
COMPARATIVE ANALYSIS OF SEVERAL MODEL-DRIVEN CONTROL DESIGN AND VALIDATION CONCEPTS

	IEC 61131	Simulink	SCADE	Open RTM	IEC 61499
Distributed languages concepts	0	0	1	1	2
Automation design models	2	0	0	0	2
Software models (UML, State charts)	1	2	2	2	1
Control models (block diagrams)	2	2	2	2	2
Integrated simulation	1	2	1	1	1
Deployment to distributed nodes	0	0	0	0	2
Integrated HMI/Visualisation	0	0	0	2	2
Total score	*6*	*6*	*6*	*8*	*12*

VIII. CONCLUSION

This paper surveyed the developments related to MDD and validation of distributed automation systems. Various engineering concepts and software tools were evaluated and discussed, including IEC 61131, MATLAB Simulink, SCADE, OpenRTM, and IEC 61499. Engineering approaches such as block-diagram programming language and MVC concept are considered valuable to the design and validation process for distributed systems. Table I summarized the capabilities of each model-driven control design and validation concepts, and an attempt to quantify differences between some of the surveyed was made. The technologies were evaluated against their capabilities to support design models and features using a simple 0–2 scale, where 0 means "not supported," 1 is "partially supported" and 2 is "can support." The IEC 61499 technology scores the highest among the surveyed list, attributed to the fact it is a new generation development that aims at combining "best of many worlds" features.

One of the conclusions that can be drawn from this paper is that there is inherent intertwining between modeling in the control sense and model-driven software engineering. It has been shown that the FB architecture of IEC 61499 incorporates several trends from both domains and allows for most natural implementation of DCSs, where validation capabilities are built in along with the features addressing classic distributed systems design challenges. The IMC architecture, built "on top" of IEC 61499, aims at filling the gap in validation and verification environment for distributed automation systems.

REFERENCES

[1] N. N. Chokshi and D. C. McFarlane, *A Distributed Coordination Approach to Reconfigurable Process Control*. London, U.K.: Springer, 2008.

[2] U. H. Felcht, R. C. Darton, R. G. H. Prince, and D. G. Wood, "The future shape of the process industries," in *Chemical Engineering: Visions of the World*. Amsterdam, The Netherlands: Elsevier Science B.V., 2003, pp. 41–66.

[3] C.-H. Yang and V. Vyatkin, "Design and validation of distributed control with decentralized intelligence in process industries: A survey," in *Proc. 6th IEEE Int. Conf. Ind. Inf.*, Daejeon, Korea, 2008, pp. 1395–1400.

[4] A. Tsuchiya, Y. Ikkai, and N. Komoda, "Development of a distributed process control programming tool for function block description," in *Proc. 7th IEEE Int. Conf. Emerging Technol. Factory Autom.*, Barcelona, Spain, 1999, pp. 1321–1325.

[5] N. Shah, "Process industry supply chains: Advances and challenges," *Comput. Chem. Eng.*, vol. 29, no. 6, pp. 1225–1236, Jan. 2005.

[6] V. Vyatkin, "Intelligent mechatronic components: Control system engineering using an open distributed architecture," in *Proc. 9th IEEE Conf. Emerging Technol. Factory Autom.*, Lisbon, Portugal, 2003, pp. 277–284.

[7] V. Vyatkin, J. H. Christensen, J. L. M. Lastra, and F. Auinger, "OOONEIDA: An open, object-oriented knowledge economy for intelligent industrial automation," *IEEE Trans. Ind. Inform.*, vol. 1, no. 1, pp. 4–17, Feb. 2005.

[8] S. D. Panjaitan, *Development Process for Distributed Automation Systems Based on Elementary Mechatronic Functions*. Maastricht, Germany: Shaker Verlag GmbH, 2008.

[9] J. L. M. Lastra, *Reference Mechatronic Architecture for Actor-Based Assembly Systems*. Tampere, Finland: Tampere Univ. Technol., 2004.

[10] C. Secchi, M. Bonfe, C. Fantuzzi, R. Borsari, and D. Borghi, "Object-oriented modeling of complex mechatronic components for the manufacturing industry," *IEEE/ASME Trans. Mechatron.*, vol. 12, no. 6, pp. 696–702, Dec. 2007.

[11] K. Thramboulidis, "Comments on object-oriented modeling of complex mechatronic components for the manufacturing industry," *IEEE/ASME Trans. Mechatron.*, vol. 13, no. 4, pp. 485–487, Aug. 2008.

[12] K. Thramboulidis, "Challenges in the development of mechatronic systems: The mechatronic component," in *Proc. 13th IEEE Int. Conf. Emerging Technol. Factory Autom.*, Hamburg, Germany, 2008, pp. 624–631.

[13] I. Terzic, A. Zoitl, B. Favre, and T. Strasser, "A survey of distributed intelligence in automation in European industry, research and market," in *Proc. 13th IEEE Int. Conf. Emerging Technol. Factory Autom.*, Hamburg, Germany, 2008, pp. 221–228.

[14] A. S. Tanenbaum and M. van Steen, *Distributed Systems*. Reading, MA, USA: Addison Wesley, 2004.

[15] S. Haridi, P. Van Roy, P. Brand, and C. Schulte, "Programming languages for distributed applications," *New Generation Comput.*, vol. 16, no. 3, pp. 223–261, Sep. 1998.

[16] R. Tirtea, G. Deconinck, V. De Florio, and R. Belmans, "QoS monitoring at middleware level for dependable distributed automation systems," in *Proc. 13th Int. Symp. Softw. Reliab. Eng.*, Annapolis, MD, USA, 2002, pp. 217–218.

[17] K. Trkaj, "Users introduce component based automation solutions," *Comput. Control Eng. J.*, vol. 15, no. 6, pp. 32–37, Dec.–Jan. 2004.

[18] M. Fowler and K. Scott, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Reading, MA, USA: Addison-Wesley Longman, 2000.

[19] T. Weilkiens, *Systems Engineering With SysML/UML: Modeling, Analysis, Design*. San Mateo, CA, USA: Morgan Kaufmann, 2007.

[20] International Electrotechnical Commission, *Programmable Controllers—Part 3: Programming Languages*, 2nd ed, IEC 61131-3 Standard, International Electrotechnical Commission, Geneva, Switzerland, 2003.

[21] M. Pechoucek, M. Rehak, and V. Marik, "Expectations and deployment of agent technology in manufacturing and defence: Case studies," in *Proc. AAMAS*, 2005, pp. 100–106.

[22] K. Thramboulidis, "IEC 61499: Back to the well proven practice of IEC 61131?" in *Proc. 17th IEEE Conf. Emerging Technol. Factory Autom.*, Cracow, Poland, 2012, pp. 1–8.

[23] V. Vyatkin, H. M. Hanisch, C. Pang, and C.-H. Yang, "Closed-loop modeling in future automation system engineering and validation," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 39, no. 1, pp. 17–28, Jan. 2009.

[24] H.-M. Hanisch, "Closed-loop modeling and related problems of embedded control systems in engineering," in *Abstract State Machines 2004. Advances in Theory and Practice*, vol. 3052, W. Zimmermann and B. Thalheim, Eds. Berlin/Heidelberg, Germany: Springer, 2004, pp. 6–19.

[25] V. Vyatkin and H.-M. Hanisch, "Verification of distributed control systems in intelligent manufacturing," *J. Intell. Manuf.*, vol. 14, no. 1, pp. 123–136, 2003.

[26] J. M. Machado, B. Denis, J. J. Lesage, J. M. Faure, and J. C. L. F. D. Silva, "Increasing the efficiency of PLC program verification using a plant model," in *Proc. 6th Ind. Eng. Prod. Manage.*, Porto, Portugal, 2003, pp. 10–16.

[27] H.-M. Hanisch and A. Lüder, "Modular modeling of closed-loop systems," in *Proc. Colloq. Petri Net Technol. Modeling Commun. Based Syst.*, Berlin, Germany, 2000, pp. 103–126.

[28] M. Perin and J.-M. Faure, "Coupling timed plant and controller models with urgent transitions without introducing deadlocks," in *Proc. 17th IEEE Conf. Emerging Technol. Factory Autom.*, Cracow, Poland, 2012, pp. 1–9.

[29] S. Preuse, H. Lapp, and H. Hanisch, "Closed-loop system modeling, validation, and verification," in *Proc. 17th IEEE Conf. Emerging Technol. Factory Autom.*, Cracow, Poland, 2012, pp. 1–8.

[30] J. Peltola, J. Christensen, S. Sierla, and K. Koskinen, "A migration path to IEC 61499 for the batch process industry," in *Proc. 5th IEEE Int. Conf. Ind. Inform.*, Vienna, Austria, 2007, pp. 811–816.

[31] M. Pěchouček and V. Mařik, "Industrial deployment of multiagent technologies: Review and selected case studies," *Int. J. Autonomous Agents Multi-Agent Syst.*, vol. 17, no. 3, pp. 397–431, 2008.

[32] G. Cândido and J. Barata, "A multiagent control system for shop floor assembly," in *Proc. 3rd Int. Conf. Ind. Appl. Holonic Multi-Agent Syst. Holonic Multi-Agent Syst. Manuf.*, Regensburg, Germany, 2007, pp. 293–302.

[33] A. Ranganathan and R. H. Campbell, *What Is the Complexity of a Distributed System?* Urbana, IL, USA: Univ. Illinois at Urbana-Champaign, 2003.

[34] K. H. Hall, R. J. Staron, and A. Zoitl, "Challenges to industry adoption of the IEC 61499 standard on event-based function blocks," in *Proc. 5th IEEE Int. Conf. Ind. Inf.*, Vienna, Austria, 2007, pp. 823–828.

[35] D. A. Bader and R. Pennington, "Cluster computing: Applications," *Int. J. High Perform. Comput.*, vol. 15, no. 2, pp. 181–185, May 2001.

[36] E. Tatara, A. Cinar, and F. Teymour, "Control of complex distributed systems with distributed intelligent agents," *J. Process Control*, vol. 17, no. 5, pp. 415–427, 2007.

[37] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[38] M. Obitko, P. Vrba, V. Marík, M. Radakovic, and P. Kadera, "Applications of semantics in agent-based manufacturing system," *Informatica (Slovenia)*, vol. 34, no. 3, pp. 315–330, Oct. 2010.

[39] M. Vall, H. Kaindl, M. Merdan, W. Lepuschitz, E. Arnautovic, and P. Vrba, "An automation agent architecture with a reflective world model in manufacturing systems," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, San Antonio, TX, USA, 2009, pp. 305–310.

[40] J. Barata, L. Camarinha-Matos, and G. Cíndido, "A multiagent-based control system applied to an educational shop floor," *Robot. Comput.-Integr. Manuf.*, vol. 24, no. 5, pp. 597–605, Oct. 2008.

[41] A. Aldea, R. Banares-Alcantara, L. Jimenez, A. Moreno, J. Martinez, and D. Riano, "The scope of application of multiagent systems in the process industry: Three case studies," *Expert Syst. Appl.*, vol. 26, pp. 39–47, 2004.

[42] V. Villasenor Herrera, A. Vidales Ramos, and J. L. Martinez Lastra, "A framework for modeling agent-based control of service-enabled manufacturing systems," in *Proc. 10th IFAC Workshop Intell. Manuf. Syst.*, Lisbon, Portugal, 2010, pp. 49–54.

[43] V. Mařík, P. Vrba, K. H. Hall, and F. P. Maturana, "Rockwell automation agents for manufacturing," in *Proc. 4th Int. Joint Conf. Autonomous Agents Multiagent Syst.*, Amsterdam, The Netherlands, 2005, pp. 107–113.

[44] F. P. Maturana, P. Tichý, P. Slechta, F. Discenzo, R. J. Staron, and K. Hall, "Distributed multiagent architecture for automation systems," *Expert Syst. Appl.*, vol. 26, no. 1, pp. 49–56, 2004.

[45] L. Mönch, M. Stehli, and J. Zimmermann, "FABMAS: An agent-based system for production control of semiconductor manufacturing processes," in *Holonic and Multi-Agent Systems for Manufacturing*, vol. 2744, V. Marík, D. McFarlane, and P. Valckenaers, Eds. Berlin/Heidelberg, Germany: Springer, 2004, p. 1085.

[46] R. W. Brennan and W. O, "A simulation test-bed to evaluate multiagent control of manufacturing systems," in *Proc. 32nd Conf. Winter Simul.*, Orlando, FL, USA, 2000, pp. 1747–1756.

[47] P. Vrba and V. Marik, "Simulation in agent-based manufacturing control systems," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2005, pp. 1718–1723.

[48] M. Bonfe and C. Fantuzzi, "Design and verification of mechatronic object-oriented models for industrial control systems," in *Proc. 9th IEEE Conf. Emerging Technol. Factory Autom.*, Lisbon, Portugal, 2003, pp. 253–260.

[49] K. C. Thramboulidis, "Using UML in control and automation: A model driven approach," in *Proc. 2nd IEEE Int. Conf. Ind. Inf.*, Berlin, Germany, 2004, pp. 587–593.

[50] V. Dubinin, V. Vyatkin, and T. Pfeiffer, "Engineering of validatable automation systems based on an extension of UML combined with function blocks of IEC 61499," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 3996–4001.

[51] J. H. Christensen, "Design patterns for systems engineering with IEC 61499," in *Proc. Verteilte Automatisierung—Modelle und Methoden für Entwurf, Verifikation Engineering Instrumentierung*, Magdeburg, Germany, 2000, pp. 63–71.

[52] M. Rausch and H. M. Hanisch, "Net condition/event systems with multiple condition outputs," in *Proc. Joint IEEE/INRIA Symp. Emerging Technol. Factory Autom.*, Paris, France, 1995, pp. 592–600.

[53] C. Sünder, A. Zoitl, and C. Dutzler, "Functional structure-based modeling of automation systems," *Int. J. Manuf. Res.*, vol. 1, no. 4, pp. 405–420, 2006.

[54] T. Strasser, M. Rooker, I. Hegny, M. Wenger, A. Zoitl, L. Ferrarini, A. Dede, and M. Colla, "A research roadmap for model-driven design of embedded systems for automation components," in *Proc. 7th IEEE Int. Conf. Ind. Inf.*, Cardiff, U.K., 2009, pp. 564–569.

[55] T. Strasser, M. Rooker, G. Ebenhofer, I. Hegny, M. Wenger, C. Sunder, A. Martel, and A. Valentini, "Multidomain model-driven design of industrial automation and control systems," in *Proc. 13th IEEE Int. Conf. Emerging Technol. Factory Autom.*, Hamburg, Germany, 2008, pp. 1067–1071.

[56] T. Strasser, C. Sunder, and A. Valentini, "Model-driven embedded systems design environment for the industrial automation sector," in *Proc. 6th IEEE Int. Conf. Ind. Inf.*, Daejeon, Korea, 2008, pp. 1120–1125.

[57] K. Thramboulidis, "IEC 61499 in factory automation," in *Proc. IEEE Int. Conf. Ind. Electron. Technol. Autom.*, Bridgeport, CT, USA, 2005.

[58] C. Tranoris and K. Thramboulidis, "From requirements to function block diagrams: A new approach for the design of industrial applications," in *Proc. 10th IEEE Mediterranean Conf. Control Autom.*, Lisbon, Portugal, 2002, pp. 1–10.

[59] C. Tranoris and K. Thramboulidis, "Integrating UML and the function block concept for the development of distributed control applications," in *Proc. 9th IEEE Conf. Emerging Technol. Factory Autom*., Lisbon, Portugal, 2003, pp. 87–94.

[60] D. Hästbacka, T. Vepsäläinen, and S. Kuikka, "Model-driven development of industrial process control applications," *J. Syst. Softw.*, vol. 84, no. 7, pp. 1100–1113, 2011.

[61] T. Vepsalainen, S. Sierla, J. Peltola, and S. Kuikka, "Assessing the industrial applicability and adoption potential of the AUKOTON model driven control application engineering approach," in *Proc. 8th IEEE Int. Conf. Ind. Inf.*, Osaka, Japan, 2010, pp. 883–889.

[62] J. Peltola, S. Sierla, T. Vepsalainen, and K. Koskinen, "Challenges in industrial adoption of model-driven technologies in process control application design," in *Proc. 9th IEEE Int. Conf. Ind. Inf.*, Lisbon, Portugal, 2011, pp. 565–572.

[63] M. Hirsch and H.-M. Hanisch, "Systemspezifikation mit SysML für eine Fertigungstechnische Laboranlage," in *Proc. Fachtagung Zum Entwurf Komplexer Automatisierungssysteme*, Magdeburg, Germany 2008, pp. 23–34.

[64] M. Hirsch, *Systematic Design of Distributed Industrial Manufacturing Conrol Systems*. Berlin, Germany: Logos Verlag, 2010.

[65] A. Thoma, B. Kormann, and B. Vogel-Heuser, "Fault-centric system modeling using SysML for reliability testing," in *Proc. IEEE 17th Conf. Emerging Technol. Factory Autom.*, 2012, pp. 1–8.

[66] OpenRTM-aist. (2010). *OpenRTM-aist official website* [Online]. Available: http://www.openrtm.org/

[67] N. Ando, T. Suehiro, and T. Kotoku, "A software platform for component based RT-system development: OpenRTM-Aist," in *Simulation, Modeling Programming Autonomous Robots*, vol. 5325, S. Carpin, I. Noda, E. Pagello, M. Reggiani, and O. von Stryk, Eds. Berlin-Heidelberg, Germany: Springer, 2008, pp. 87–98.

[68] International Electrotechnical Commission, "International Standard Draft IEC61804-2: Function blocks (FB) for process control," in *Part 2: Specification of FB Concept and Electronic Device Description Language (EDDL)*. Geneva, Switzerland: International Electrotechnical Commission, 2004.

[69] C. Diedrich, F. Russo, L. Winkel, and T. Blevins, Function Block Applications in Control Systems Based on IEC 61804. 2001.

[70] National Instruments. (2010). *NI LabVIEW: The Software That Powers Virtual Instrumentation* [Online]. Available: http://www.ni.com/labview/

[71] V. Vyatkin and H.-M. Hanisch, "Application of visual specifications for verification of distributed controllers," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Tucson, AZ, USA, 2001, pp. 646–651.

[72] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*, ISA-Instrumentation, Systems, and Automation Society, Research Triangle Park, NC, USA, 2007.

[73] G. Black and V. Vyatkin, "Intelligent component-based automation of baggage handling systems with IEC 61499," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 2, pp. 337–351, Apr. 2010.

[74] J. Yan and V. Vyatkin, "Distributed software architecture enabling peer-to-peer communicating controllers," *IEEE Trans. Ind. Inf.*, vol. PP, no. 99, p. 1, Apr. 2013.

[75] W. Lepuschitz, A. Zoitl, M. Valle, and M. Merdan, "Toward self-reconfiguration of manufacturing systems using automation agents," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 41, no. 1, pp. 52–69, Jan. 2011.

[76] I. Hegny, O. Hummer, A. Zoitl, G. Koppensteiner, and M. Merdan, "Integrating software agents and IEC 61499 realtime control for reconfigurable distributed manufacturing systems," in *Proc. IEEE 3rd Symp. Ind. Embedded Syst.*, La Grande Motte, France, 2008, pp. 249–252.

[77] X. M. Huang, "Intelligent and reconfigurable control of automatic production line by applying IEC61499 function blocks and software agent," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Changchun, China, 2009, pp. 1481–1486.

[78] G. Zhabelova and V. Vyatkin, "Multiagent smart grid automation architecture based on IEC 61850/61499 intelligent logical nodes," *IEEE Trans. Ind. Electron.*, vol. 59, no. 5, pp. 2351–2362, May 2011.

[79] J. Lunze, "What is a hybrid system?" in *Modelling, Analysis, and Design of Hybrid Systems*, vol. 279, S. Engell, G. Frehse, and E. Schnieder, Eds. Berlin-Heidelberg, Germany: Springer, 2002, pp. 3–14.

[80] E. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE Int. Symp. Object Oriented Real-Time Distributed Comput.*, Orlando, FL, USA, 2008, pp. 363–369.

[81] A. Gorod, B. Sauser, and J. Boardman, "System-of-systems engineering management: A review of modern history and a path forward," *IEEE Syst. J.*, vol. 2, no. 4, pp. 484–499, Dec. 2008.

[82] B. Zhou, A. Dvoryanchikova, A. Lobov, J. Minor, and J. L. Martinez Lastra, "Application of the generic modelling method for system of systems to manufacturing domain," in *Proc. 37th Annu. Conf. IEEE Ind. Electron. Soc.*, Melbourne, VIC, Australia, 2011, pp. 352–358.

[83] MathWorks. (2010). *The MathWorks—MATLAB and Simulink for Technical Computing* [Online]. Available: http://www.mathworks.com

[84] B. I. Silva and B. H. Krogh, "Formal verification of hybrid systems using CheckMate: A case study," in *Proc. Am. Control Conf.*, Chicago, IL, USA, 2000, pp. 1679–1683.

[85] J. H. Taylor, "A modeling language for hybrid systems," in *Proc. IEEE/IFAC Joint Symp. Computer-Aided Control Syst. Design*, Tucson, AZ, USA, 1994, pp. 339–344.

[86] J. H. Taylor and D. Kebede, "Modeling and simulation of hybrid systems," in *Proc. 34th IEEE Conf. Decision Control*, New Orleans, LA, USA, 1995.

[87] T. A. Henzinger, H. Pei-Hsin, and H. Wong-Toi, "HYTECH: The next generation," in *Proc. 16th IEEE Real-Time Syst. Symp.*, Pisa, Italy, 1995, pp. 56–65.

[88] S. Kowalewski, N. Bauer, J. Preussig, O. A. S. O. Stursberg, and H. A. T. H. Treseler, "An environment for model-checking of logic control systems with hybrid dynamics," in *Proc. IEEE Int. Symp. Comput.-Aided Control Syst. Design*, Kohala Coas, HI, USA, 1999, pp. 97–102.

[89] Ptolemy II. (2013). *Ptolemy II* [Online]. Available: http://ptolemy.berkeley.edu/ptolemyII/

[90] Modelica. (2013). *Modelica and the Modelica Association* [Online]. Available: http://www.modelica.org/

[91] OpenModelica. (2013) [Online]. Available: http://www.openmodelica.org/

[92] MapleSim. (2013). *MapleSim—High Performance Physical Modeling and Simulation—Technical Computing Software* [Online]. Available: http://www.maplesoft.com/products/maplesim/

[93] MathModelica. (2013). *MathModelica: Modeling, Simulation, Analysis, and Documentation of Multiengineering and Life Science Systems* [Online]. Available: http://www.mathcore.com/products/mathmodelica/

[94] R. A. Santos, J. E. Normey-Rico, A. M. Gomez, L. F. A. Arconada, and C. d. P. Moraga, "Distributed continuous process simulation: An industrial case study," *Comput. Chem. Eng.*, vol. 32, no. 6, pp. 1195–1205, 2008.

[95] MathWorks. (2012). *Distributed Simulation Toolbox* [Online]. Available: http://www.mathworks.com/products/connections/product_detail/product_35768.html

[96] N. P. Mahalik and K. Kim, "A prototype for hardware-in-the-loop simulation of a distributed control architecture," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 38, no. 2, pp. 189–200, Mar. 2008.

[97] S. Mishra and N. Mahalik, "Virtual DCS and specification," *Int. J. Inf. Commun. Technol.*, vol. 3, no. 4, pp. 339–353, Nov. 2011.

[98] D. Wenbin and V. Vyatkin, "Redesign distributed IEC 61131-3 PLC system in IEC 61499 function blocks," in *Proc. IEEE Conf. Emerging Technol. Factory Autom.*, Vienna, Austria, 2010, pp. 1–8.

[99] nxtControl.com. (2010) [Online]. Available: http://www.nxtcontrol.com/

[100] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State of the art review," *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 768–781, Nov. 2011.

[101] K. Thramboulidis, S. Sierla, N. Papakonstantinou, and K. Koskinen, "An IEC61499 based approach for distributed batch process control," in *Proc. 5th IEEE Int. Conf. Ind. Inf.*, Vienna, Austria, 2007, pp. 177–182.

[102] K. C. Thramboulidis, "Using UML in control and automation: A model driven approach," in *Proc. IEEE 2nd Int. Conf. Ind. Inf.*, 2004, pp. 587–593.

[103] *Standard: Batch Control. Part 1: Models and Terminology*, ISA-S88.01-1995, The International Society for Measurement and Control, 1995.

[104] J. P. Peltola, S. A. Sierla, M. P. Stromman, and K. O. A. K. K. O. Koskinen, "Process control with IEC 61499: Designers' choices at different levels of the application hierarchy," in *Proc. 4th IEEE Int. Conf. Ind. Inf.*, Singapore, 2006, pp. 183–188.

[105] S. Kuikka, "A batch process management framework: Domain-specific, design pattern and software component based approach," Doctor of Technology Dissertation, Technical Res. Centre Finland, Helsinki Univ. Technol., Espoo, Finland, 1999.

[106] Holobloc, Inc. (2011). FBDK: The function block development kit [Online]. Available: http://www.holobloc.com/doc/fbdk/index.htm

[107] K. Thramboulidis, "Development of distributed industrial control applications: The CORFU framework," in *Proc. 4th IEEE Int. Workshop Factory Commun. Syst.*, Västerås, Sweden, 2002, pp. 39–46.

[108] 4DIAC. (2010). *Framework for distributed industrial automation* [Online]. Available: http://www.fordiac.org

[109] ISaGRAF. (2013, May 26). *ICS triplex ISaGRAF Inc.—Leading IEC 61131 and IEC 61499 software* [Online]. Available: http://www.isagraf.com

[110] L. H. Yoong, P. S. Roop, V. Vyatkin, and Z. Salcic, "A synchronous approach for IEC 61499 function block implementation," *IEEE Trans. Comput.*, vol. 58, no. 12, pp. 1599–1614, Dec. 2009.

[111] P. Tata and V. Vyatkin, "Proposing a novel IEC61499 runtime framework implementing the cyclic execution semantics," in *Proc. 7th IEEE Int. Conf. Ind. Inform.*, Cardiff, U.K., 2009, pp. 416–421.

[112] L. H. Yoong, P. S. Roop, and Z. Salcic, "Efficient implementation of IEC 61499function blocks," in *Proc. IEEE Int. Conf. Ind. Technol.*, Gippsland, VIC, Australia, 2009, pp. 1–6.

[113] C.-H. Yang and V. Vyatkin, "Model transformation between MATLAB simulink and function blocks," in *Proc. 8th IEEE Int. Conf. Ind. Inf.*, Osaka, Japan, 2010, pp. 1130–1135.

[114] V. Vyatkin and V. Dubinin, "Sequential axiomatic model for execution of basic function blocks in IEC61499," in *Proc. IEEE 5th Int. Conf. Ind. Inform.*, Vienna, Austria, 2007, pp. 1183–1188.

[115] V. Dubinin and V. Vyatkin, "On definition of a formal semantic model for IEC 61499 function blocks," *EURASIP J. Embedded Syst. Embedded Syst. Design Intell. Ind. Autom.*, vol. 2008, pp. 1–10, 2008.

[116] V. Vyatkin, V. Dubinin, C. Veber, and L. Ferrarini, "Alternatives for execution semantics of IEC61499," in *Proc. IEEE 5th Int. Conf. Ind. Inform.*, 2007, pp. 1151–1156.

[117] J. Christensen, T. Strasser, A. Valentini, V. Vyatkin, and A. Zoitl, "The IEC 61499 function block standard: Overview of the second edition," *ISA Autom. Week*, 2012 [Online]. Available: http://www.isa.org/Template.cfm?Section=Shop_ISA&Template=/Ecommerce/ProductDisplay.cfm&ProductID=12502

[118] MVC XEROX PARC. (1979). Model-view-controller design pattern [Online]. Available: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[119] H. Hanisch, A. Lobov, J. L. Martinez Lastra, R. Tuokko, and V. Vyatkin, "Formal validation of intelligent-automated production systems: Towards industrial applications," *Int. J. Manuf. Technol. Manage.*, vol. 8, no. 1, pp. 75–106, 2006.
[120] G. Frey and L. Litz, "Formal methods in PLC programming," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Nashville, TN, USA, 2000, pp. 2431–2436.
[121] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.
[122] W. Farn, "Formal verification of timed systems: A survey and perspective," *Proc. IEEE*, vol. 92, no. 8, pp. 1283–1305, Aug. 2004.
[123] C. Pang and V. Vyatkin, "Automatic model generation of IEC 61499 function block using net condition/event systems," in *Proc. 6th IEEE Int. Conf. Ind. Inform.*, Daejeon, Korea, 2008, pp. 1133–1138.
[124] N. Hagge and B. Wagner, "Java code patterns for Petri net based behavioral models," in *Proc. 3rd IEEE Int. Conf. Ind. Inform.*, Perth, WA, Australia, 2005, pp. 450–455.
[125] V. Vyatkin, H. M. Hanisch, and T. Pfeiffer, "Object-oriented modular place/transition formalism for systematic modeling and validation of industrial automation systems," in *Proc. IEEE Int. Conf. Ind. Inform.*, 2003.
[126] G. Cengic and K. Akesson, "On formal analysis of IEC 61499 applications, Part A: Modeling," *IEEE Trans. Ind. Inform.*, vol. 6, no. 2, pp. 136–144, May 2010.
[127] C. Gerber and H. M. Hanisch, "Does portability of IEC 61499 mean that once programmed control software runs everywhere?" in *Proc. 10th IFAC Workshop Intell. Manuf. Syst.*, Lisbon, Portugal, 2010, pp. 29–34.

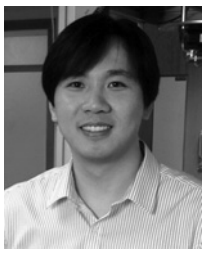
Chia-Han Yang received the B.E. (first class honors) and Ph.D. degrees in electrical and electronics engineering from the University of Auckland, Auckland, New Zealand, in 2006 and 2011, respectively.

His Ph.D. research focused on investigating methodologies of improving distributed control system design (based on IEC61499 standard) through closed-loop validation processes. In 2011, he was a Research Engineer with the Department of Electrical and Computer Engineering, University of Auckland, Auckland, New Zealand, continuing research work in the simulation of distributed control systems. He is currently a Research Engineer doing robotics-related research and development at the Centre for Autonomous System (CAS), University of Technology Sydney, Sydney, NSW, Australia. His current research interests include distributed control, industrial automation, modeling and simulation, and robotics.

Valeriy Vyatkin received the B.E. degree in applied mathematics, the Ph.D. and Dr. Sci. degrees from the Taganrog State University of Radio Engineering (TSURE), Taganrog, Russia, in 1988, 1992 and 1998, respectively, and the Dr. Eng. degree from the Nagoya Institute of Technology, Nagoya, Japan, in 1999.

He is a Chaired Professor of dependable computation and communication systems at the Luleå University of Technology, Sweden, and a Visiting Scholar at Cambridge University, Cambridge, U.K. He is on leave from the University of Auckland, Auckland, New Zealand, where he has been an Associate Professor and the Director of the InfoMechatronics and Industrial Automation Laboratory (MITRA), Department of Electrical and Computer Engineering. He was in faculty positions with the Martin Luther University of Halle-Wittenberg, Germany, as a Senior Researcher and a Lecturer from 1999 to 2004, and with TSURE as an Associate Professor and Professor from 1991 to 2002. His current research interests include dependable distributed automation and industrial informatics, including software engineering for industrial automation systems, distributed architectures and multiagent systems applied in various industry sectors: smart grid, material handling, building management systems, reconfigurable manufacturing, etc. He is also active in research on dependability provisions for industrial automation systems, such as methods of formal verification and validation, and theoretical algorithms for improving their performance.

Dr. Vyatkin was a recipient of the Andrew P. Sage Award for the Best IEEE Transactions Paper in 2012.



Cheng Pang (S'08–M'13) received the B.E. (Hons.) and M.E. (Hons.) degrees in computer systems engineering and the Ph.D. degree in electrical and electronic engineering from the University of Auckland, Auckland, New Zealand, in 2005, 2007, and 2013, respectively.

He is a Post-Doctoral Research fellow at the Laboratory of Advanced Computing and Communications for Industrial Applications, Luleå University of Technology, Sweden. His current research interests include model-driven engineering for industrial automation systems, building automation and control systems, and distributed control for the Internet of things.