Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 232 (2024) 446–455

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Industry 4.0 and Smart Manufacturing

A reference architecture to implement Self-X capability in an industrial software architecture

Walter Quadrini[a,*], Francesco Alessandro Cuzzola[a], Luca Fumagalli[a], Marco Taisch[a], Gabriele De Luca[b], Marta Calderaro[b], Mattia Giuseppe Marzano[b], Angelo Marguglio[b]

[a]*Department of Management, Economics and Industrial Engineering, Politecnico di Milano, p.zza Leonardo da Vinci, 32, 20133 Milan, Italy*
[b]*Engineering Ingegneria Informatica S.p.A., Via Monteroni s.n., C/O Edificio Dhitech – Ecotekne, I-73100, Lecce, Italy*

Abstract

The recent advancements in the industrial communities which have taken place from 2011 and are nowadays referenced as "Industry 4.0" allowed the scientific community to highlight the benefits of data-based approaches when dealing with the industrial operations. The most impressive outcomes, in particular, leveraged on the tools implementing Machine Learning algorithms, which gave companies new tool to understand current and future statuses of operations and processes through a data-driven approach. These tools often rely on historical data and can indeed cluster or label events basing on existing or past measurement, while, when dealing with unforeseeable events, are limited to alert warning. In order to overcome these burdens, the concept of Autonomous Computing is nowadays gaining more and more importance, as a Computer Science discipline dealing with the resilience of a system with respect to unexpected and unregistered behaviours. The adoption of this kind of technology would hence allow to enhance the existing data analytics industrial infrastructures reaction capabilities, able to maximise the assets' availability beyond the structural limits of traditional Machine Learning approaches. This work proposes a reference architecture to embody these functionalities in an industrial software architecture based on Artificial Intelligence processes and pipelines, to pave the way for a full implementation of autonomous responsiveness from the software modules constituting the architecture. Despite the rising interest in Autonomous Computing, its adoption in industry is indeed slowed by the fear of complexity in its implementation. The (numerically) poor body of knowledge about software architectures devoted to Autonomous Computing is hence enriched, and a reference implementation is presented as well, in order to encourage the practitioners' community towards the adoption of this technology.

* Corresponding author. Tel.: +390223999269
E-mail address: walter.quadrini@polimi.it

1877-0509
10.1016/j.procs.2024.01.044

Keywords: Industry 4.0; software architecture; Machine Learning; MAPE-K; Self-X;

1. Introduction

Since its introduction at the 2011 Hannovermesse, the so-called "Industry 4.0" has been identified as a game changer in the industrial environment [1]. The idea behind this new approach to manufacturing, is to enhance the interactions between shop-floor devices and high-level automation systems, leveraging on the pre-existing generation of industrial monitoring and control systems, embodied by the so-called Computer-Integrated Manufacturing pyramid (CIM pyramid, represented in Fig. 1a), which, on the other hand, reproposes the traditional rigorous hierarchical approach of the well-known ISA95 [2]. This structure expresses the typical model of production systems automation before the Industry4.0 paradigm, when each level of the pyramid relied on the layer below, with the lower levels providing the data and control necessary for the higher levels to monitor, make decisions, and take actions. This hierarchical structure proposes a clear separation of responsibilities and enables the integration of different systems and technologies [3]. The pyramidal layers are built on a "layer 0", which includes the hardware (sensors, actuators) involved in the production process that are responsible for the direct control and the collection of data from the field devices, and then build up "layer 1" (the Programmable Logic Controller, PLC layer, which is responsible for the closed-loop control of the equipment, the collection of data, and the monitoring of alarms and events), "layer 2" (Supervisory Control And Data Acquisition, SCADA layer, which synchronises, oversees and routes the data produced by PLCs), "layer 3" (Manufacturing Execution System, MES layer, responsible of scheduling and coordination across the entire shopfloor activities) and "layer 4" (Enterprise Resource Planning, ERP layer, managing the business-related and long-term functions of an industrial company).

The proposed hierarchy is also referenced by the standard IEC 62264, established by the International Electrotechnical Commission, and which serves as a complementary standard to ISA 95, providing a comprehensive framework for the integration and communication of enterprise and control systems in the industrial context, defining a set of functional requirements and protocols for communication between office floor and shop floor, deemed critical for effective industrial control and automation [3].

This hierarchical and layered approach is embedded in the right dimensions of the "Reference Architecture Model for Industrie4.0" (RAMI4.0, Fig. 1b), whose three-axes structure serves as a guide for aligning the needs of various fields with various standards, to implement Industry 4.0 methods, and to enable the recognition of overlapping requirements, gaps and their resolution in accordance with related industrial standards [4]. To the aforementioned layers of the CIM, RAMI4.0 adds two further ones, one self-explaining and named "Product" and another one named "Connected world", which usually represents the cloud-based infrastructure connecting the company with the external stakeholders. For what concerns the other axes, the left one frames the product life cycle and the value stream it

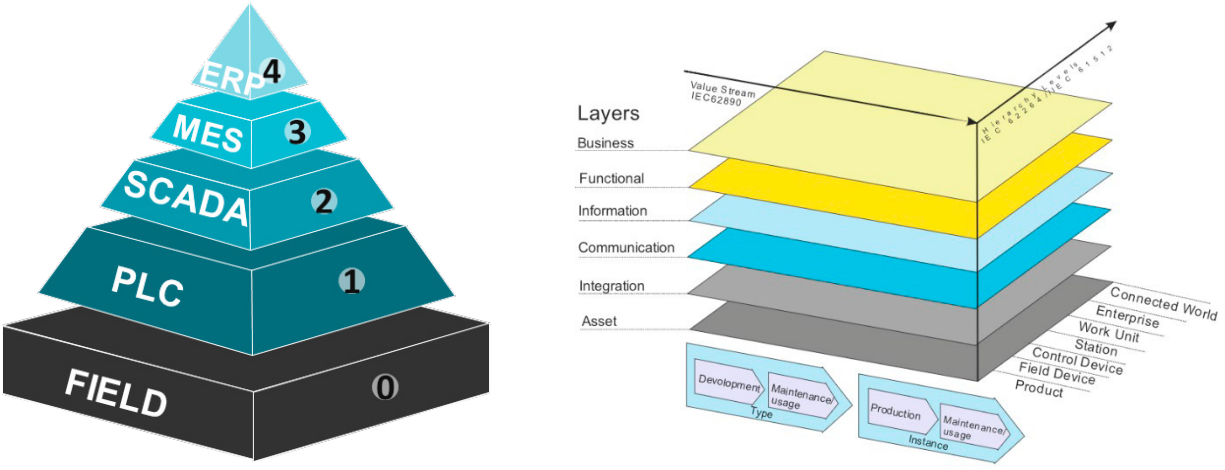

Fig. 1. (a) CIM representation; (b) RAMI4.0.

comprises, deemed as crucial criteria in contemporary manufacturing, since the complete digitalization of the entire development-market chain offers enormous possibilities for improving goods, machinery, and other Industry 4.0 architecture layers over their entire lifecycle [5]. The vertical axis, on the other hand, indicates the appearance of Industry4.0 components from various angles (a look from the market aspect, a look from a perspective of functions, information, communication, a look from an integration ability of the components) [6]. In 2016, Zezulka et al. provided a detailed insight into the layers composing this axis [5].

Both CIM and RAMI4.0 act as frameworks the software architects can refer to implement any solution connecting shopfloor assets to digital services. Software architectures are indeed the backbone through which data are generated and are able to feed high-level services providing companies different and documented advantages, e.g., the interoperability of different suppliers' equipment [7], the transparency of machine states [8], the quality control [9], and the maintenance [10], often relying on Machine Learning algorithms [11].

These techniques have however already demonstrated their lack to handle unexpected events, given the limitations of their training datasets, or their intrinsic incapability to derive actions apart from warn the users about these events (depending on the specific implementation technique). With respect to the subset of Supervised Learning Methods [8], the computer science community has tried to overcome this issue by proposing new sub-classes of algorithms under the umbrella of the so-called "Lifelong Machine Learning" [12]: these algorithms, after an initial training phase, continuously re-train themselves with occurred events, increasing their knowledge or improving their performances with respect to recent trends. However, this approach has already shown some weaknesses, when it led the algorithms to drift from their initial behaviour. This particular feature, which in Neural Networks is addressed as "catastrophic interference" [13], could, for example, make the algorithm to provide different outputs when dealing with identical inputs received in different moments.

This paper is organised as it follows: Section 2 provides an overview of the body of knowledge this article relies on as well as other works dealing with the same problem; Section 3 depicts a reference architecture able to respond to this problem; Section 4 suggests a reference framework relying on open-source solutions to implement the proposed solution; Section 5 concludes the article, summarising the main findings and debating possible limitations.

2. State of the art

This work lays on a previously published article by the same authors [14] which indeed defined a software architecture specifically designed to implement Industry 4.0-based services in the process industry context. As depicted in Fig. 3, the architecture aimed at making the production data integrated and available for a set of applications, through the implementation and deployment of several functional blocks.

This layered architecture has also been functional to host ML-based services, which are generally able to identify patterns and events from data, or to categorise data/events according to pre-defined clusters [11]. In other words and simplifying, these services usually define sets of clusters on historical data and allow to label new events according to the pre-defined clusters [15].

This approach, however, leave uncovered the resilience of the process with respect to unforeseeable events which may occur and may lead to misbehaviour of the nominally designed process. To overcome this issue, a new approach, namely "Autonomous Computing", derived from the computer science is gaining more and more importance in the manufacturing domain [16].

Autonomous computing pretends to implement four different abilities (namely "Self-X" or "Self-CHOP" [17]) to implement its resilience requirements: these abilities are known as Self-Configuration (the ability of a system to reconfigure itself, adapting to external changes through the installation and configuration of new software components), Self-Healing (the ability to maximise the availability by detecting diagnosing, and repair whenever possible), Self-Optimization (the ability to improve performances in a reactive and proactive way), and Self-Protection (the ability to detect in advance external factors which could lead to any kind of damage)[18].

Since its earliest appearance in scientific literature, the Self-X capabilities have been developed relying on inner control loops implementing the so-called MAPE-K framework, which realises the aforementioned abilities basing on four sequential phases (Monitor, Analyse, Plan, and Execute) relying, on turn, on the pre-existing shared Knowledge available to the designers/users of the system [19].

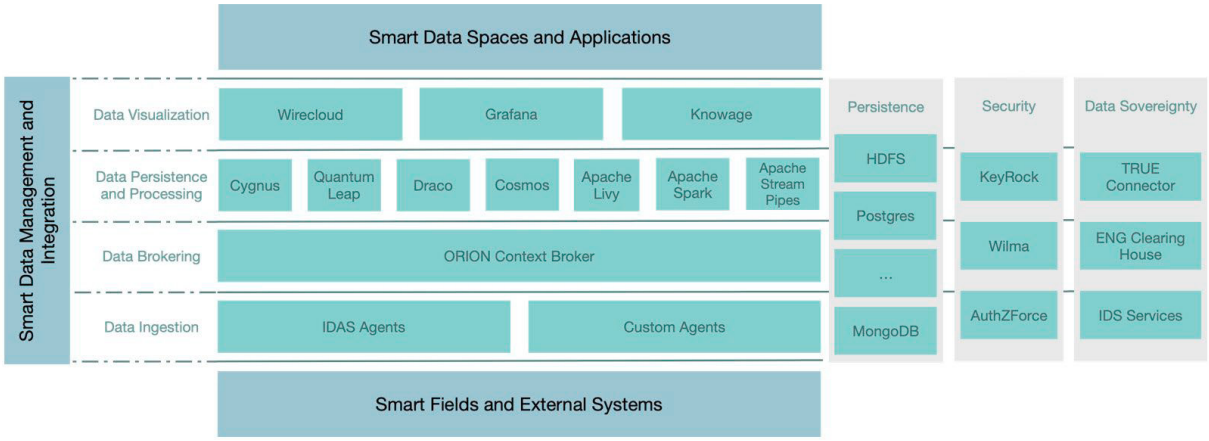

Fig. 2. Example of layered architecture [14].

This research frames itself hence in the design of a software architecture adding Self-X capabilities in a typical Industry 4.0-compliant one. From a research perspective, this is justified by the fact that MAPE-K framework is gaining increasing popularity in the academic and practitioners' community: indeed, several recently published works highlighted the benefits of this approach in the manufacturing domains. However, the majority of these publications reported advancements in control loops applied to a single machine (such as IEC 61499-compliant PLCs [20], or Digital Twins applied to the control optimisation of a single asset [21]) and, according to the authors' knowledge, only two works debated the integration of MAPE-K framework in a pervasive software architecture for production systems [22,23]. Among other reasons, a recent and uplifting article attributed the lack of adoption of this framework to the huge gap between the conceptual (and "inspirational") concept of the MAPE-K and its actual implementation, which could dishearten companies to adopt Self-X solutions [24]. The Self-X-coherent architecture here described seeks hence to reinforce the existing body of knowledge with a deeper description of the software modules enabling Self-X functionalities, and to provide a reference implementation, in order to demonstrate the easiness its adoption in manufacturing companies.

3. Reference Architecture

The proposed architecture is therefore declined on horizontal and vertical layers, as for the one presented in Fig. 3, but enriches it with additional blocks representing the AI functionalities and the Autonomous Computing capabilities.

Starting from the lowest levels, the proposed system depicts as it follows:

- Execution and External Layers: named as "Execution Layer" and External Systems in Fig. 4, are conceptual layers listing all the software interfaces collecting data from the shopfloor devices (as well as from any external component) and feeding back instructions to the actuators. This layer can be seen as the interface between PLC and Field layer of the CIM pyramid.
- Integration Layer: constituted by the two blocks named "Data Ingestion and Transformation Layer" and "Adapter", with the first block, this layer transforms the data coming from the aforementioned layer into format/standard readable by the upper layers, while the second block adapts it in terms of communication protocol. This functionality can be framed in the Information domain of IIRA.
- Data Brokering and Persistence Layer: this layer grants a collection of historical data from different and, possibly, synchronous sources, decoupling their continuous stream from real-time use cases. A Data Broker aggregates information from a variety of sources dividing them on topics designed to provide the required information to the upper layers' modules. This layer also embodies a module named "ML/DL Datasets": a list of persistent datasets ready to feed ML implementations. These datasets, hosting historical

data and aggregated information, are the modules embodying the Knowledge attribute of the MAPE-K framework.

- Data Processing Layer: this macro-layer is the place supposed to host the traditional Industry 4.0-compliant software modules, able to generate adding value from raw data, and it is sub-divided in several sub-layers. Data Exploration layer makes data ready to be sent on the upper layers and/or to be consumed on the lower ones. It is the first step of the data analysis used to explore and visualize data to uncover insights from the start or identify areas or patterns, allowing a deeper understanding of a dataset. ML/DL Catalogue hosts the libraries needed to create ML models from scratch. Trained Models Catalogue is an indexed list of the trained ML implementations, complete of metadata about their training and application domain. Pipelines catalogue, on the other hand, is a list of untrained ML applications ready to be trained and accompanied by instructions to be easily deployed and scaled. AI Engine is, on the other hand, a set of tools that makes possible the creation, execution and scalability of AI pipelines and models: it accesses all catalogues, acting as both an experiment platform and a design one; it also orchestrates data flow from the input through the different ML modules of the pipeline, and finally collects results and making them available for the above layers. AI Pipelines is the core "asset", result of interconnected and streamlined collection of operations which can be built from scratch by assembling pieces through dedicated program interfaces and executed; in doing this, a further instance of the Knowledge MAPE-K framework is here embodied. AI Interfaces is the layer including both data/results/explanation dashboarding and AI programming interfaces, allowing users to visualize and understand model results and behavior, as well as customize AI assets for training and build AI pipelines.
- Meta Layer: it is the layer used to drive the communication between the Autonomic Manager and Data Processing Layers. The main feature of this layer is to enable Autonomic Manager for the on-the-fly problem solving, performing its Self-X abilities. On the other hand, it complements the processing layer acting as a "repository" (push and pull) of information.
- Autonomic Manager Layer: this macro-layer has the role of autonomous AI Data pipeline coordinator and decision maker adopting MAPE-K framework and implementing the Self-X abilities. It has the possibility also to interact with the applications layer to improve its functionalities and support the AI pipeline processing. It is composed by two sub-layers: one implementing the Self-X abilities and one embodying the MAPE-K framework, composed by the four blocks implementing the "MAPE" agents. Self-X Coordination orchestrates indeed the AI Pipelines, implementing the abilities of Self-Configuration, Self-Healing, Self-Optimisation, and Self-Protection. MAPE sub-layer assumes the availability of gathered information from the AI Pipelines and the execution layer (Knowledge): it influences the future behaviour of managed resources, but requires functionality to monitor the data, analyze the current status, and plan corrective actions.
- Applications Layer: all the modules which suppose an interaction with users are conceptually hosted in this layer. In particular, this layer can host whatever application fed by/insisting on AI Interfaces or the Autonomic Manager (e.g., dashboards), but, in any case, are supposed to involve the human in the data generation process, such as labelling events for ML training, providing feedbacks with respect to the effectiveness and transparency of AI pipelines, and building artificial or synthetic datasets to train ML algorithms and to validate them artificially constructing a real-like unexpected event.

Additionally, the architecture relies on "vertical" or "pervasive" layers, accomplishing to the following roles:

- Digital Models and Vocabularies: a set of metadata representing the information about the structures and syntaxes of data exchanged in the different layers of the architecture, useful for faster configuration and transparency. This layer can also include descriptive information about data sources, as well as documentation over different modules (e.g., APIs).
- Data in Motion and Data at Rest: this layer represents the actual raw data source for the Perceptors, the Data Ingestion and Transformation layers, as well as for every other module in other layers. Data in Motion are here addressed as data flowing from the field (or from external systems) and Data at Rest are referred as static data supposed to be shared just once (e.g., models of data sources or files mapping data sources).

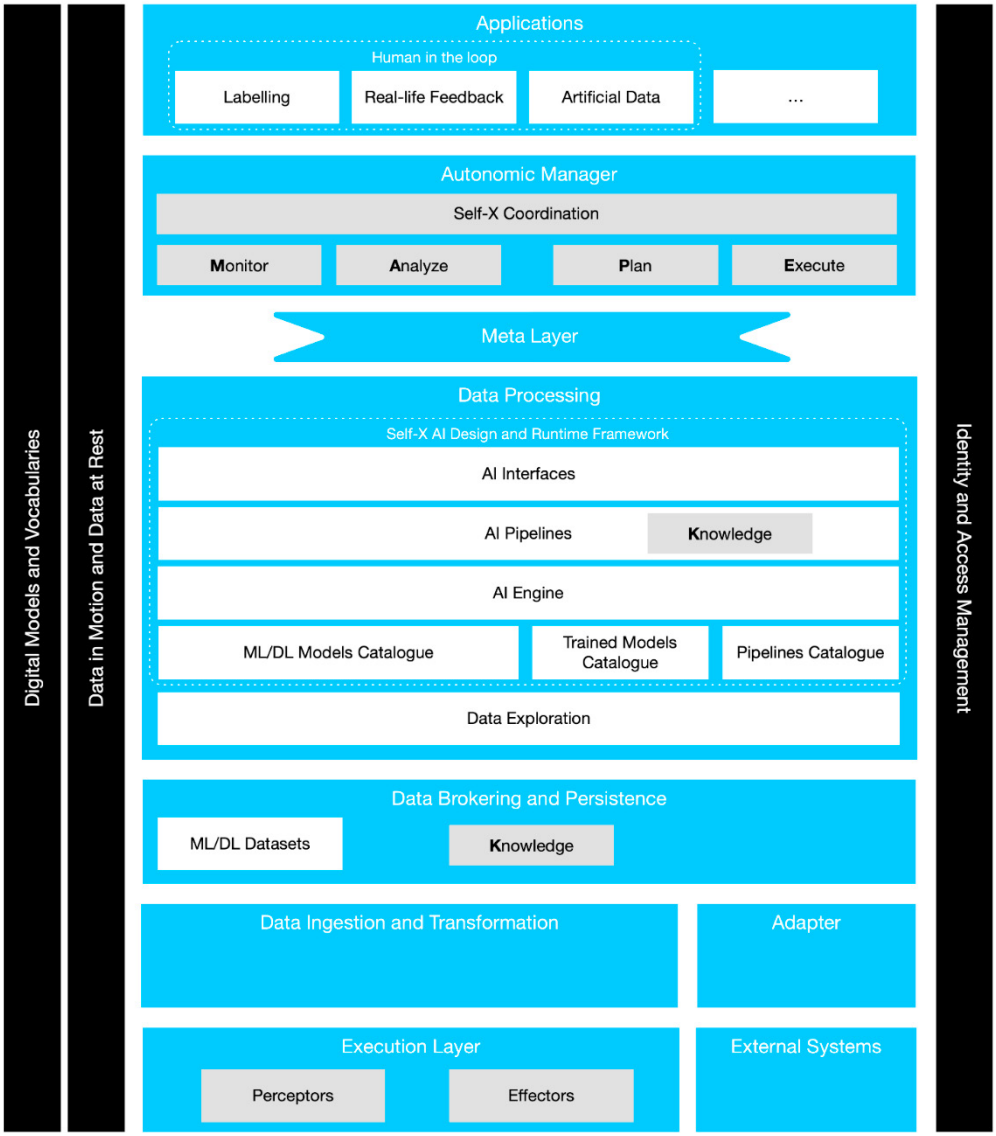

Fig. 3. Proposed architecture.

- Identity and Access Management: this layer includes all the applications ensuring trusted access to the different modules deployed in the architecture implementation.

Fig. 4 depicts in detail the aforementioned layers, highlighting the sequence of layers.

4. Reference Implementation

Some technical tools selected by the authors for various reasons are presented in this section, so as to ease the adoption of the proposed architecture and to better frame its operational details. This is supposed to represent a reference, so adopters can select alternative tools or rely on custom implementations.

Adopting a top-down approach, Apache Superset [25], as an enterprise-ready business intelligence web application, is the suggested tool for the implementation of Applications layer, given the features and the options that make it easy

for users of all skill sets to explore and visualize their data, from simple line charts to highly detailed geospatial charts, in an easy and interactive way.

The Autonomic Manager layer is supposed to be implemented by a set of instances materialized by a Complex Event Processing (CEP) tool. Among the different ones available in the various open source communities, possible alternatives are represented by Perseo CEP, a product of the FIWARE portfolio based on Esper [26] and natively interfaceable with Orion Context Broker in its NGSIv2 configuration [27], by DRACO [28], another FIWARE module based on Apache NiFi (a platform provided by Apache Foundation whose aim is to automate data flow basing on dynamically-set conditions [29]), or by Apache Airflow, a software again by Apache Foundation which can serve to the same purposes, but with a perspective oriented on the orchestration of data flows [30].

Despite being on different conceptual levels, since they accomplish purposes of data brokering, both the Meta Layer and the Data Brokering and Persistence can rely on the same instance of a single tool. The suggested tools for implementation are middleware embracing the publisher/subscriber communication policy: notable examples are constituted by Orion Context Broker (a middleware maintained by FIWARE Foundation, based on NGSIv2 or NGSI-LD REST APIs [31]) and by Apache Kafka (a message-oriented middleware maintained by the Apache Foundation and relying on binary TCP protocols [32]). Other possible implementations are constituted by Mosquitto (the most known MQTT broker [33]) or RabbitMQ [34] (the most known AMQP broker [35]). Despite Orion Context broker is the preferred choice, because of its APIs able to natively merge and integrate data from different topics, it lacks persistence features. Anyways, in order to implement the persistence in the Data Brokering and Persistence layer, some database technology is supposed to be installed, to guarantee the MAPE-K Knowledge feature after the consumption of an event-based entity.

For what concerns the Data Processing layer, since it mainly relies on AI implementations, a set of alternative (or complementary) AI-specific tools is here presented: the entire macro-layer can be embodied in the execution of a pipeline toolbox such as StreamPipes [36] or, in case of server-hosted implementation, StreamSets [37]. Both the toolboxes provide drag-and-drop user interfaces for low (or no) code integration. AI-based modules are suggested to be built on runtime execution tools, in order to be hosted on the catalogue modules and to be executed in the AI Engine module: some suggested library sets are Tensorflow [38], its alternative Pytorch [39], and Caffe (a deep learning framework specifically designed for image analysis, developed by Berkeley AI Research) [40]. When implementing Tensorflow-based AI engines, interfaces can be deployed in Keras, which provides modular interfaces for Neural Networks (NN) [41]. This module can be framed in the AI Interfaces sub-layer. D2Lab, as a solution to detect unusual service behaviours, can be deployed in the Data Exploration layer [42].

In the lowest part of the architecture, no reference tool is given for the Execution Layer and the External Systems layer, since the low position in the layered architecture and the high variety of data sources would lead to a cumbersome catalogue. The implementation policy for execution layers and external systems relies on the usage of drivers and firmware provided with the data sources and effectors, which will be interfaced with Data Ingestion and Transformation layer and Adapter layer, while, if Orion Context Broker is selected as per Data Brokering & Persistence, IDAS Agents (the IoT connectors of the FIWARE catalogue) can be easily deployed to interface the most-used industrial IoT protocols such as OPC UA, MQTT and LoRa [43].

Apart from Data in Motion and Data at Rest layer, which represents the raw data sources and do not require any implementation, Digital Models and Vocabularies can be represented through Asset Administration Shell (AAS), a de facto standard strongly linked to RAMI4.0 [44], or another object-oriented modelling language (such as AutomationML [45]); on the other side of the architecture, several tools are available to implement cyber-security functionalities, but, in case of usage of FIWARE catalogue tools in other modules, KeyRock, Wilma, and AuthZForce respectively for token-based rights, proxy functions and authentication/identity validation [46].

5. Conclusion and further works

The architecture described in the present work represents a reference framework to add self-X abilities to a CIM/RAMI4.0 compliant software architecture. In particular, with respect to the CIM pyramid of Fig. 1a, the proposed layers are supposed to gather data from the assets (Field-PLC levels), implement filtering and routing tasks (SCADA level) and contribute with knowledge and actions at a production level (MES level). In a similar way, for what concerns the layers of the RAMI4.0 model of Fig. 1b, the proposed architecture functionalities cover the layers from the Asset

to the Functional one, while, in the right-handed one, it ranges from the Field Device to the Enterprise levels, but with the possibility of including the Connected world in case the AI Pipelines are intended to leverage on a federated learning [47] approach (e.g., if the AI Engines are trained with data coming from analogue processes in different physical plants).

The paper is however supposed to be interpreted as a positioning work and, in its implementation part, does not pretend to tie eventual deployments to the set of presented tools. Also, the modules described in the single layers of the architecture have been defined under the perspective of an exhaustive solution to the shopfloor digitalization problem, but the actual implementation in factories' informative systems could diverge depending on specific requirements and contextual limitations (some modules can hence be added, some other ones could not be implemented). Additionally, the approach used in this design phase has been driven by a deployment-agnostic policy, leaving to the deployers the opportunity to implement the different layers on local or cloud servers.

Furthermore, the architecture here presented depicts a monolithic structure, which is a well-established format for the manufacturing community, but as highlighted in the scientific debate [24], a more decentralized approach would be more intrinsically coherent with Self-X functionalities. This approach is however an intended choice by the authors, who wanted to encourage the implementation of MAPE-K approaches in the industrial community (positioning indeed Self-X functions on the edge level would have led to an ensemble of service-specific microcontrollers which could be hard to accept).

A final open issue that this article is not covering is the security aspect: despite in Section 5 some security tools are provided, these modules are network-related packages, intended to provide protection against untrusted applications. The autonomous computing implementation, given its adaptability, requires however a series of specific measures to prevent itself form deviating from its designed behaviour [48], with serious consequences on the safety aspects. Despite several attacks have already targeted Autonomous Computing modules (mainly compromising sensors) [49], the implementation of specific counter-measures is still on a maturity level too low to be adopted in industrial environments [49].

Being a positioning paper, this work lacks implementation evidence: despite relying on established practices to integrate the different components no Key Performance Indicators are assessed. As per other previous works by the authors [50], a further research is supposed to take place, implementing the existing architecture in different industrial domains, in order to evaluate the performances of the implemented architecture with respect to existing production and network infrastructures.

Additionally, further investigations will concern the experimentation of the proposed Self-X architecture in industrial pilot cases that present a non-trivial complexity: for instance in the steel industry that represents a case where process anomalies need to be distinguished in a precise way with respect to other conventional production outliers (e.g. normal product changes related to quality degradations) require Self-X capabilities in order to coordinate that knowledge that is widespread in geographically distributed production areas whose data-flows are by construction each other asynchronous. In this case therefore the implementation of the Self-X architecture appears naturally to be experimented in the MES layer of the CIM pyramid being not only the most flexible in terms of interoperability but also the first layer from the bottom having the capability to coordinate the data-fluxes generated by different production areas.

Acknowledgements

This work has been supported by the project "self-X Artificial Intelligence for European Process Industry digital transformation" (s-X-AIPI), which has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101058715.

References

[1] Rüßmann M, Lorenz M, Gerbert P, Waldner M, Justus J, Engel P, et al. Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries. Business and Information Systems Engineering 2015. https://doi.org/10.1007/s12599-014-0334-4.

[2] The Road to Integration: A Guide to Applying the ISA-95 Standards in Manufacturing, Second Edition. IsaOrg n.d. https://www.isa.org/products/the-road-to-integration-a-guide-to-applying-th-1 (accessed June 29, 2023).

[3] IEC. IEC 62264-2: Enterprise-control system integration – Part 2: Objects and attributes for enterprise-control system integration. IEC 62264-2 Standard 2015. https://doi.org/10.1002/asia.201300717.
[4] Kannan SM, Suri K, Cadavid J, Barosan I, Van Den Brand M, Alferez M, et al. Towards Industry 4.0: Gap Analysis between Current Automotive MES and Industry Standards Using Model-Based Requirement Engineering. 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), 2017, p. 29–35. https://doi.org/10.1109/ICSAW.2017.53.
[5] Zezulka F, Marcon P, Vesely I, Sajdl O. Industry 4.0 – An Introduction in the phenomenon. IFAC-PapersOnLine 2016;49:8–12. https://doi.org/10.1016/j.ifacol.2016.12.002.
[6] Heinze R, Manzei C, Schleupner L. Industrie 4.0 im internationalen kontext: kernkonzepte, ergebnisse, trends. Beuth Verlag; 2017.
[7] Quadrini W, Negri E, Fumagalli L. Open interfaces for connecting automated guided vehicles to a fleet management system. Procedia Manufacturing 2020;42:406–13. https://doi.org/10.1016/j.promfg.2020.02.055.
[8] Nucera DD, Quadrini W, Fumagalli L, Scipioni MP. Data-Driven State Detection for an asset working at heterogenous regimens. 17th IFAC Symposium on Information Control problems in Manufacturing, IFAC-PapersOnLine; 2021, p. In print.
[9] Carvajal Soto JA, Tavakolizadeh F, Gyulai D. An online machine learning framework for early detection of product failures in an Industry 4.0 context. International Journal of Computer Integrated Manufacturing 2019;32:452–65. https://doi.org/10.1080/0951192X.2019.1571238.
[10] Polenghi A, Cattaneo L, Macchi M, Pasanisi D, Pesenti V, Borgonovo A. Development of an advanced condition-based maintenance system for high-critical industrial fans in a foundry. IFAC-PapersOnLine 2022;55:48–53. https://doi.org/10.1016/j.ifacol.2022.04.168.
[11] Cattaneo L, Fumagalli L, Macchi M, Negri E. Clarifying Data Analytics Concepts for Industrial Engineering. IFAC-PapersOnLine 2018. https://doi.org/10.1016/j.ifacol.2018.08.440.
[12] Liu B. Lifelong machine learning: a paradigm for continuous learning. Front Comput Sci 2017;11:359–61. https://doi.org/10.1007/s11704-016-6903-6.
[13] van de Ven GM, Tolias AS. Three scenarios for continual learning 2019. https://doi.org/10.48550/arXiv.1904.07734.
[14] Salis A, Marguglio A, De Luca G, Razzetti S, Quadrini W, Gusmeroli S. An Edge-Cloud based Reference Architecture to support cognitive solutions in Process Industry. Procedia Computer Science 2023;217:20–30. https://doi.org/10.1016/j.procs.2022.12.198.
[15] Tavola G, Caielli A, Taisch M. An "additive" architecture for industry 4.0 transition of existing production systems. Studies in Computational Intelligence, 2020. https://doi.org/10.1007/978-3-030-27477-1_20.
[16] Gamer T, Hoernicke M, Kloepper B, Bauer R, Isaksson AJ. The autonomous industrial plant – future of process engineering, operations and maintenance. Journal of Process Control 2020;88:101–10. https://doi.org/10.1016/j.jprocont.2020.01.012.
[17] Sanz R, López I, Bermejo J, Chinchilla R, Conde RP. SELF-X: THE CONTROL WITHIN. IFAC Proceedings Volumes 2005;38:179–84. https://doi.org/10.3182/20050703-6-CZ-1902.01071.
[18] Mo F, Monetti FM, Torayev A, Rehman HU, Mulet Alberola JA, Rea Minango N, et al. A maturity model for the autonomy of manufacturing systems. Int J Adv Manuf Technol 2023;126:405–28. https://doi.org/10.1007/s00170-023-10910-7.
[19] Kephart JO, Chess DM. The vision of autonomic computing. Computer 2003;36:41–50. https://doi.org/10.1109/MC.2003.1160055.
[20] Prenzel L, Steinhorst S. Towards Resilience by Self-Adaptation of Industrial Control Systems. 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), 2022, p. 1–8. https://doi.org/10.1109/ETFA52439.2022.9921597.
[21] Feng H, Gomes C, Gil S, Mikkelsen PH, Tola D, Larsen PG, et al. Integration Of The Mape-K Loop In Digital Twins. 2022 Annual Modeling and Simulation Conference (ANNSIM), 2022, p. 102–13. https://doi.org/10.23919/ANNSIM55834.2022.9859489.
[22] Müller T, Kamm S, Löcklin A, White D, Mellinger M, Jazdi N, et al. Architecture and knowledge modelling for self-organized reconfiguration management of cyber-physical production systems. International Journal of Computer Integrated Manufacturing 2022;0:1–22. https://doi.org/10.1080/0951192X.2022.2121425.
[23] Malburg L, Hoffmann M, Bergmann R. Applying MAPE-K control loops for adaptive workflow management in smart factories. J Intell Inf Syst 2023;61:83–111. https://doi.org/10.1007/s10844-022-00766-w.
[24] Lemos R de. Software Self-adaptation and Industry: Blame MAPE-K. 2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2023, p. 88–9. https://doi.org/10.1109/SEAMS59076.2023.00021.
[25] Michele P, Fallucchi F, De Luca EW. Create Dashboards and Data Story with the Data & Analytics Frameworks. In: Garoufallou E, Fallucchi F, William De Luca E, editors. Metadata and Semantic Research, Cham: Springer International Publishing; 2019, p. 272–83. https://doi.org/10.1007/978-3-030-36599-8_24.
[26] Zacheilas N, Kalogeraki V, Zygouras N, Panagiotou N, Gunopulos D. Elastic complex event processing exploiting prediction. 2015 IEEE International Conference on Big Data (Big Data), 2015, p. 213–22. https://doi.org/10.1109/BigData.2015.7363758.
[27] Zamora-Izquierdo MA, Santa J, Martínez JA, Martínez V, Skarmeta AF. Smart farming IoT platform based on edge and cloud computing. Biosystems Engineering 2019;177:4–17. https://doi.org/10.1016/j.biosystemseng.2018.10.014.
[28] Velasquez W, Tobar-Andrade L, Cedeno-Campoverde I. Monitoring and Data Processing Architecture using the FIWARE Platform for a Renewable Energy Systems. 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, p. 1383–7. https://doi.org/10.1109/CCWC51732.2021.9376026.
[29] Vikash, Mishra L, Varma S. Performance evaluation of real-time stream processing systems for Internet of Things applications. Future Generation Computer Systems 2020;113:207–17. https://doi.org/10.1016/j.future.2020.07.012.
[30] Kotliar M, Kartashov AV, Barski A. CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language. GigaScience 2019;8:giz084. https://doi.org/10.1093/gigascience/giz084.
[31] da Cruz MAA, Rodrigues JJPC, Lorenz P, Solic P, Al-Muhtadi J, Albuquerque VHC. A proposal for bridging application layer protocols to HTTP on IoT solutions. Future Generation Computer Systems 2019;97:145–52. https://doi.org/10.1016/j.future.2019.02.009.
[32] Le Noac'h P, Costan A, Bougé L. A performance evaluation of Apache Kafka in support of big data streaming applications. Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017, vol. 2018- Janua, Institute of Electrical and Electronics Engineers Inc.; 2017, p. 4803–6. https://doi.org/10.1109/BigData.2017.8258548.
[33] Kashyap M, Sharma V, Gupta N. Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. Procedia Computer Science 2018;132:1611–8. https://doi.org/10.1016/j.procs.2018.05.126.
[34] Dobbelaere P, Esmaili KS. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations:

Industry Paper. Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, New York, NY, USA: Association for Computing Machinery; 2017, p. 227–38. https://doi.org/10.1145/3093742.3093908.
[35] Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials 2015;17:2347–76. https://doi.org/10.1109/COMST.2015.2444095.
[36] StreamPipes | Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems n.d. https://dl.acm.org/doi/10.1145/2675743.2776765 (accessed June 29, 2023).
[37] Goss R, Subramany L. Journey to a Big Data Analysis Platform: Are we there yet? 2021 32nd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), 2021, p. 1–7. https://doi.org/10.1109/ASMC51741.2021.9435701.
[38] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A system for large-scale machine learning, 2016, p. 265–83.
[39] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. vol. 32, 2019.
[40] Deep Learning Tools and Libraries | SpringerLink n.d. https://link.springer.com/chapter/10.1007/978-3-031-01821-3_8 (accessed June 29, 2023).
[41] Grattarola D, Alippi C. Graph Neural Networks in TensorFlow and Keras with Spektral [Application Notes]. IEEE Computational Intelligence Magazine 2021;16:99–106. https://doi.org/10.1109/MCI.2020.3039072.
[42] Stojanovic L, Stojanovic N. PREMIuM: Big data platform for enabling self-healing manufacturing. 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), 2017, p. 1501–8. https://doi.org/10.1109/ICE.2017.8280060.
[43] Armando N, Sinche S, Rodrigues A, Silva JS, Boavida F. IoT Management Services: A Comparative Assessment of Popular FIWARE Agents. 2022 IEEE Ninth International Conference on Communications and Electronics (ICCE), 2022, p. 81–6. https://doi.org/10.1109/ICCE55644.2022.9852047.
[44] Quadrini W, Cimino C, Abdel-Aty TA, Fumagalli L, Rovere D. Asset Administration Shell as an interoperable enabler of Industry 4.0 software architectures: a case study. Procedia Computer Science 2023;217:1794–802. https://doi.org/10.1016/j.procs.2022.12.379.
[45] Drath R, Luder A, Peschke J, Hundt L. AutomationML - the glue for seamless automation engineering. 2008 IEEE International Conference on Emerging Technologies and Factory Automation, 2008, p. 616–23. https://doi.org/10.1109/ETFA.2008.4638461.
[46] Munoz-Arcentales A, López-Pernas S, Pozo A, Alonso Á, Salvachúa J, Huecas G. Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE. Sustainability 2020;12:3885. https://doi.org/10.3390/su12093885.
[47] Yang Q, Liu Y, Chen T, Tong Y. Federated Machine Learning: Concept and Applications. ACM Trans Intell Syst Technol 2019;10:12:1-12:19. https://doi.org/10.1145/3298981.
[48] Marshall A, Jahan S, Gamble R. Toward evaluating the impact of self-adaptation on security control certification. Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, New York, NY, USA: Association for Computing Machinery; 2018, p. 149–60. https://doi.org/10.1145/3194133.3194139.
[49] Pekaric I, Groner R, Witte T, Adigun JG, Raschke A, Felderer M, et al. A systematic review on security and safety of self-adaptive systems. Journal of Systems and Software 2023;203:111716. https://doi.org/10.1016/j.jss.2023.111716.
[50] Quadrini W, Galparoli S, Nucera DD, Fumagalli L, Negri E. Architecture for Data Acquisition in Research and Teaching Laboratories. Procedia Computer Science 2021;180:833–42. https://doi.org/10.1016/j.procs.2021.01.333.