Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 180 (2021) 656–665

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing

IEC 61499 Device Management Model through the lenses of RMAS

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

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

Abstract

The RMAS (relational-model multi-agent system) architecture has been proven promising towards the introduction of autonomic intelligence for industrial agents, and suitable for the enforcement of some parts of the reference model in the IEC 61499 standard, for distributed adaptive automation and control systems, namely the Resource model and the Function Block model. In this paper, the analysis is extended to the Device Management model. The RMAS is overlaid on the structures of this model in order to demonstrate that it can constitute factual and promising means for the expression of essential self-* capabilities of autonomic industrial agents. The analysis here conducted covers also some aspects of the XAI (explainable artificial intelligence), due to their relevance in industrial context. RMAS results a fundamental step for the factual introduction of AI in industrial playground, as a technology framework for the explainability, controllability, and quality of the intelligent automation solutions for industry.

Keywords: multi-agent systems; relational model; IEC 61499; autonomic computing; explainable AI

1. Introduction

Nomenclature

RMAS	relational-model multi-agent system
RM	relational model (as treated and defined in [27])
OC	oracular computation
XAI	explainable artificial intelligence
MAS	multi-agent systems

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

1877-0509
10.1016/j.procs.2021.01.288

FB	function block
SIFB	Service Interface Function Block
LLC	low level control
HLC	high level control

The IEC 61499 standard [1] for distributed, reconfigurable industrial automation and control is the most prominent framework for the development of field-level agents and for the factual introduction of artificial intelligence (AI) solutions in industrial contexts. This standard tries to determine a factual and persistent evolution of the widespread and consolidated IEC 61131 [2], concerning PLC (Programmable Logic Controller) systems, and of the IEC 61804 series, concerning different aspects of Distributed Control System (DCS).

IEC 61499 was stimulated by the requirements coming from open and standardized architectures featuring Function Blocks (FB), applied in distributed Industrial Process Measurement and Control Systems (IPMCS) [3]. In addition, IEC 61499 conveyed ideas coming from holonic and agile manufacturing systems [4],[5].

The uptake of IEC 61499 has been rather long because the underlying technology is new, unlike in the previous IEC 61131-3 standard developments [6]. Moreover, some possible foundational limitations or flaws of this new standard have been spotted in [7] and references therein. Nonetheless, recently the IEC 61499 has received research interests and standardization efforts concerning the introduction of distributed intelligence in industrial systems by industrial multi-agent systems (MAS) [8], [9]. MAS are promising technological means in order to design large scale distributed control systems, provide intelligence, flexibility, robustness, and the self-* capabilities, as required from the concept of cyber-physical systems (CPS) [10] and their compliance to industrial specific needs [11].

The reference model of IEC 61499 provides a solid ground for the developments of MAS. In this paper, this grounding is leveraged and then reinforced with the introduction of RMAS (relational-model multi-agent system) [12]. The RMAS is proposed as a novel methodology that has the ambition to be used as an overlay architecture over the models of the IEC 61499. The introduction of RMAS enables and reinforces the valuable features of the standard towards complex CPPS (cyber-physical production system) design [13]. The IEC 61499 already provides basic but unified support for dynamic re-configurability. As explained in [10], the connection between automation and autonomic computing is increasingly pressing. Autonomic entities has to feature a minimum set of self-* capabilities (self-configuration, self-healing, etc.) achieved through technological support for autonomous program rewriting, triggered proactively by artificial agents. Re-configurability, along with other self-* capabilities, are essential requirements for the introduction of AI (artificial intelligence) in the very industrial realm, which inherently presents many barriers for the factual introduction of agents [14].

In this paper, it is supported that the Device Management model of IEC 61499 is a key feature towards AI, and that it holds enough expressivity in order to start the path towards autonomic computations in industry. RMAS represents an enhancement over the already well-established Device Management model provisions, in order to render industrial AI solutions within reach.

The RMAS architecture has been introduced first as a means for factual realization of networked and distributed automations with an event-driven database-centric paradigm [15], towards advanced holonic MAS, basing on best available technologies [12]. Subsequently RMAS has been challenged against the expectations of autonomic computing for multi-agent systems [10]. A first contribution of RMAS to the IEC 61499 has been achieved in [16], though limited to the Resource and the Function Block models.

In this work, the study is continued in order to cover also the Device Management model. Moreover, whilst the good features of RMAS over IEC 61499 models are analyzed, it is in parallel considered the additional problem of the development of explainable AI in industrial applications that increasingly are expected to feature autonomy, emergence, and intelligent behavior.

In section 2, some related work is presented in order to establish the context and background with respect to state-of-the-art perspectives in distributed intelligent industrial applications. This necessarily aims to introduce the relevance of the IEC 61499 Device Management model that is briefly recalled. In section 3, the fundamentals of RMAS are summarized, in order to present a methodological setup that is used for the study, analysis, and discussion that aim to cover the adoption RMAS and its implications on XAI. The expected contributions of the application of RMAS architecture to the IEC 61499 standard are collected in section 4. Section 5 is reserved to conclusions.

2. The role of the IEC 61499 Device Management model for autonomic applications, and related work.

Due to the growing interest around the holistic approach taken from the RAMI 4.0 framework (Industry 4.0 reference architecture [17]), the design, maintenance, life cycle, and other dimensional aspects of the CPPSs are growingly integrated. Any state-of-the-art system-engineering environment aims at supporting the complete life cycle of distributed component-based automation from design to operation and reconfiguration [18]. The digital models (and digital twins) are increasingly participating in the core layers of runtime environments, with providing the suitable plug-and-play, service-oriented components, and models for controllers and procedures [5],[19],[20]. In [19] authors propose a methodology for the generation of the PLC code based on a digital model within a virtual engineering tool; others explored knowledge-driven reasoning for program code generation [21].

Another growing concept in distributed knowledge context is the asset administration shell (AAS). Asset administration shells provide a unified and self-describing models and sub-models conforming to common meta-models for all Industrie 4.0 entities [22]. The basic idea in the AAS is to apply the type/instance distinction for all elements such as material type/material instance, product type/product instance, machine type/machine instance, and more. The same categorization is also at the core of the ARTI reference architecture [23] in holonic approaches that now consider also the digital twin's role [5]. Recently the connection between the AAS and the Function Block (FB) expressivity has been keenly explored [24], [25]. In [24], it is noted that current industrial automation systems integrate statically information and component configuration into the program code, which prevents a dynamic use of the information provided by the AAS. It is instead highly desirable that event-driven run-time access is granted to automatic self-configuration and adaptation of running component instances.

In [25], FBs are specified for reading and writing data and properties, and for invoking operations from an AAS, with the future perspective to possibly host a whole AAS, or some of its sub-models, locally in the LLC (low-level control) device boundary. The possible integration of high-level control (HLC) and low-level control can be rendered compliant with IEC 61499 FB specifications [4]; it is the main focus of current best practices in the implementation of industrial agents [8], [11]. In [4], it is also shown how to transform an IEC 61499-compliant LLC device into holonic device, as the definition of FBs can be used for encapsulation, reuse, distribution and integration of both LLC and HLC functions. Usually, this can be obtained by suitable combination of FBs in the IEC 61499 that are of three types [3], [26]:

- Basic Function Block (BFB), which event-driven functionality is defined by an Event Control Chart (ECC), a state machine with internal variables and algorithms written in programming languages stated in IEC 61131-3 or compliant with IEC 61499.
- Service Interface Function Block (SIFB), which behaviour is user-defined and is normally used to access hardware resources or platform dependent features like inputs, outputs or communication (non-algorithmic parts).
- Composite Function Block (CFB), which are constituted by a networked composition of BFBs, SIFBs, and (nested) CFBs.

FBs are used by the Resource model. A resource can be considered as a functional unit that is encapsulated in a device and has independent control of its operations, in order to be managed (created, configured, parameterized, started, deleted, etc.) without affecting other resources within the same device [3]. In turn, Resources are part of a Device model that describes the structure of a control equipment capable of executing FB networks, grouped in resource execution entities. It contains either the process and/or the communication interface [3].

The Device Management model is defined after FBs, resources, devices, and uses them as domain objects. It is at the basis of the definition of management applications that are responsible for managing the life cycle of resources and of FB networks within a resource. The management application can construct/destruct parts of other applications by creating/deleting function blocks and connections, and includes [3], [26]:

- creating/deleting function block instances within a resource with creating/deleting data and event connections between FB instances;
- setting parameters of resources and FB instances;
- initiating/terminating/change the execution of FB instances;
- providing services to support queries from communication links on the status of FB instances.

The full definition of the functionality and implementation of a management application is beyond the scope of IEC 61499, and this leaves room for unforeseen further developments [26]. Nevertheless, management applications can be modelled in exactly the same way as other applications using networks of FBs and SIFBs. Management applications can be implemented as separate sections of a device or be embedded into resources as shown in Fig. 1 (inspired from [3]). In Fig. 1 (a), a (resource) section of the device is dedicated to enforce management functionalities of the whole device and of other resources contained by the device. In Fig.2 (b), management is spread across different resources.

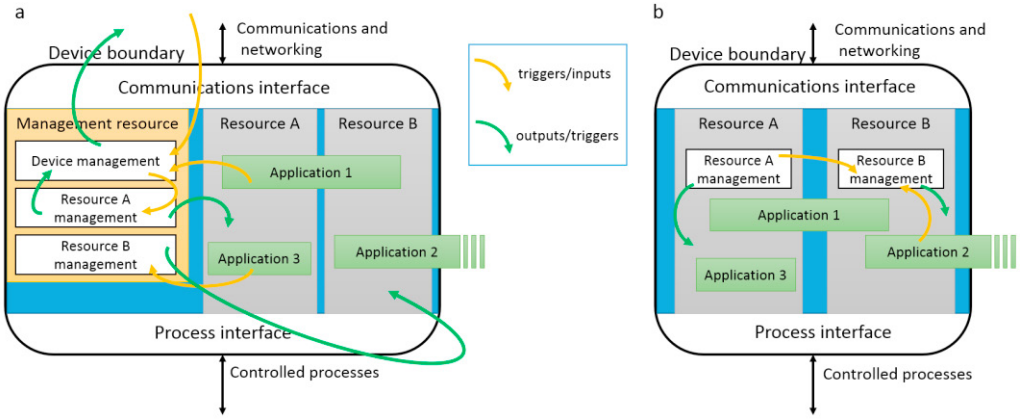

Fig. 1. (a) Device-bound and shared management resource; (b) Resource-wise distributed management applications.

3. Methodology: the RMAS architecture overlay on Device Management model, towards explainable AI

The methodology used in this work considers a new interpretation of the Device Management model and its involved objects. The new interpretation will depend and be oriented to the structures of the RMAS architecture. The RMAS architecture is overlaid across the elements of the Device Management model.

With making again reference to Fig. 1, the set of arrows there superimposed aim to show pictorially (and not exhaustively) how events and data, coming from different elements (also heterogeneous) outside or inside the boundaries of the device, respectively represent triggers and inputs for the activation of management procedures. In turn, data and events produced by the management functions constitute activation/deactivation and commands that manage the configuration and the execution of the applications. The triggering and inputs of management actions and their effects should be able to occur on every element inside the device (resources, FBs, applications), but also from outside (through SIFBs) in order to achieve unrestricted autonomic capabilities.

Essentially, the Device Management changes the overall functionality of the control program and adds functionalities to the control program by adding additional software components or rewire (rewrite) existing ones. It can be noted here that these functionalities are at the basis of any autonomic application, if they can be initiated by the application itself (embodied agent) depending on the environment state, interactions and contexts.

The RMAS (relational-model multi-agent system) architecture was first proposed in [12] as a synthesis and evolution of a framework grounded on database-centric, event-based, and publish-subscribe paradigms [15]. RMAS is based on studies about the role of the *full* Relational Model (RM) as a unifying model of computation for distributed holonic MAS and associated AI [27]. Thereafter, RMAS has been challenged in terms of autonomic computation issues in [10] and, at the same time, with the requirements of the models in IEC 61499 [16]. From [16] the Resource model and the Function Block model have been covered; in the following, the Device Management model will be the next. Albeit reader is suggested to refer to formerly cited works and references therein for due details, here we provide a summary list of the fundamentals concerning RMAS, barely enough to continue with the present scope:

- 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). This combination allows the Relational Model (RM), model of computation, being emulated with best available technologies [27]. First proof-of-concept, and complete, RMAS used simply Matlab as host language and SQLite as database management system and SQL language. In RMAS context, RM is not to be confused with any of the existing relational database technologies [27]. A pragmatic, and possibly restricted, version of an ideal RM language is obtained by synergizing the capabilities and the expressivity of a combination of host and database manipulation languages.
- All the events and information communications are carried on by queries: any kind of query language (implementing RM's algebra) is allowed, but standard SQL is already expressive enough to carry on.
- An RMAS unit is made of relational (database) objects (tables, constraints, triggers, stored procedures, schemas, etc.) and libraries of procedural code in the host language invoked by means of database triggers.
- An RMAS unit starts as a genotype, containing only minimum provisions to grow: a *v_bootstrap* table, a trigger (on INSERT) associated to it, and a communication channel as its umbilical cord. By subsequent injection of relational objects, the RMAS unit reaches (by ontogeny) a full-fledged status when needed (Fig. 2 (a)). In general RMAS units undergo both allopoiesis and autopoiesis processes.
- The host language allows *oracular computation* (OC) to be leveraged in order to render an RMAS unit a super-Turing machine [10]. Querying an oracle can produce effects on the world of interest (state of affairs) being it physical or virtual, by following the well-known CRUD (create, read, update, delete) paradigm. This involves also the effectors of a system (physical, robotic, mechatronic, automation, etc.), sensors, or non-algorithmic computations like neuromorphic or quantum computing. If the CRUD capability is fulfilled, we can define the oracles as *materialized* [10].

In Fig. 2 (a), a full-fledged RMAS unit is divided into its major parts, associated to colors: yellow is the part called the Input Stage; green in the Output Stage; red is the RM language expressions' domain (i.p. relational computing); blue is the host language computing part that accesses the materialized OC through suitable APIs. With more details in [16], this color convention is used to pictorially express the mapping between IEC 61499 entities and RMAS.

In Fig. 2 (b), we can see how any IEC 61499 Resource in general is a CFB, made of networks of SIFBs, CFB, and FBs, which can be functionally and recursively mapped with the aforementioned parts of the RMAS architecture. Note, in Fig. 2 (b), that the ECC (Execution Control Chart) can be handled completely by the RM provisions, as any state machine can be easily implemented in the relational database language domain (even with the tiny SQLite, without complete RM). On the other hand, the algorithmic part could need a link with other languages, in particular with the IEC 61131-3, for compatibility with existing PLC practice. Nevertheless, RMAS does not prescribe the use of any particular host or database language; it is a matter or practical convenience to be handled case by case. Usually ECC is implemented in Sequential Function Chart (SFC) for legacy reasons. In that case, the ECC would be part of the host SFC language.

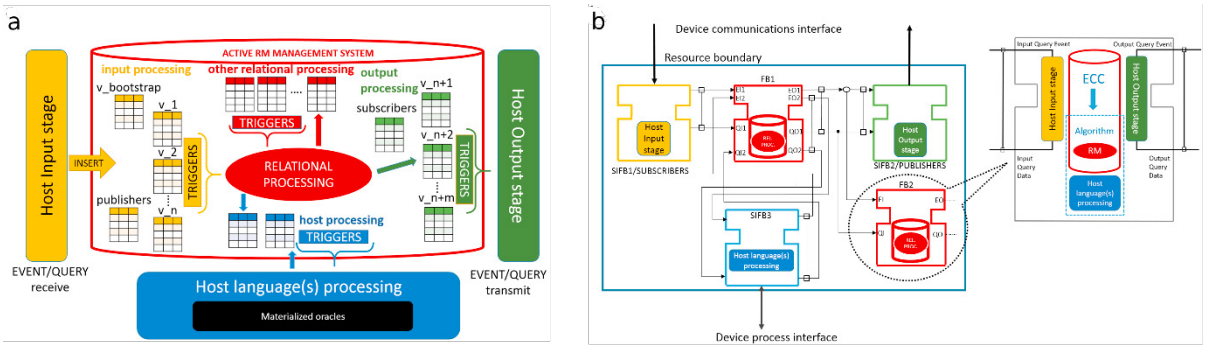

Fig. 2. (a)Full-fledged RMAS unit; (b) IEC 61499 Resource with RMAS interpretation, and recurrence of RMAS on FBs.

3.1. The dynamics of the deployment and the evolution of applications in RMAS over IEC 61499.

A feature embedded in the design of RMAS architecture is the capability to let an application self-evolve, self-replicate, and grow dynamically, basing on the evolutions and the interactions of an entity (or agent) with the environment and its contingencies and unexpected facts. This capability is also known as autopoiesis.

In order to fulfil this requirement, at the core of RMAS stands the definition of a generic agent skeleton structure that contains the genotype of all the RMAS units [10], [12]. This unit has initially only minimal provisions for being reached, bootstrapped and configured in order to be provided with suitable and minimum necessary facilities to develop into its full-fledged status. A value of the *v_bootstrap* variable (see Fig 2 (a)) can be transmitted by an INSERT operation as its carrier. A database trigger parses and executes the value content. Depending on the nature of the content, the trigger may perform typical database manipulations (e.g. creating/deleting other objects/tables, inserting or updating contents) or, through the power of the host language, the trigger may perform meta operations affecting the whole database (e.g, operations at the database schema level) or other parts of the software or hardware.

A new RMAS unit can be spawn from another (autopoiesis) and then be inflated into some resource of the IEC 61499 (comprising the Device Management resource itself, with a recursive approach). All this process in RMAS is performed by RM queries (or SQL queries whenever satisficing). Note that RM operations are not covered by current SQL standards, and here is the main need for the host meta language (or a suitable RM language).

3.2. A brief methodological discussion on explainability of AI in RMAS perspective.

As just observed, RMAS is in essence a prescription and a facilitator towards the RM. RM itself can be proved expressive enough [27] to cover at least the state of the art in GOFAI (Good Old-Fashioned AI) that has resulted in numerous symbolicist (or cognitivist) frameworks and related domain-defined languages (covering inference and knowledge-based representation and reasoning). The symbolicist contrasts the connectionist (emergentist) approach to AI that appears, today, at least in hybridized forms, in most of the prominent cognitive architectures of AI [28]. In recent years, and in MAS context, the connectionist approach has taken most of the AI scene, also thanks to the advancements in technologies as deep learning or swarm-based control, mostly relying on the emergence of unforeseen phenomena as systematic features.

While emergentism is promising and supportive, it unavoidably hinders explainability and controllability of AI solutions. Some research efforts and programs are currently trying to go deeper into these aspects in order to sanitize the current flaws and obscure zones in AI methodologies applied to engineering and other fields, and in order to come up with a common solid framework for researchers and practitioners [29], [30], [31]. In particular, there is a growing concern on the concept of trust between humans and the machines, and solid frameworks like game theory are increasingly used in the field of MAS to model trust between artificial agents [32].

In this context, Explainable Artificial Intelligence (XAI) emerged as a field of study that aims to provide a path towards a more transparent AI [31]. Explainable AI (XAI) should allow a machine to provide an explanation as to why it is acting in the way that it is. Currently, there is no unique and unified technical formal definition of the terms explainability and interpretability [29], [30], [31]. Interpretability and explainability are often used interchangeably [31]. Nevertheless, as reported in [30]: the act of interpretation is the mapping of abstract concept into a domain humans can make sense of; explanation is the collection of features of interpretable domain that have contributed for a given example to produce a decision. However, an intelligent system is interpretable if it can produce information about why and how it reached a result in a form understandable by people [32], [31].

Notably the DARPA XAI program is trying, since 2017, to turn the black boxes in Deep Learning into sets of explainable models and associated explainable interfaces, in order to enable human users to understand, appropriately trust, and effectively manage the emerging generation of artificially intelligent partners [33].

The leveraging of FBs and the other models of IEC 61499 allows creating networks and hierarchies of trusted components. If an FB is trusted, when it is used in an application the application in turn it can be trusted. Moreover, this process can be reiterated if also all the remaining of FB elements in the application can be trusted. Therefore, the component-based grounding of the standard is an intrinsic and built-in source of explainability and reputation.

With overlaying RMAS over IEC 61499 as a MAS application, as shown above, the "black-boxes" (the part concerning connectionism in the application) are restricted to the OC part that is accessed by the host language APIs (Fig. 2 (a)). The rest of the algorithmic processing performed by RM (if source code is available to users or certifiers) can be traced down, certified and explained, with reduced complicatedness through leveraging the component-based nature of the IEC 61499 models. In case of unmanageable number of components in applications, the explainability allows automated model-checking techniques to be applied.

4. Study results: the RMAS in IEC 61499, to take full advantage of Device Management model.

As remarked in [10], a relevant engineering challenge is posed by the installation and configuration of autonomic systems, in which the elements entail a bootstrapping process that begins when an element registers itself in a directory service by publishing its capabilities and contact information. Moreover, the element might also use a directory service to discover suppliers or brokers that may provide information or the services it needs to complete its initial configuration [18]. These features can be conveniently covered by RMAS but also by the IEC 61499 through the Device Management model.

Similar provisions for bootstrapping an application are foreseen by the Device Management model in IEC 61499, although in [26] they inquire: "If the management application loads normal applications, how is the management application itself loaded?". RMAS gives a prompt answer.

As surveyed in [34] the IEC 61499 does not strictly define its execution semantics and thus different implementations are possible. Authors present an exhaustive collection of runtime environments that have been implemented since the inception of the standard, and note that debate on open issues on formalisms for execution models are still open. Actually, the standard does not limit the implementations to a specific set of languages and tools. It embraces the whole set of IEC 1131-3 PLC programming provisions, along with a range of programming languages ranging from imperative to object-oriented (e.g. C, C++, Java), and other relevant computational paradigms for distributed computing as Erlang [35]. Nonetheless, this lack of unification does not hinder the strength of the IEC 61499 reference model. On the contrary, it is a living proof of its robustness across decades of numerous challenges. The model holds general and valid under unavoidable change of contexts, application and engineering methodologies and related technologies.

A first contribution obtained by the overlaying of RMAS's event-based computational model over the Device Management model, is to restrict the range of possible realizations of the aforementioned capabilities in order to achieve a practical specification on how to achieve them without sacrificing the expressivity of the standard's model.

To authors' knowledge, what have been still remained unexplored, is the adoption of other kind of computational models covered by logic and functional languages (e.g. Prolog, Haskell, Lisp, …) that are the usual means in the developments of AI systems under the symbolicist or cognitivist approaches.

The RMAS architecture can represent an opportunity for a gentle introduction of these symbolicist AI techniques, along with the prominent connectionist ones, brought about and confined to the OC part of the architecture. In the algorithmic part, the inclination on the use of the RM can be a means for the embracing of logic programming [27] in addition to the widespread imperative, declarative, functional, and object-oriented programming techniques. But most of all, RMAS provides at the same time the possibility to consider algorithmic (Turing-complete machines) and oracular parts that can include all kinds of computing (deep learning, machine learning, analog circuits, Quantum computing etc.). This added flexibility and expressivity provides IEC 61499 devices with AI and MAS applications in a way that potentially is tightly coupled to the LLC layer, avoiding the latencies, the infrastructural complexity, and the security issues of HLC and cloud solutions. A vision of how this coupling is obtained is presented in more detail in [36], where it is discussed how the RM provisions on even computationally tiny industrial devices can constitute a full-fledged intelligent multi-agent application stack.

Some examples on the use of Device Management of IEC 61499 can be found in [24], [25], [37], and [38], with an FB or external tool managing the conversion between XML-based or JSON representations into configurations or programs. RMAS can be proposed as a facilitator, in all these cases, by using the CRUD approach for any re-programming action. CRUD is immediately enforced by the RM – in particular by the database management language of choice in any RM implementation. In [38], it is also made a focus on low-resources devices, which is

compelling issue in edge and industrial IoT deployments of the IEC 61499. As shown in [27], and [36], the RM is particularly prone for deployment of intelligence solutions in such tiny automation devices, as well.

4.1. Implications on explainability of AI applications.

As just discussed, the introduction of the RMAS architecture over the IEC 61499 framework can constitute a handy bridge towards MAS and then AI for industrial-rank applications. This step has unavoidably to include the requirements and the issues of explainability that have been anticipated in section 3.2. In order to foresee the actual extent to which the introduction of RMAS can intervene into explainability, it useful to recall the two aspects defined as *intrinsic* and *post hoc* interpretability of AI applications (as treated by [31] and references therein). The intrinsic interpretability deals with constraints applied to the computational model in order to apply domain knowledge, to pragmatically restrict, or detect unforeseen behaviors or outcomes, and keep them confined under a certain safety threshold. On the other hand, the post hoc interpretability is the capability to trace down meanings and interpretations in a way that not necessarily opens an AI black box, but that can account for its emergent features.

The RM model, at the grounds of RMAS, is being implemented with languages that mostly extend database manipulation languages [27]. These languages are generally based on declarative and more natural-like language with respect to other programming language solutions. This is expected to bring an additional benefit with an increase in intelligibility and interpretability. As noted by [32], turning data into a concise natural language explanation can turn an otherwise opaque system into one that can be understood and predicted. In database-centric approaches like the RMAS, data and relations carry all the domain information in a declarative and interpretable way. By the introduction of RMAS, there is an inherent prescription on how to widen the explainable and interpretable parts of AI computing.

For example, in the research program described in [33], there is an effort to bring some intrinsic explainability into the computational models by appropriate structuring and classification. In addition, the post hoc interpretability is assigned to a specific sense-making interface to the psychological model of the human actors. The expressivity of the RM allows to bring most of information and knowledge across the symbolicist border of an otherwise opaque monolithic OC part. In the case of inherently opaque machine learning as in [33] it is difficult to foresee complete explainability, because it trades off for predictive performance in such applications. The best effort that RMAS and its RM can do is to offer a means for any explanations to be handled conveniently and grow in size and quality. This growth will tend to confine the amount of misbehaviors and uncertainties within a manageable practical size.

In the case of MAS applications, the effect of RMAS might be even more effective. Potentially, all the communications and the evolutions between agents, embodied into RMAS units, are tracked-down and stored by the relational tables in a natural way – see for example the input, output, host and other relational tables depicted in Fig. 2 (a). This characteristic constitutes some sort of intrinsic interpretability. At the same time, the dynamics and the history of the information and knowledge stored, processed, and transmitted between agents remains transparent and prone to post hoc analysis; a good and favorable feature when emergent meanings have to be extracted from a certain unexpected evolution of the MAS system.

The definite result of our analysis is that with RMAS over IEC 61499, there are some perspectives for effective XAI to be enforced in the industrial real field.

5. Conclusion

The Device Management model is a fundamental part of the IEC 61499 standard reference model. The model is open enough to accept a set of completely different implementations of the functional specifications it establishes. Thanks to this openness, it has been here possible to propose the RMAS as an architectural and technological overlay on IEC 61499 that aims to extend the reach of the standard towards the inclusion of autonomic automation solutions for industry, which require the factual introduction of explainable, trustable, and well-controllable AI solutions.

The RMAS architecture, and its inherent prescriptions, have been discussed in order to provide some hints to the communities of researchers and practitioners. The RMAS practice is originally focused on multi-agent systems,

which can be adopted in order to work out distributed intelligent industrial applications that realize the vision of the industrial artificial autonomic agency of next generation.

In this paper, it has been shown how RMAS shares some objectives with IEC 61499. The RMAS architecture can offer to this standard a technological framework that aims to reduce the current gap on factual adoption of industrial multi-agent systems and supportive means towards AI applications in industry. This objective necessarily brings about the issues and the efforts in the XAI vision, being this an essential feature for the applications of the future. The aspects of XAI have been discussed in order to show and exemplify how the introduction of the RMAS architecture is a promising step in this direction.

Nonetheless, the experimentation of RMAS is still in its early phases. Though promising, it should be thoroughly compared and challenged against the numerous use cases about the IEC 61499 Device Management already present in literature first, and then against the high expectation and hopes for autonomic computing and XAI in the industrial real field.

Future realization of RMAS instances will be realized by coupling a lightweight database management system with the languages adopted by major runtime environments in IEC 61499, in order to test and highlight the potential or the limits of this architecture.

Acknowledgements

This research is supported by: the ENCORE Horizon 2020 project Grant Agreement No. 820434; the REACT project, "REACTive Product Design and Manufacturing", funded by Italian MISE; "HD3Flab - Human Digital Flexible Factory of the Future Laboratory", ERDF, POR MARCHE region FESR 2014/2020.

Authors would like to express sincere gratitude to the anonymous reviewers, whose very valuable suggestions and comments really helped in the overall improvement of this work.

References

[1] International Electrotechnical Commission. (2012) "International Standard IEC61499, Function Blocks, Part 1 - Part 4", IEC.

[2] International Electrotechnical Commission. (2003) "International Standard IEC61131-3: Programmable Controllers, Part 3: Programming Languages", IEC.

[3] Zoitl, Alois, and Thomas Strasser, eds. (2017) *Distributed control applications: guidelines, design patterns, and application examples with the IEC 61499*. CRC Press.

[4] Christensen, James H. (2003) "HMS/FB architecture and its implementation." In *Agent-based manufacturing:* 53–87. Springer, Berlin, Heidelberg.

[5] Derigent, William, Olivier Cardin, and Damien Trentesaux. (2020) "Industry 4.0: contributions of holonic manufacturing control architectures and future challenges." *Journal of Intelligent Manufacturing* : 1–22.

[6] Vyatkin, Valeriy. (2011) "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review." *IEEE transactions on Industrial Informatics* **7** (**4**): 768–781.

[7] Thramboulidis, Kleanthis. (2012) "IEC 61499: Back to the well proven practice of IEC 61131?." In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*: 1–8.

[8] Leitão, Paulo, Stamatis Karnouskos, Luis Ribeiro, Panayiotis Moutis, José Barbosa, and Thomas I. Strasser. (2017) "Common practices for integrating industrial agents and low level automation functions." In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*: 6665–6670.

[9] Ribeiro, Luis, Stamatis Karnouskos, Paulo Leitão, and Thomas I. Strasser. (2017) "A community analysis of the IEEE IES industrial agents technical committee." In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*: 6139–6144.

[10] Bonci, Andrea, Sauro Longhi, and Massimiliano Pirani. (2019) "RMAS Architecture for Autonomic Computing in Cyber-Physical Systems." In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*: 2996–3003.

[11] Leitão, Paulo, and José Barbosa. (2016) "Building a robotic cyber-physical production component." In *Service Orientation in Holonic and Multi-Agent Manufacturing:* 295–305. Springer, Cham.

[12] Bonci, Andrea, Massimiliano Pirani, Carlo Bianconi, and Sauro Longhi. (2018) "RMAS: relational multiagent system for CPS prototyping and programming." In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*: 1–6.

[13] Ribeiro, Luis, and Mats Björkman. (2017) "Transitioning from standard automation solutions to cyber-physical production systems: an assessment of critical conceptual and technical challenges." *IEEE systems journal* **12** (**4**): 3816–3827.

[14] Karnouskos, Stamatis, and Paulo Leitao. (2016) "Key contributing factors to the acceptance of agents in industrial environments." *IEEE Transactions on Industrial Informatics* **13** (**2**): 696–703.

[15] Bonci, Andrea, Massimiliano Pirani, and Sauro Longhi. (2018) "A database-centric framework for the modeling, simulation, and control of cyber-physical systems in the factory of the future." *Journal of Intelligent Systems* **27** (**4**): 659–679.

[16] Bonci, Andrea, Sauro Longhi, Emanuele Lorenzoni, and Massimiliano Pirani. (2020) "RMAS architecture for industrial agents in IEC 61499." *Procedia Manufacturing* **42**: 84–90.

[17] VDI/VDE Society. (2015) "Reference architecture model Industrie 4.0 (RAMI4.0)," Status Report.

[18] Harrison, Robert, Daniel Vera, and Bilal Ahmad. (2016) "Engineering methods and tools for cyber–physical automation systems." *Proceedings of the IEEE* **104** (**5**): 973–985.

[19] Jbair, Mohammad, Bilal Ahmad, Ahmad Mus'ab H, Daniel Vera, Robert Harrison, and Tony Ridler. (2019) "Automatic PLC code generation based on virtual engineering model." In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*: 675–680.

[20] Dai, Wenbin, Valeriy Vyatkin, James H. Christensen, and Victor N. Dubinin. (2015) "Bridging service-oriented architecture and IEC 61499 for flexibility and interoperability." *IEEE Transactions on Industrial Informatics* **11** (**3**): 771–781.

[21] Li, Ruoqi, Wenbin Dai, Sheng He, Xiaosheng Chen, and Genke Yang. (2019) "A Knowledge Graph Framework for Software-Defined Industrial Cyber-Physical Systems." In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*: 2877–2882.

[22] Plattform Industrie 4.0 (2019) "Details of the Asset Administration Shell, Details of the Asset Administration Shell Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 2.0.1)" Platform Industrie 4.0, Berlin, Germany, Techical Report, www.plattform-i40.de

[23] Valckenaers, Paul. (2018) "ARTI reference architecture–PROSA revisited." In *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing:* 1–19. Springer, Cham.

[24] Wenger, Monika, Alois Zoitl, and Thorsten Müller. (2018) "Connecting PLCs with their asset administration shell for automatic device configuration." In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN):* 74–79.

[25] Terzimehić, Tarik, Andreas Bayha, and Kirill Dorofeev. (2019) "Function Blocks for the Interaction with the Asset Administration Shell." In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA):* 1235–1238.

[26] Zoitl, Alois, and Robert Lewis (2014) *Modelling control systems using IEC 61499*. Vol. 95. IET.

[27] Bonci, Andrea, Massimiliano Pirani, Aldo Franco Dragoni, Alessandro Cucchiarelli, and Sauro Longhi. (2017) "The relational model: In search for lean and mean CPS technology." In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN):* 127–132.

[28] Kotseruba, Iuliia, and John K. Tsotsos. (2020) "40 years of cognitive architectures: core cognitive abilities and practical applications." *Artificial Intelligence Review* **53** (**1**): 17–94.

[29] London, Alex John (2019) "Artificial intelligence and black-box medical decisions: accuracy versus explainability." *Hastings Center Report* **49** (**1**): 15-21.

[30] Došilović, Filip Karlo, Mario Brčić, and Nikica Hlupić (2018). "Explainable artificial intelligence: A survey." In *2018 IEEE 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*: 0210-0215.

[31] Carvalho, Diogo V., Eduardo M. Pereira, and Jaime S. Cardoso. (2019) "Machine learning interpretability: A survey on methods and metrics." *Electronics* **8** (**8**): 832.

[32] Andras, Peter, Lukas Esterle, Michael Guckert, The Anh Han, Peter R. Lewis, Kristina Milanovic, Terry Payne et al. (2018) "Trusting intelligent machines: Deepening trust within socio-technical systems." *IEEE Technology and Society Magazine* **37** (**4**): 76–83.

[33] Gunning, David, and David W. Aha. (2019) "DARPA's explainable artificial intelligence program." *AI Magazine* **40** (**2**): 44–58.

[34] Prenzel, Laurin, Alois Zoitl, and Julien Provost. (2017) "IEC 61499 runtime environments: A state of the art comparison." In *International Conference on Computer Aided Systems Theory:* 453–460. Springer, Cham.

[35] Prenzel, Laurin, and Julien Provost. (2019) "FBBeam: an erlang-based IEC 61499 implementation." In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN):* 629–634.

[36] Bonci, Andrea, Massimiliano Pirani, and Sauro Longhi. (2018) "Tiny cyber-physical systems for performance improvement in the factory of the future." *IEEE Transactions on Industrial Informatics* **15** (**3**): 1598–1608.

[37] Strasser, Thomas, Filip Andrén, Felix Lehfuss, Matthias Stifter, and Peter Palensky. (2013) "Online reconfigurable control software for IEDs." *IEEE Transactions on Industrial Informatics* **9** (**3**): 1455–1465.

[38] Pinto, Leandro Israel, Cristiano D. Vasconcellos, Roberto Silvio Ubertino Rosso, and Gabriel Hermann Negri. (2016) "Icaru-fb: An IEC 61499 compliant multiplatform software infrastructure." *IEEE Transactions on Industrial Informatics* **12** (**3**): 1074–1083.