# Automation with Decentralised Logic in Industrial Internet

*Valeriy Vyatkin*
Aalto University, School of Electrical Engineering, FI-00076 Aalto, Finland
Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, Sweden
vyatkin@ieee.org

*Cheng Pang*
Aalto University, School of Electrical Engineering, FI-00076 Aalto, Finland
cheng.pang.phd@ieee.org

*Ari Kaukovirta*
Microteam Oy, Manager of Automation Systems, Hatanpäänkatu 3G, FI-33900 Tampere, Finland
Tel: + 358 20 766 0724, ari.kaukovirta@microteam.fi

*Wenbin Dai*
Shanghai Jiao Tong University, China
w.dai@ieee.org

KEY WORDS Decentralized Control, Industrial Internet, Cyber-Physical Systems, IEC 61499, PLC

## ABSTRACT

The advent of Internet of Things in Industrial Automation challenges the conventional automation software architectures based on PLCs and IEC 61131-3 in terms of control decentralization, design flexibility, and system productivity. The IEC 61499 standard was introduced as an extension of IEC 61131 to address these issues by establishing a new event-driven control architecture for developing reusable, interoperable, and reconfigurable distributed industrial automation systems.

## 1 INTRODUCTION

Today's industrial automation industry is facing challenges from both the market and the advancement of new technologies in materials, processes, and ICT. The Internet revolution strongly influences the industrial landscape, in particular the future of industrial automation. During the last few years several visionary concepts have appeared that are based on the ubiquitous connectivity of devices, sensor networks, and cloud computing, known as Internet of Things (IoTs) /1/, Cyber-Physical Systems /2/, and Industrial Internet /3/. The leading academic institutions, industries, and governments world-wide have come up with plans and strategies for the long-term development. For example, in Germany it resulted in an ambitious program called "Industrie 4.0" /4/ that aims at creating the 4$^{th}$ industrial revolution. "Industrie 4.0" and other similar concepts aim at mass customization. This requires substantial changes in manufacturing machinery: modular and reconfigurable machines more resembling intelligent robots, if not in appearance, then in behaviour. This, in turn, requires decentralized and reconfigurable control. The conventional automation systems are using centralized control architecture that works very well in

mass production scenarios with centralized control but do not fit the new reality of distributed automation configuration for the IoTs. The IEC 61499 reference architecture /5/ has been introduced to complement the insufficiencies and extend the functionality of IEC 61131-3 /6/ to distributed systems, which is achieved by using a component model with event-driven execution semantics.

## 2 KEY CONCEPTS OF IEC 61499 STANDARD

In IEC 61499, the well-known concept of function block (FB) is extended to become a software component that can be executed in an arbitrary compliant device. An application is represented as a network of FBs, connected via data and control signals. The main "magic" of the IEC 61499 technology is its ability to seamlessly deploy such a network of function block to distributed control devices. As long as the PLCs or micro-controllers embedded into intelligent actuators and sensors are compliant with IEC 61499, they can receive and execute a part of the application. Reallocation of the components between available devices can be done in no time and with zero effort. This paves the way to using IoTs technologies widely in industrial automation environments. IEC 61499 promotes a top-down system engineering flow and provides a system-level view on the intelligence distributed across complex automation systems. This is especially helpful when it comes to verification and validation of a system's behaviours. On the level of a single component, IEC 61499 provides developers with well-known languages from IEC 61131-3. In a way, a single component can be seen as a model of a single PLC, small or big. But, it is even more beneficial to represent a set of functionalities, related to a particular machine, or a part thereof, as a single component. Then, a change in a factory floor layout or a change of functionality can be easier translated to modifications of automation programs.

This concept was convincingly demonstrated in many research projects. For example, a novel shoe manufacturing facility /7/ built at the ITEA-CNR research institute in Italy is automated in such a way as shown in Figure 1. This facility is able to produce individually tailored shoes with similar costs as massively manufactured ones. The facility is composed of highly flexible material handling modules named *terns*, which can be further combined to form *molecular lines*. Each molecular line is composed of arbitrary number of terns for a particular sequence of operations. As indicated in Figure 1 (a), this shoe facility can accommodate multiple product paths. Thanks to IEC

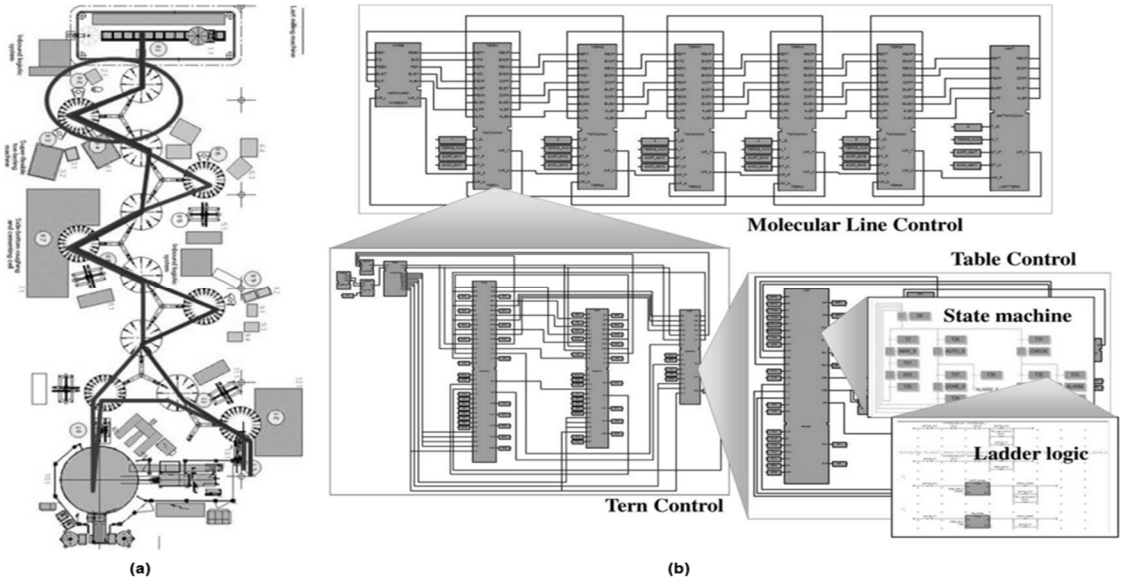


**Figure 1**. (a) Flexible shoe manufacturing facility and (b) Its IEC61499 implementation /7/.

61499, as shown in Figure 1 (b), the control software of a molecular line can closely follow its physical structure. This high degree of code modularity enables reusability and re-configurability of the equipment. Another interesting feature of IEC 61499 is the extended use of state-machines inside FBs. State-based control fits most naturally many types of applications, thus this feature has been very positively received by researchers and developers. The AutomationIT group at Aalto University is experimenting with implementation of control based on IoTs in several application areas exemplified in the following sections.

# 3 PILOT DEMONSTRATIONS

## 3.1 Plug and Play Mechatronics

In this case study /8/ we chose a simplified version of a pick and place manipulator with remote control, as shown in Figure 2 (a). The manipulator is composed of two pneumatic cylinders and a joystick for remote control. Each cylinder is equipped with two valves enabling forward and backward motion, and two end position sensors, indicating the extended and retracted positions. There could be great variety of valves and sensors that can be provided by suppliers, each of which may set specific requirements to the control implementation. We will encapsulate these details into the corresponding software components, to demonstrate how the particulars can be masked if they are not essential at the top integration level. For the purposes of this study, we will assume that each constituent part of the manipulator can have its own control and wireless communication capabilities.

The FB application implementing this control program is presented in Figure 2 (b). The application can be clearly divided into three parts: one joystick model and two cylinder models. Each cylinder is implemented as a program with five FBs: one for cylinder, two for sensors, and two for valves. The FB application includes four models of *intelligent valves*, which have embedded controllers. They are represented by the FB type *iValve*. The advantage of this solution is that no central controller is required at all.

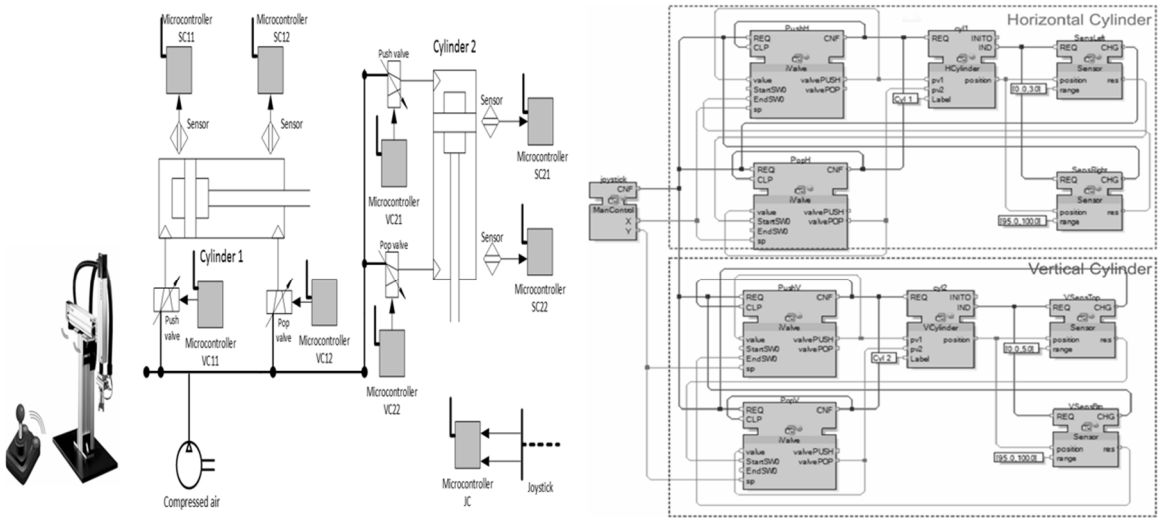


**Figure 2**. (a) Pick and place manipulator with extreme distributed control architecture and (b) Function block model of the manipulator control system.

## 3.2 Process Automation

Another promising application area for distributed automation is process technologies. The collaboration and interoperability gain of the Internet-based control is demonstrated by distributing control logic of water process testbed at Aalto. As indicated in Figure 3 (a), the controllers are connected via standard Ethernet in a daisy chain topology. Each controller is responsible for some physical devices in the system, with the extreme case being that each such device (valve, pump, sensor, etc.) can have its own embedded controller. The functionality of the distributed control has been designed in a service-oriented manner, where each physical device is providing a set of services, encapsulated in FBs. The service-oriented software architecture follows the SOA stack as described in /9/. In this approach, upper layer services trigger lower layer services by sending request signals and collect response data from lower layer services. The presentation layer in the SOA approach provides a semantic view of system configurations. The core service layer is designed for FBs provided by vendors. Those services are platform-dependent and usually implemented in as service interface FBs for IEC 61499. The user defined services layer are reusable functions developed in this system configuration which could be utilized in the future. Those services provide an abstract layer for core services and a bridge between processes and base functions. The process layer composes basic functionalities from user-defined services and core services to form simple sequences. Finally, the presentation layer is the "brain" of the control system. Service sequences indicating control flow are implemented in a single or a network of FBs to provide semantic information by arranging simple sequences in the correct order. For example, an implementation of the process service layer is illustrated in Figure 3 (b). In the process layer, services are grouped by functionalities in the process, namely tank control (*FB_TankControl*), PID control (*FB_PIDControl*), heater control (*FB_HeaterControl*), pump control (*FB_PumpControl*) and valve control (*FB_ValveControl*). The tank control service collects alarm signals from sensor measurement services and generates tank status, such as "is tank ready for feed in and out water", "can tank be heated and is tank over pressured" and so on. The PID control service reads process value from the flow measurement service and recalculates control values for valve and pump control services. The heater, pump, and valve control services check that the control value is in the valid range and produce the output command to the actuator control services.

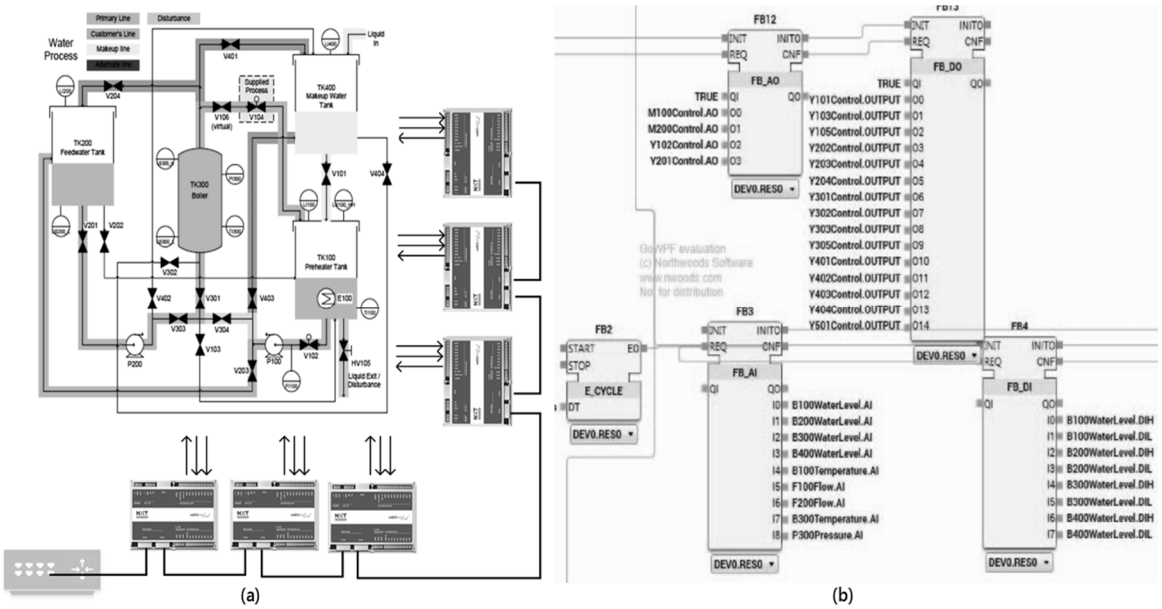


**Figure 3**. (a) Water process testbed with distributed control and (b) Function block implementation of the process service layer in SOA stack /9/.

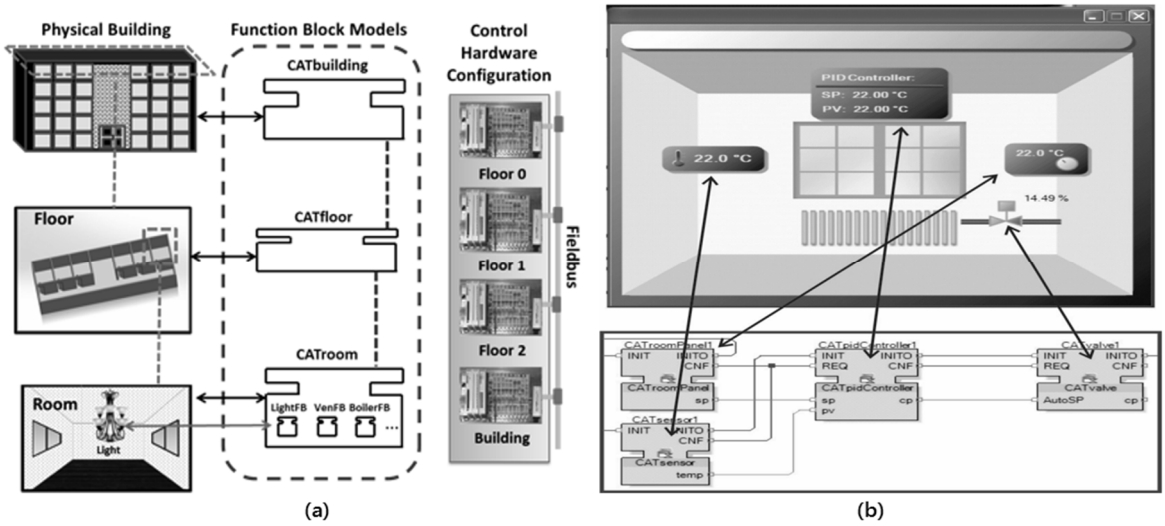


**Figure 4**. (a) Design architecture for decentralized BMS (b) Internal Structure and HMIs of BoilerFB.

## 3.3 Building Management Systems

Decentralised control can be successfully applied as a system-level design architecture for decentralized *Building Management Systems* (BMS) with *Demand Side Management* (DSM) using the IEC 61499 standard /10/. This architecture facilitates the development of automation software of BMS, which has an embedded capability to conduct simulation in the loop. By employing IEC 61499 in BMS domain, all subsystems can be autonomously controlled and coordinated at various levels. More importantly, the control application can be developed independently from the deployment configurations. Figure 4 (a) schematically presents the overall structure of an IEC 61499 control application for a decentralized BMS. The proposed BMS design follows a bottom-up approach. First of all, functions of each subsystem for a room are implemented in a corresponding FB. For example, *LightFB* is used to control the light switch and brightness level of the room; *BoilerFB* adjusts the room's heating; and *VenFB* manages the room's ventilation. The *LightFB*, *BolierFB*, *VenFB*, and other required FBs constitute the *CATroom* FB, which provides control functions and HMIs for all room appliances. The individually configured *CATroom* FB instances can then be coordinated by a *CATfloor* FB. This *CATfloor* FB provides a central control point for all rooms in the same floor. Commands received by the *CATfloor* FB will be interpreted and then sent to all the corresponding rooms in the floor. Similarly, following the control hierarchy, the *CATbuilding* FB is used to interpret building-level control commands for *CATfloor* FBs. For example, when a "building shutdown" command is received, the *CATbuilding* FB will dispatch corresponding floor-level commands to all the associated *CATfloor* FBs. Subsequently, these *CATfloor* FBs will command their *CATroom* FBs to switch off all room appliances. The floor- and building-level control hierarchies are not compulsory for the BMS's proper functioning. They are used to improve manageability of large buildings. For small buildings, such as the nine-room building shown in Figure 4 (b), the BMS is implemented as a network of nine instances of *CATroom* FB.

## 3.4 Automotive Control Systems

The benefits of distributed logic for managing device-independent control systems can be exemplified by the following automotive control system developed for Volvo buses. There were several reasons why this system was developed by Microteam for Volvo. One of the biggest challenges in automotive software development is the

amount of configurations that need to be maintained. In every such industry, where products are varying a lot, the management of control systems and software life cycles are challenging. Quite often the interfaces between product development, production, and aftermarket are poorly defined and inefficient. By building control software from separate components and functions, control solutions can be easily reused and adapted in new projects with different topologies of hardware and networks. To achieve this goal, software component descriptions and interfaces must be standardized in a way similar to the philosophy proposed in IEC 61499. Another challenge is hardware independency. When an automation product manufacturer chooses a software development platform, he often becomes dependent on a specific hardware family. For example, the traditional way of programming Electronic Control Units (ECUs) with C code implies target dependency. Higher level abstraction and programming languages can help mitigate such hardware dependency. These issues led to the development of a system, as indicated in Figure 5, with the following major functions.

**Application development tool**: This is the tool to create software components that are required for building the complete control software package for a product. Vehicle electrical functions are defined as function specifications where functionality is described using IEC 61131-3 languages. Other software components are I/O descriptions, architecture descriptions, user interface components, event control chains, and so on.

**PDM/PLM/BOM manager subsystem**: From here comes the software bill of material (BOM) based on the order breakdown. The PDM controls also the product lifecycle management with component part numbers and replacement chains.

**Component data management/Package generator**: The software package for the product is generated based on components referred by BOM. According to the given rules, the functions are split and distributed to the available ECUs in the product. Executable code for ECUs is generated and configuration for all required communication between ECUs is compiled. Product parameters are constructed to software package. The timing constraints against ECU's capabilities and communication networks capabilities are analysed. AES encryption and key and password handling is managed by CDM. The software package is embedded with onboard web documentation with support in the languages requested by the order.

The output of this automated process is a software package that is delivered to shop floor where it is downloaded to the vehicle on the production line and the tests for the delivering the vehicle can start. Each product can have its variations in functionality but there is no need for software engineering in production line and yet production

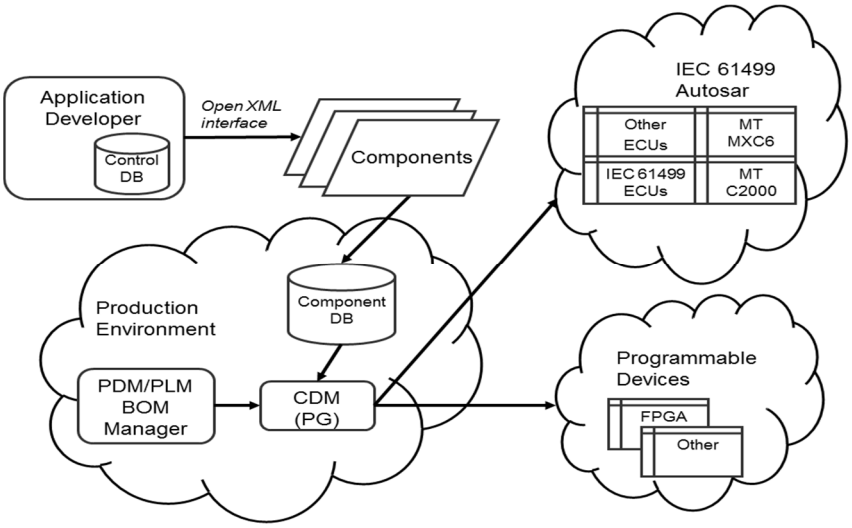


**Figure 5**. Process flow from order to product.

receives a functional entity of software and hardware that fulfils the requirements set to the particular instance of the product. Benefits of this system are plenty, for example:

- Code design work is easily distributed globally due to the principle that the software development process is divided in to development of individual functions.
- Strict rules govern that the generation and compilation of distributed code produces functional entities.
- Software components are kept separate from the hardware platform. This system connects software to hardware through specifications and requirements. As long as those are obeyed, the components can be changed and developed freely. For instance, by keeping stable interface and requirement specifications of ECUs, when new ECU versions emerge the old ones are not needed as spare parts and migrating to the next generation ECUs is easy. Since software is not bound with hardware version, the full control of software life cycle is achieved and the software is independent from hardware. Target-independent software components provide independence from hardware components and suppliers.
- Distribution of control code to ECUs is automated and can be repeated at any time. The system also knows what code should be in which ECU and enabling the automated programming of ECUs. This means that ECUs can be replaced in the field without the needs of programming and any manual software installation.
- The system also automates maintenance, as like other components the software components have replacement chains, which enables automatic software update for devices. The system is integrated with PDM/PLM/ERP systems. Therefore, the software components are seen as similar objects as any other component in the system by the higher level production management.

## 5 ADOPTION AND BENEFITS

The adoption of any new technology is not easy, especially in automation with long investment cycles and strong legacy solutions. Nevertheless, IEC 61499 is making good progress. There are notable early adopters in building automation and process technologies, in which the appeal of distributed automations is especially strong. The system integrators who used IEC 61499 have reported benefits in terms of reduced engineering cycles on account of reusing a good deal of solutions and the ability to adapt them in new projects with different configurations of hardware and networks. The key to successful mastering of new technology is through proper education and training. As the experiences showed, the concepts of component-based design of IEC 61499 were very quickly learned by the new generation of developers. However, re-educating seasoned PLC developers requires a bit more systematic approach. Another crucial factor for the success is the availability of mature hardware platforms and software tools. Currently, dozens of PLCs are available on the market with full support of IEC 61499 technology. One such software platform is ISaGRAF /11/, which has been ported to countless number of hardware targets. Another platform, nxtSTUDIO /12/, is from nxtControl, with available PLCs from Beckhoff, Mitsubishi, Advantech, Wago, Bosch, SIEMENS, and many more. The open source 4DIAC platform /13/ is also quite popular among smaller vendors of PLCs.

For example, ISaGRAF is a powerful standard-compliant programming solution for industrial automation. It provides a comprehensive set of software technologies for developing a wide range of distributed automation systems. ISaGRAF addresses both technical and usability aspects in the design of automation systems to satisfy expectations of industrial markets for standards, performance, and functionality. With the compliance with IEC

61131 and IEC 61499, ISaGRAF allows products to be certified according to international automation standards. ISaGRAF consists of two utilities: Application Workbench and Runtime Target. The Application Workbench provides a full-fledged integrated development environment for developing, executing, debugging, online monitoring, offline simulating, and generating highly portable control applications. It fully supports all IEC 61131-3 and IEC 61499 programming languages. The Runtime Target is a portable execution engine that can run on any operating systems including Linux, VxWork, OS-9, INtime, RTX, Windows, etc., and any hardware platforms, such as Intel, Motorola, ARM, PowerPC, etc. By leveraging the ISaGRAF technologies, control applications only need to be developed once for all operating systems and hardware platforms. Recently, special attention has been paid to the support of up-coming standards related to SmartGrid, such as IEC 61850 /14/, whose stack has been tightly integrated with the runtime target. ISaGRAF also provides interfaces to field devices using standard protocols, such as ModbusTCP, CANopen, and EtherCAT. Since I/O drivers are implemented as C functions linked to the ISaGRAF firmware, OEMs have the flexibility of developing their own drivers with ISaGRAF's easy-to-use I/O development kit, which includes step-by-step instructions and various samples. ISaGRAF can also be customized and extended in a very flexible way. It is possible to extend the development environment with new elements (called plugins), which can implement custom editors of programming languages, or support of vendor-specific features, etc. For example, the German vendor Infoteam Software AG has decided to base on ISaGRAF technologies the new version of their established OpenPCS development environment. In Finland and the Nordic countries, MicroTeam is the first operator providing automation control solutions using the IEC 61499 standard. Combined with the profound expertise in IEC 61499 available at Aalto University, it promises great perspectives of fruitful collaboration in research and education to the benefits of Finnish automation industry in the IoTs era.

## 6 REFERENCES


[1] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," IEEE Transactions on Industrial Informatics, vol. 10(4), pp. 2233-2243, 2014.

[2] E. A. Lee, "Cyber Physical Systems: Design Challenges," in 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC 2008), Orlando, FL, US, 2008, pp. 363-369.

[3] P. C. Evans and M. Annunziata. (2012). Industrial Internet: Pushing the Boundaries of Minds and Machines [Online]. Available: http://www.ge.com/docs/chapters/Industrial_Internet.pdf

[4] R. Drath and A. Horch, "Industrie 4.0: Hit or Hype?," IEEE Industrial Electronics Magazine, vol. 8(2), pp. 56-58, 2014.

[5] Function blocks — Part 1: Architecture, IEC Standard 61499-1, 2012.

[6] Programmable controllers — Part 3: Programming languages, IEC Standard 61131-3, 2013.

[7] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," IEEE Transactions on Industrial Informatics, vol. 7(4), pp. 768-781, 2011.

[8] D. Kleyko, E. Osipov, S. Patil, V. Vyatkin, and P. Zhibo, "On Methodology of Implementing Distributed Function Block Applications using TinyOS WSN nodes," in 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2014), Barcelona, Spain, 2014, pp. 1-7.

[9] W. Dai, J. Peltola, V. Vyatkin, and C. Pang, "Service-Oriented Distributed Control Software Design for Process Automation Systems," in 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2014), San Diego, CA, US, 2014, pp. 3637-3642.

[10] V. Vyatkin, C. Pang, Y. Deng, M. Sorouri, and H. Mayer, "System-level Architecture for Building Automation Systems: Object-Orientated Design and Simulation," in 39th Annual Conference on IEEE Industrial Electronics Society (IECON 2013), Vienna, Austria, 2013, pp. 5334-5339.

[11] ICS Triplex ISaGRAF. ISaGRAF Workbench [Online]. Available: http://www.isagraf.com

[12] nxtControl. nxtSTUDIO [Online]. Available: http://www.nxtcontrol.com/en/engineering/

[13] Framework for Distributed Industrial Automation (4DIAC) [Online]. Available: http://www.fordiac.org

[14] Communication networks and systems for power utility automation - Part 1: Introduction and overview, IEC/TR 61850-1, 2013.