Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 42 (2020) 84–90

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019)

RMAS architecture for industrial agents in IEC 61499

Andrea Bonci[a], Sauro Longhi[a], Emanuele Lorenzoni[a], Massimiliano Pirani[a,*]

[a]*Dept. of Information Engineering, Polytechnic University of Marche, Brecce Bianche 12, Ancona 60131, Italy*

* Corresponding author. Tel.: +39-0712204666 ; fax: +39-0712204224. *E-mail address:* m.pirani@univpm.it

Abstract

This paper analyses an architecture, recently proposed and under development, that aims to render the design and the development of multi-agent systems viable for the vision and the needs of the industry in the cyber-physical systems era. Prominent standardization work is currently focusing on the IEC 61499 standard as a most promising basis for the integration of industrial agents in the real field. In this context, authors propose RMAS (Relational Multi-Agent System) as a possible useful aid and means for the researches and the practices in the field. On this purpose, the compliances and the possible overlaps between the major features and capabilities of RMAS and IEC 61499 are assessed and discussed. This work aims to constitute a position paper for the future possible developments around IEC 61499.

Keywords: multi-agent systems; cyber-physical systems; relational model; actor model; IEC 61499

1. Introduction

The problems of industrial production and related business management solutions currently span a wide context. This context is constituted of the several technological endeavors and visions such as Industry 4.0, industrial cyber-physical systems (ICPS), and industrial IoT (IIoT) where cyber-physical systems (CPS) monitor and orchestrate physical processes [1]. In this scenario, multiagent systems (MAS) are a suitable technology to develop modular, flexible, robust, and adaptive systems based on the decentralization of functions, in order to solve interoperability problems in a lean and clever way [2,3,4,5]. A CPS is designed to promote the symbiosis and fusion between a physical element, its controller, and its abstract or logical representation existence. Industrial agent systems can be seen as a restricted case of CPSs [6].

Industrial agents have their specific application domains in industrial environments. This forces them to adhere to industrial requirements as the specific hardware integration, reliability, fault-tolerance, scalability, industrial standard compliance, quality assurance, resilience, manageability, and maintainability [4,7]. These requirements have originated great efforts towards standardization activities. Standardization efforts are in search of consistent performance guarantees, involving digitalization, in relation to the multidimensional life cycle aspects that model the industrial business of the future [1]. A prominent contribution in this sense has been provided by the International Electrotechnical Commission (IEC) 61499 standard [8]. This standard tries to determine a strong and general common ground for the efficiency of distributed realizations of industrial automation. In addition, IEC 61499 is designed to keep the door open to the easy integration of the developments that are likely to occur at the higher tiers of industrial production business technology, seen as a whole.

The interest in the IEC 61499 is testified from the developments and the efforts of two major task forces. The first is the working group P2660.1 of the IEEE SA on "Recommended Practice for Industrial Agents: Integration of Software Agents and Low Level Automation Functions" [2]. The second is the IEEE IES Technical Committee on Industrial Agents (TCIA) [3].

Since its origins, the design of IEC 61499 was intertwined with intelligence of the agents in the context of the framework of holonic management systems [6,9]. The application of function blocks for the building of holonic systems had been envisaged since their inception [10,11,12]. As reported in [13], since the beginning of the works on the IEC 61499 standard, many design methods, architectures, computing platforms, networking technologies, and programming languages have been proposed to help improve automation systems. They used a multiagent approach, some of which incorporating the use of IEC 61499 Function Blocks [14,15].

In close link and adherence with the aforementioned scenario, the work here presented creates a bridge between the IEC 61499 and an already existing MAS research framework. In this framework, the authors proposed new holonic interpretations of intelligent automation and control of systems of systems, specially developed for industrial applications of the future, and backed by the CPS playground and motivations [16,17,18,19,20]. The computational and technical aspects in the realization of suitable holonic systems has led authors to the proposition of an architecture called RMAS (Relational MAS) [21]. RMAS has been proposed as a viable means for the implementation of said intelligent control vision.

In this article, we discuss to what extent the RMAS architecture is compliant with the vision and the works of the IEC 61499. In Fig. 1 a Venn diagram is used to render the motivations of the framework about the relationships between RMAS and IEC 61499 that starts with this position paper. The figure shows and possibly denotes that the goal is to demonstrate that RMAS constitutes a possible complete implementation of IEC 61499. Moreover, the figure expresses that, in general, RMAS can be also a partial solution to the complex problem of CPS design and realization. A rather strong (and maybe a little controversial) position in the same figure is that IEC 61499 spans completely all the Industry 4.0. This has been voluntarily stressed (and probably exaggerated) here to remark optimistically the potentialities of IEC 61499 while acknowledging its important role.

Since IEC 61499 is, by design, already expressive enough to integrate heterogeneous types of computing and their implementations, the claiming that RMAS is compliant with the standards is some sort of truism. Nonetheless, it would be

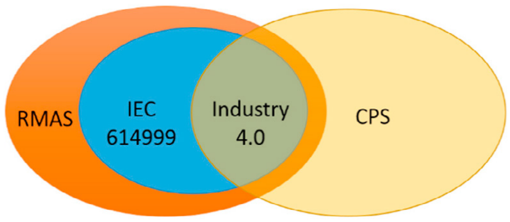

Fig. 1. Venn diagram of RMAS, IEC 61499, and CPS intersections.

apparent, in the following, how RMAS can be rather tightly and straightforwardly mapped to the models and the design paradigms of the standard. Indeed, this work aims to strengthen the connection of the author's ongoing work with the state-of-the-art developments in IEC 61499. At the same time, it aims to call the attention of the industrial community on a possibly promising approach that requires further collaboration and effort to be improved and developed.

In section 2, a brief recall of the main directives and features of the IEC 61499 standard is made. In section 3, the RMAS architecture is summarily introduced in order to proceed to the section 4, in which it is discussed in what sense and to what extent the RMAS is compliant with the models of the IEC 61499 standard, in order to represent an advisable means for the factual introduction of MAS in industrial automation. Section 5 is dedicated to conclusion and to some perspectives on future work.

2. The IEC 61499 basics

The IEC 61499 standard aims to provide a guideline for the design of distributed, modular and flexible industrial process control. More specifically, IEC 61499 addresses [22]:

- *Portability*, understood as easy exchange of control applications and library elements between different software tools;
- *Configurability*, as the ability to configure control devices by multiple software environments;
- *Interoperability*, with targeting the standardized information and data exchange between control devices.

This is achieved by the provision of several reference models in the IEC 61499 specification, which was released in its first edition in 2005 and in its second edition in 2012 [22].

These specifications let the former IEC 61131 standard be endowed with direct support for distribution of computing and flexible configuration and organization. Another big difference, between the former IEC 61131 PLC-based approach and the IEC 61499, is a new definition for its execution model.

In the usual PLC-based context, the execution of the function blocks (FB) is cyclic. In the IEC 61499, major part of the execution relies on events. IEC 61499 is event-based, by means of the definition of suitable event interfaces for the FBs. This new standard supports the asynchronous execution of function blocks and networks of function blocks through the usage of suitable control of special events. Nonetheless, the synchronous concept is still supported [7,22,23].

In the following, the main components of the general model suggested by IEC 61499 are recalled. In order to easily identify these components, the reader can refer to the scheme of Fig. 2, adapted from [7].

The model follows a granular approach and can be split into the *Application model*, the *System model*, the *Device model*, the *Resource model*, the *Function block model* and the *Device management model* [7,23].

The *Application model* builds on a framework based on function blocks that perform control functionalities detached from any specific implementation issue. In Fig. 2, the collection of FBs constituting the control application *App_1* is shown. Such an application can be shared between two (or more in general) devices by means of the *resources* Res_{1m} and Res_{n1}. This property achieves the distribution of an application across different *devices*. Resources are associated to devices.

The basic function block (BFB) can be an atomic entity derived from the PLC languages of FB of IEC 61131-3 [24] but can also be extended in order to fulfil the new requirements. Actually, unlike the IEC 61131 standard, IEC 61499 does neither define any implicit cycle-based execution behavior for the FB, neither a time base. On the contrary, the event-based execution model is defined, which relies on an interface characterized by two semantically different types of inputs and outputs [25], namely events and data. With these provisions, asynchronous execution of FBs is supported.

The model of the FB encapsulates data and algorithms whose execution is handled by a state machine defined by the Execution State Chart (ECC). Once an algorithm terminates its execution, output events are triggered, and the associated data are available to the downstream FB. In addition, the Composite FB (CFB) is defined, which can be used for functional aggregation of BFBs and the Service Interface FBs (SIFB). SIFB are interfaces to communication services or to non-compliant IEC 61499 devices.

The *System model* defines at the top level the architecture by means of which a set of devices can cooperate to realize one or more distributed applications. It defines the *devices* and their *inter-device communication links*, as shown in Fig. 2.

A *device* represents a physical entity able to execute control functions (e.g., embedded controller, PLC) containing one or more independent functional units called *resources* [22]. It is within the *system model* abstraction that the *device* model can represent the structure of any physical control device, which possibly lodges one or more *resources*. The Fig. 2 shows two devices which host *m resources* and *q resources* respectively.

By definition, a *resource* is the execution environment for a network of function blocks. More rigorously, portions of the FB network, constituting the whole application, are allocated on suitable *resources*. For example, Fig. 2 shows that the application *App_1* is realized as an FB network. This network is in turn split across two resources (Res_{1m} and Res_{n1}) on two different devices.

The *resource* is responsible of several tasks:

- provides the correct scheduling of the algorithms embedded on the FBs and triggered by input events;
- retains the values of the FB's internal variables;
- propagates events and data among FBs within the resource and provides communication services to remote resources through the *communication interface*.
- maps the read/write requests to and from the FB to local device I/Os (*physical process*) onto *the process interface* (Fig. 2).

Within the FBs' framework, SIFBs are a special kind of FB used as links between FBs and the *Communication interface* or the *Process interface*. These links are defined by means of two different high-level communication patterns, namely the publish/subscribe and the client/server models [7].

The *Device management model* is defined by the standard in order to provide the automation system with adaptive

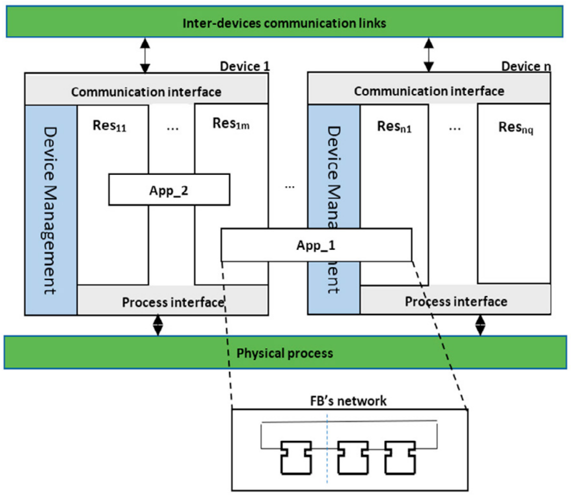

Fig. 2. The IEC 61499 main model.

reconfigurability features. It consists of a *device management application* mapped on a special type of resource, named the *Device Management* (see Fig. 2) resource, devoted to the reconfiguration at run-time of other resources and of the network of FBs [7,22,23]. A high-level application can use a management application with privileges in order to create and cancel FBs on a local and on remote resources through the communication interface. It can be considered a meta-application. The IEC 61499 device management, with its management commands, provides an open interface to other tools or agents. This mechanism should be used to reconfigure the control applications during execution. This is also the very feature of the standard that leaves the door open to a future of autonomic computing for the industrial context.

3. A brief recall of major RMAS architecture concepts

The RMAS (relational multi-agent system) architecture was firstly presented from the authors in [21]. RMAS has been conceived as a natural evolution of a previous framework based on database-centric, event-based, and publish-subscribe techniques [16]. It aimed to render the design and deployment of holonic distributed elements and components in a way that meets industrial automation requirements.

By leveraging the findings about the capabilities of the relational model, as a basis for logic programming in CPS [17], the RMAS established a subsequent consistent architecture for the dynamic configuration, programming, simulation, maintenance, and deployment of MAS.

RMAS has been specially designed for the embodiment of reasoning and control in CPS, by introducing an effective combination of host languages (imperative, object-oriented, functional or other computing languages) and database manipulation languages (mostly declarative languages as the SQL).

The core of the architecture starts with the definition of a germinal RMAS unit, which can be considered as the container of the genotype of RMAS, as reported in Fig. 3. This unit has initially only minimal provisions for being contacted, configured and programmed, namely:

- an input stage, typically implemented by means of the hosts language (for example, code for Internet socket connections and for the interpretation of the message);
- the *v_bootstrap*, database variable;
- the trigger *AFTER_INSERT_ON_v_bootstrap*, which executes host or database language code after a new value is inserted into the *v_bootstrap* relational variable (table).

This specific bootstrap mechanism is made essentially of a database variable and a trigger (*v_bootstrap* and *AFTER_INSERT_ON_v_bootstrap* in Fig. 2) that is able to trigger some further relational processing. While, for more details, we refer to [21], we have to remark here that the creation, deletion or modification of the database catalog or schema is considered part of the relational processing (which is not the case in the normal understanding of commercial relational implementations).

When a new value for the variable *v_bootstrap* arrives, it is inserted into the variable table. At this point, the trigger can interpret and execute any logic contained in the value of the variable expressed in the host or the database language. This allows embedding in the new value of the database variable a whole set of transactional actions to be committed by the triggered relational processing. In this way, a value of the variable transmitted by an INSERT query operation as its carrier, can perform typical manipulations (e.g. creating or deleting other database objects or tables, inserting or updating contents) or, through the power of the host language, can perform also meta operations affecting the whole database (e.g., operations at the schema level) or other parts of the software or the hardware.

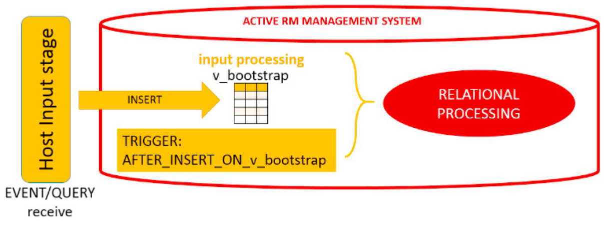

Fig. 3. Genotype RMAS unit.

Through this mechanism, the internal structures and programs of a RMAS unit can be "inflated" by a parent unit or by another kind of program or system. This process lets the RMAS unit reach a complete and full-fledged structure, which features all the capabilities to express any Turing-complete processing, made of a combination of database manipulation language and one or more host programming languages. The generic structure of such a full-fledged RMAS unit is shown in Fig. 4.

In the next section, it will follow a discussion on how such an architecture can provide means compliant to the IEC 61499 standard. A peculiarity of RMAS is indeed its particular choice on the relational structure that complies with many of the publish/subscribe schemes of networking, the *actor model* of computation, and event-driven processing, which are some of the milestones in the IEC 61499 standard.

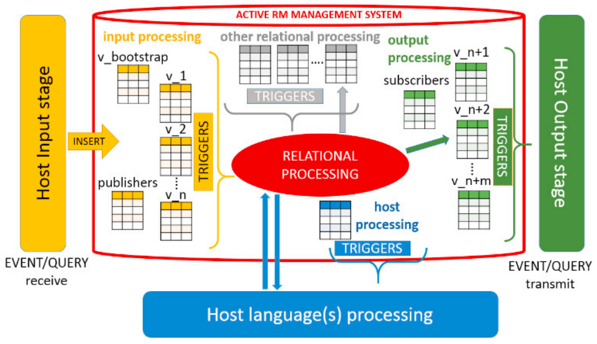

Fig. 4. Full Fledged RMAS unit.

4. Relationships between RMAS and IEC 61499

As envisioned in previous sections, the full-fledged RMAS unit depicted in Fig. 4 is proposed as a candidate architecture for the realization of (at least some) of the models and of the structures of the IEC 61499. To achieve this result, it is necessary to craft some suitable mappings between the functionalities and the characteristics of the standard and the provisions of the RMAS architecture.

Indeed, it can be easily maintained in the following that at least four of the basic models of the IEC standard recalled in section 2 are prone to some straightforward isomorphism with the RMAS components. In particular, the focus of the mappings proposed henceforth will be applied to the *resource model* and the *function-block model*. Although possible, the mapping with the application model and the management model will not be discussed in detail in this paper, but left for the future extensions of this work, deserving them a deeper inspection, experimentation, and more space.

4.1. Mapping RMAS with the IEC 61499 resource model

A first starting point comes from the definition of *resource* in the standard. The *resource* can be considered a suitable structure for the mapping with the essential parts of a RMAS unit.

In Fig. 5, the authors propose a possible *resource* configuration in which some of the basic RMAS components, as depicted in Fig. 4, may be mapped. Note that a corresponding color pattern has been used for the components of RMAS and the resource in order to immediately identify them projected into the new structures of Fig. 5 from Fig. 4.

The role envisioned in the standard's models for a *resource* is defined in order to enable the communications with the physical process and, at the same time, with the network of resources. This happens by the appropriate instantiation of SIFBs (Service Interface Function Blocks) for the communications with the process and the network. The algorithmic part of the resource can in turn be a composition of function blocks [23].

In Fig. 5, the algorithmic part of the function block FB1 has been used to encapsulate the relational processing of the RMAS unit. We note that this FB1 part, in general, can be easily split or distributed to a set of interconnected FBs – but it is not shown here to keep the figure more readable. In FB1, the tables and database objects, with associated triggers, can be considered equivalent to some algorithmic and logic programming stage. This property has been demonstrated and discussed in [17]. The FB1 executes this part as a query arrive in form of event (EI1 or EI2 in Fig. 5) carrying some data (the query text, as QI1 or QI2 in Fig. 5). In this case, the IEC 61499 FB model results fundamental, as it perfectly complies with the event-driven paradigm for the exchanging of data to and from the RMAS unit. It is obtained simply with the association of an event to the execution of an update query in the relational processing section of the RMAS (e.g. INSERT, UPDATE, CREATE, and DELETE in SQL-like manipulation language).

After the fetching of the first event, other internal events can be issued by a suitable cascade of triggers, until eventually FB1 generates an output event at the end of the relational processing (EO1 or EO2 and the associated data QO1 or QO2, in Fig. 5). At this point, output data and events can be available for other FBs owned by the *resource*. Other FBs can be of different natures and types, being them other generic FBs or parts of other RMASs. In Fig. 5, a most simple and minimal configuration is shown, having FB1 flanked by SIFB1, SIFB2, and SIFB3. With these four elements, a complete resource functionality can be obtained.

The information elaborated by FB1 is distributed to remote subscribers by means of the Service Interface Function Block SIFB2 (colored in green) that embodies the *host output stage* of the RMAS. On the other hand, FB1 is able to receive events and data queries from remote publishers through the (yellow colored) SIFB1 which embodies the RMAS *host input stage.*

The service interface SIFB3 (colored in blue) is used to link the *resource*/RMAS unit with a host computing process that, in turn, usually constitutes the set of calls to procedures or functions connected to transduction of sensors and effectors in contact with the physical processes.

In this regard, and as a particular case, SIFB3 can be used for the adaptation and the integration of the standard IEC 61131 PLC-based model of computing. It can happen if the SIFB3 includes as a legacy PLC-based code as a host processing stage. This processing stage is, by construction, connected to the physical layer of the system. Nonetheless, the details on the specific implementation of the blocks are beyond the architectural realm and scope. Implementation details have to concern with the organization and the nature of the (cyber-physical) computing infrastructure. This is left out of the scope of this positional work. Nonetheless, RMAS has been conceived particularly to scale down a powerful and expressive architecture also on devices with very limited resources, by leveraging the results about the relational model of computation performed in [17,20].

4.2. Mapping RMAS with IEC 61499 function block

So far, it has been highlighted how the RMAS architecture matches easily the IEC 61499 *resource model*, by means of the embedding of a composition of FBs and SIFBs.

A further step can be made towards the inner elements of this model, to focus on its fundamental elements.

We continue with proposing an alternative mapping of the RMAS unit that directly involves a function block, as shown in Fig. 6.

Typically, the scheme of Fig. 6 expresses another possibility of mapping of a RMAS unit directly into a

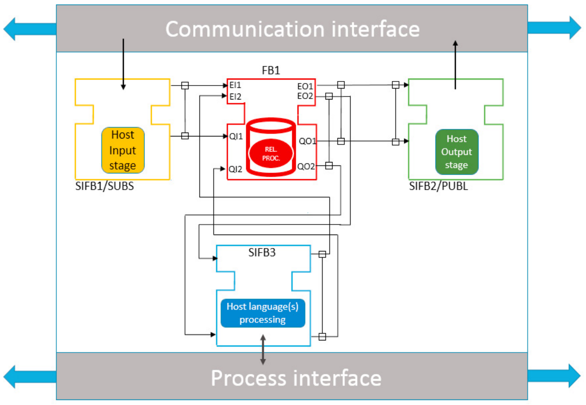

Fig. 5. Mapping RMAS units to IEC 61499 resource model.

function block, which is to be considered the minimal and fundamental element in the IEC 61499. By the way, this mapping constitutes also a link to the older and widespread IEC 61131 [22].

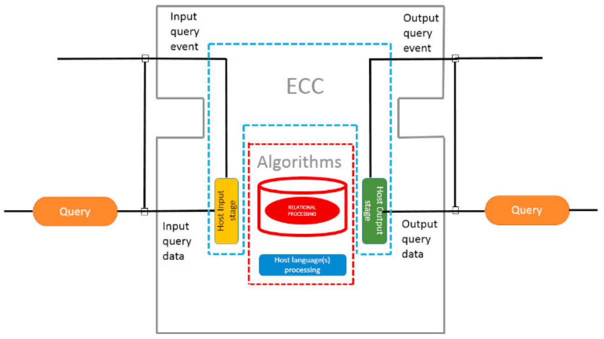

Fig. 6. Mapping RMAS units to IEC 61499 Function block

The function block shown in Fig. 6 will receive in input, and then produce in output, a database manipulation language query. The input query can be decoupled into data and an associated event by means of the WITH construct from the standard [7,23], denoted with a square connector symbol, over the connection point in the wire that links the input query event with the input query data (as in Fig. 6).

In addition, it is easy at this point to figure out a mechanism that decouples the event of query message arrival from its content. As an input, a query can be decoupled into data and its associated event of arrival and fetching of the input message payload.

In the output of the FB the mechanism is simply reversed. Events and data are merged into a query message to be transmitted.

If it is the case, the *host input stage* and the *host output stage* of the RMAS unit can constitute the typical execution control part of FBs. This part is usually associated to an execution control chart (ECC) that permits the algorithms to be triggered and controlled by events. Moreover, the ECC produces events in output as soon as the algorithmic part is completed (run to completion). This is exactly what the input and output stages of the RMAS are designed to.

The algorithmic part of the FB can instead be associated, at the same time, with the relational processing part of RMAS and with the host language processing part. The perfect mix of the two languages will depend on how the implementation on a specific device will balance between relational manipulation language and a specific host language of the device. It depends on the capabilities and the technologies that the device renders available – and their costs.

In practice, if the device underlying the FB is based on recent embedded electronics, it will be possible to deploy on it some sort of database management system. In [17,20] some experiments have been performed in this sense using an SQLite as an implementation of the database part, and using the C language as a very thin layer of host language. An interesting outcome from that test was the low cost of the device obtained with respect to the relatively high level of thee functionality achieved.

4.3. Hints on other possible mappings of RMAS with IEC 61499

Note that with the previous discussion we achieved the mapping of the RMAS architecture with two (somewhat nested) models and the related layers of the IEC 61499. This means that RMAS can potentially scale at different levels of the standard.

A similar exercise could be continued to map RMAS to the *Application model*. This model foresees the distribution of the FBs into different *devices* and *resources* [7,23]. In this regard, and as a mere hint for the future discussion, a remark can be made about the distribution of the RMAS architecture.

Being the core of the RMAS unit constituted by an active database management system, if this database system is distributed, then the distribution of an RMAS unit or application is immediately achieved. This is the easiest route for the mapping of RMAS to the *application model* of the standard, as the major peculiarity of that model is the distribution on the computing.

Another important part, foreseen from the IEC 61499, is its capability of adaptation, evolution, and reconfiguration obtained through the *management model* and *management applications* [7,22]. These models have been created in the standard with the purpose of allowing an automated reconfiguration and reprogramming of the applications. Once again, for this capability, RMAS can be useful. With reference to the creation and configuration mechanism recalled in section 3, the RMAS germinal unit (see Fig. 3) can be programmed and "grown" by another RMAS unit or by other kind of external software. This capability of the architecture matches in a major way the design of the management models of the IEC 61499.

5. Conclusion

The treatise in this paper aimed to stimulate both the researchers and practitioners involved in the IEC 61499 standard in order to make the community acquainted with some new developments that can make a contribution and provide possible support to the overall framework of the incoming standards in the area. In this sense, this work aims to constitute a position paper for the future developments of both the RMAS and the IEC 61499 implementations.

In particular, the proposed affinity between the RMAS architecture and the models of the IEC 61499 need to be more deeply discussed from the community. However, more details, examples and tests should constitute the ground for the viability and feasibility of the proposed techniques.

It is here just maintained that RMAS can be proven to represent another tool which the standard can count on. Mostly, in view of the already long path and the numerous efforts that have currently been spent in the factual introduction and integration of multi-agent systems in the industrial playground.

Many experiments in this sense are already undergoing from the industrial agents' community and, without doubt, the IEC 61499 has already to be considered the best candidate battlefield.

Typically, the features of distributed computing and life cycle management that the IEC 61499 standard keenly adopts are prone, by design, to close the gap between the flexibility of agents and the deterministic and dependability requirements in industrial processes.

Future work is needed to extend the propositions here put forth. In particular, more details should be given about the matching of the semantics of IEC 61499 and the RMAS at the different levels of resource, function-block, application, management, and distribution models of this standard.

Moreover, the choice of the host and the database manipulation languages can be a sensitive argument when the implementations have to meet the current proposals from the major device and systems vendors in the industrial automation playground.

To this end, RMAS has to be considered as an architecture completely agnostic about the choice of the languages and the implementations. It can be commented that this agnostic approach is by itself another aspect of compliance with the philosophy of the IEC 61499.

Acknowledgements

This research is supported by funding pertaining to three research projects. The first is the ENCORE project that receives funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 820434. The second is the REACT project, funded by

Italian National Ministry of Economic Development - MISE, for developing Methods and Tools for the “REACTive Product Design and Manufacturing”. The third supporting project is funded by the European Regional Development Fund ERDF, POR MARCHE region FESR 2014/2020, – AXIS 1 – Specific Objective 2 – ACTION 2.1, in the Regional Operational Program of the European Regional Development Fund: “HD3Flab - Human Digital Flexible Factory of the Future Laboratory”.

References

[1] Givehchi O, Landsdorf K, Simoens P, Colombo AW. Interoperability for industrial cyber-physical systems: An approach for legacy systems. IEEE Transactions on Industrial Informatics 2017; 13(6):3370–3378.

[2] Leitão P, Karnouskos S, Ribeiro L, Moutis P, Barbosa J, Strasser TI. Common practices for integrating industrial agents and low-level automation functions. In: 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON). IEEE; 2017. p. 6665-6670.

[3] Ribeiro L, Karnouskos S, Leitão P, Strasser TI. A community analysis of the IEEE IES industrial agents technical committee. In: IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society. IEEE; 2017. p. 6139-6144.

[4] Karnouskos S, Leitão P. Key contributing factors to the acceptance of agents in industrial environments. IEEE Transactions on Industrial Informatics 2016; 13(2):696-703.

[5] Unland R. Industrial agents. In: Industrial Agents. Morgan Kaufmann; 2015. p. 23-44.

[6] Ribeiro L. The design, deployment, and assessment of industrial agent systems. In: Industrial Agents. Morgan Kaufmann; 2015. p. 45-63.

[7] Zoitl A, Strasser T. Distributed control applications: guidelines, design patterns, and application examples with the IEC 61499. Boca Raton: CRC Press; 2017.

[8] IEC-61499-1. International and Electrotechnical Commission. IEC-61499-1, Function Blocks - Part 1: Architecture. Edition 2.0. Geneva: IEC; 2012.

[9] Van Leeuwen EH, Norrie D. Holons and holarchies [intelligent manufacturing systems]. Manufacturing Engineer 1997; 76(2): 86-88.

[10] Fletcher M. Building holonic control systems with function blocks. In: Proceedings of 5th International Symposium on Autonomous Decentralized Systems; 2001. p. 247-250.

[11] Christensen J. Holonic manufacturing systems-initial architecture and standards directions. In: 1st European Conference on HMS, Hannover (Germany); 1994.

[12] Fletcher M, Garcia-Herreros E, Christensen JH, Deen SM, Mittmann R. An open architecture for holonic cooperation and autonomy. In: Proceedings 11th International Workshop on Database and Expert Systems Applications; 2000. p. 224-230.

[13] Black G, Vyatkin V. Intelligent component-based automation of baggage handling systems with IEC 61499. IEEE Transactions on Automation Science and Engineering 2009; 7(2):337-351.

[14] Brennan RW, Fletcher M, Norrie DH. An agent-based approach to reconfiguration of real-time distributed control systems. IEEE transactions on Robotics and Automation 2002; 18(4): 444-451.

[15] Ferrarini L, Veber C. Design and implementation of distributed hierarchical automation and control systems with IEC 61499. In INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics. IEEE; 2005. p. 74-79.

[16] Bonci A, Pirani M, Longhi S. A database-centric framework for the modeling, simulation, and control of cyber-physical systems in the factory of the future. Journal of Intelligent Systems 2018; 27(4):659-679.

[17] Bonci A, Pirani M, Dragoni AF, Cucchiarelli A, Longhi S. The relational model: In search for lean and mean CPS technology. In: IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE; 2017. p. 127-132.

[18] Bonci A, Pirani M, Carbonari A, Naticchia B, Cucchiarelli A, Longhi S. Holonic Overlays in Cyber-Physical System of Systems. In: IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Vol. 1. IEEE; 2018. p. 1240-1243.

[19] Bonci A, Pirani M, Cucchiarelli A, Carbonari A, Naticchia B, Longhi S. A Review of Recursive Holarchies for Viable Systems in CPSs. In: IEEE 16th International Conference on Industrial Informatics (INDIN). IEEE; 2018. p. 37-42.

[20] Bonci A, Pirani M, Longhi S. Tiny Cyber-Physical Systems for Performance Improvement in the Factory of the Future. IEEE Transactions on Industrial Informatics 2018; 15(3):1598-1608.

[21] Bonci A, Pirani M, Bianconi C, Longhi S. RMAS: Relational Multiagent System for CPS Prototyping and Programming. In: 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA). IEEE:2018. p. 1-6.

[22] Strasser T, Zoitl A. Distributed real-time automation and control-reactive control layer for industrial agents. In: Industrial Agents. Morgan Kaufmann; 2015. p. 89-107.

[23] Zoitl A, Lewis R. Modelling control systems using IEC 61499 (Vol. 95). IET; 2014.

[24] Lewis RW. Programming industrial control systems using IEC 1131-3. 2nd edition. Stevenage (UK): IET; 1998.

[25] Vyatkin V. The IEC 61499 standard and its semantics. IEEE Industrial Electronics Magazine 2009; 3(4):40-48.