Model-driven Development of Control Software for Distributed Automation: a Survey and an Approach

Chia-Han Yang, Valeriy Vyatkin, Cheng Pang

***Abstract* – This paper presents a survey on model-driven design and validation approaches for distributed automation and control systems with essentially decentralised logic. Driven by the goals of flexibility and performance improvement, researchers have explored several approaches to distributed systems design, including multi-agent 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 modelling 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 modelling, implementation and validation.**
[1]

Keywords: Modelling, Distributed systems, Simulation, IEC 61499, Function Blocks.

I. INTRODUCTION

The ever increasing demand for flexibility and reconfigurability of control system in manufacturing and process industries is an undisputable fact, as indicated in many publications, e.g. [1-3]. The requirement to react on the ever changing market demands by producing small quantities of many customised products rather than mass production of a single product [2, 4, 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) is known in process industries since 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 centralised control unit implementing control algorithm. However, while increasing the flexibility of hardware maintenance, this traditional DCS approach has little to do with flexibility of the production.

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" in order to improve the flexibility of the production systems [6-8]. Such components are individually equipped with embedded controllers and can be aggregated into machines and systems, arguably easier than traditional mechanical and mechatronic components. This results in easier reconfiguration. The use of such intelligent components and object-oriented design promises to bring essential benefits for design and re-configuration 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 (for example, in [9]) 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 decentralised 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 formalisation, software construction, verification and validation, 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 modelling frameworks that are used for verification and validation of complex automation and control systems.

The rest of this paper is structured as follows. Section II attempts to summarize various sources of distributed intelligence 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 verification and validation (Section V). Section VI examines how the trends from all these streams have been addressed in the international standard IEC 61499. Section VII presents elements of Intelligent Mechatronic Component (IMC) architecture that uses IEC 61499 as an enabling technology and attempts 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 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 visualisation and human interface software, programmable logic controller (PLC) with control software

Please cite as follows: C.-H. Yang, V. Vyatkin, C. Pang, "Model-driven Development of Control Software for Distributed Automation: a Survey and an Approach", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 2013, in print

and one or several motor drives implementing motion control functionality. Ensuring correct operation of these components requires addressing the issues of correct coordination, synchronisation 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 [10], many distributed programming languages [11] have been developed by computer scientists, however none of these is implemented on the existing PLC platforms.

Another approach is to implement the DI concept with loosely coupled Programmable Logic Controllers (PLC) interconnected via networks and middleware, e.g. [12], or DCOM based PROFInet-CBA [13]. However, researchers face various difficulties in software development for distributed systems in this way. One of the challenges is the lack of system-level view at heterogeneous automation systems which 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 Modelling Language (UML) [14] and its derivative, SysML [15].

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 organisation of automation software to virtually and physically distributed configurations. However, the existing PLC programming architecture (standardized in the form of IEC 61131-3 standard [16]) does not provide such mechanisms.

The International Electrotechnical Commission has addressed these issues in the IEC 61499 standard [17] by defining a concept of distributed control system design using event-driven modules called "Function Blocks" (FB). This standard establishes a reference architecture for the implementation of distributed control, allowing the design to be software-centric and vendor-independent while achieving flexibility in terms of both software and hardware.

Along with the software construction challenges, another main challenge of distributed systems design is their verification and validation. When controllers are independent of each other and distributed across the system network, the data exchange to the devices and the communication intensity between the controllers 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 behaviour 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 distributed systems is essential for the successful designs of distributed systems.

Closed-loop modelling and simulation is standard in control engineering. The "plant" model describes the behaviour 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 [18], the benefit of verification and validation using closed-loop models is as follows: "The validation of controller design by itself has no meaning and does not guarantee the correct behaviour 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 verification and validation 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 [19-22], 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 familiarise with the new design approach and new design tools [23]. This problem also leads to an idea of linking existing tools and languages with the new ones. Therefore improving efficiency of verification and validation can increase the industrial adoption of distributed automation.

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 organisation. 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 distributed control systems originates from the control implementation in process industry. This is where each physical element such as heaters, motors, pumps and valves is directly connected in closed-loop to its own automatic control unit with a 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 eventually communicate in a peer-to-peer manner. This will make the entire system more flexible and reliable. In [1], the re-configurability was considered as the target 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:

- The risks that accompany every new technology that has not been proven in large scale industrial applications [24];

- Lack of mature enough design and development tools for industrial development [24];
- Paradigm misunderstanding due to the small number of successful industrial applications [25];
- 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;
- Lack of methodologies that can simplify the integration of the new technology;
- Lack of industrial recognition of the potentials of the new technology, due to absence of publicity of successful industrial projects [17].

According to [26], 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 [27] that runs a single process.

The complexity of distributed systems design made the researchers looking for self-organisation mechanisms, that has given rise to the agent-based control concept [28] discussed in the following section.

B. Multi-agent Systems (MAS)

Therefore more modern techniques have been investigated, describing the notion of an intelligent, autonomous and goal-oriented agent in a multi-agent system [29]. An agent is a concept that represents an autonomous combination of software and hardware interacting with the environment. A multi-agent system 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 [30]. 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) [28]. A framework for modelling agent-based control of service-enabled manufacturing systems is presented in [31].

In a multi-agent system (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 [32], distributed control of ship equipment in US Navy shipboard systems [33] and production control of semiconductor wafer fabrication facilities called FABMAS [34]. The low level of those systems comprises device-specific implementations with IEC 61131-3 compliant PLCs communicating via technologies such as DCOM (Distributed Component Object Model) and Ethernet. Works [25, 35] present examples of modelling distributed agents communicating through a network for simulation purposes.

Another research stream has been dealing with making multi-agent systems more intelligent based on biological principles, for example, the holonic manufacturing concept [30, 36]. The work [30] also 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-organisation issues via multi-agent 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 is a dominating technology in general software engineering. In control and automation, the specific of this approach is that objects are often associated with some components of the controlled plant.

Bonfé and Fantuzzi [37] have introduced the use of UML in automation and Thramboulidis [38] in particular in the IEC 61499 context. The latter work proposed generation of function blocks from UML diagrams, while Dubinin in [39] proposed the UML-FB architecture supporting round-trip engineering of UML diagrams generation from function block designs and vice versa.

One characteristic concept in the mentioned works is encapsulation of hardware and software models of an automation and control system (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. Moreover, the software model provides all the necessary domain specific information of the ACS.

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 new ACS, the automation objects modelling the components are selected from a knowledge 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 *Intelligent Mechatronic Component* (IMC) in [18], where the closed-loop modelling methodology and corresponding model transformation approaches are introduced. Each IMC is internally organized following the *Model-View-Controller*

(MVC) design pattern [40]. Different languages and formalisms are employed to develop the domain specific models. For instance, Matlab/Simulink is used to create the hybrid model capturing both continuous and discrete dynamics of the IMC and IEC 61499 is used to model the executable behaviour, which can be deployed to the controllers directly. The *Net Condition/Event System* formalism [41] was adopted to define the IMC's formal model for the purposes of formal verification.

Sünder *et al.* presented a similar idea in [42], where the *Automation Component* concept as a variant of automation object is proposed. The focus of automation component is the universal component interface and unified hierarchical architecture allowing flexible reconfiguration of ACSs. This automation component model is later adopted and extended in the MEDEIA project [43-45], 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 introduced the *Model-Integrated Mechatronics* paradigm in [46] 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 modelling of mechatronic components in the *mechatronic layer*, which is vertically projected to the *application layer* for modelling the control software; the *resource layer* for modelling the control system infrastructure; and finally the *mechanical process layer* for modelling the mechanical composition. The UML profile introduced in [47, 48] is used to support this MDD process to generate the control application developed in IEC 61499 function blocks.

A similar MDD idea has also been implemented in the AUKOTON research project [49-51]. 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 et al. [52, 53] provide a pathway for linking function block technology with SysML.

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 is a RT-middleware proposed for robot system integration [54]. The term "RT" represents "Robot Technology." OpenRTM also uses a component-based approach in designing robotics applications. The component's model is shown in Fig. 1.

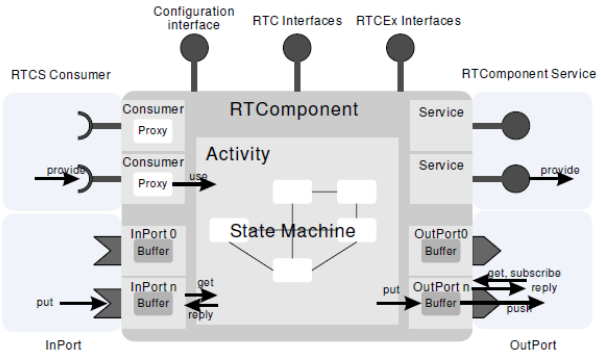


Fig. 1. The proposed architecture of RT-Component model [55].

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

- Component metadata for dynamic component assembly.
- Component action and execution context for business logic execution.
- Data ports for data exchange between RTCs.

There are already various industrial application built based on this framework, including 3D recognition, tracking, dynamic simulation, learning systems, etc. Fig. 2 shows the 3D recognition and tracking implementation example based on OpenRTM. However, even though there are various extensive works completed based on OpenRTM, this concept has not been formally standardised and is so far only applied in the robotics or related industries.

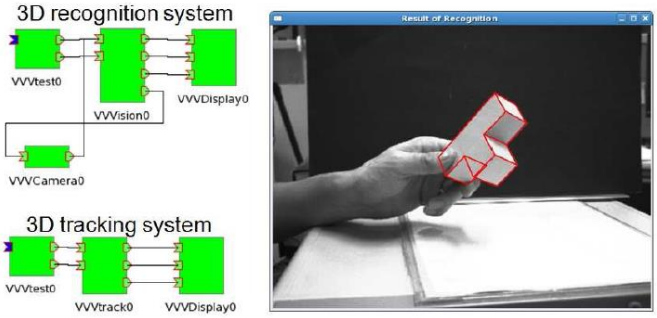


Fig. 2. 3D recognition and tracking using the proposed OpenRTM development tools [55].

The IEC 61804 standard draft [56], describes the specification and requirements of distributed process control systems based on Function Blocks [57]. The Electronic Device Description Language (EDDL) is the language that is stated in the part 2 of IEC 61804 standard and describes the properties of automation system component [56], such as vendor information, version of firmware/hardware and data format, etc. Through this language, all the information will be carried between the devices (controllers, sensors, actuators and engineering stations, etc.) by a fieldbus. This language fills in the gap between the Function Block specification and product implementation by allowing manufacturers to use the same description method for devices of different technologies and platforms. The Function Block design of a process control system for example, is only an abstract representation which may be implemented differently with different device types [57], such as Field Devices (FD), PLCs, visualisation stations and Device Description (DD).

There are also other popular modelling and simulation tools such as LabView [58] and SCADE [59]. These tools all follow the block-diagram based modelling approach. For example, SCADE is a model based development environment

dedicated to 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, model-driven design, 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 from 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.

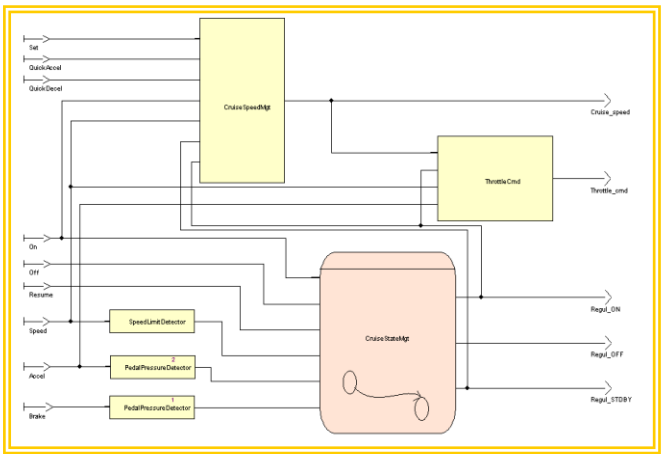


Fig. 3. Block-diagram design in SCADE.

C. IEC 61499 Function Blocks

The international Electrotechnical Commission (IEC) established the international standard IEC 61499 [17] to provide a reference software and system architecture for truly distributed control design. This architecture enhances the interoperability and re-usability 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 *Function Blocks* (FB), to address the increasing demand of flexibility, reusability, reconfigurability and distributed control applications [60]. Function Block can enhance software-reusability and simple reconfiguration through its object-oriented nature. IEC 61499 is appealing to developers because of the simplified specification approach and benefits related to the language's abstraction. The basic design unit or function block provides a graphical method for control flow design, and uses algorithms written in any programming language, including the PLC legacy ones.

With such distributable modules, it allows design to be completed before considering physical hardware layout. This is very different to the traditional design approach with PLC where hardware layout needs to be considered prior to the control software implementation. The Function Blocks design can be broken down into different subsystems and loaded onto different controllers where each controls an individual device or a subsystem.

A number of works explored implementation of multi-agent control with IEC 61499 distributed architecture. The fully distributed approach to baggage handling systems automation was demonstrated in [61]. A hierarchical multi-agent architecture based on IEC 61499 which enables elements of self-configuration in manufacturing systems was developed in [62], and [63] investigates the use of IEC 61499 to implement multi-agent control in material handling systems. The work [64] discusses the architectural solutions for joining IEC 61499 lower-level agents into upper multi agent manufacturing platform. In [65] multi-agent control for SmartGrid automation was reported.

V. Concepts and Environments for Modelling and Simulation

A. Concepts

This section presents some concepts for modelling of distributed automation and control systems. This discussion is followed by a survey of several modelling tools.

In general, a system in process industry is not purely discrete or purely continuous, as 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" [66]. 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. Modelling, verification and validation of systems with hybrid, i.e. continuous and discrete dynamics, executing intelligent control algorithms in decentralised nodes, are highly sophisticated. A big challenge will be incurred when introducing truly distributed control approach into such process control systems.

Another emerging modelling and design concept is called Cyber Physical System (CPS) [67]. 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 modelling adequately concurrent physical processes.

One more trend in modelling 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 [4] and they are stated as follows:

- Operational independence of the constituent systems;
- Managerial independence of the constituent systems;
- Geographic distribution;
- Emergent behaviours;
- Evolutionary and adaptive development.

Works [31, 68] introduced the SoS concept to the domain of industrial automation, having demonstrated its value for system-level parameters estimation.

B. Modelling Tools

MATLAB is a numerical modelling environment developed by MathWorks [69]. 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 computing, there are toolboxes available, allowing accesses to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Driven Design 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 non-linear continuous dynamics, under Simulink environment [70].

A language called Hybrid System Modelling Language (HSML) is created specifically for modelling hybrid systems. It is able to construct state events and discrete time modules in MATLAB and SIMULINK [71, 72]. In [72], it is indicated that HSML is particularly useful for time- and state-event handling. There are also some other software tools being developed to handle simulation and verification of different types of hybrid systems, such as *HyTech* [73] which uses symbolic model checking techniques in continuous state space to verify systems modelled with linear hybrid automata (LHA) and *VERDICT* [74] that provides an environment for modular modelling and simulation for timed and hybrid systems.

The *Ptolemy II* modelling environment also aims at modelling and simulation of hybrid systems [32]. Here, a hybrid system can be modelled by a finite state machine (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.

C. Simulation environments for Distributed Systems

The work [75] presents an industrial case study of a distributed continuous process simulation of a beet sugar factory. This simulation work is done by using distributed component object model (DCOM) components written with a modelling language called "EcosimPro." DCOM is the Microsoft solution for a component software bus. The work [76] also presents an object-oriented framework for process simulation.

Another example is distributed simulation (DS) toolbox for Matlab [77]. 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 distributed simulation: 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 Kiseon [78] address specification of requirement, design, and development of a hardware-in-the-loop simulation (HILS)-based tool for the configuration, validation, and management of distributed control systems. 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 [79] this work is extended by support of virtual distributed control systems 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 is able to handle and provide both simulation and verification of distributed control systems. It would be beneficial if a software package contained pre-set models specifically for process control (i.e. models for pipe-line, valve, etc.). The tools handling hybrid systems only work with their own modelling language and are not yet combined for implementation purpose to the real system. Therefore, a challenge here is to find a path of integrating models for hybrid validation with programming codes for deployment purpose, for distributed systems. This idea is also suggested in [18].

Most of the tools described above can handle hybrid systems, but they are not intended to deal with systems with decentralised intelligence. This brings an opportunity for Function Blocks, since the IEC 61499 standard and Function Blocks are established specifically to handle decentralised design with direct deployment functionality.

VI. IEC 61499 Distributed Function Block 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 organisation to distributed organisation (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. Support of HiL or Software in the Loop (SiL) simulation.

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 [16] 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. 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:

- Mapping of IEC 61499 FB applications to different devices is enhanced by tools which insert the required communication code as required. This provides platform independence as different controllers may be selected at the end of development to deploy to. In the case of typical PLC development, knowing the exact layout of PLCs and connected I/Os is mandatory at the beginning of development;
- The existence of global variables in PLCs makes it difficult to design software in parallel by multiple developers and re-use the developed components;
- Human-machine interfaces 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 [80].

C. Pilot applications of IEC 61499

The benefits of the Function Block architecture are being explored by researchers both in discrete manufacturing systems and in the process industry [81]. For example in [4], specific distributed process control programming tools for Function Block 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 [4] are seeking a migration path to this recently developed standard for the distributed batch process industry. In [82], another attempt to exploit the IEC 61499 model in the batch process is described. 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 [83]. This approach aims to transform and reduce switching cost from the ISA SP88 [84], an industrially accepted family of standards in batch control, to IEC 61499.

Also, the work [23] 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 [23, 85]. It is also suggested the component based approach for the batch process industry presented in [86] 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 process industry still remains to be a challenge.

D. Tools

This section describes some design environments and design approaches developed for IEC 61499 Function Blocks. 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:

- Function Block Development Kit (FBDK) [87];
- Engineering Support System CORFU [88];
- 4DIAC IDE [89];
- ISaGRAF workbench [90];
- nxtStudio [80];
- Synchronous Compiler [91];
- Cyclic run-time [92].

FBDK is one of the earliest well-developed tools for Function Block development. Even though it is considered mainly as a research tool, it is capable for demonstrating various benefits of Function Blocks in practice, such as distributed architecture and direct deployment. The tool is Java-based and relies on a Java-based run-time called Function Block 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 approach for the development of automation control systems that integrates UML with the Function Block concept.

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

ISaGRAF [90] is tool based on a combination of IEC 61131-3 and IEC 61499 standards. Due to this fact, the specification of the Function Block execution model is based on cyclic execution, similar to that the IEC 61131-3 PLC programming environments.

The NxtStudio tool is developed by NxtControl [80]. The tool uses a customized FORTE run-time. The tool introduces a novel Composite Automation Type (CAT) that is a Function Block including visualisation functionality for visualised simulation purposes, by directly following the MVC concept.

The Synchronous Compiler compiles function block applications into C code. This compiler is based on the synchronous execution model [91] and is proven to be very efficient in terms of the target code performance [93]. 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.

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 with 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 IEC 61499 tools existing on the market do not address properly the interoperability feature of IEC 61499, nevertheless basic cross platform communications via TCP/IP have been implemented.

The portability promise of IEC 61499 largely depends on implementation of platform dependent interfaces to device peripherals and networks.

There are certain semantic gaps in the 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 modelled in IEC 61499 by using the event abstraction. However, its implementation at the low level can be done in different ways.

Detailed discussion of the gaps is clearly outside the scope of this survey. They are being discussed in the research publications and solutions get implemented in the newer generation of tools. Therefore, in the authors' opinion, the IEC 61499 architecture represents a sound base for bringing the model-driven design 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 verification and validation. In this section, an attempt of synergy of these activities is presented that results in the concept of Intelligent Mechatronic Component (IMC) architecture. An IMC is composed according to the MVC pattern described in the following subsection.

A. MVC design pattern

Model-View-Control (MVC) design pattern [94] is adapted by Christensen in [40] 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 core interconnected components and several auxiliary ones. The core components are:

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

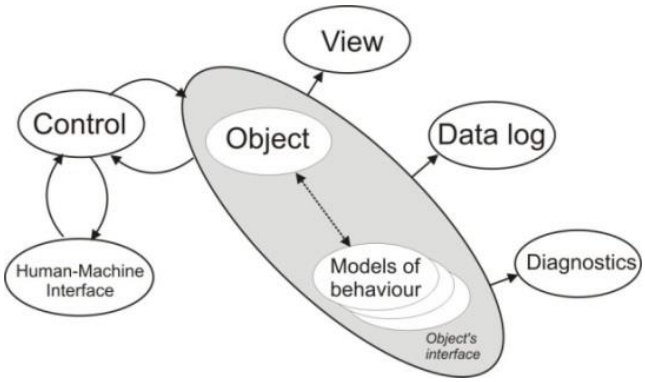


Fig. 4. MVC design pattern architecture [18].

The behavioural *Models*, provided in a repository, can be used for verification of the IMC's behaviour, or as building blocks for creating the behavioural 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' re-use.

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 re-used 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 as *Diagnostics* and *Database Logger* are also fed by the data from the *Object* or the *Model*. In contrast, the *Human-Machine Interface* (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 [18] 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 re-using 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.**

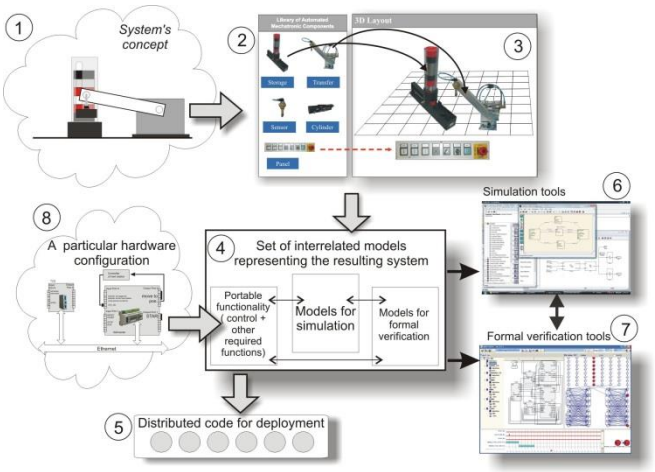


Fig. 5. Sketch of the IMC-based engineering framework supporting the integrated validation [18].

As a result of using this framework, the model of the system's behaviour can be designed with 100% re-use of the component models.

C. Use case examples

Distributed controls of baggage handling system (BHS) have been implemented in the authors' research group. This include a simulation for real-time tracking based on time prediction [61] and also using ISaGRAF tool with an OPC server (see Fig. 6) [95]. Both the examples shown here are the representations of small airport BHSs.

The nxtControl company has also demonstrated some visualised simulation examples with the development of their IEC 61499 compliant development tool called nxtStudio [80]. They implement the view component by using the CAT concept where a FB model is linked with a visualised element. Fig. 7 shows one of the examples from the building management systems sector.

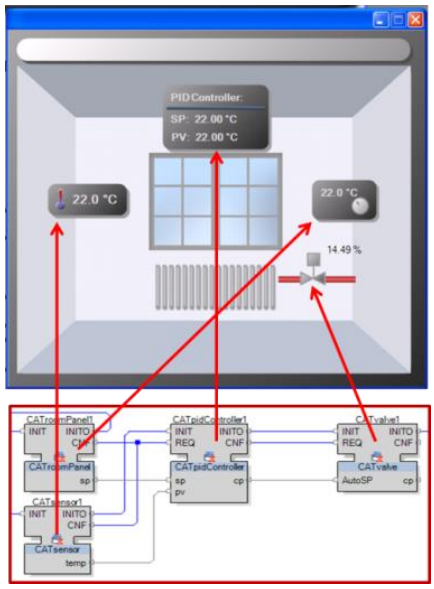


Fig. 7. Visualisation of a water tank system with distributed control implemented with nxtControl [80].

Here a set of controls and displays in a room is represented by function blocks connected to a FB implementing PID controller. The network of function blocks 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 vs. embedded control devices.

D. Verification and validation

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 verification and

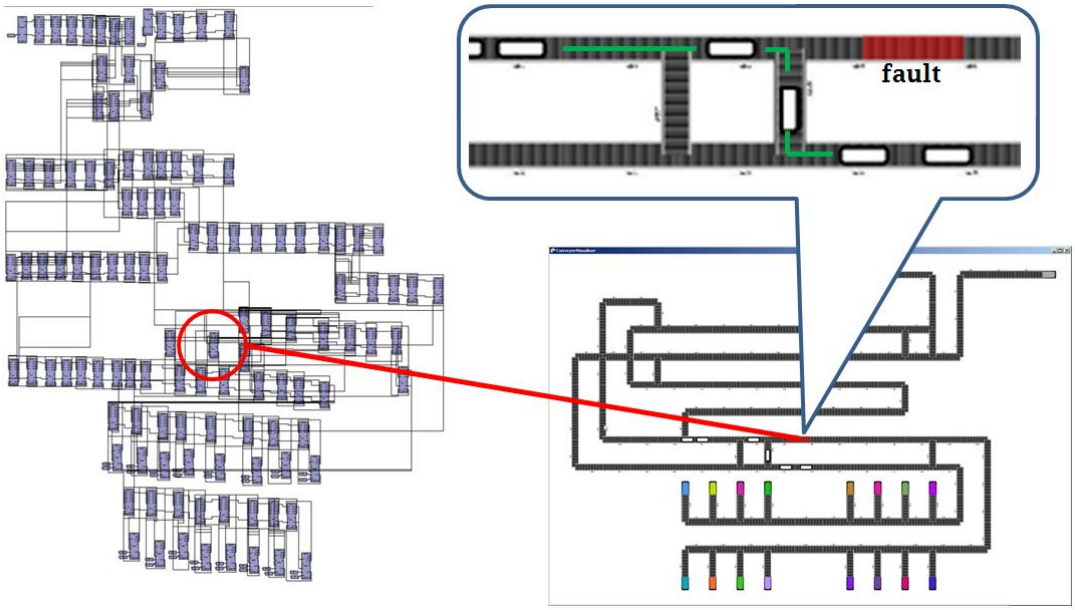


Fig. 6. BHS visualisation communicating via OPC server with ISaGRAF distributed control.

TABLE 1
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*

validation 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 function blocks with UML or SysML, e.g. [8, 39, 40, 53, 83, 96].

Simulation and model-checking are both verification techniques used in industrial automation, e.g. as exemplified in [97]. The use of simulation in a FB integrated development environment was demonstrated in [40]. A simulation run helps to check system's correctness for one specific behaviour scenario (i.e. with a fixed set of input parameters). Formal verification via model-checking [98-100] proves the system's correctness for all scenarios. The use of model-checking in industrial automation was exemplified in [97]. The formal verification of function blocks is done through their modelling 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 [101-105].

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 [18], because of the benefits of modular design (the software re-usability 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 immediately to the system validation model also corresponds to a change in the codes for deployment. Once the validation is completed, the software program is ready to be deployed into the real hardware.

One idea, proposed in [106] is to link Function Block design environments with popular tools such as MATLAB where proper validation and verification can already be performed. This may immediately show benefits by saving time and cost for re-developing 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 [40, 60, 61], that using function blocks as a modelling language is feasible and beneficial. Another possible use of models is implementation of model-predictive control in a FB-compliant embedded controller.

VIII. CONCLUSIONS

This paper surveyed the developments related to model-driven design and validation of distributed automation systems. Various engineering concepts and software tools have been evaluated and discussed, including IEC 61131, MATLAB Simulink, SCADE, OpenRTM, and IEC 61499. Engineering approaches such as block-diagram programming language and MVC concept is considered as valuable to design and validation process for distributed systems. Table 1 summarizes the capabilities of each model-driven control design and validation concepts, and an attempt to quantify differences between some of the surveyed is made. The technologies are 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 'supported'. 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 study is that there is inherent intertwining between modelling in the control sense and model-driven software engineering. It has been shown that the Function Block architecture of IEC 61499 incorporates several trends from both domains and allows for most natural implementation of distributed control systems, where validation capabilities are built in along with the features addressing classic distributed systems design challenges. The Intelligent Mechatronic Component (IMC) architecture, built "on top" of IEC 61499, is able to fill the gap in validation and verification environment for distributed automation systems.

IX. REFERENCES

[1] N. N. Chokshi and D. C. D. C. McFarlane, *A distributed coordination approach to reconfigurable process control*. Berlin ; London: Berlin ; London : Springer, 2008.

[2] U. H. Felcht, *et al.*, "The Future Shape of the Process Industries," in *Chemical Engineering: Visions of the World*, ed Amsterdam: 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," presented at the Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on, Daejeon, 2008.

[4] A. Tsuchiya, *et al.*, "Development of a distributed process control programming tool for function block description," in *Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA '99. 1999 7th IEEE International Conference on*, 1999, pp. 1321-1325 vol.2.

[5] N. Shah, "Process industry supply chains: Advances and challenges," *Computers & Chemical Engineering*, vol. 29, pp. 1225-1236, 2005.

[6] V. Vyatkin, "Intelligent mechatronic components: control system engineering using an open distributed architecture," presented at

the Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference, 2003.
[7] V. Vyatkin, *et al.*, "OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Distibuted Automation," *IEEE Transactions on Industrial Informatics*, vol. 1, pp. 4-17, Feb. 2005 2005.
[8] S. D. Panjaitan, *Development Process for Distributed Automation Systems based on Elementary Mechatronic Functions*: Shaker Verlag GmbH, Germany, 2008.
[9] I. Terzic, *et al.*, "A survey of distributed intelligence in automation in european industry, research and market," 2008, pp. 221-228.
[10] A. S. Tanenbaum and M. van Steen, *Distributed Systems*: Addison Wesley, 2004.
[11] S. Haridi, *et al.*, "Programming languages for distributed applications," *New generation computing*, vol. 16, pp. 223-261, 1998.
[12] R. Tirtea, *et al.*, "QoS monitoring at middleware level for dependable distributed automation systems," in *Suppl. Proc. 13th Int. Symp. on Software Reliability Engineering (ISSRE-2002)*, Annapolis, Marryland, 2002, pp. 217-218.
[13] K. Trkaj, "Users introduce component based automation solutions," *Computing & Control Engineering Journal*, vol. 15, pp. 32-37, 2004.
[14] M. Fowler and K. Scott, *UML distilled: a brief guide to the standard object modeling language*: Addison-Wesley Longman Publishing Co., Inc., 2000.
[15] T. Weilkiens, *Systems engineering with SysML/UML: modeling, analysis, design*: Morgan Kaufmann, 2007.
[16] "IEC 61131-3, Programmable controllers - Part 3: Programming languages, International Standard, Second Edition," ed, 2003.
[17] M. Pechoucek, *et al.*, "Expectations and deployment of agent technology in manufacturing and defence: case studies," *AAMAS Industrial Applications*, 2005.
[18] V. Vyatkin, *et al.*, "Closed-Loop Modeling in Future Automation System Engineering and Validation," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, pp. 17-28, 2009.
[19] H.-M. Hanisch, "Closed-Loop Modeling and Related Problem of Embedded COntrol Systems in Engineering," presented at the Proc Intl Conf Abstract State Machines: advances in theory and practice (ASM' 04), Lutherstadt Wittenberg, 2004.
[20] V. Vyatkin and H.-M. Hanisch, "Verification of Distributed Control Systems in Intelligent Manufacturing," *Journal of Intelligent Manufacturing*, vol. 14, pp. 123-136, 2003.
[21] J. M. Machado, *et al.*, "Increasing the efficiency of PLC program verification using a plant model," presented at the Proc Intl Conf on Industrial Engineering and Production Management (IEPM's 2003), Porto, Portugal, 2003.
[22] H.-M. Hanisch and A. Lüder, "Modular modeling of closed-loop systems," presented at the Proc of Colloquium on Petri Net Technologies for Modeling Communication Based Systems, Berlin, Germany, 2000.
[23] J. Peltola, *et al.*, "A Migration Path to IEC 61499 for the Batch Process Industry," presented at the 5th IEEE International Conference on Industrial Informatics, 2007 Vienna, Austria, 2007.
[24] M. Pechoucek and V. Marik, "Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies.," *International Journal on Autonomous Agents and Multi-Agent Systems*, 2008.
[25] G. Cândido and J. Barata, "A Multiagent Control System for Shop Floor Assembly," presented at the Proceedings of the 3rd international conference on Industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent Systems for Manufacturing, Regensburg, Germany, 2007.
[26] K. H. Hall, *et al.*, "Challenges to Industry Adoption of the IEC 61499 Standard on Event-based Function Blocks," presented at the 5th IEEE International Conference on Industrial Informatics, 2007.
[27] D. A. Bader and R. Pennington, "Cluster Computing: Applications," *The International Journal of High Performance Computing*, vol. vol. 15, pp. pp. 181-185, MAY 2001.
[28] E. Tatara, *et al.*, "Control of complex distributed systems with distributed intelligent agents," *Journal of Process Control*, vol. 17, pp. 415-427, 2007.
[29] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2 ed.: Upper Saddle River, NJ: Prentice Hall, 2003.
[30] A. Aldea, *et al.*, "The scope of application of multi-agent systems in the process industry: three case studies," *Expert Systems with Applications*, vol. 26, pp. 39-47, 2004.
[31] V. Villasenor Herrera, *et al.*, "A framework for modeling agent-based control of service-enabled manufacturing systems," in *10th IFAC Workshop on Intelligent Manufacturing Systems IMS '10*, Lisbon, Portugal 2010, pp. 49-54.
[32] V. Mařík, *et al.*, "Rockwell automation agents for manufacturing," presented at the Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, The Netherlands, 2005.
[33] F. P. Maturana, *et al.*, "Distributed multi-agent architecture for automation systems," *Expert Systems with Applications*, vol. 26, pp. 49-56, 2004.
[34] L. Mönch, *et al.*, "FABMAS: An Agent-Based System for Production Control of Semiconductor Manufacturing Processes," in *Holonic and Multi-Agent Systems for Manufacturing*. vol. 2744, V. Marík, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2004, pp. 1085-1085.
[35] R. W. Brennan and W. O, "A simulation test-bed to evaluate multi-agent control of manufacturing systems," presented at the Proceedings of the 32nd conference on Winter simulation, Orlando, Florida, 2000.
[36] P. Vrba and V. Marik, "Simulation in agent-based manufacturing control systems," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, 2005, pp. 1718-1723 Vol. 2.
[37] M. Bonfe and C. Fantuzzi, "Design and verification of mechatronic object-oriented models for industrial control systems," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, 2003, pp. 253-260 vol.2.
[38] K. C. Thramboulidis, "Using UML in control and automation: a model driven approach," in *Industrial Informatics, 2nd IEEE International Conference on*, 2004, pp. 587-593.
[39] V. Dubinin, *et al.*, "Engineering of validatable automation systems based on an extension of UML combined with function blocks of IEC 61499," *IEEE International Conference on Robotics and Automation (ICRA'06)*, pp. 3996-4001, 2005.
[40] J. H. Christensen, "Design patterns for systems engineering with IEC 61499," presented at the Verteilte Automatisierung - Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung, Magdeburg, Germany, 2000.
[41] M. Rausch and H. M. Hanisch, "Net condition/event systems with multiple condition outputs," in *Emerging Technologies and Factory Automation, 1995. ETFA '95, Proceedings., 1995 INRIA/IEEE Symposium on*, 1995, pp. 592-600 vol.1.
[42] C. Sünder, *et al.*, "Functional structure-based modeling of automation systems," *International Journal of Manufacturing Research*, vol. 1, pp. 405-420, 2006.
[43] T. Strasser, *et al.*, "A research roadmap for model-driven design of embedded systems for automation components," in *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, 2009, pp. 564-569.
[44] T. Strasser, *et al.*, "Multi-domain model-driven design of Industrial Automation and Control Systems," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, 2008, pp. 1067-1071.
[45] T. Strasser, *et al.*, "Model-driven embedded systems design environment for the industrial automation sector," in *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, 2008, pp. 1120-1125.
[46] K. Thramboulidis, "IEC 61499 in Factory Automation," in *IEEE International Conference on Industrial Electronics, Technology and Automation (CISSE-IETA'05)*, Bridgeport, USA, 2005.
[47] C. Tranoris and K. Thramboulidis, "From Requirements to Function Block Diagrams: A new Approach for the design of industrial applications," in *10th IEEE Mediterranean Conference on Control and Automation, MED'02*, Lisbon, Portugal, 2002.
[48] C. Tranoris and K. Thramboulidis, "Integrating UML and the function block concept for the development of distributed control applications," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, 2003, pp. 87-94 vol.2.
[49] D. Hästbacka, *et al.*, "Model-driven development of industrial process control applications," *Journal of Systems and Software*, vol. 84, pp. 1100-1113, 2011.
[50] T. Vepsalainen, *et al.*, "Assessing the industrial applicability and adoption potential of the AUKOTON model driven control

application engineering approach," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, 2010, pp. 883-889.
[51] J. Peltola, *et al.*, "Challenges in industrial adoption of model-driven technologies in process control application design," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, 2011, pp. 565-572.
[52] M. Hirsch and H.-M. Hanisch, "Systemspezifikation mit SysML für eine Fertigungstechnische Laboranlage," in *Fachtagung zum Entwurf komplexer Automatisierungssysteme (EKA 08)*, Magdeburg, Germany 2008, pp. 23-34.
[53] M. Hirsch, *Systematic Design of Distributed Industrial Manufacturing Conrol Systems*. Berlin: Logos Verlag 2010.
[54] OpenRTM-aist. (2010, *OpenRTM-aist official website*. Available: http://www.openrtm.org/
[55] N. Ando, *et al.*, "A Software Platform for Component Based RT-System Development: OpenRTM-Aist," in *Simulation, Modeling, and Programming for Autonomous Robots*. vol. 5325, S. Carpin, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2008, pp. 87-98.
[56] "International Standard Draft IEC61804-2: Function blocks (FB) for process control," in *Part 2: Specification of FB concept and Electronic Device Description Language (EDDL)*, ed: International Electrotechnical Commission, 2004.
[57] C. Diedrich, *et al.*, "Function Block Applications in Control Systems Based on IEC 61804," 2001.
[58] NI. (2010, *NI LabVIEW - The Software That Powers Virtual Instrumentation - National Instruments*. Available: http://www.ni.com/labview/
[59] V. Vyatkin and H.-M. Hanisch, "Application of Visual Specifications for Verification of Distributed Controllers," in *2001 IEEE International Conference on Systems, Man, and Cybernetics*, Tucson, AZ, USA, 2001, pp. 646-651
[60] V. Vyatkin, *IEC 61499 function blocks for embedded and distributed control systems design*. Research Triangle Park, NC: ISA-Instrumentation, Systems, and Automation Society, 2007.
[61] G. Black and V. Vyatkin, "Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499," *IEEE Transactions on Automation Science and Engineering*, vol. 7, pp. 337-351, Apr 2010.
[62] W. Lepuschitz, *et al.*, "Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, pp. 52-69, 2011.
[63] I. Hegny, *et al.*, "Integrating Software Agents and IEC 61499 Realtime Control for Reconfigurable Distributed Manufacturing Systems," *2008 International Symposium on Industrial Embedded Systems*, pp. 249-252, 2008.
[64] X. M. Huang, "Intelligent and Reconfigurable Control of Automatic Production Line by Applying IEC61499 Function Blocks and Software Agent," in *IEEE International Conference on Mechatronics and Automation*, 2009, pp. 1481-1486.
[65] G. Zhabelova and V. Vyatkin, "Multi-agent Smart Grid Automation Architecture based on IEC 61850/61499 Intelligent Logical Nodes," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 2351 - 2362 2011.
[66] J. Lunze, "What Is a Hybrid System?," in *Lecture Notes in Control and Information Science 279: Modelling, Analysis, and Design of Hybrid Systems*, ed Berlin Heidelberg: Springer-Verlag, 2002.
[67] E. Lee, "Cyber Physical Systems: Design Challenges," University of CaliforniaJanuary 23 2008.
[68] B. Zhou, *et al.*, "Application of the generic modelling method for system of systems to manufacturing domain," presented at the IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, 2011.
[69] MathWorks. (2010, *The MathWorks - MATLAB and Simulink for Technical Computing*. Available: http://www.mathworks.com
[70] B. I. Silva and B. H. Krogh, "Formal verification of hybrid systems using CheckMate: a case study," in *American Control Conference, 2000. Proceedings of the 2000*, 2000, pp. 1679-1683 vol.3.
[71] J. H. Taylor, "A modeling language for hybrid systems," in *Computer-Aided Control System Design, 1994. Proceedings., IEEE/IFAC Joint Symposium on*, 1994, pp. 339-344.
[72] J. H. Taylor and D. Kebede, "Modeling and simulation of hybrid systems," presented at the Decision and Control, 1995., Proceedings of the 34th IEEE Conference on, 1995.
[73] T. A. Henzinger, *et al.*, "HYTECH: the next generation," in *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, 1995, pp. 56-65.
[74] S. Kowalewski, *et al.*, "An environment for model-checking of logic control systems with hybrid dynamics," in *Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on*, 1999, pp. 97-102.
[75] R. A. Santos, *et al.*, "Distributed continuous process simulation: An industrial case study," *Computers & Chemical Engineering*, vol. In Press, Corrected Proof.
[76] J. Chen and R. A. Adomaitis, "An object-oriented framework for modular chemical process simulation with semiconductor processing applications," *Computers & Chemical Engineering*, vol. 30, pp. 1354-1380, 2006.
[77] (2012, *Distributed Simulation Toolbox*. Available: http://www.mathworks.com/products/connections/product_detail/product_35768.html
[78] N. P. Mahalik and K. Kim, "A prototype for hardware-in-the-loop simulation of a distributed control architecture," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 38, pp. 189-200, 2008.
[79] S. Mishra and N. Mahalik, "Virtual DCS and specification," *International Journal of Information and Communication Technology*, vol. 3, pp. 339-353, 2011.
[80] nxtControl.com. (2010. Available: http://www.nxtcontrol.com/
[81] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review," *IEEE Transactions on Industrial Informatics*, vol. 7, pp. 768-781, 2011.
[82] K. Thramboulidis, *et al.*, "An IEC61499 Based Approach for Distributed Batch Process Control," presented at the INDIN, 2007.
[83] K. C. Thramboulidis, "Using UML in control and automation: a model driven approach," in *Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on*, 2004, pp. 587-593.
[84] "ISA-S88.01-1995, Standard: Batch Control. Part 1: Models and Terminology," ed: The International Society for Measurement and Control, 1995.
[85] J. P. Peltola, *et al.*, "Process Control with IEC 61499: Designers' Choices at Different Levels of the Application Hierarchy," in *Industrial Informatics, 2006 IEEE International Conference on*, 2006, pp. 183-188.
[86] S. Kuikka, "A batch process management framework: Domain-specific, design pattern and software component based approach," Doctor of Technology Dissertation, Technical Research Centre of Finland, Helsinki University of Technology, Espoo, Finland, 1999.
[87] Holobloc. *Function Block Development Kit*. Available: http://www.holobloc.com
[88] K. Thramboulidis, "Development of Distributed Industrial Control Applications: The CORFU Framework," in *4th IEEE International Workshop on Factory Communication Systems*, Sweden, 2002.
[89] 4DIAC. (2010, *Framework for Distributed Industrial Automation (4DIAC)* Available: http://www.fordiac.org
[90] ISaGRAF. *ICS Triplex ISaGRAF Inc. - leading IEC 61131 and IEC 61499 software*. Available: http://www.isagraf.com
[91] L. H. Yoong, *et al.*, "A synchronous approach for IEC 61499 function block implementation," *IEEE Transactions on Computers*, vol. 58, pp. pp. 1599–1614, December 2009 2009.
[92] P. Tata and V. Vyatkin, "Proposing a novel IEC61499 runtime framework implementing the Cyclic Execution semantics," presented at the 7th IEEE International Conference on Industrial Informatics (INDIN 2009), Cardiff UK, 2009.
[93] L. H. Yoong, *et al.*, "Efficient implementation of IEC 61499function blocks," presented at the IEEE International Conference on Industrial Technology (ICIT), Gippsland, 2009.
[94] (1979, *Model-View-Controller design pattern*. Available: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html
[95] J. Yan and V. Vyatkin, "Distributed Execution and Cyber-Physical Design of Baggage Handling Automation with IEC 61499," presented at the 9th International IEEE Conference on Industrial Informatics, (INDIN'11), Lisbon, Portugal, 2011.
[96] V. Dubinin and V. Vyatkin, "On Definition of a Formal Semantic Model for IEC 61499 Function Blocks," *EURASIP Journal of Embedded Systems*, vol. 2008, p. 10, 2008.
[97] H. Hanisch, *et al.*, "Formal validation of intelligent-automated production systems: Towards industrial applications,"

International Journal of Manufacturing Technology and Management, vol. 8, pp. 75-106, 2006.

[98] G. Frey and L. Litz, "Formal methods in PLC programming," presented at the Systems, Man, and Cybernetics, 2000 IEEE International Conference, 2000.

[99] E. M. Clarke, *et al.*, *Model Checking*: Cambridge: The MIT Press, 1999.

[100] W. Farn, "Formal verification of timed systems: a survey and perspective," *Proceedings of the IEEE*, vol. 92, pp. 1283-1305, 2004.

[101] P. Cheng and V. Vyatkin, "Automatic model generation of IEC 61499 function block using net condition/event systems," presented at the 6th IEEE International Conference on Industrial Informatics, 2008.

[102] N. Hagge and B. Wagner, "Java code patterns for Petri net based behavioral models," presented at the 3rd IEEE International Conference on Industrial Informatics, 2005.

[103] V. Vyatkin, *et al.*, "Object-oriented modular place/transition formalism for systematic modeling and validation of industrial automation systems," presented at the Industrial Informatics, 2003. INDIN 2003. Proceedings. IEEE International Conference on, 2003.

[104] G. Cengic and K. Akesson, "On Formal Analysis of IEC 61499 Applications, Part A: Modeling," *Industrial Informatics, IEEE Transactions on*, vol. 6, pp. 136-144, 2010.

[105] C. Gerber and H. M. Hanisch, "Does portability of IEC 61499 mean that once programmed control software runs everywhere?," presented at the 10th IFAC Workshop on Intelligent Manufacturing Systems, Lisbon, Portugal, 2010.

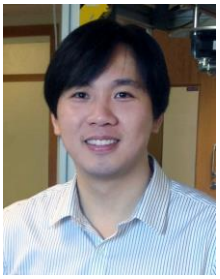
[106] C. Yang and V. Vyatkin, "Model transformation between MATLAB Simulink and Function Blocks," in *IEEE International Conference on Industrial Informatics (INDIN'10)*, Osaka, Japan, pp. 1130-1135.

Chia-han Yang received the B.E. (1st class honours) and Ph.D. in Electrical and Electronics Engineering from University of Auckland, Auckland, New Zealand in 2006 and 2011 respecitvely. His Ph.D. research focused on investigating methodologies of improving distributed control system design (based on IEC61499 standard) through closed-loop validation process. His research interests are in the area of distributed control, industrial automation, modelling & simulation, and robotics.

In 2011, he was a research engineer with the department of Electricial and Computer Engineering, University of Auckland, continuing the research work in the simulation of distributed control systems. He is currently a research engineer doing robotics related research and development at Centre for Autonomous System (CAS), University of Technology Sydney, Australia.

Valeriy Vyatkin is Chaired Professor of Dependable Computation and Communication Systems at Luleå University of Technology, Sweden, and visiting scholar at Cambridge University, U.K., on leave from The University of Auckland, New Zealand, where he has been Associate Professor and Director of the InfoMechatronics and Industrial Automation lab (MITRA) at the Department of Electrical and Computer Engineering. He graduated with the Engineer degree in applied mathematics in 1988 from Taganrog State University of Radio Engineering (TSURE), Taganrog, Russia. Later he received the Ph.D. (1992) and Dr. Sci. degree (1998) from the same university, and the Dr. Eng. Degree from Nagoya Institute of Technology, Nagoya, Japan, in 1999. His previous faculty positions were with Martin Luther University of Halle-Wittenberg in Germany (Senior researcher and lecturer, 1999–2004), and with TSURE (Associate Professor, Professor, 1991–2002).

Research interests of professor Vyatkin are in the area of dependable distributed automation and industrial informatics, including software engineering for industrial automation systems, distributed architectures and multi-agent systems applied in various industry sectors: Smart Grid, material handling, building management systems, reconfigurable manufacturing, etc. Dr Vyatkin 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. In 2012, Prof Vyatkin has been awarded Andrew P. Sage Award for best IEEE Transactions paper.



Cheng Pang (S'08-M'13) received the BE (Hons) and ME (Hons) degrees in Computer Systems Engineering and the PhD degree in Electrical and Electronic Engineering from the University of Auckland, New Zealand in 2005, 2007, and 2013 respectively. He is a postdoctoral research fellow in the Laboratory of Advanced Computing and Communications for Industrial Applications at Luleå University of Technology, Sweden. His main research interests include model-driven engineering for industrial automation systems, building automation and control systems, and distributed control for the Internet of Things.