V. Vyatkin, C. Pang, S. Tripakis, "Towards Cyber-Physical Agnosticism by Enhancing IEC 61499 with PTIDES Model of Computations", International Annual conference of IEEE Industrial Electronics Society, Yokohama, November, 2015

Towards Cyber-Physical Agnosticism by Enhancing IEC 61499 with PTIDES Model of Computations

Valeriy Vyatkin[1,3], Cheng Pang[1], and Stavros Tripakis[2,4]
[1]Department of Electrical Engineering and Automation, Aalto University, Finland
[2]Department of Computer Science, Aalto University, Finland
[3]Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Sweden
[4]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA
{vyatkin, cheng.pang.phd}@ieee.org, stavros@eecs.berkeley.edu

***Abstract*—This paper addresses software design for cyber-physical automation systems that enables invariant properties of the physical system in case of software reallocation to different hardware. The proposed approach is based on the distributed reference architecture of IEC 61499 standard enhanced with a time-stamping mechanism. It is demonstrated that the proposed approach complements the abilities of IEC 61499 to maintain correct causality of distributed system execution with improved performance of physical system property called cyber-physical agnosticism. The time-stamped event semantics of IEC 61499 is introduced and mapped to the PTIDES execution model of Ptolemy II. We have experimentally validated that changing the model of computation in distributed automation to a time-stamped event-driven one can bring substantial improvements in flexibility and reconfigurability of cyber-physical automation systems.**

Keywords— distributed automation, time-stamped events, IEC 61499, PTIDES, Ptolemy II

I. INTRODUCTION

Applications of the Internet of Things (IoT) architecture in industrial automation raise the question whether automation systems can be efficiently implemented using distributed and possibly wirelessly networked sensors and actuators that communicate asynchronously, as opposed to the currently used rigid wired fieldbus connections based on synchronous communication protocols. As the automation systems become more software intensive, it is desirable to design and verify system-level models of software and guarantee invariance of physical system behaviour when the same software is executed on different IoT configurations. Hardware agnostic software is the term introduced in reconfigurable computing [1] for software that the same logical and timing behaviour when executed on different hardware platforms. In Cyber-Physical Systems (CPS) based on the IoT architecture, software applications are executed on networked devices, whose performance may change during operations due to, for example, battery discharge, wireless communication distortions, and changes of physical location or environment parameters. Therefore, we call this property Cyber-Physical Agnosticism, or CPA, instead of hardware agnosticism. In particular, we will consider (a bit futuristic) IoT architecture in which all distributed nodes have tightly synchronised clocks and asynchronously exchange messages that are tagged with timestamps. This model of computation has been recently investigated in the PTIDES/Ptolemy II framework [2] for CPS modelling and our aim is to examine it in the distributed automation system context.

In a recent work [3] we have prototyped a fully distributed approach to automate a manipulator with an extreme case of distributed automation architecture as shown in Fig. 1, where each sensor and actuator is a kind of intelligent device with embedded microcontroller and wireless connectivity. This can be considered as IoT inside a single machine, where the "things" are sensors and actuators. It has been demonstrated that using the distributed function block (FB) language of IEC 61499 standard architecture [4, 5] it is possible to port the code from being executed in one device to the network of six devices without any modifications and the system behaviour remains the same. This is an important step towards reducing the development efforts in flexible and reconfigurable systems, where changes in hardware specifications and network configurations occur on a regular basis. However, the quality of control is certainly changing dependent on the performance of control nodes and bandwidth and load of the network. Achieving a solution also agnostic to those factors would mean that not only the logic of the behaviour will remain the same, but also the quality of control will not depend (to the limits determined by the physical system properties) on the hardware and network parameters.

The rest of this paper is structured in the following way. Section II motivates the goal of achieving CPA by considering an example of a simple automation system fully based on wireless communication. Section III presents a brief survey of related works in control, computing, and automation. Then, Section IV elaborates the modifications to existing IEC 61499 FB syntax in order to use time stamped events. Section V elaborates the rules for manipulating timestamps. The concept of CPA is experimentally validated in Section VI. The paper is finally concluded in Section VII.

II. DISTRIBUTION TRANSPARENCY AND IEC 61499

The IEC 61499 standard introduces a system-level reference software architecture for distributed automation

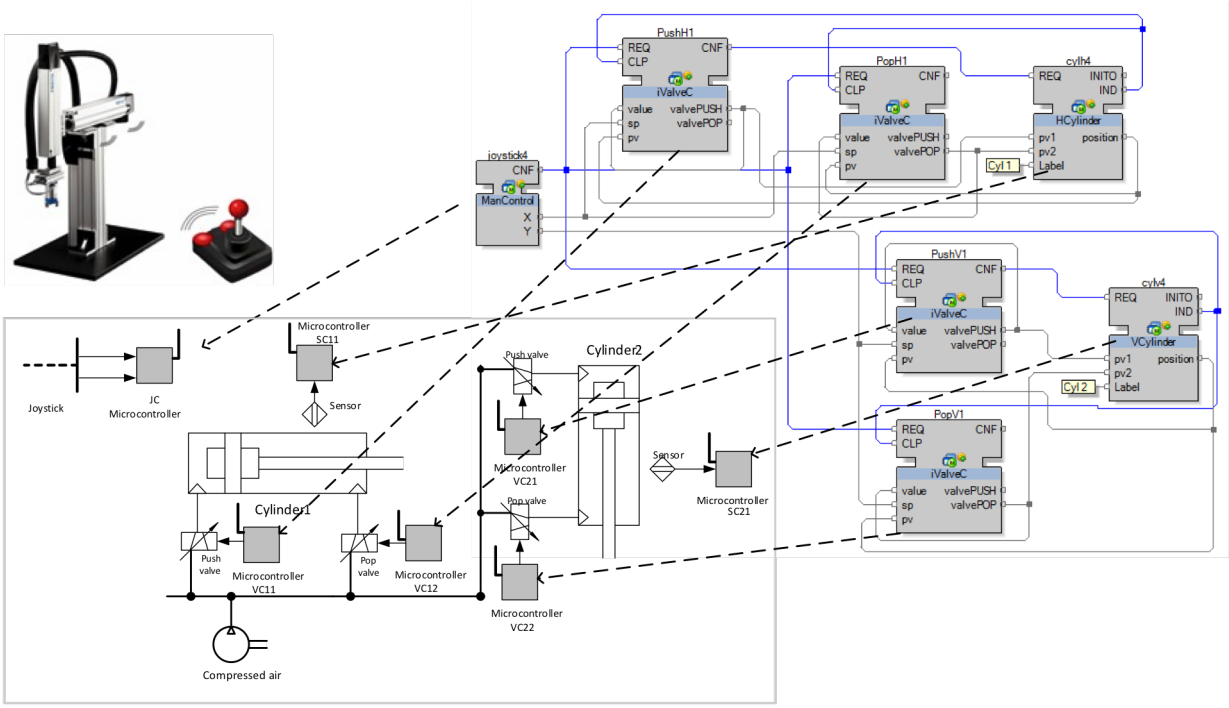

Fig. 1. Decentralized control logic of a “fully wireless” IoT based manipulator [3].

systems. The most essential claim of the IEC 61499 architecture is about minimizing developers’ efforts in deploying automation software to different distributed architectures of hardware. The event-driven activation mechanism of FBs helps to preserve causality in distributed systems, which is an important enabler of this distributed deployment transparency. Fig 1 illustrates this idea in a nutshell using the pick and place manipulator from [3] with slightly modified control that is based on continuous control of the cylinder positions according to position sensors rather than the mere end position observation in [3]. The manipulator is operated by a joystick device that determines the desired target coordinates of the gripper. The control logic is highly modular following the mechatronic modularity of the manipulator, which consists of two identical pneumatic cylinders. Each of the cylinders is controlled by two proportional pneumatic valves: pop and push. The valves are intelligent in the sense that they have their own controllers that decide how much they should be open in order to reach the desired position. Thus, each valve and position sensor form a closed control loop. Besides there is setpoint feeding from the joystick to the four valve controllers.

For simplicity, we will be further considering an example that is very close to just one out of those 4 closed loops, as shown in Fig 2 (a). Here the FB application implements the control of a “pneumatic cylinder with a retracting spring”, using a position sensor and proportional valve actuator. It is supposed that the control goal is to track a certain desired position of the cylinder as provided by the setpoint input SP of the FB "Error". The sensor reading is obtained and initially processed in the FB "Sensor". Then it is passed to the FB "Error" by emitting event. The FB "Error" calculates the difference between SP and S and passes it to the FB "Controller" that recalculates the control signal and passes it to the "Actuator" FB.

If the controller implements a continuous control algorithm, such as proportional-integral-derivative (PID) control, it relies on the periodic sampling of sensor readings. The maximum sampling period is selected based on, for example, the Nyquist frequency, while its minimum value is limited by the computational delay of the controller hardware, in principle the smaller, the better. In any case, the fixed sampling rate value DT is important to know in order to recalculate the value of process variables in the controller. In order to achieve the fixed sampling rate in our example, the FB "Sensor" needs to be activated periodically by the FB “Clock” with period RT=DT.

In the most traditional central control hardware architecture case, the application can be deployed to a single microcontroller connected to both sensor and actuator (Fig 2 (b)). In this case, it is easier to estimate the worst case computational delay. The sampling rate parameter DT can be chosen to be greater than the estimated delay. Alternatively, the very same application can be deployed to a distributed network of two microcontrollers, one residing closer to the position sensor and the other to the proportional valve actuator, with wired or wireless communication between them, as indicated in Fig 2(c). The sensor-attached microcontroller may be sending the updated position readings only in case its change is significant. Such a distributed architecture has many benefits in automation systems, such as reducing wiring and improving flexibility of

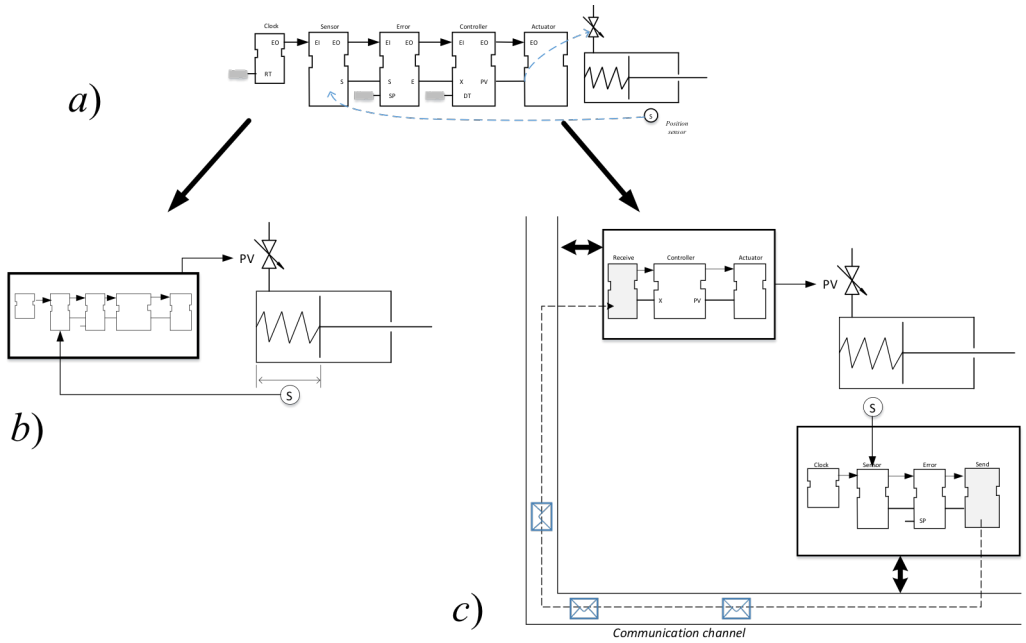

Fig. 2. Cylinder control function block application: (a) centralized deployment and (b) distributed deployments of the function block control application.

automation systems. Therefore, industrial network technologies have been proliferating in the last couple of decades. However, in order to ensure the correct causality in execution of this distributed software, devices need to be synchronised between each other and with the communication channel. The event-driven execution mechanism of FBs maps well to the message based network protocols.

Hence, sequence of FBs invocations will remain the same after distribution without additional efforts. However, performance will certainly be affected as networks introduce communication delays, which may be variable due to jitter. In case of a distributed architecture, such as the one in Fig 2 (b), sampled data are transmitted via network in packets, and times of their arrivals to the controller are variable due to the network jitter. There are several ways to deal with this situation. The most obvious one is to estimate the upper bound value of the communication delay and select the sampling interval to be greater than this upper bound. In case if the next sensor reading arrives sooner than the upper bound, the activation of the controller FB is delayed. This approach, however, has obvious drawbacks, as it will affect the quality of control responses to the changing values of sensors.

Real deployment transparency of distributed control applications could be achieved if the physical system's behaviour would not substantially change after changing the topology of the target system. In this paper we investigate if the knowledge of exact time taken by each packet with sensor data can help in improving control quality when a classic control algorithm is applied in the networked architecture. For that, we will enhance the event mechanism of FBs with timestamps. Knowing the time of packet's sending, it will be possible to calculate the transfer time at the destination. The control application with "classic" PID control not enhanced with any advanced techniques will be taken for comparison.

III. Related Developments

Traditionally, the problems related to CPA of automation software have been addressed separately in computer science and control science, but rarely in a synergetic conjunction. This is insufficient in a view of IoT concept becoming a major driver for many industrial applications. In manufacturing, it leads to flattening of the control pyramid thus increasing flexibility and enabling unprecedented level of production flexibility and adaptability. This makes it feasible to produce products in smaller amounts with shorter time to markets and higher economic efficiency. For example, according to the German development agenda Industrie 4.0 [6], the main driving forces of the new industrial revolution are IoT and CPS. In the manufacturing environment, CPS comprise smart machines, storage systems and production facilities capable of autonomous and collaborative actions. Another promising application area for IoT is SmartGrid: energy generation, distribution, and consumption infrastructure based on the wide use of renewables and ICT intertwined with classic power system control. The growing interest in using distributed hardware architectures in automation has led to the creation of the IEC 61499 standard, which presents a component-based software reference architecture. This architecture uses the concept of event-driven invocation of components that maps well to the message passing mechanism of network communication. This helps to achieve a certain degree of transparency in mapping software to hardware.

The control systems theory has reacted to the networking revolution first by the avalanche of works on networked control, and later, more specifically, with theories of event-

based control, for example developed by Aström [7] and Årzen [8]. The networked control methods are based on robust control algorithms with the purpose of reducing impact of the variability of sampling period on the quality of control. Such advanced control methods help to achieve better control quality in the case of uncertain sampling rates. Such methods, however, are more complex in implementation and more resource hungry. One possible way of handling uncertainty is using a model-based prediction of system behaviour. The obvious limitation of any such prediction method is that the real system's behaviour may be hard to predict.

The efficient use of event-driven methods by automation system developers is hampered by the lack of support in programming languages, commercial control devices, and design tools. Computer scientists have been more concerned with investigating and guaranteeing timing properties of software executed on a single computer node or on a network thereof. This is an important prerequisite for, but does not explicitly address, meeting requirements for the performance of physical systems under control. Synergy of these aspects is in the core of research on CPS. The term CPS has been coined to emphasise the cross-disciplinary nature of computer interactions with the physical world, in which both parts influence each other and efficient design is impossible staying within classical disciplines of computer science, or control theory, or communications. Modelling CPS with the purpose of validating their behaviour is an intrinsically hard problem due to the need of taking into account computational correctness aspects, control correctness aspects, dynamic behaviour of the physical environment along with communication, and especially the cross-dependencies between these factors.

A notable development in CPS is the Ptolemy II/PTIDES approach [9] that has been gaining momentum in the general embedded systems arena. The PTIDES programming model is designed as a coordination language for CPS, providing robust distributed real-time software models, whose behaviour is independent of hardware deployment. PTIDES also provides deterministic execution semantics, in which variability in clock synchronisation and network latencies can be eliminated from the physical plant model of any CPS. The PTIDES semantics is based on a tagged-signal model, which provides deterministic temporal semantics. Interactions of control programs and physical processes are represented in the same system model without specifying hardware details. As real-time constraints are met at sensors, actuators, and network interfaces, modifications in hardware details or small variations in program execution time will not affect the system behaviours. This characteristic is extremely important for design and analysis of CPS. The use of the PTIDES semantics in the implementation language of FBs could achieve the desired CPA in the example presented in Fig 1 and in other similar IoT architectures. The basic assumption for applying PTIDES, however, is quite strong: tight synchronisation of clocks in all distributed cyber-nodes. However, there has been substantial progress to this end in the industrial domain, for example, associated with adoption of the IEEE 1588 standard [10] that provides clock synchronisation technology for mass applications. There are plenty of cost effective hardware devices on the market, which enable clock synchronisation to the microsecond level.

As far as the design of distributed automation systems with essentially distributed logic is concerned, the FB architecture introduced in the IEC 61499 remains the most credible system level reference architecture with sufficient industrial adoption prospects and potential of defining appropriate formal semantics [11]. Recognizing this technological trend, there has been growing interest in industrial automation domain in efficient applications of this technology to derive tangible benefits in design of dependable distributed systems. For example, several developments have been conducted towards proving feasibility of automation systems based on peer-to-peer communicating actuators [12, 13]. Multi-agent architectures have been applied at the application level to achieve autonomy and self-configuration of systems, in particular in SmartGrid automation [14-17]. The service-oriented architecture (SOA) has been extended and bridged with current automation practices [18-20] to enable interoperability and simplify design, and the cyber-physical component architecture (CPCA) was introduced in [21] to facilitate the design and validation of IoT-applications. The IEC 61499 reference component architecture for distributed automation systems was attempted to be applied as a "glue" between these two and the legacy automation approaches, and the follow-up works, e.g. [22], suggested how software systems made of such components can be made self-configurable similar to eco-systems of living organisms. This raises concerns as to what extent system-level software models can define behaviour of such systems in a manner that is independent of the underlying computer hardware.

In this paper we enhance the IEC 61499 standard with event time-stamping mechanism, which simplifies the achievement of cyber-physical invariance of system-level models of distributed systems.

IV. Modified Syntax of Function Blocks

Some minor additions to the syntax of FBs are required in order to use time-stamped events. It is proposed to treat event variables as complex data types.

```
EVENT= STRUCT
    value     : BOOL;
    ts_last   : TIME;
    ts_born   : TIME;
    SID       : FBI_ID;
END_STRUCT;
```

The logical value of an event input of an FB is referred to as *value* of type BOOL. There are two timestamps associated with an event, each being a variable of the data type TIME. The *ts_born* is assigned to the current system clock when the event is created at the event-source FB. Then this value is copied along the event chain. The *ts_last* contains the timestamp of the last FB when the event is emitted. SID is

the identifier of the FB instance where the event was created. Other required syntax extensions are as follows.

The proposed rule for timestamp assignment requires determination of what is invocation event for the current run of FB. For that, we introduce the reference to event by which the current run of a basic FB was invoked.

1) INVOKEDBY : EVENT;

The assignment of timestamps will require access to system clocks of devices. We assume it is provided by the variable:

2) systemclock: TIME;

V. Rules for Timestamp Manipulation

A. Timestamp Creation and Modification

As depicted in Fig. 1, if FB1 is invoked by event FBSRC.EO and emits event FB1.EO, the event properties can propagate through the chain as follows:

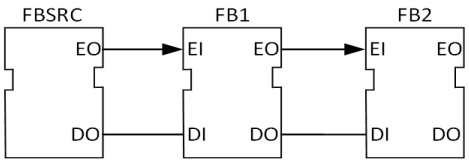

Fig. 1. Simple event chain of function blocks.

1) // At source event, timestamps are initialised with the current reading of system clocks.

FBSRC.EO.ts_born := FBSRC.systemclock;

FBSRC.EO.ts_last := FBSRC.systemclock;

2) // Invocation of FB1 by event EI

// Timestamp is copied from FBSRC.EO;

FB1.EI1 := FBSRC.EO;

FB1.INVOKEDBY:=FB1.EI1;

...

// Before output event EO is emitted by FB1

// Its birth time is assigned from that of the input event.

// The last timestamp is assigned with the current reading of system clocks.

FB1.EO.ts_born := FB1.INVOKEDBY.ts_born;

FB1.EO.ts_last := FB1.systemclock;

// Activation of FB2 is similar to that of FB1.

3) FB2.EI:=FB1.EO;

B. Timestamps in Communication FBs

The Send/Rcv pair of communication function blocks implements event and data transfer between devices (in real implementations these can be PUBLISH/SUBSCRIBE and CLIENT/SERVER FBs). Event transfer from Send to Rcv happens as if they were connected by direct event link as shown in Fig. 2. As a result:

Rcv.EO.ts_born = Clock.EO.ts_born; //and

Rcv.EO.ts_last = current D2 system clock at the moment of packet arrival.

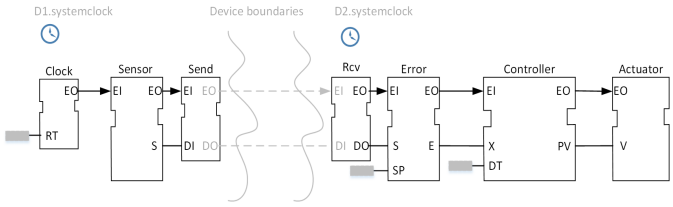

Fig. 2. Event chain of function blocks distributed across two devices.

C. Usage of Timestamps

To determine sampling duration in algorithm invoked by event REQ:

timegap:= REQ.ts_born - systemclock;

It is assumed that the event's source is the same FB where the sensor value is sampled. In order to measure the duration of event propagation from any point, the unique FB instance identifier SID is used, and an operation that allows overwriting of the ts_born parameter in FB algorithms is applied.

VI. Experimental Validation of CPA

Let us consider what would be the performance of the distributed system in case if the actual communication delay was known for each sensor value sampled and compare it with the case when the delay is uncertain but a fixed DT value is used in the controller. These two cases will be compared with the ideal case of fixed time delay and fixed DT parameter of the controller. In order to make the comparison, we created models in the Ptolemy II environment corresponding to all three cases as shown in Fig. 3.

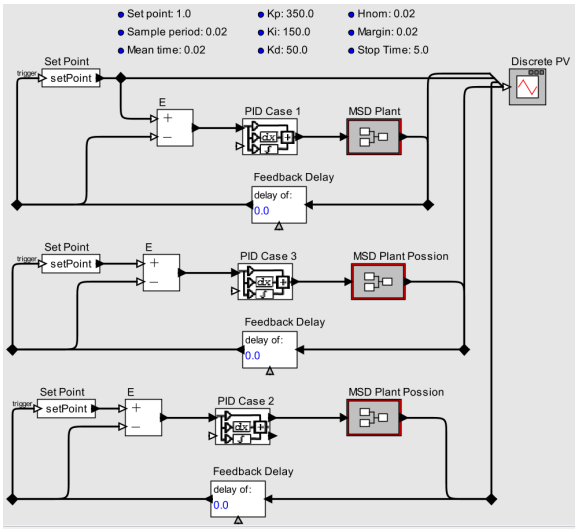

Fig. 3. PtolemyII model of three communication delay cases for comparison.

Ptolemy II enables heterogeneous semantics of modelling and simulation. The Discrete Event execution model of Ptolemy II is based on event-driven invocation of blocks and supports time stamping of events. Therefore, it is possible to calculate the time interval between two consecutive events within a module. The results of the comparison are presented in Fig. 4 below.

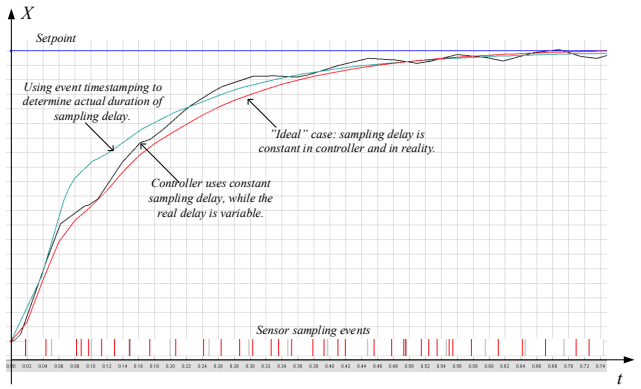

Fig. 4. Plot of process variables in three comparison cases.

The ideal reference case is plotted in red. It exhibits smooth tracking of the setpoint. The impact of running the same application in presence of variable communication delay is the black plot, which shows good deal of oscillations This is undesirable impact on the quality of control and physical system properties. The green plot represents behaviour of the physical system after changing the model of computation to the time-stamped events. The properties of tracking are certainly worse than the ideal case but free of oscillations, which is a great improvement in the quality of control. This result inspires optimism and justifies the effort on implementing the proposed model of computation in IEC 61499.

VII. Conclusion

In this paper we estimated the impact of introducing time-stamped event mechanism of IEC 61499 function blocks on achieving cyber-physical agnosticism of distributed automation software. An initial experimentation was conducted using the Ptolemy II simulation environment without implementation of dedicated new compiler or runtime of IEC 61499. The results demonstrated tangible benefits of enhancing IEC 61499 with the PTIDES model of computation for maintaining cyber-physical system properties in case of hardware/software reconfigurations.

References

[1] M. Vuletid, L. Pozzi, and P. Ienne, "Seamless Hardware-Software Integration in Reconfigurable Computing Systems," *IEEE Design & Test of Computers,* vol. 22(2), pp. 102-113, 2005.

[2] Z. Jia, S. Matic, E. A. Lee, T. H. Feng, and P. Derler, "Execution Strategies for PTIDES, a Programming Model for Distributed Embedded Systems," in *15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2009)*, San Francisco, CA, US, 2009, pp. 77-86.

[3] D. Kleyko, E. Osipov, S. Patil, V. Vyatkin, and Z. Pang, "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.

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

[5] 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.

[6] Industrie 4.0 Working Group. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0* [Online]. Available: http://www.plattform-i40.de/finalreport2013

[7] K. J. Åström, "Event Based Control," in *Analysis and Design of Nonlinear Control Systems*, A. Astolfi and L. Marconi, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 127-147.

[8] K.-E. Årzén, "A Simple Event-based PID Controller," in *14th IFAC World Congress*, Bejing, China, 1999, pp. 423-428.

[9] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and Z. Jia, "Distributed Real-Time Software for Cyber-Physical Systems," *Proceedings of the IEEE,* vol. 100(1), pp. 45-59, 2012.

[10] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008.

[11] V. Vyatkin, "Software Engineering in Industrial Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics,* vol. 9(3), pp. 1234-1249, 2013.

[12] M. Sorouri, S. Patil, and V. Vyatkin, "Distributed Control Patterns for Intelligent Mechatronic Systems," in *10th IEEE International Conference on Industrial Informatics (INDIN 2012)*, Beijing, China, 2012, pp. 259-264.

[13] J. Yan and V. Vyatkin, "Distributed Software Architecture Enabling Peer-to-Peer Communicating Controllers," *IEEE Transactions on Industrial Informatics,* vol. 9(4), pp. 2200-2209, 2013.

[14] P. Vrba, V. Marik, P. Siano, P. Leitao, G. Zhabelova, V. Vyatkin, *et al.*, "A Review of Agent and Service-Oriented Concepts Applied to Intelligent Energy Systems," *IEEE Transactions on Industrial Informatics,* vol. 10(3), pp. 1890-1903, 2014.

[15] G. Zhabelova, V. Vyatkin, and V. N. Dubinin, "Toward Industrially Usable Agent Technology for Smart Grid Automation," *IEEE Transactions on Industrial Electronics,* vol. 62(4), pp. 2629-2641, 2015.

[16] M. Degefa, A. Alahäivälä, O. Kilkki, I. Seilonen, and M. Lehtonen, "MAS-based Active Network Model for State Estimation and Beyond," *IEEE Transactions on Industrial Electronics*, accepted, 2015.

[17] S. Patil, V. Vyatkin, and B. McMillin, "Implementation of FREEDM Smart Grid distributed load balancing using IEC 61499 function blocks," in *39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)*, Vienna, Austria, 2013, pp. 8154-8159.

[18] W. Dai, V. Vyatkin, J. H. Christensen, and V. Dubinin, "Function Block Implementation of Service Oriented Architecture," in *12th IEEE International Conference on Industrial Informatics (INDIN 2014)*, Porto Alegre, Brazil, 2014, pp. 112-117.

[19] 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.

[20] G. Zhabelova, C.-W. Yang, S. Patil, C. Pang, J. Yan, and V. Vyatkin, "Cyber-Physical Components for Heterogeneous Modeling, Validation and Implementation of Smart Grid Intelligence," in *12th IEEE Conference on Industrial Informatics (INDIN 2014)*, Porto Alegre, Brazil, 2014, pp. 411-417.

[21] J. Yan, C. Pang, C.-W. Yang, and V. Vyatkin, "Adaptable Software Components: Towards Digital Ecosystems and Software Evolution in the Industrial Automation Domain," in *40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014)*, Dallas, TX, US, 2014, pp. 2512-2518.

[22] A. Choudhari, H. Ramaprasad, T. Paul, J. W. Kimball, M. Zawodniok, B. McMillin, *et al.*, "Stability of a Cyber-physical Smart Grid System Using Cooperating Invariants," in *37th IEEE Annual Computer Software and Applications Conference (COMPSAC 2013)*, Kyoto, Japan, 2013, pp. 760-769.