Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

An industrial agent-based customizable platform for I4.0 manufacturing systems

Alejandro López[a,*], Oskar Casquero[a], Elisabet Estévez[b], Aintzane Armentia[a], Darío Orive[a], Marga Marcos[a]

[a] Systems Engineering and Automatic Control Department, University of the Basque Country (UPV/EHU), Bilbao, Spain
[b] Electronics and Automation Engineering Department, University of Jaén, Jaén, Spain

A R T I C L E I N F O

Keywords:
Industrial agents
Design patterns
Asset administration shell (AAS)
I4.0 Platform
RAMI 4.0

A B S T R A C T

The fourth industrial revolution paradigm places value on the management of information related to the manufacturing process. Reference Architectural Model for Industrie 4.0 (RAMI 4.0), proposed by Plattform Industrie 4.0 (I4.0), provides a starting point for the development of I4.0 systems, based on: (1) international standards organized in a cubic model; (2) a set of key concepts to define the system participants, called I4.0 components; and (3) a list of infrastructure services required to manage I4.0 components and support them in the execution of manufacturing applications. However, the terms in which RAMI 4.0 is stated are generic and neutral from a technological point of view, without actually providing support for the practical development of concrete management platforms (I4.0 Platforms). This work contributes an I4.0 platform for the manufacturing domain that offers the infrastructure services required to manage an I4.0 system. The objective is to bring Industry 4.0 closer to companies, bridging the existing gap by providing a platform aligned with RAMI 4.0 which also offers tools and resources to facilitate the development of I4.0 components. To that end, this I4.0 platform is based on industrial agents, which have an inherent ability to negotiate and cooperate with each other and address the integration of assets. The applicability of the proposed I4.0 platform is evaluated by means of a testing scenario.

1. Introduction

The fourth industrial revolution paradigm places value on the management of information related to the manufacturing process, whose origin can be in the factory itself or come from any other point in the value chain (suppliers, clients, other actors involved in the process, etc.) (Schwab, 2016). Governmental institutions and private organizations around the world have been working over the last decade to define standards and concepts that regulate and contextualize the fourth industrial revolution. Reference Architectural Model for Industrie 4.0 (RAMI 4.0), Industrial Internet Reference Architecture (IIRA), Intelligent Manufacturing System Architecture (IMSA), and Industrial Value Chain Reference Architecture (IVRA) are some of the most representative examples of reference architectures (Nakagawa et al., 2021; Fraile et al., 2019; Li et al., 2018). In addition, as the mere provision of technology referenced by these architectures does not guarantee digitization, efforts have also been made to define different methods and models that allow practitioners to design roadmaps for digitization, and analyze their confluence (Sassanelli et al., 2020).

RAMI 4.0, proposed by the German Plattform Industrie 4.0, can be considered the main reference on this topic. RAMI 4.0 was the first reference architecture to be released (2015), and has exerted a strong influence on subsequent reference architectures, which have sought to collaborate or be aligned with RAMI 4.0 (Lin et al., 2017; Alignment Report for Reference Architectural, 2018). RAMI 4.0 is based on a three-dimensional map defined by three axes: (1) Layers, representing the different functional levels present in the implementation of I4.0 systems; (2) Life cycle & value stream, based on the IEC 62890 standard (IEC 0, 6289, 2020), which provides models to describe the operational state of the product; and (3) Hierarchy levels, based on IEC 62264 (IEC, 62264–1, 2013) and IEC 61512 (IEC, 61512–1, 1997) standards, which propose a structure to define how manufacturing systems are organized (DIN SPEC 5, 9134, 2016). This cubic model provides a common understanding to all participants of an I4.0 system, known as I4.0 components. I4.0 components consist of an asset (in this context, any physical or logical entity with value for a company) and an asset

* Correspondence to: Escuela de Ingeniería de Bilbao, Plaza Ingeniero Torres Quevedo 1, 48013 Bilbao, Spain.
E-mail address: alejandro.lopez@ehu.eus (A. López).

https://doi.org/10.1016/j.compind.2023.103859
Received 21 October 2022; Received in revised form 29 December 2022; Accepted 9 January 2023
Available online 19 January 2023

administration shell (AAS) (Glossary, 2021). The AAS is in charge of representing its asset, providing it with communication capabilities and managing the access to its information and functionalities by participants from multiple parties throughout the life cycle of the asset (Wagner et al., 2017).

I4.0 systems are service oriented, i.e., I4.0 components in the system offer services concerning asset functions or data (e.g., executing manufacturing operations, querying information related to asset maintenance or process quality, etc.) by means of an Application Programing Interface (API) provided by their AASs. These services can be combined to compose manufacturing applications. For that reason, Plattform Industrie 4.0 calls these services *application relevant services* (Miny et al., 2022). However, I4.0 systems also require an infrastructure to manage I4.0 components and support them in the execution of manufacturing applications. To that end, in addition to *application relevant services*, Plattform Industrie 4.0 identifies the so-called *infrastructure services*. This paper will focus on the latter.

Infrastructure services are defined as *"software services that are used by different application relevant services or applications in the same way,* e.g., *mediate, enable and support the interaction with and between I4.0 components"* (IEC, 62264–1, 2013), p. 7. They are responsible for access control to *application relevant services*; AAS information management; AAS creation, registration and deregistration; AAS controlled exposure; etc. *Infrastructure services* are classified into two categories: (1) *AAS services*, that can be performed by the AASs themselves (i.e., those concerning AAS management of I4.0 component information and services); (2) *AAS infrastructure services*, that are related to the management of the AAS as a whole (i.e., those intended to create AASs and make them findable to each other). These latter are performed by management platforms for I4.0 systems (hereafter, I4.0 platforms). It can therefore be concluded from the above that I4.0 systems are hybrid by nature, having a centralized part carried out by the I4.0 platform through the *AAS infrastructure services*, and a distributed part developed by the AASs themselves through the *AAS services*.

Despite the efforts made by Plattform Industrie 4.0 to generate documentation related to its cubic model, I4.0 components and infrastructure services, the lack of technological support is holding back the adoption of this reference architecture by companies. This can be seen in studies such as the one conducted in April 2020 by the German National Academy of Science and Engineering (Acatech) (Schuh et al., 2020a), in which it evaluates 70 companies according to its Industrie 4.0 Maturity Index (Schuh et al., 2020b). The results of this study reveal that, at the time of the survey, only 4% of the companies surveyed were engaged in a large-scale implementation of Industrie 4.0 (see subsection 2.1 of reference (Schuh et al., 2020a) for further details).

As for the scientific community, many researchers continue to address the challenges of the fourth industrial revolution without adhering to RAMI 4.0 or other reference architectures (Kovalenko et al., 2022; Tang et al., 2018a). Among those that do, much attention is being paid to the concept of the I4.0 component, and more specifically, the AAS, with papers such as (Sakurada et al., 2022; Ye et al., 2021a; Pribiš et al., 2021). However, as far as the authors know, there is a lack of works that offer a holistic response, covering the joint development of the I4.0 components and the I4.0 platform that manages and supports them. For this reason, this paper presents an I4.0 platform aligned with RAMI 4.0 for the manufacturing domain. This I4.0 platform offers, on the one hand, the *infrastructure services* required to manage an I4.0 system; on the other hand, it provides tools and resources to implement I4.0 components.

For this purpose, it is based on the industrial agent paradigm (Vogel-Heuser et al., 2020; Karnouskos and Leitão, 2017). Industrial agents (or simply agents, for short) are software entities with an inherent capability to compete and/or collaborate with each other to achieve their goals. This negotiation and decision-making capabilities make industrial agents naturally meet the requirements to implement *proactive AASs* (i.e., AASs with the ability to interact as equals among themselves, offering and requesting services when necessary to meet their goals) (Miny et al., 2022). They also stand out for naturally addressing the integration of assets, which is fundamental in I4.0 systems and is supported by international standards (IEEE Recommended Practice for Industrial Agents, 2021). Furthermore, agents have their own Agent Communication Language (ACL), developed by the Foundation for Intelligent Physical Agents (FIPA). ACL incorporates the use of mechanisms such as ontologies and performatives, which facilitates the definition of different scenarios to ensure interoperability in the system. In addition, agents can also fit other protocols and build well-known APIs such as REST.

Specifically, the I4.0 platform presented in this work contributes a core, formed by a set of agents responsible for providing the *AAS infrastructure services* (e.g., creation and registration of AASs, system state management, etc.). In addition to this core, another set of agents intended to implement *proactive AASs*, taking advantage of their features (distributed intelligence, decision making capabilities etc.). These agents implement the *AAS services* (i.e., support the interaction between I4.0 components and manage access to *application relevant services*). In addition, they include a generic interface for the integration of physical assets and, by extension, for the implementation of *application relevant services*, so they can be customized without requiring extensive knowledge of industrial agents. In this way, the user is provided with a base that:

- Guarantees interoperability between I4.0 components without the need to master the industrial agent paradigm.
- Enables the development of customized I4.0 components by extending the agents provided within the platform with the required *application relevant services*.

The rest of the article is structured as follows: Section 2 discusses different management platforms for the manufacturing domain, analyzing what infrastructure they offer and what they lack when it comes to supporting the development of customized I4.0 systems. Section 3 outlines the design research method followed in this work. Section 4 describes the proposed I4.0 platform, showing how it is structured to deliver the infrastructure services required by RAMI 4.0, and how it supports the development of I4.0 components. Section 5 presents the case study used for testing the I4.0 platform, and Section 6 discusses the results obtained from these tests. Finally, Section 7 will present the conclusions of the paper, as well as the authors' future work.

2. Related work

This section analyzes the literature about management platforms for the manufacturing domain. The objective is to check how they organize factory management, what services they offer, and whether their solutions are closed or they facilitate their extension and customization according to the users' needs. The characteristics of industrial agents make them suitable for implementing these manufacturing management platforms, as evidenced by the volume of literature published on the subject. For that reason, the analysis has focused mainly on industrial agent-based solutions.

Within this group, there are still works that are ad hoc, meaning they do not adhere to any reference architecture. In most of these works, the management of the manufacturing process relies almost exclusively in two types of agents (normally referred with these or similar names): Product Agent, dedicated to managing one or several products, and Resource Agent, in charge of representing the assets in the factory. In (Bennulf et al., 2020), Product Agents manage process plans, understood as sequences of skills they require to fulfil their goals. These skills are offered by the Resource Agents, who fulfill their designated demands (specific, parameterized skill requests), collaborating with each other if necessary. PROSA architecture (Van Brussel et al., 1998) also proposes using Product Agents and Resource Agents, and incorporates Order

Agents for handling scheduling and logistical aspects. This work also considers incorporating Staff Agents to supervise the system and assist the other agents with expert knowledge. ADACOR architecture (Leitao and Restivo, 2006) follows a similar approach and proposes a Product Agent, Task Agent and Operational Agent corresponding to the Product Agent, Order Agent and Resource Agent in PROSA, respectively. In addition, ADACOR includes a Supervision Agent that provides coordination and optimized scheduling capabilities. Another example is CASOA architecture (Tang et al., 2018b), where there are four different agents: Product Agents, Machine Agents, Conveying Agents, in charge of managing the manufacturing process; and Suggestion Agents, whose function is to assign order tasks to Product Agents and subsequently suggest updates based on data processing in the cloud. All these approaches have in common that they identify the basic entities present in a manufacturing system and define agents associated with them. Thus, the management of the system is based on the interactions between these agents for the supply and demand of services (generally with Product Agents requesting the operations needed to meet their goals and Resource Agents satisfying the requests). However, these works do not point to any additional infrastructure or the one they have is only focused on task planning. This makes system management dependent only on the partial system information that each agent has, with no possibility of checking whether this is outdated or incorrect.

Other papers present solutions that go deeper into the functionalities that support the management of the manufacturing process, with services such as component creation and registration (Peres et al., 2018). proposes the IDARTS framework, which consists of several parts, including an agent-based manufacturing system. This system has a data model that registers, among other things, plant topology information concerning the existing assets and their relevant information. This information is periodically checked by a set of Deployment Agents, which launch and kill agents as required to match the plant topology (Munkelt and Krockert, 2018). presents a concept based on up to eight different types of agents divided into transient agents (linked to task execution) and persistent agents (alive during the whole process). The latter include agents that do not intervene directly in the process but provide infrastructure: the system agent (in charge of registering production plans and launching them by creating the necessary agents) and the directory agent (manages a directory of the agents available in the system). Similarly, the MAS-RECON architecture proposes a series of agents, called system supervisory agents, that manage the creation and registration of agents associated with system resources and applications (Gangoiti et al., 2022). MAS-RECON is not focused on the manufacturing domain, but it provides tools to customize the resource and application to a specific domain by means of information models. These works go a step further, proposing functionalities and means that are essential to support the management of the manufacturing process. However, their main drawback is that they do not follow RAMI 4.0 or any of the reference architectures for the fourth industrial revolution, which makes interoperability with other systems difficult.

Among the works that are aligned with RAMI 4.0, many focus on the concepts of AAS and I4.0 Component from different points of view. For instance, (Sakurada and Leitão, 2020) focuses on the implementation of the AAS, identifying the following challenges: supporting interoperability, interconnecting AASs and assets, and providing distributed intelligence to I4.0 components. Similarly, (Baumgärtel and Verbeet, 2020) proposes the use of industrial agents to implement the AAS, obtaining what they have called Active Component Shell. In turn, (Contreras et al., 2017) proposes an implementation of industrial agent-based AASs based on three types of agents (resources, products and coordinators), as well as which technologies to use to implement the different layers of the cubic model (OPC for the communication layer, FDI for the information and functional layers, and AutomationML for end-to-end engineering). These works are interesting because they identify the features of I4.0 components and which technologies fit with them, but they do not go into detail on how to develop I4.0 components or what services they should offer.

Other authors go a step further with the implementation of their AAS concepts (Arm et al., 2021). proposes a procedure for facilitating the creation of AASs by means of a configuration wizard. In this proposal, two basic types of AASs are defined, service requesters and service providers, with specific functionalities. However, they share a common code that supports bidding, structured access to data, and data logging. In contrast, (Cavalieri and Salafia, 2020) focuses on modeling AASs to represent PLCs programmed according to the IEC 61161–3 standard. This work proposes to implement AASs as OPC UA nodes, accompanied by a web application that allows to perform AAS creation and update services manually. These works present AAS implementations with specific services, accompanied by complementary infrastructure, conceived to create or update AASs manually. However, they do not contemplate an active and automated management of AASs in this infrastructure.

In other works, I4.0 platform proposals are proposed to provide infrastructure to support I4.0 components. The work presented in (Leitão et al., 2016) proposes an architecture for managing manufacturing reconfiguration. This architecture is based on a network of heterogeneous components interconnected by a middleware that guarantees their interoperability. To this end, the middleware offers functionalities such as service registration and exposure, exception handling and data persistence, etc. In (Trunzer et al., 2019) the requirements for I4.0 architectures are defined based on the analysis of three research projects that approach the topic from different perspectives: data analytics and security, service discovery, and real-time capability. As a result, an architecture is proposed that combines elements from each of the projects to meet all the requirements. These works corroborate that there is a real need to develop architectures for the management of I4.0 systems and indicate what aspects to focus on. However, after identifying these needs, they do not indicate how to implement their architectures to address them.

In contrast, there are works that present specific proposals oriented to different kinds of services (Pisching et al., 2018). proposes an architecture based on RAMI 4.0 layers that focuses on the discovery of manufacturing assets. For this purpose, two databases are proposed: Virtual Entity, which keeps a virtual representation of any physical asset in the system, and the Hierarchical Equipment Network, which manages two tables: one that indicates the operations available in the system, and another that relates the operations that each machine can do and their priority. These priorities are used as a fixed criteria to designate operations (Ye et al., 2021b). features an architecture comprising three layers: field assets, edge AAS deployment and cloud AAS management. In this approach, AASs are manually modeled and then transformed into OPC UA nodes that can exchange data with each other for realizing application-specific functions. AASs are managed through two cloud applications: a web browser to view and edit AAS submodels and a user interface to monitor the data acquired from the assets. These two works offer, from different perspectives, functional I4.0 platforms to address the problems they propose to solve. However, although they show implementation examples, they do not seem to provide guidelines or templates from which to use their approach.

In (Cruz Salazar et al., 2019), design patterns (i.e., categories of agents with a specific name and functionality) are proposed to implement an industrial agent-based architecture aligned with RAMI 4.0. Specifically, in this proposal, the AAS is interpreted as a combination of agents that play specific roles in the different layers of RAMI 4.0. This work is reviewed and extended in the MARIANNE architecture (Cruz Salazar and Vogel-Heuser, 2022), focused on providing a solution aligned with both RAMI 4.0 and the main standards concerning industrial agents (IEEE Recommended Practice for Industrial Agents, 2021; VDI/VDE, 2021). In contrast, in (López et al., 2021) a skeleton pattern, based on a single agent, is proposed as the basis for implementing all the AAS of the system. The paper defines functionalities and a state machine for the skeleton pattern, and gives examples of how to extend it to develop different design patterns. These papers do not focus so much on the services to be offered by the I4.0 components and infrastructure, but instead give more detail on what kind of components should make up the system.

To sum up, these works have been compared considering several aspects: (1) the compliance with RAMI 4.0, (2) the focus of the works (components, infrastructure, or both), (3) the kind of patterns for implementing system components (fixed or customizable) if any is proposed, and (4) the *infrastructure services* offered (the *application relevant services* are not considered, since they depend on the domain). As a reference, the *infrastructure services* considered necessary by Plattform Industrie 4.0 itself have been taken into account (IEC, 62264–1, 2013), Fig. 18. In this sense, in those works not aligned with RAMI 4.0, services with functionalities equivalent to those required by Plattform Industrie 4.0 have been considered. Table 1 summarizes the analysis of related works.

3. Research method

Systems Engineering is dedicated to the realization of successful systems by documenting requirements, analyzing functionality, designing architectures, developing and validating prototypes, and commissioning the solutions while considering all business and technical concerns in the process (Caillaud et al., 2016). The research method in this area of knowledge can be experimental, design, empirical, analysis, or a combination of these methods (Muller, 2013). This work follows a design research method, as the authors design a software artifact, in this case a platform, in support of the management of manufacturing systems at plant level.

As part of the design research approach, a literature review was conducted focusing on the management platforms for the manufacturing domain available in the literature and their compliance with RAMI 4.0, considered the main reference on this topic. As it can be seen in Table 1, none of the reviewed approaches meet all the requirements, either because they lack some or many functionalities, or because they offer functionalities equivalent to those required, but they are not framed within RAMI 4.0. In order to fill this gap, this work presents an I4.0 platform for manufacturing systems aligned with RAMI 4.0. This platform is based on the industrial agent paradigm, leveraging its intrinsic capabilities to not only provide the required infrastructure to support the management of manufacturing systems, but also to provide a solid foundation from which to develop *proactive AASs*. This approach aims to provide a platform that offers the infrastructure proposed by RAMI 4.0 to ensure the scalability and interoperability of the system, while helping practitioners by offering them a base from which to develop concrete solutions with no need to be experts in industrial agents.

To do so, this platform provides a core infrastructure that implements the *AAS infrastructure services* and supports the creation and registration of new AASs on the fly at the request of an operator or other AASs in the system. In addition, the platform also provides a set of agents devoted to implementing the AASs in the system. These agents serve to a double purpose: on the one hand, they implement the *AAS services* to ensure a seamless understanding between the I4.0 platform and the I4.0 components (granted by their AASs); on the other hand, they have been designed in such a way that they can be extended and customized according to the needs of each use case. To this end, they have a generic interface to integrate physical assets, enabling access to their information and functionalities (i.e., allowing the execution of the *application relevant services*).

The proposed platform is based on the MAS-RECON architecture (Gangoiti et al., 2022). MAS-RECON was not designed in accordance with RAMI 4.0, but it has the potential to manage service-oriented components with an application-centered approach. Therefore, the generic core provided by MAS-RECON (e.g., state machine-based lifecycle management, service-oriented interactions) has been adapted to fit RAMI 4.0. In addition, new functionalities have been added to meet the requirements of I4.0 systems (e.g., manufacturing resource integration). Besides MAS-RECON, this work also picks up some of the most interesting concepts from other works, such as the division between transient and persistent agents (Munkelt and Krockert, 2018) and the use of design patterns (Cruz Salazar et al., 2019; Cruz Salazar and Vogel-Heuser, 2022).

Finally, after presenting the proposed platform and justify its compliance with RAMI 4.0, the work is completed with an evaluation of the design in real-world settings in the context of an industrial application. The testing follows a case study approach (Aberdeen, 2013), where evidence of the functionality and data of the performance of a prototype of the platform are provided.

4. Industrial agents for manufacturing systems I4.0 platform

This section presents the Industrial Agents for Manufacturing Systems (IAMS) I4.0 platform. This platform provides a central infrastructure that implements the *AAS Infrastructure Services* and allows the creation of new AASs both by the operator and by other AASs (*AAS Create Service*), keeping an updated state of the whole system in an internal repository (*AAS Registry Services*). In addition, the platform also provides a set of agents dedicated to implement the AASs in the system based on design patterns that can be extended and customized according to the needs of each use case. As explained above, this platform is based on the MAS-RECON architecture (Karnouskos and Leitão, 2017), Fig. 1. MAS-RECON is neither characterized for the manufacturing domain nor aligned with RAMI 4.0, but it provides the System Supervisory Agents (SSAs), which present some useful features for managing I4.0 systems. It also offers two templates from which to develop the remaining agents of the architecture, which can be either Resource Agents (RAs) or Application Agents (AppAs). In addition, MAS-RECON provides a set of tools and resources to facilitate domain customization. This includes, on the one hand, the extension or addition of more SSAs if needed, and, on the other hand, the characterization of the types of RAs and AppAs and how they interact with each other.

The remainder of this section is divided into two parts. The first one presents an overview of the IAMS I4.0, describing all the agents that compose it. The second one is devoted to explaining the process to develop I4.0 components.

4.1. IAMS I4.0 platform overview

Fig. 1 shows the IAMS I4.0 platform. From bottom to top, it can be seen how it is built in terms of implementation: first, the Java Agent DEvelopment Framework (JADE[1]) provides basic classes and methods for the creation of agents and ensures compliance with FIPA (Standard Status Specifications, 2022) standard for agent communication; next, the MAS-RECON core, formed by the agents highlighted in black, provides support and supervision functionalities for the management of the I4.0 system; finally, the IAMS customization, made up of the agents in green, provides the user with (1) an interface to interact with the I4.0 system and (2) a set of agent categories with specific names and infrastructure functionalities (hereafter, design patterns) from which to develop customized AASs.

From left to right, instead, the agents are organized according to their functionalities. In the first group are the System Supervisory Agents, which comprise the core of the IAMS I4.0 platform (see Fig. 1). They are in charge of managing and supervising the process from a global point of view, providing both functional and computational support. Moreover, their implementation is independent of the I4.0 components that the system will have.

Within this group, the most prominent agent is the System Repository Agent (SRA). It is in charge of managing the System Repository

[1] https://jade.tilab.com/

Table 1
Summary of related work analysis.

Proposal	RAMI 4.0 compliance	Focus	Patterns	I4.0 Infrastructure Services (or equivalent)					
				AAS Infrastructure Services			AAS Services		
				AAS Create	AAS Registry	AAS Exposure and Discovery	Submodel Registry	Meta Information Management	Submodel Exposure and Discovery
(Bennulf et al., 2020)	No	C	Fixed	No	No	No	Ms	Ms	Yes
(Van Brussel et al., 1998; Leitao and Restivo, 2006; Tang et al., 2018b)	No	C+I	Fixed	No	No	Yes	Yes	Yes	Yes
(Munkelt and Krockert, 2018; (Peres et al., 2018)	No	C+I	Fixed	Yes	Yes	Yes	Yes	Yes	Yes
(Gangoiti et al., 2022)	No	C+I	Customizable	Yes	Yes	Yes	Yes	Yes	Yes
(Sakurada and Leitão, 2020; Baumgärtel and Verbeet, 2020)	Yes	C	Not specified	No	No	No	No	No	No
(Contreras et al., 2017)	Yes	C	Fixed	No	No	No	No	No	No
(Arm et al., 2021)	Yes	C	Fixed	Ms	No	No	Yes	Yes	Yes
(Cavalieri and Salafia, 2020)	Yes	C	Not specified	Ms	No	No	Mp	Mp	Yes
(Leitão et al., 2016; Trunzer et al., 2019)	Yes	C+I	Not specified	No	No	No	No	No	No
(Pisching et al., 2018)	Partially	I	Fixed	No	Yes	Yes	No	No	Yes
(Ye et al., 2021b)	Yes	C+I	Not specified	Ms	Yes	Yes	Mp	Mp	Yes
(Cruz Salazar et al., 2019; Cruz Salazar and Vogel-Heuser, 2022)	Yes	C+I	Fixed	Ms	No	Yes	Yes	Yes	Yes
(López et al., 2021)	Yes	C	Customizable	No	No	No	Yes	Yes	Yes

C = Focus on components, I = Focus on infrastructure, C+I = Focus on components and infrastructure; Ms = Manually, before startup; Mp = Manually, during process.

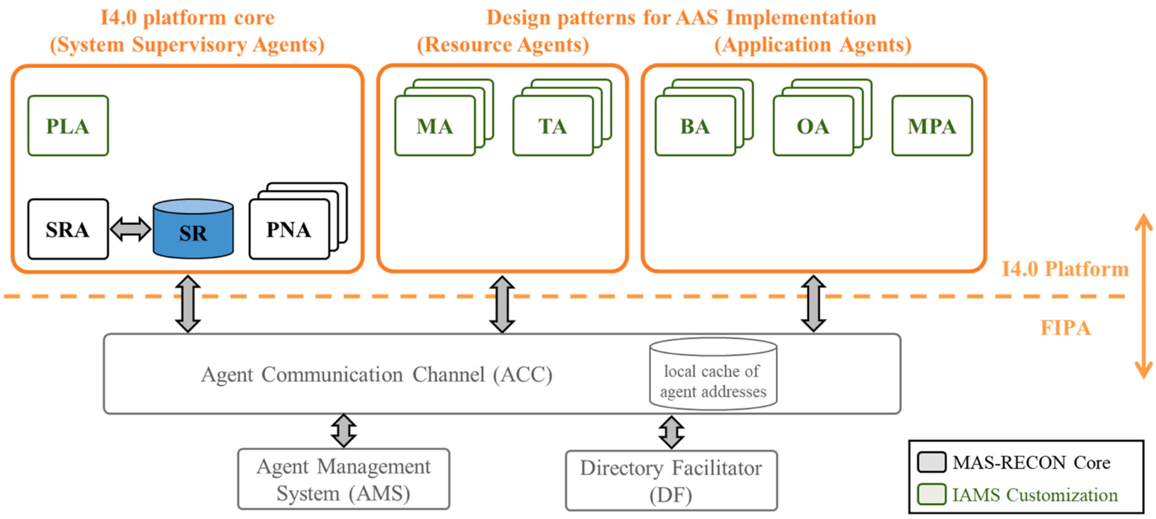
I4.0 platform core
(System Supervisory Agents)
Design patterns for AAS Implementation
(Resource Agents)
(Application Agents)
PLA
SRA
SR
PNA
MA
TA
BA
OA
MPA
I4.0 Platform
FIPA
Agent Communication Channel (ACC)
local cache of agent addresses
Agent Management System (AMS)
Directory Facilitator (DF)
MAS-RECON Core
IAMS Customization

Fig. 1. IAMS I4.0 platform overview.
Adapted from (Karnouskos and Leitão, 2017), Fig. 1. SRA = System Repository Agent, SR = System Repository, PLA = Planner agent, PNA = Processing Node Agent, MA = Machine Agent, TA = Transport Agent, BA = Batch Agent, OA = Order Agent, MPA = Manufacturing Plan Agent.

(SR), which contains essential information about all the I4.0 components running in the system. Thus, putting the *infrastructure services* considered by Plattform Industrie 4.0 into perspective (see (IEC, 62264–1, 2013), Fig. 18), the SRA is responsible for providing the *AAS infrastructure services*: when creating and registering I4.0 components, it is responsible for assigning them a unique identifier and registering their information in the SR (thus providing the *AAS create service* and the *AAS registry services*). The SRA can also query, edit and delete I4.0 components from the SR. In addition, it can manage access to this information: any participant in the system can access the information of another I4.0 component by asking for its identifier or if it meets some specific required characteristic (attributes, services offered, etc.). This functionality corresponds to the *AAS exposure and discovery services*.

To complement the SRA, the Planner Agent (PLA) has been developed. The PLA serves as an interface for the user, allowing the use of the *AAS infrastructure services* offered by the SRA. In this way, a user of the IAMS I4.0 platform can create and register I4.0 components, and consult their state individually or jointly by means of the PLA.

Finally, the Processing Node Agent (PNA) aims to represent the processing nodes available in the system, ensuring proper awareness and access to the system's computing capacity. PNAs can negotiate with each other based on different criteria (e.g., available memory, CPU load, etc.) to offer computation and communication capabilities to the agents in the system.

On the other hand, the RAs and AppAs are intended to implement the AASs of the I4.0 system. There are two main differences between these groups of agents: the type of asset they integrate and their life cycle. RAs integrate physical assets, while AppAs integrate logical assets. This implies a difference in the way of accessing the data and functionalities offered by the asset: in the case of RAs, integration is limited by the communicative capabilities of their assets, while in AppAs integration can usually be done directly. As for the life cycle, the assets represented by RAs can take part at different manufacturing applications when requested, providing the services required by the applications to meet their objectives. For this reason, RAs are persistent in nature, i.e., once they are created, they remain in the system indefinitely unless there is a problem, or they are removed on purpose. Instead, AppAs represent assets whose life is linked to a specific task within a manufacturing application. Therefore, AppAs are transient in nature, i.e., once their task is completed, they have no other function in the system. Beyond these differences, specific RAs and AppAs have been defined for the manufacturing domain. For this purpose, the works analyzed in Section 2 have been taken as a reference.

With respect to RAs, the simplest approach consists on defining a single agent type for any manufacturing asset (Bennulf et al., 2020; Munkelt and Krockert, 2018), although a higher granularity can be considered (e.g., Robot Agents (Kovalenko et al., 2022), Conveyor Agents (Kovalenko et al., 2022; Tang et al., 2018b), or AGV Agents (Tang et al., 2018a), among others). Having these works in mind, a balance has been sought between being too generic (considering all factory resources as one) and being too specific (defining too many types). As a result, in this work manufacturing resources are classified in two basic types: machines (i.e., resources that transform products) and transports (i.e., resources that move products). Consequently, two design patterns have been developed from the RA template: the Machine Agent (MA) and the Transport Agent (TA). Regarding the AppAs, manufacturing applications are usually narrowed to product orders, so that the Product Agent (PA) is the only agent of this type considered (Kovalenko et al., 2022; Bennulf et al., 2020; Tang et al., 2018b). Although these approaches allow unitary traceability, (López et al., 2020) presents a complementary perspective that facilitates tracking the status of complex orders and the factory in general. Hence, three application entities (batch, order and manufacturing plan) have been defined for tracking the manufacturing process at different levels, each one of them with their respective design pattern: the Batch Agent (BA) manages traceability at the product or batch level; the Order Agent (OA) tracks at the customer level (i.e., monitoring all the batches that are part of the same customer order); finally, the Manufacturing Plan Agent (MPA) monitors the development of the manufacturing plan as a whole. These application entities (and, by extension, their design patterns) have a hierarchical relationship, so that one or more BAs report information to a single OA, and the existing OAs report to a single MPA.

As for the compliance of these design patterns with RAMI 4.0, they are the basis provided by IAMS for the user to develop AASs capable of integrating manufacturing assets. Note that I4.0 components (or, more specifically, their AASs) should implement, on the one hand, the *AAS services* required to manage interoperability between AASs (access control to *application relevant services*, negotiation mechanisms between AASs, etc.) and, on the other hand, the *application relevant services* to give access to assets' functions and data. The implementation of the *AAS services* has to be the same for all I4.0 components to ensure interoperability between them and with the core of the I4.0 platform. For this reason, *AAS services* are already implemented in the set of RAs and AppAs presented in this work, leveraging the negotiation and decision-making capabilities inherent to industrial agents. On the contrary, the *application relevant services* cannot be implemented in advance since they will depend on the communication and functional capabilities of the asset to be integrated. However, RAs and AppAs do include interfaces to help the user to integrate physical assets in a generic way, giving access to functions and data to develop customized *application relevant services*.

4.2. Development of I4.0 components

This subsection illustrates how to develop I4.0 components from the RA and AppA templates provided by MAS-RECON. This process consists of three steps. First, the RA and AppA templates have been adjusted for the IAMS I4.0 platform to fit the manufacturing domain. Next, these templates have been extended to develop the design patterns (the MA and TA from the RA template, and the BA, OA, and MPA from the AppA template, respectively). This step covers the implementation of the *AAS services*, and their use to support the interactions of these design patterns. Finally, the last step is to develop I4.0 components from the design patterns. This step requires customization by the user, since the characteristics of the assets to be integrated and the *application relevant services* they offer are particular to each case. Nevertheless, this work shows the set of tools and mechanisms provided by IAMS to assist the user in this step.

4.2.1. Fitting Mas-Recon templates to the manufacturing domain

The templates provided by the MAS-RECON architecture for RAs and AppsAs are classes that provide basic methods (e.g., for communicating with the SRA) and implement the finite state machines (FSM) depicted in (Karnouskos and Leitão, 2017) Fig. 4. Each state of these FSMs is implemented by means of one or several behaviors (i.e., a class that defines the actions performed by an agent depending on its interactions). In the case of the RA template, the FSM proposed in MAS-RECON consists of three states: a *booting state* that performs the initialization tasks implemented in a *boot behavior*; a *running state* composed of two behaviors, one to participate in negotiations for service allocation (*negotiating behavior*) and one to manage the allocated service requests (*running behavior*); and a *stopping state* that performs the finalization tasks implemented in an *end behavior*. This RA template has been fitted in IAMS for the manufacturing domain by applying the following changes:

- A new *asset management behavior* has been included in the *running state*. This behavior manages the interactions with the physical assets of the factory in a homogeneous way. It allows decoupling the management of service requests (processed by the *running behavior*) from the interactions with the physical asset necessary to fulfil the services (handled by the *asset management behavior*).
- In addition, a new *idle state* has also been included. This state represents situations in which the I4.0 component is alive, but its asset is unavailable (e.g., in case of a mechanical breakdown or a maintenance stop). The *idle state* can be reached from the *running state*, and it can transition either to the *running state* or to the *stopping state*. The actions corresponding to this state can be defined by means of an *idle behavior*.

Fig. 2 shows the resulting FSM for the RA template proposed in the IAMS I4.0.

As for the AppA template, it presents the three states already seen in the RA template (*booting*, *running* and *stopping*), plus two other states used for replica management (tracking and waiting for decision), which are not of interest for this work. In this case, it has been considered that it is not required to add further states or behaviors with respect to the FSM provided by MAS-RECON.

4.2.2. Developing the design patterns from the Ra and Appa templates

In this step, RA and AppA templates are extended to obtain the MA, TA, BA, OA and MPA design patterns, which allow implementing the AASs of the I4.0 components of the system. To this end, specific content has been given to the states and behaviors from the FSMs discussed in the previous subsection. These design patterns provide the *AAS services* and support SSAs in the execution of the *AAS infrastructure services* (see (IEC, 62264–1, 2013), Fig. 18).

In the case of the MA and TA-based AASs, they receive at booting all the information related to their asset from a configuration file. This ensures the fulfilment of the *submodel registry services*. This is followed by registration in the system, as illustrated in Fig. 3 for the case of the MA.

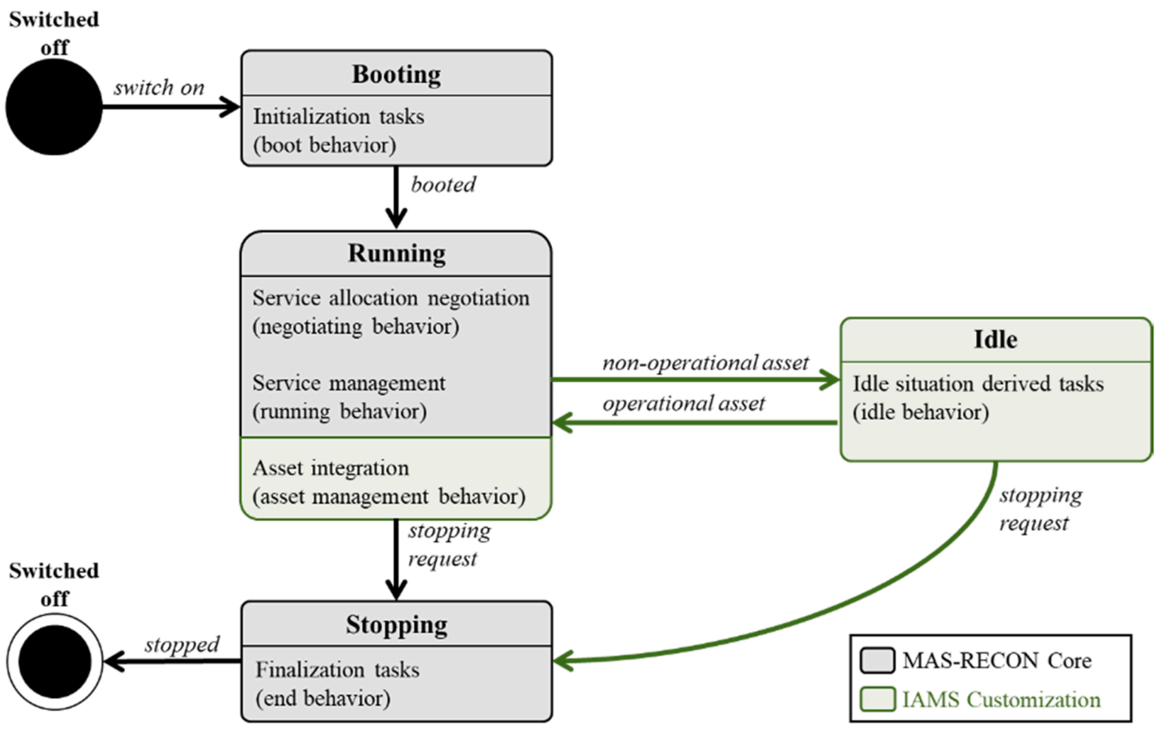
Switched off
switch on
Booting
Initialization tasks (boot behavior)
booted
Running
Service allocation negotiation (negotiating behavior)
Service management (running behavior)
Asset integration (asset management behavior)
non-operational asset
operational asset
Idle
Idle situation derived tasks (idle behavior)
stopping request
stopping request
Switched off
stopped
Stopping
Finalization tasks (end behavior)
MAS-RECON Core
IAMS Customization

Fig. 2. FSM for the RA template proposed in the IAMS I4.0 platform.

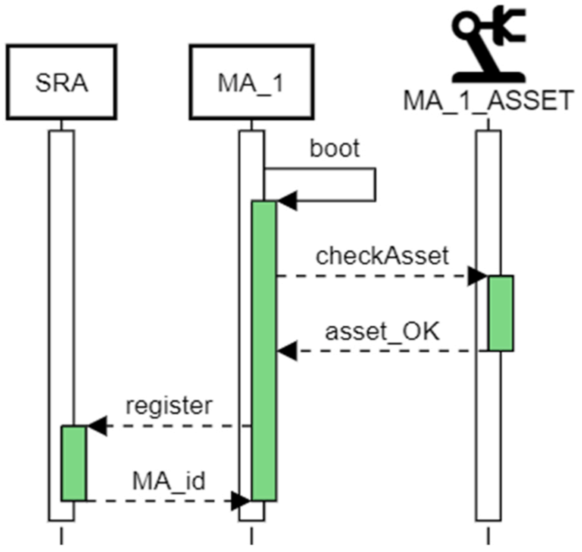

Fig. 3. Sequence diagram representing the interactions of the MA design pattern during the booting state.

As a precondition, it is checked the availability of the physical asset by sending it a message (*checkAsset* in Fig. 3). In case there is not response, or it indicates the asset is not ready to work, the registration is aborted. Otherwise, the SRA is requested to assign a unique identifier to the new AAS and to register its information in the SR (*register* in Fig. 3). In addition to registration, the user can add the required initialization tasks in the *boot behavior*.

Once startup is completed, MA and TA-based AASs enter the *running state*. From this point on, they should be ready to handle requests for *application relevant services*. These requests can be directed to a specific AAS (e.g., to inquire about the status of its asset, such as its current temperature or speed), or they can be the result of a negotiation (e.g., the requester requires an *application relevant service*, but it is not aware of which is the most suitable candidate to do so). For the latter case, the *negotiating behavior* implements a negotiation mechanism by which AASs of the same kind (i.e., which integrate the same type of asset) can determine in a decentralized way who should accomplish the requested *application relevant services*. As for the mechanism to calculate the negotiating value of each AAS, it will be strongly dependent on the criteria required by the user. Similarly, the actions to be taken by the winning AAS will depend on the purpose of the negotiation. In both cases, the user will be able to customize these aspects as required thanks to two methods, *calculateNegotiationValue* and *checkNegotiation*, included in the *negotiating behavior*.

Requests for *application relevant services* received by the AAS, either directly or as a result of a negotiation, are managed by the *running behavior*. Specifically, requests concerning *submodel services* (i.e., *application relevant services* which can be performed by retrieving all or part of the information within a submodel) are handled directly. As for requests regarding *asset related services* (i.e., *application relevant services* which require an interaction with the asset), they are registered in a submodel so that they are addressed in order. This implies that this behavior allows an AAS to make its submodels accessible to others (complying with the *exposure and discovery services*).

As for the BA, OA and MPA-based AASs, their functionalities are founded on the same *AAS services*, although they present some differences in their *booting* and *running states*. First, they receive the information with the initial state from a manufacturing plan loaded into the system by an operator through the PLA (*loadPlan* in Fig. 4). Before creating and registering the AAS, the manufacturing plan is checked for conformity to the domain. This is done in two phases: a first one in which each entity of the plan is checked individually (*seRegister* in Fig. 4), and a second one in which it is validated if its hierarchy is correct (*iValidate* in Fig. 4). This process is explained in more detail in (Casquero et al., 2020). If the validation is successful, then the first entity in the hierarchy is started. In our customization of the domain, it is the MPA (*appStart* in Fig. 4). The MPA AAS is deployed on the most suitable processing in terms of system utilization (e.g., available memory, CPU load, etc.). This is determined by a negotiation between PNAs (*negotiate* in Fig. 4). To complete the startup of the entire manufacturing plan, the MPA AAS is

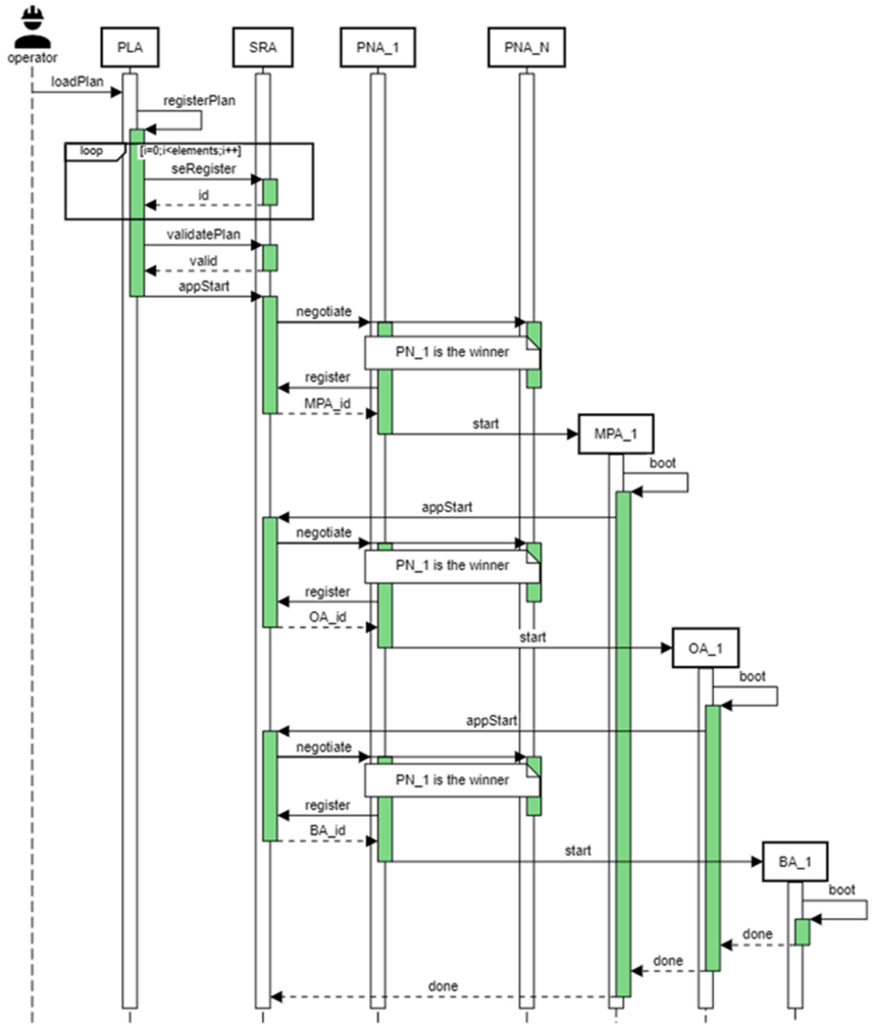

Fig. 4. Sequence diagram representing the interactions between the MPA, OA and BA design patterns during booting state.

responsible for starting the OA AASs that depend on it, and these, in turn, for starting the BA AASs that depend on them. In addition, they all wait to receive a confirmation that their children have been successfully booted before moving to the *running state*. As it can be seen in the lower part of Fig. 4, an OA AAS remains in the *booting state* until it receives confirmation that its children BA AASs have transitioned to the *running state*. The same is true for the MPA AAS as long as it does not receive confirmation from its associated OA AASs.

Once in the *running state*, the BA, OA and MPA-based AASs work in the same way: they update the submodel in which they record the traceability of their application entity based on the information they receive (*traceUpdate* in Fig. 5). Furthermore, as they collect information, they report it to the higher level of the hierarchy (*reportUpdate* in Fig. 5). Finally, when the entity they are tracing has completed its manufacturing process, they transmit their complete traceability submodel and then transition to the *stopping state*, where they are unregistered and removed from the system (*terminate* in Fig. 5).

4.2.3. *Supporting the development of I4.0 components*

The previous step explains how the design patterns proposed in this paper implement AASs that incorporate the *AAS services* and use them to develop their functionality in the *booting* and *running states*. However, to complete the development of I4.0 components, it is necessary for these AASs to integrate their assets to provide *application relevant services*. This last step is responsibility of the user, since the factory assets to use and the *application relevant services* they offer depend on each use case. In this sense, the integration of BA, OA and MPA AASs with their assets is considered direct: being logical assets (in this case, process information at different levels) they can be directly embedded as submodels or deployed in a database with the appropriate access. Thus, this subsection focuses on the tools and mechanisms provided to facilitate the user the development of I4.0 components based on the MA and TA AASs.

Regarding the interactions between AASs and physical assets, there may be cases where different assets offer the same services but have different communication capabilities. This would require several implementations of the same type of AAS for different physical assets. To avoid this, the *JadeGateway* and *GatewayAgent* classes, provided by JADE, have been used to decouple the high-level functionality of the AAS (i.e., management of *application relevant services* and interaction with other AASs) from the communication capabilities of its physical asset. By means of these classes, a gateway is created that allows connecting entities using different communication protocols (Vogel-Heuser et al., 2020). In this case, it serves as a bridge between the AAS, which uses ACL, and the physical asset, which uses one of the communication protocols defined by its manufacturer. Beyond abstracting the asset's communication protocol, the gateway also allows the AAS to be deployed both on and off the asset. This provides users with flexibility in applying generic integration practices, regardless the asset to be integrated and its communication capabilities (López et al., 2022).

In addition, the RA template customization for the manufacturing domain (see subsection 4.2.1) includes an interface with two methods used to standardize interactions between AASs and their gateways:

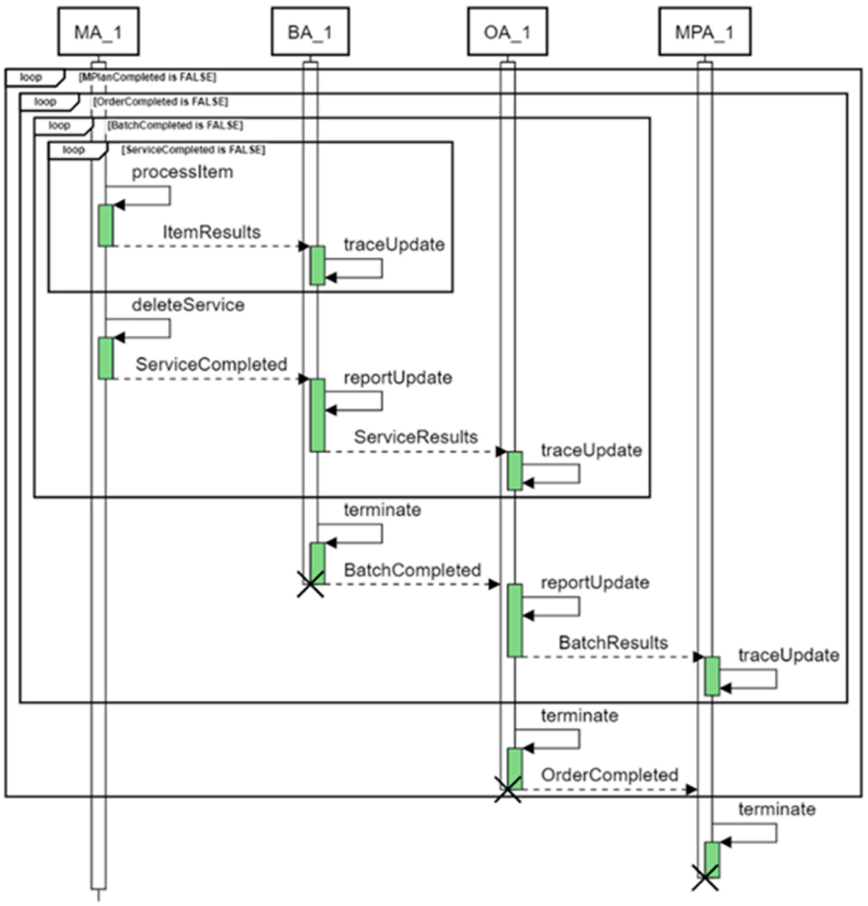

Fig. 5. Sequence diagram representing the interactions between the MPA, OA and BA design patterns during running state.

sendDataToAsset and *rcvDataFromAsset*. These methods, invoked from the *asset management behavior* in the *running state*, work as shown in Fig. 6:

- *sendDataToAsset*, evaluates a work in progress flag to determine whether the physical asset is free. If so, it also checks for pending *asset related services* requests (*opt statement* in Fig. 6). If these conditions are met, information concerning the next *asset related service* request is fetched from the corresponding submodel (*fetchServiceRequestInfo* in Fig. 6) and sent to the gateway. Once the information has been sent, the work in progress flag is activated to block the sending of new information until the service has been completed (*workInProgress=TRUE* in Fig. 6).
- *rcvDataFromAsset* exclusively processes messages received from its gateway. These can be either acknowledgment messages to confirm the physical asset has received some information, or they can contain partial or total results from the execution of an *asset related service*. If the latter is the case, the results received are transmitted to the BA that is tracing the product(s) involved (*processItem* in Fig. 6). In addition, if the service has been completed, its information is deleted from the list of queued *asset related services* and the work in progress flag is deactivated (*workInProgress=FALSE* in Fig. 6).

5. Case study

This section illustrates the applicability of the IAMS I4.0 platform for managing manufacturing processes by deploying it on a demonstrator used as a test bed. The demonstrator is equipped with two manufacturing cells that perform assembly operations of a set of 3D printed parts emulating the shaft of a stepper motor. These assembly operations are processed in batches of variable size, ranging from one to six items. For this purpose, each cell comprises a KUKA KR3 R540 robot, in charge of handling the parts, and a Siemens ET 200SP Open Controller. This controller is made up of a software PLC, which allows controlling the robot, and an integrated Windows 10 operating system, which allows deploying third-party software. In addition to the manufacturing cells, the demonstrator also features a cluster of PCs to provide computing capacity to enable the deployment of the software infrastructure required by the user. This cluster was made up of five nodes. Each node was a Dell Optiplex 760 with Ubuntu 20.04 as the operative system. All nodes were connected through a gigabit switch.

Fig. 7 shows the deployment of the IAMS I4.0 platform in the demonstrator for this testing scenario. On the one hand, one of the cluster PCs has been reserved to host the SRA and the PLA, serving as the interface with the system. The rest of the PCs in the cluster act as processing nodes, each represented by a PNA. These nodes will be used to deploy the BA, OA and MPA-based AASs (for this concrete testing scenario, two BAs, one OA and one MPA, respectively). On the other hand, each MA-based AAS corresponding to the two manufacturing cells have been deployed in the Windows environment of each ET 200SPs. Their gateways have also been deployed on the same devices, and exchange information with their PLC environments through shared memory.

Regarding the metrics, the execution times of the *application relevant services* were not considered, as these depend more on the demonstrator than on the performance of the IAMS I4.0 platform itself. Instead, we focused on evaluating infrastructure measures, specifically the deployment time of the AASs (i.e., the time required for them to register on the I4.0 platform, complete their startup tasks and reach the running state, when they are considered fully operational), to see if it is carried out in a manageable time. In order to have more detail on what happens during

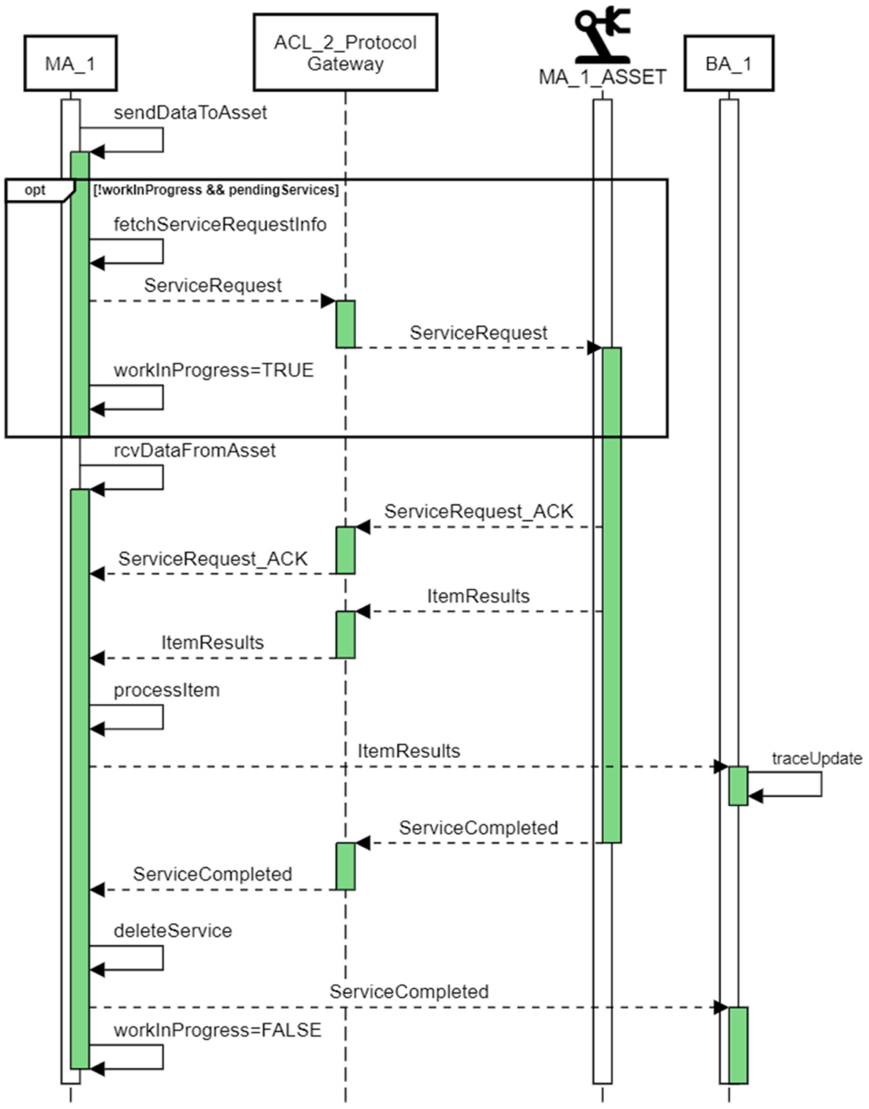

Fig. 6. Sequence diagram representing the operation of the sendDataToAsset and rcvDataFromAsset methods to standardize the interactions between AASs and gateways.

this deployment process, it has been decided to obtain disaggregated measures. These measures will be different depending on the type of AASs, as they follow different startup processes:

- MA and TA-based AASs (illustrated in Fig. 8.a for the case of the MA). First, the timestamp *tr0* is measured at the time of AAS creation. A timestamp *tr1* will then be collected when confirmation of connectivity to the asset is received. Finally, timestamp *tr2* will represent the instant when the AAS enters the running state. With these timestamps, three time intervals were defined: *asset check time* (*tr1-tr0*), *registration time* (*tr2-tr1*) and *deployment time* (*tr2-tr0*).
- BA, OA and MPA-based AASs (illustrated in Fig. 8.b for the case of the MPA). The timestamp *ta0* will indicate the time at which the user requests the deployment of a manufacturing plan through the PLA. This request initiates a negotiation between the PNAs to determine on which processing node to deploy each AAS. The timestamp *ta1* is collected for each AAS in the application when its negotiation process ends (i.e., when they have already been assigned a processing node). Next, timestamp *ta2* is measured at the beginning of the booting state to know that the AAS has already been created, and finally, *ta3* is measured when the AAS enters the running state. With these timestamps, four time intervals were defined: *scheduling time* (*ta1-ta0*), *registration time* (*ta2-ta1*), *booting time* (*ta3-ta2*) and *deployment time* (*ta3-ta0*).

6. Results and discussion

The purpose of this section is to present and discuss the results presented in Table 2, which show the deployment times of all the AAS used in the case study presented in the previous section see Fig. 7). The main interest of this test is in the differences between the deployment times of each agent and in reasoning the causes behind them, rather than in the overall figures (considering that all agents deploy in about 1 s or less). To ensure a correct data fitting distribution, the tests have been performed 20 times.

As can be seen, there is a significant difference between the deployment time used by MA-based AASs (around 125 ms), and BA, OA and MPA-based AASs (ranging from 873 ms for BA_2 to 1003 ms for MPA_1). This difference can be explained as follows: in the case of MA-based AASs, the deployment process is performed in a straightforward manner, without any interactions other than checking that the asset is alive. In contrast, in the case of BA, OA and MPA AASs, the deployment process is conditioned by the negotiation process between PNAs to decide where to deploy each AAS. In addition, the hierarchical

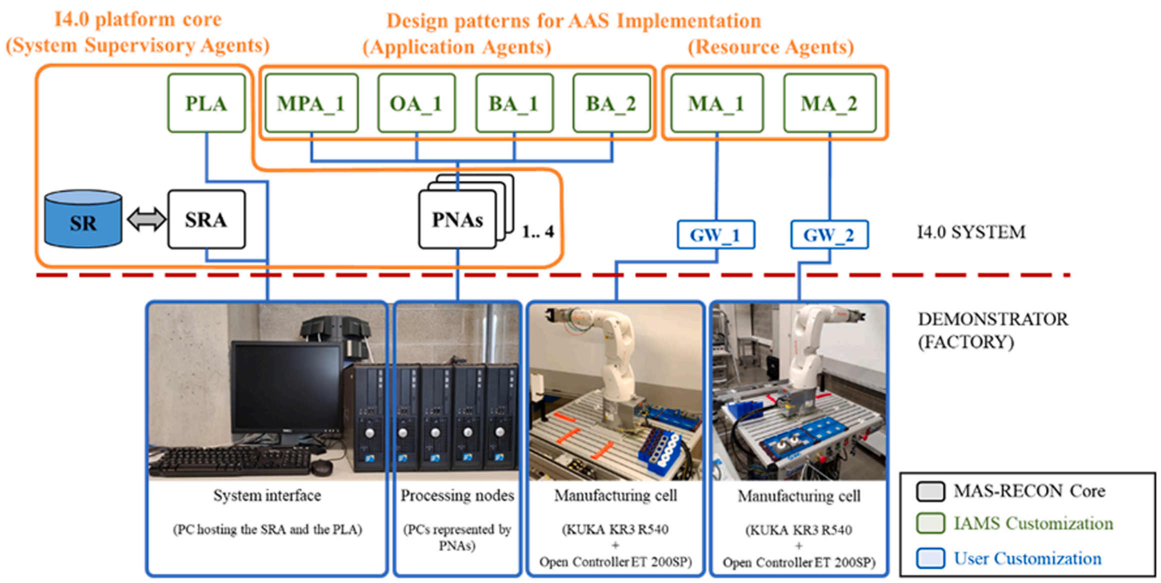

Fig. 7. Diagram of the I4.0 platform deployment on the demonstrator.

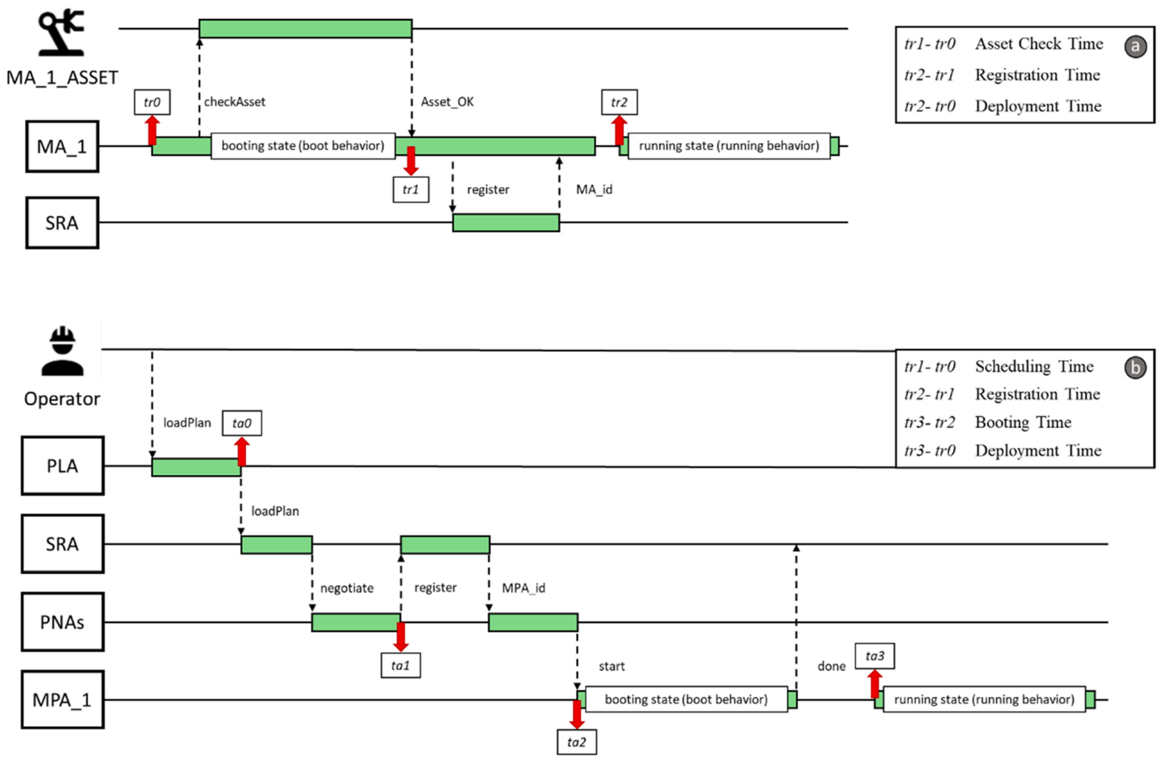

Fig. 8. Timing diagram indicating the instants at which timestamps are captured for: a) AASs based on the MA design pattern; b) AASs based on the MPA design pattern.

Table 2
Average deployment time after conducting the testing scenario (N = 20).

AAS	Average Deployment Time (ms)
MA_1	132
MA_2	119
MPA_1	1003
OA_1	1002
BA_1	924
BA_2	873

relationship between them also conditions their deployment times.

To better understand these differences, Fig. 9 and Fig. 10 break down in detail how the deployment of each type of AAS occurs. Fig. 9 shows the deployment time of the MA-based AASs divided into intervals. Both cases show a similar response, with approximately 30% of the time spent confirming that the asset is accessible, and the remaining time spent registering the AAS in the SR.

As for Fig. 10, it shows the results obtained for BA, OA and MPA AASs, which allow contrasting the deployment procedure detailed in Fig. 4. First, the longest deployment time corresponds to MPA_1 (1003 ms), followed by OA_1 (1002 ms), and finally BA_2 (924 ms) and

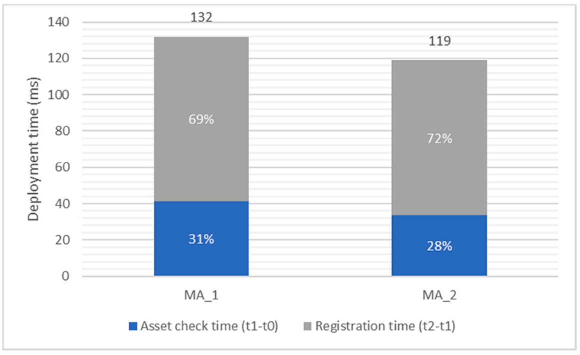

Fig. 9. Disaggregated times of deployment time for MA-based AASs.

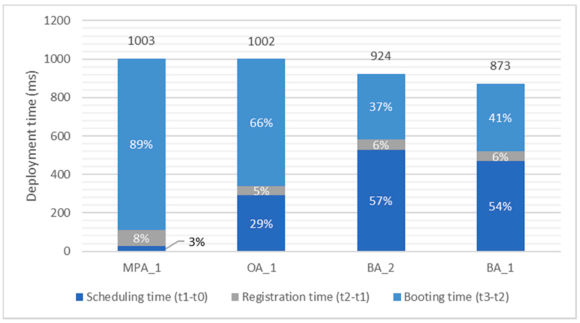

Fig. 10. Disaggregated times of deployment time for BA, OA and MPA-based AASs.

BA_1 (873 ms). This makes sense, since MPA_1 cannot leave its booting state until it receives confirmation from OA_1, and the same happens to OA_1 with respect to BA_2 and BA_1. This cascading booting process also explains the differences between the time intervals for these AASs. In the case of MPA_1, the planning time is minimal (only 3% of the deployment time), since it is the first negotiation to take place, followed by an also short registration time (8% of the total). In contrast, its booting time accounts for 89% of the deployment time, not because it requires this time to perform all its booting tasks, but because it remains waiting to receive confirmation from OA_1. Proof of this is that the average deployment time of MPA_1 is only 1 ms longer than that of OA_1 (as soon as it receives the confirmation message, it transitions to the running state). As the hierarchy descends, the scheduling time increases. This is because the negotiations to determine which node should host which AAS are resolved sequentially, so the elapsed time from *ta0* is increasing (note that *ta0* is the same for OA_1, BA_1 and BA_2 as for MPA_2 but their *ta1* are different). At the same time, the booting times are getting smaller and smaller, since OA_1 only must wait for confirmation from BA_1 and BA_2 and they perform their boot tasks directly.

7. Conclusions

There is active work in the development of platforms based on the reference architectures, but with approaches that are not mature enough to be adopted by the industry, and vice versa. Besides, there are many works that focus on some of the different technologies that, together, can enable I4.0 systems to meet all their requirements. However, most of these works are based on particular approaches, and therefore do not follow reference architectures. For this reason, this work proposes a platform that takes into account the evolution of reference architectures and their key aspects, and includes methodological and technological resources that allow the implementation of customized solutions. These resources, based on the paradigm of industrial agents, provide a solid foundation that allows the platform to be used without the need to have a deep knowledge of the technology. In this way, companies are supported in the transition towards I4.0.

Specifically, the aim of this work is to offer a platform for the management of manufacturing systems aligned with the precepts of the idea of Industrie 4.0 proposed by RAMI 4.0 that also provides mechanisms that facilitate the user the development and customization of the AAS that participate in the system representing the assets of its factory. To this end, it employs the paradigm of industrial agents, which innately contemplates the ability to cooperate between system participants, as well as the integration of assets. In the opinion of the authors, the novelty of this proposal resides in two different aspects. On the one hand, in offering within the I4.0 platform design patterns for the development of customizable AASs. These design patterns implement the *infrastructure services*, but can be customized to offer the *application relevant services* required by the user, which helps reducing the development times of customized I4.0 systems. On the other hand, in the way of providing through the different agents of the I4.0 platform all the infrastructure services required by the Plattform Industrie 4.0.

However, the services indicated by Plattform Industrie 4.0 only contemplate a scenario in which manufacturing runs smoothly. In the authors' opinion, it is essential that the I4.0 platforms have the necessary resources to be able to keep manufacturing applications operational despite breakdowns or delays in any of the factory's assets. Thus, future work will be devoted to increase the functionalities of the I4.0 platform presented in this work to ensure seamless execution of manufacturing applications.

CRediT authorship contribution statement

Alejandro López: Conceptualization, Software, Validation, Writing – original draft. **Oskar Casquero:** Conceptualization, Supervision, Writing – review & editing. **Elisabet Estévez:** Conceptualization, Supervision, Writing – review & editing. **Aintzane Armentia:** Software, Validation, Writing – review & editing. **Darío Orive:** Validation, Resources, Writing – review & editing. **Marga Marcos:** Conceptualization, Supervision, Writing – review & editing. All authors have read and agreed to the current version of the manuscript.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ministerio de Ciencia, Innovación y Universidades (MCIU)/Agencia Estatal de Investigación (AEI)/Fondo Europeo de Desarrollo Regional (FEDER), Unión Europea (UE) (grant number RTI2018-096116-B-I00).

Data Availability

Data will be made available on request.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.compind.2023.103859.

References

Aberdeen, T., 2013. Yin, R. K. (2009). Case study research: Design and methods (4th Ed.). Thousand Oaks, CA: Sage. Can. J. Action Res. 14 (1), 69–71. https://doi.org/10.33524/cjar.v14i1.73.

Alignment Report for Reference Architectural Model for Industrie 4.0/ Intelligent Manufacturing System Architecture. Apr. 2018. [Online]. Available: 〈https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/hm-2018-manufacturing.html〉.

Arm, J., et al., 2021. Automated design and integration of asset administration shells in components of industry 4.0. Sensors 21 (6), 2004. https://doi.org/10.3390/s21062004.

Baumgärtel H. and Verbeet R., Service and Agent based System Architectures for Industrie 4.0 Systems, in NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, Apr. 2020, pp. 1–6. doi: 〈10.1109/NOMS47738.2020.9110406〉.

Bennulf, M., Danielsson, F., Svensson, B., Lennartson, B., 2020. Goal-oriented process plans in a multi-agent system for Plug & Produce. IEEE Trans. Ind. Inform. 1. https://doi.org/10.1109/TII.2020.2994032.

Caillaud, E., Rose, B., Goepp, V., 2016. Research methodology for systems engineering: some recommendations. IFAC-Pap. 49 (12), 1567–1572. https://doi.org/10.1016/j.ifacol.2016.07.803.

Casquero O., A. Armentia, E. Estevez, A. López, M. Marcos, Customization of agent-based manufacturing applications based on domain modelling, in 21st IFAC World Congress, Jul. 2020, vol. 4.

Cavalieri, S., Salafia, M.G., 2020. Asset administration shell for PLC representation based on IEC 61131–3. IEEE Access 8, 142606–142621. https://doi.org/10.1109/ACCESS.2020.3013890.

Contreras, J.D., Garcia, J.I., Diaz, J.D., 2017. Developing of industry 4.0 applications. Int. J. Online Eng. IJOE 13 (10), 30. https://doi.org/10.3991/ijoe.v13i10.7331.

Cruz Salazar, L.A., Vogel-Heuser, B., 2022. A CPPS-architecture and workflow for bringing agent-based technologies as a form of artificial intelligence into practice. Autom 70 (6), 580–598. https://doi.org/10.1515/auto-2022-0008.

Cruz Salazar, L.A., Ryashentseva, D., Lüder, A., Vogel-Heuser, B., 2019. Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. Int. J. Adv. Manuf. Technol. https://doi.org/10.1007/s00170-019-03800-4.

DIN SPEC 91345. 2016.

Fraile, Sanchis, Poler, Ortiz, 2019. Reference models for digital manufacturing platforms. Appl. Sci. 9 (20), 4433. https://doi.org/10.3390/app9204433.

Gangoiti, U., López, A., Armentia, A., Estévez, E., Casquero, O., Marcos, M., 2022. A customizable architecture for application-centric management of context-aware applications. IEEE Access 10, 1603–1625. https://doi.org/10.1109/ACCESS.2021.3138586.

Glossary. 〈https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/Glossary/glossary.html〉 (Accessed Feb. 13, 2021).

IEC 61512–1. 1997.

IEC 62264–1. 2013.

IEC 62890. 2020.

IEEE Recommended Practice for Industrial Agents, 2021. Integration of software agents and low-level automation functions, 26601-2020 IEEE Std 1–43. https://doi.org/10.1109/IEEESTD.2021.9340089.

Karnouskos, S., Leitão, P., 2017. Key contributing factors to the acceptance of agents in industrial environments. IEEE Trans. Ind. Inform. 13 (2), 696–703. https://doi.org/10.1109/TII.2016.2607148.

Kovalenko, I., Balta, E.C., Tilbury, D.M., Barton, K., 2022. Cooperative product agents to improve manufacturing system flexibility: a model-based decision framework. IEEE Trans. Autom. Sci. Eng. 1–18. https://doi.org/10.1109/TASE.2022.3156384.

Leitao, P., Restivo, F., 2006. ADACOR: a holonic architecture for agile and adaptive manufacturing control. Comput. Ind. https://doi.org/10.1016/j.compind.2005.05.005.

Leitão P., J. Barbosa, A. Pereira, J. Barata, A.W. Colombo, "Specification of the PERFoRM architecture for the seamless production system reconfiguration," in IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Oct. 2016, pp. 5729–5734. doi: 〈10.1109/IECON.2016.7793007〉.

Li, Q., et al., 2018. Smart manufacturing standardization: architectures, reference models and standards framework. Comput. Ind. 101, 91–106. https://doi.org/10.1016/j.compind.2018.06.005.

Lin S.-W. et al., Architecture Alignment and Interoperability. Dec. 2017. [Online]. Available: 〈https://www.iiconsortium.org/iic-i40-joint-work.htm〉.

López A., E. Estévez, O. Casquero, M. Marcos, Using industrial standards for modeling flexible manufacturing systems, in 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), Jun. 2020, 1, pp. 41–46. doi: 〈10.1109/ICPS48405.2020.9274785〉.

López A., O. Casquero, M. Marcos, Design patterns for the implementation of Industrial Agent-based AASs, in 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), May 2021, pp. 213–218. doi: 〈10.1109/ICPS49255.2021.9468129〉.

López, A., Estévez, E., Casquero, O., Marcos, M., 2022. A methodological approach for integrating physical assets in industry 4.0. IEEE Trans. Ind. Inform. 1–9. https://doi.org/10.1109/TII.2022.3230714.

Miny T., G. Stephan, T. Usländer, J. Vialkowitsch, Functional View of the Asset Administration Shell in an Industrie 4.0 System Environment. Apr. 13, 2021. Accessed: Jan. 17, 2022. [Online]. Available: 〈https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Functional-View.html〉.

Muller, G., 2013. Systems engineering research methods. Procedia Comput. Sci. 16, 1092–1101. https://doi.org/10.1016/j.procs.2013.01.115.

Munkelt T. and Krockert M., Agent-based self-organization versus central production planning, in 2018 Winter Simulation Conference (WSC), Gothenburg, Sweden, Dec. 2018, pp. 3241–3251. doi: 〈10.1109/WSC.2018.8632305〉.

Nakagawa, E.Y., Antonino, P.O., Schnicke, F., Capilla, R., Kuhn, T., Liggesmeyer, P., 2021. Industry 4.0 reference architectures: State of the art and future trends. Comput. Ind. Eng. 156, 107241 https://doi.org/10.1016/j.cie.2021.107241.

Peres, R.S., Dionisio Rocha, A., Leitao, P., Barata, J., 2018. IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0. Comput. Ind. 101, 138–146. https://doi.org/10.1016/j.compind.2018.07.004.

Pisching, M.A., Pessoa, M.A.O., Junqueira, F., dos Santos Filho, D.J., Miyagi, P.E., 2018. An architecture based on RAMI 4.0 to discover equipment to process operations required by products. Comput. Ind. Eng. 125, 574–591. https://doi.org/10.1016/j.cie.2017.12.029.

Pribiš, R., Beňo, L., Drahoš, P., 2021. Asset administration shell design methodology using embedded OPC unified architecture server. Electronics 10 (20), 2520. https://doi.org/10.3390/electronics10202520.

Sakurada L. and Leitão P., Multi-Agent Systems to Implement Industry 4.0 Components, in 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), Jun. 2020, vol. 1, pp. 21–26. doi: 〈10.1109/ICPS48405.2020.9274745〉.

Sakurada, L., Leitao, P., la Prieta, F.D., 2022. Agent-based asset administration shell approach for digitizing industrial assets. IFAC-Pap. 55 (2), 193–198. https://doi.org/10.1016/j.ifacol.2022.04.192.

Sassanelli, C., Rossi, M., Terzi, S., 2020. Evaluating the smart maturity of manufacturing companies along the product development process to set a PLM project roadmap. Int. J. Prod. Lifecycle Manag. 12 (3) https://doi.org/10.1504/IJPLM.2020.109789.

Schuh G., R. Anderl, R. Dumitrescu, A. Krüger, Industrie 4.0 Maturity Index. Apr. 22, 2020b. [Online]. Available: 〈https://en.acatech.de/publication/industrie-4–0-maturity-index-update-2020/〉.

Schuh G., R. Anderl, R. Dumitrescu, A. Krüger, Using the Industrie 4.0 Maturity Index in Industry. Apr. 22, 2020a. [Online]. Available: 〈https://en.acatech.de/publication/using-the-industrie-4–0-maturity-index-in-industry-case-studies/〉.

Schwab K., The Fourth Industrial Revolution. 2016.

Standard Status Specifications. 〈http://www.fipa.org/repository/standardspecs.html〉 (Accessed Sep. 02, 2022).

Tang, D., et al., 2018a. Using autonomous intelligence to build a smart shop floor. Int. J. Adv. Manuf. Technol. 94 (5), 1597–1606. https://doi.org/10.1007/s00170-017-0459-y.

Tang, H., Li, D., Wang, S., Dong, Z., 2018b. CASOA: an architecture for agent-based manufacturing system in the context of industry 4.0. IEEE Access 6, 12746–12754. https://doi.org/10.1109/ACCESS.2017.2758160.

Trunzer, E., et al., 2019. System architectures for Industrie 4.0 applications: Derivation of a generic architecture proposal. Prod. Eng. 13 (3–4), 247–257. https://doi.org/10.1007/s11740-019-00902-6.

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P., 1998. Reference architecture for holonic manufacturing systems: PROSA. Comput. Ind. 37 (3), 255–274. https://doi.org/10.1016/S0166-3615(98)00102-X.

VDI/VDE. 2021. 2653 Sheet 4: Multi-agent systems in industrial automation – Selected patterns for field level control and energy systems," Feb. 2022.

Vogel-Heuser, B., Seitz, M., Cruz Salazar, L.A., Gehlhoff, F., Dogan, A., Fay, A., 2020. Multi-agent systems to enable Industry 4.0. Autom 68 (6), 445–458. https://doi.org/10.1515/auto-2020-0004.

Wagner C. et al., The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant, in 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Sep. 2017, pp. 1–8. doi: 〈10.1109/ETFA.2017.8247583〉.

Ye, X., Yu, M., Song, W.S., Hong, S.H., 2021a. An asset administration shell method for data exchange between manufacturing software applications. IEEE Access 9, 144171–144178. https://doi.org/10.1109/ACCESS.2021.3122175.

Ye, X., Hong, S.H., Song, W.S., Kim, Y.C., Zhang, X., 2021b. An industry 4.0 asset administration shell-enabled digital solution for robot-based manufacturing systems. IEEE Access 9, 154448–154459. https://doi.org/10.1109/ACCESS.2021.3128580.