2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE) | 979-8-3503-9971-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISIE51358.2023.10227964

A Methodology for Integrating Asset Administration Shells and Multi-agent Systems

Lucas Sakurada*, Fernando De la Prieta†, and Paulo Leitao*‡
* Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal
‡ Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal
† BISITE Digital Innovation Hub, University of Salamanca, Edificio Multiusos I+D+i, 37007, Salamanca, Spain
Email: {lsakurada, pleitao}@ipb.pt, fer@usal.es

***Abstract*—Industry 4.0 (I4.0) is promoting the digitization of industrial environments towards intelligent and distributed industrial automation systems based on Cyber-physical Systems (CPS). Currently, this digitization process is being leveraged by the Asset Administration Shell (AAS), which digitally describes an asset in a standardized and semantically unambiguous form throughout its lifecycle. However, more robust solutions based on autonomous AASs endowed with collaborative and intelligent capabilities, also called proactive AASs, are still in the early stages. In this context, Multi-agent Systems (MAS) are a key enabler to provide the required autonomy, intelligence and collaborative capabilities for the AASs. With this in mind, this paper presents a methodology positioned with respect to the Reference Architecture Model Industrie 4.0 (RAMI4.0) layers, which provides guidelines for integrating AASs and MAS, aiming to support the development of proactive AASs. The applicability of the proposed methodology was tested through the integration of AASs and MAS for a small-scale CPS demonstrator.**

I. Introduction

Modern manufacturing systems are facing strong demands for flexible, reconfigurable and intelligent systems to meet the ever-changing market, characterized by the high-customized, low-cost and high-quality products. To address the mentioned issues, Industry 4.0 (I4.0) is promoting the digitization of industry towards intelligent and distributed industrial automation systems based on Cyber-physical Systems (CPS). CPS enable the design of large-scale systems endowed with intelligent, highly automated and rapidly adaptable functions based on a set of distributed and autonomous entities [1]. In such digitization perspective, the Reference Architecture Model Industrie 4.0 (RAMI4.0) [2] introduces the concept of I4.0 components, a specific CPS category, which enables the integration of physical or logical assets in the I4.0 through their digital representations, the Asset Administration Shells (AASs) [3].

The AAS is perceived as an enabler in developing Digital Twins for industrial applications, describing an asset in a standardized and semantically unambiguous form along its lifecycle and enabling interoperable communication among I4.0 components [3]. Additionally, the AASs offer the digital basis for future autonomous systems, where AASs can collaborate with each other without the human intervention. However, usually these AASs are developed as passive/reactive entities, only storing and providing the asset information without the autonomy and intelligence capabilities to make decisions, with applications based on the intelligent and collaborative AASs still being at an early stage.

In this context, Multi-agent Systems (MAS) [4] are a key enabler to provide the required autonomy, intelligence and collaborative capabilities for the AASs. MAS comprise a set of intelligent, autonomous and cooperative agents, representing physical or logical objects in the system (e.g., open, complex or ubiquitous systems [5]). The agents are able to perceive and act in dynamic environments, as well as interact with each other, e.g., exchanging information in order to self-adapt or perform distributed tasks following interaction strategies, namely collaboration, negotiation and self-organization [4].

Having this in mind, this paper proposes a methodology positioned with respect to the RAMI4.0 layers, which provides guidelines for integrating AASs and MAS, aiming to support the development of proactive AASs. In this perspective, the MAS are responsible for providing the required intelligence and autonomy to the AASs, and at the same time, the AASs provide the standard knowledge representation for the agents. The proposed methodology was applied in a small-scale CPS demonstrator [6] comprised of several fischertechnik based automation assets to verify the applicability of this approach.

The rest of the paper is organized as follows. Section II overviews the related works in the AAS field, particularly MAS-based AAS solutions. Section III presents the proposed methodology to integrate AASs and MAS. Section IV describes a practical example of applying the proposed methodology to a small-scale CPS demonstrator. Finally, Section V rounds up the paper with the conclusions and future work.

II. Digitizing Assets Using AASs

The AAS is a standard digital representation of an asset (i.e., every logical or physical object that needs to be connected to the I4.0 network). Regarding its structure, the AAS comprises several submodels, where the asset information, e.g., characteristics, properties and capabilities, is stored in a standard manner [3]. Recognized by the Plattform Industrie 4.0, the AAS can be classified as passive, reactive or proactive, where the main difference is regarding its interaction pattern and degree of autonomy to make decisions [7].

The passive AAS, specified in [8], acts as a static file that holds the asset information along its lifecycle and can be exchanged digitally across the I4.0 network. On the other hand, the reactive AAS, specified in [7], acts as an Application Programming Interface (API) that enables the online access to the asset information in a technology-neutral way, e.g., via HTTP/REST, MQTT and OPC UA. Although the passive and reactive AASs present great benefits by promoting interoperability across different suppliers' solutions, they are not endowed with autonomy and intelligence to make decisions.

In this regard, the proactive AASs are decision-making entities that interact with each other autonomously to exchange information. In the state-of-the-art, this type of AAS implementation is still in the early stage. However, some works are investigating the viability of using MAS to implement the proactive AAS itself or to extend its functionalities, e.g., data collection, collaborative functions, autonomous decision-support and embedding artificial intelligence (AI) algorithms [9], [10]. Aligned with this research direction, other works propose a pattern for implementing AASs based on industrial agents [11] and an architecture for integrating AASs (implemented as industrial agents) and physical assets [12]. Moreover, the authors in [13] discuss that the asset information described in AAS submodels may be used to create the agent's knowledge representation in a standard manner, which is also demonstrated by the research works developed in [14], [15].

III. Methodology for integrating AASs and MAS

As aforementioned, although there are some research works related to MAS-based AAS approaches, for the better knowledge of the authors, they do not provide a methodology of how to integrate AASs with MAS, considering them as separate entities, i.e., not using agents to implement the AASs, but having agents to complement the already implemented AASs based on the specifications described in [7], [8]. Bearing this in mind, this section proposes a methodology that presents the guidelines related to the main aspects to be considered for the integration of AASs and MAS. The objective is not to present how the implementation should be performed, but to guide developers to create their own solutions considering the technologies, standards and recommended practices. These aspects are not mandatory and others ones can be considered according to the application requirements.

As seen in Figure 1, this methodology is positioned with respect to the RAMI4.0 layers, where MAS are combined with passive/reactive AASs in order to extend their functionalities, including autonomy, intelligence and collaborative capabilities. In this context, the benefits of MAS are more directed to the *functional* and *business layers*, since a passive/reactive AAS commonly cover the others lower layers. For instance, the AAS establishes the transition between the physical and digital worlds (*integration layer*) through the digitization of assets, e.g., manufacturing equipments and enterprise systems (*asset layer*). This digitization process results in the description of the asset information in a standardized and semantically unambiguous form along its lifecycle (*information layer*), and enables an interoperable communication among I4.0 components across the value-added network (*communication layer*).

This not means that an AAS itself or embedded with AI capabilities can not cover all the layers. However, MAS-based approaches are a suitable solution to distribute the intelligence and perform collaborative tasks [4]. In this sense, MAS may provide novel functionalities for the AAS (*functional layer*), encapsulating AI algorithms/data analysis and offer as services, e.g., to perform monitoring, diagnosis and optimization. Moreover, the agents can be specified (through behavior models, interaction protocols, etc.) to perform decision-making based on the company business strategy (*business layer*). For this purpose, agents can be part of artificial societies [16], collaborating with each other and adapting their behavior and configuration, aiming to achieve the company goals, i.e., the overall goals of the society which the agents are part.

Aiming to facilitate the integration of AASs and MAS, Figure 1 (box on the right side) presents the guidelines to be considered to carry out this integration. Although the focus is on the integration between AASs and MAS, the guidance on the asset arrangement and AAS development is also discussed. These aspects are detailed in the following subsections.

A. Analysis of Available Assets

In industry, different arrangements of assets can be considered. In general, an asset may involve multiple auxiliary assets, e.g., a robotic arm integrated with a gripper. In this case, it is possible to have an AAS for each individual asset (robotic arm and gripper) or an AAS for the multiple assets (set formed by the robotic arm and gripper). There are situations where it is more beneficial to have an unique AAS for each individual asset, or others where it can be more adequate to have an unique AAS to manage multiple assets. For instance, assuming an application in which the unique interest is the pick-and-place task, an AAS for the multiples assets may seen as a suitable solution, since the robotic arm and gripper can offer this capability. However, in some scenarios where the robotic arm can be equipped with both the gripper and other tools, e.g., cutting and welding tools, performing different functions, maybe an AAS for each individual asset can be more suitable, allowing to simplify the reconfiguration process. How this association between AASs and assets will be made will depend on the application requirements.

B. Development of AAS and Interface with Agents

The AAS can be developed in several ways. For instance, some approaches adopt AutomationML and OPC UA to provide means to model the asset information and develop an I4.0 communication interface. Furthermore, there are several open source platforms based on the AAS specifications [7], [8] that can be used to develop the AASs, namely FA^{3}ST, AASX Package Explorer & AASX Server, Eclipse BaSyx and Eclipse AAS Model for Java [17]. However, these platforms do not implement proactive AASs.

When deciding which technology to adopt, it is important to choose one that provides means for the agents to be

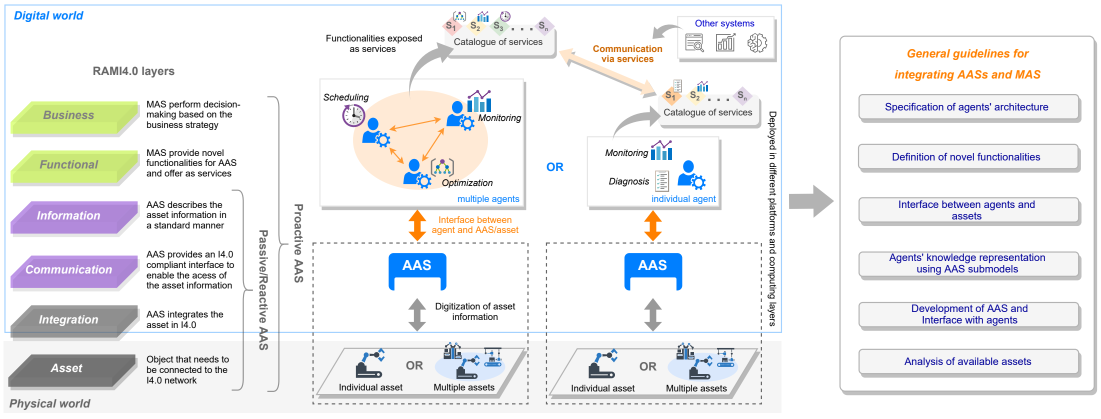
Digital world
RAMI4.0 layers
Business
MAS perform decision-making based on the business strategy
Functional
MAS provide novel functionalities for AAS and offer as services
Information
AAS describes the asset information in a standard manner
Communication
AAS provides an I4.0 compliant interface to enable the acess of the asset information
Integration
AAS integrates the asset in I4.0
Asset
Object that needs to be connected to the I4.0 network
Physical world
Passive/Reactive AAS
Proactive AAS
Functionalities exposed as services
S1 S2 S3 . . . Sn
Catalogue of services
Scheduling
Monitoring
Optimization
multiple agents
Interface between agent and AAS/asset
AAS
Digitization of asset information
Individual asset
OR
Multiple assets
OR
Other systems
Communication via services
S1 S2 . . . Sn
Catalogue of services
Monitoring
Diagnosis
individual agent
AAS
Individual asset
OR
Multiple assets
Deployed in different platforms and computing layers
General guidelines for integrating AASs and MAS
Specification of agents' architecture
Definition of novel functionalities
Interface between agents and assets
Agents' knowledge representation using AAS submodels
Development of AAS and Interface with agents
Analysis of available assets

Figure 1. Methodology for the integration between AASs and MAS positioned with respect to the RAMI4.0 layers.

able to interface with the AASs to get information from the submodels, whether through communication protocols, such as OPC UA, MQTT and HTTP/REST, or even file-exchange formats, e.g., JSON and XML.

C. Agents' Knowledge Representation Using AAS Submodels

The agents' knowledge representation comprises all the information that the agent has about the environment in which it is operating. Based on this information, agents know their goals and how they should make decisions. Considering the large amount of heterogeneous assets in the industry, one of the main challenges is to create the agents' knowledge representation in a simple, standardized and error-prone manner. In this context, the AASs provide standardized information about their assets that can be used as knowledge representation for the agents, where each submodel holds specific information, e.g., asset identification, maintenance needs and capabilities.

It is important to note that there are situations where the AAS is already implemented, but the agent may need more information than the AAS does not provide. In this case, it is necessary to define new submodels according to the needs.

D. Interface Between Agents and Assets

In general, the AAS is designed to represent an asset digitally, only storing the static data (e.g., technical data, identification, maintenance instructions, etc.) of the asset along its lifecycle. However, some applications may also require dynamic data, e.g., operational data from the asset during its operation, or even sending commands to the asset, e.g., to execute a service, requiring a kind of integration between the asset and the AAS.

The heterogeneity of assets in the industry makes complex the development of a generic solution to integrate every asset with its AAS, since the assets have their particularities and proprietary technologies. In this context, a suitable approach is to describe how to perform the integration with the assets in the AAS submodels, e.g., informing which communication protocol to use, the required settings, which methods/services the asset offers, etc.

As discussed in the subsection III-C, the agents are able to use the information from the AAS submodels as knowledge to perform specific tasks. In this case, based on this acquired information (i.e., how to integrate with assets), the agents can be designed to integrate with the assets, aiming to gather operational data or adapt control of physical assets. For this purpose, the IEEE 2660.1-2020 standard [18], which provide recommended practices for the interface of software agents and low-level automation devices, can support this solution.

E. Definition of Novel Functionalities

MAS can offer additional functionalities for the AASs based on data analysis and AI techniques to cover the operational and business levels. The selection of functionalities will depend on the specific needs and requirements of the organization and their assets. For instance, at the operational level, it is common to carry out monitoring and control tasks to ensure the efficiency and safety of the production processes. The monitoring of production processes allow to detect failures or wear that could affect the quality or efficiency of the processes. On the other hand, the control task is related to the process automation. Regarding the business level, the tasks are usually related to simulation, planning and optimization. These tasks enable to predict the behavior of the industrial processes in different scenarios and identify possible problems or opportunities for improvement, e.g., optimizing the energy efficiency, reducing downtimes and improving maintenance/production schedules.

After defining the functionalities that the agents will offer, it is important to define the multiplicity of associations between agents and AASs. One agent for one AAS is a simpler and more straightforward approach, since it is easier to centralize all the functionalities in one agent. However, this approach may not be able to handle the workload or provide the necessary level of flexibility. Multiple agents for one AAS can provide greater flexibility and scalability, as different agents can be specialized to handle specific functions or work in parallel to improve performance. However, this approach also increases the complexity of the system and may require more resources to manage and maintain the multiple agents.

F. Specification of Agents' Architecture

The specification of the agents architecture is a fundamental step in the development of MAS, since it allows clearly defining the behaviors, functions and goals of each agent, and the interactions between them. In this sense, the adoption of agents methodologies, namely Gaia, TROPOS, INGENIAS, etc., can help developers to design and develop agent-based systems for general applications [19]. On the other hand, a common practice when developing MAS for the smart production domain is to define specializations for the agents, their behaviors, ontologies and interaction protocols.

In this methodology, each agent can be associated to an AAS and its asset (individual or multiple). In this regard, a different specialization for the agent can be defined according to the type of asset, e.g., resources and products, increasing the modularity and flexibility of the system. After defining the agents and their types, it is important to define their behavior models, which describes how the agents perceive the environment in which it is placed, process information, make decisions and act to achieve their goals. Moreover, other aspects also need to be considered namely the interaction protocols (i.e., how the agents will interact with each other), the interaction strategy, e.g., based on negotiation, cooperation or delegation, and also an ontology to provide a common vocabulary and taxonomy for agents to communicate.

IV. Experimental Testing

This section aims to present the applicability of the proposed methodology in a small-scale CPS demonstrator based on a fischertechnik infrastructure by integrating the AASs developed for the assets of the demonstrator with the MAS.

A. CPS Demonstrator

As illustrated in Figure 2, the demonstrator comprises several assets, namely two punching machines responsible for performing punching functions, two indexed line machines responsible for performing milling and drilling functions and one industrial robot IRB 1400 ABB to perform the transport of products between the stations. Moreover, there are multiple auxiliary assets, e.g., conveyors, motors, sensors and actuators, which are integrated to the main assets (i.e., robot, punching machines and indexed line machines).

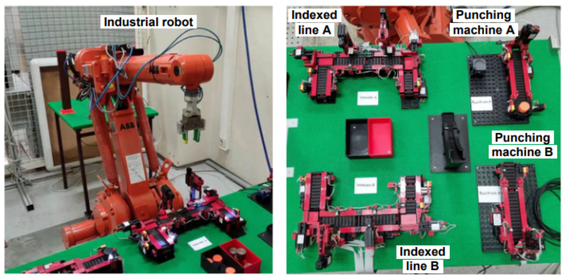

Figure 2. Small-scale CPS demonstrator.

In this example, there are several possibilities for the association between AASs and assets. For instance, an AAS for an auxiliary asset (e.g., motor) could contribute to the analysis of its health condition throughout its lifecycle. Another possibility would be to have an AAS for all the assets in the demonstrator. However, as the interest is to distribute the intelligence and perform collaborative tasks, an AAS was developed for each main asset. Regarding the products, an AAS was developed for each individual product (see Table I).

Table I
Description of assets, AASs and agents in the system.

Asset + AAS	AAS submodel	Agent	Agent role
PMA + AAS_{PMA}	*Capabilities*: information of the capabilities offered by a resource and how to interface with the asset	RA_{PMA}	Offer the capabilities as services
PMB + AAS_{PMB}		RA_{PMB}	Interface with the asset
ILA + AAS_{ILA}		RA_{ILA}	Monitor the health condition of the asset
ILB + AAS_{ILB}		RA_{ILB}	
TR + AAS_{TR}		RA_{TR}	
P_n + AAS_{P_n}	*ManufacturingProcesses*: information of the product production plan; *Location*: information of the product location	PA_n	Manage the product production plan

PMA - punching machine A; PMB - punching machine B; ILA - indexed line A; ILB - indexed line B; TR - transport robot; P - product.

B. Development of the AASs

The AASs were developed based on the specifications described in [7], [8]. Initially, for each asset considered above, a file-based AAS was created using the AASX Package Explorer tool (https://github.com/admin-shell-io/aasx-package-explorer). In this sense, some submodels were defined, namely "ManufacturingProcesses", "Location" and "Capabilities" submodels. The "ManufacturingProcesses" submodel contains information related to the production process, e.g., it is informed that the product is obtained by introducing a raw material in a punching machine and an indexed line. On the other hand, the "Location" submodel provides information regarding where the product is initially located and its final destination after the process. Finally, the "Capabilities" submodel includes information about the capabilities offered by a resource, e.g., punching, drilling and milling.

The information contained in these AASs will be used as a knowledge representation for the agents, and therefore needs to be accessed by the agents. For this purpose, the AASX Server (https://github.com/admin-shell-io/aasx-server) was adopted, which automatically parses these file-based AASs and provides the asset information using several protocols, namely HTTP/REST, OPC UA and MQTT. For instance, by executing a HTTP/REST GET method `/aas/{aas-id}/submodels/ManufacturingProcesses`, it is possible to obtain the information of the "ManufacturingProcesses" submodel.

In this case, the adopted AAS platforms does not provide means to send commands directly to the asset, e.g., start a operation of a resource. Therefore, as the demonstrator already have these functions implemented as services [6], the information of how to invoke these services were included in the "Capabilities" submodel. In this context, the agents are responsible to get this information and perform the interface with the asset.

C. Deployment of the MAS

The MAS system was implemented using the Java Agent DEvelopment (JADE) framework (https://jade.tilab.com/). JADE can be easily deployed in different computational platforms, providing advanced mechanisms to support the development of MAS for general applications.

Based on the demonstrator assets, two types of agents were created, namely product agents (PAs) representing the products, and resource agents (RAs) representing the punching machines, indexed lines and the robot. Regarding to the association between agents and AASs, an agent was defined for each AAS, as the agents are only responsible for performing collaborative tasks and rule-based monitoring, not requiring multiple agents to distribute complex tasks into small simple tasks. Table I presents the agents and their roles in the system.

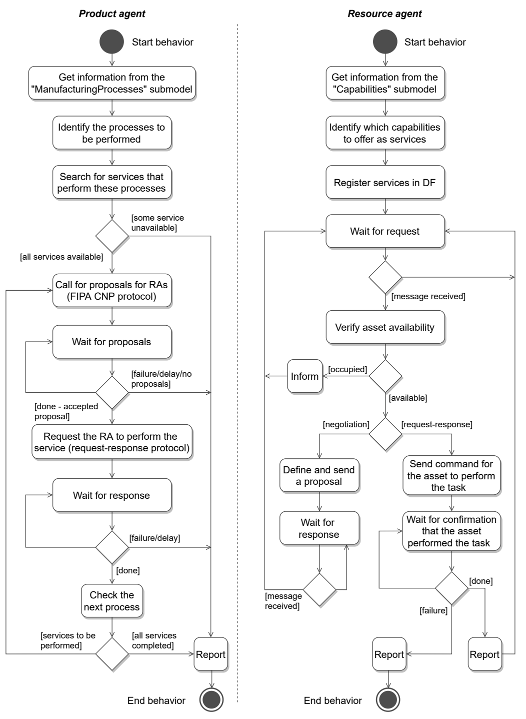
Product agent
Start behavior
Get information from the "ManufacturingProcesses" submodel
Identify the processes to be performed
Search for services that perform these processes
[some service unavailable]
[all services available]
Call for proposals for RAs (FIPA CNP protocol)
Wait for proposals
[failure/delay/no proposals]
[done - accepted proposal]
Request the RA to perform the service (request-response protocol)
Wait for response
[failure/delay]
[done]
Check the next process
[services to be performed]
[all services completed]
Report
End behavior
Resource agent
Start behavior
Get information from the "Capabilities" submodel
Identify which capabilities to offer as services
Register services in DF
Wait for request
[message received]
Verify asset availability
[occupied]
Inform
[available]
[negotiation]
[request-response]
Define and send a proposal
Send command for the asset to perform the task
Wait for response
Wait for confirmation that the asset performed the task
[message received]
[done]
[failure]
Report
Report
End behavior

Figure 3. Behavior model for the product and resource agents.

In this work, the behavior model and interaction between the agents were modeled using the Agent Unified Modeling Language (AUML). For example, Figure 3 illustrates one of the behavior models for the PAs and RAs. These behaviors consist of seeking the information described in the AAS submodels for the agents to have knowledge of their main functions in the system and execute them. The RA uses the information described in the “Capabilities” submodel to provide the capabilities of its associated asset as services and to interface with the asset. On the other hand, PA uses the information described in the “ManufacturingProcesses” submodel to know the production plan of its associated asset and the “Location” submodel to know its initial location and final destination after the execution of all process. From this, agents begin to interact with each other to achieve their goals, following proper interaction protocols.

The defined interaction protocols (see Figure 4) aim to describe the interaction of agents and assets, particularly in the case of the products manufacturing process. In this context, the PA needs to interact with several RAs to fulfill the production plan of its product. However, initially, the PA needs to know which RAs can provide the desired capabilities to produce its associated product. To do so, the RAs need to register the capabilities offered by their assets as services in the yellow pages service. Based on that, the PA can consult the yellow pages and discover which RAs offer the required services.

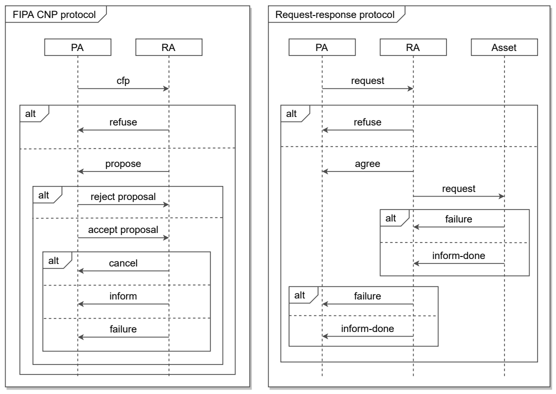

Figure 4. Interaction protocols between agents and assets.

After discovering which RAs can provide the required services, an interaction between PA and RAs is started to negotiate which RAs will execute the operations included in the PA’s production plan. The negotiation strategy was adopted since there are redundant assets in the demonstrator, i.e., more than one asset that can provide the same service. As example, Figure 4 (left) describes the interaction between the PA and RAs following the FIPA Contract Net Protocol (CNP), where the RAs offer proposals and the PA chooses the best one according to pre-defined criteria, e.g., the resource’s price, mean processing time and location.

After the negotiation process, it is defined which RAs will perform the required services to produce the product. In this sense, as illustrated in Figure 4 (right), the PA requests the services for the selected RAs, which in turn will need to interact with the physical assets to execute the process.

D. Experimental Results

In order to verify the applicability of the proposed solution, experimental tests were performed by introducing manufacturing orders in the system. These batch of orders required the same type of product, but in different quantities (ranging from 2 to 32 products), where each product has its own AAS and agent. In this case, the desired product is obtained by introducing a raw material in the punching machine and moving the resulting unfinished product to the indexed line, where the robot carries out the movements between the machines and the warehouses. For each batch of order, the manufacturing lead time (MLT) was measured (i.e., time between the moment of the order request until the end of the process). Figure 5 illustrates that even increasing the number of products, and consequently the number of interactions and negotiations between agents, it was possible to complete all the processes with the MLT increasing proportionally as the number of products increases.

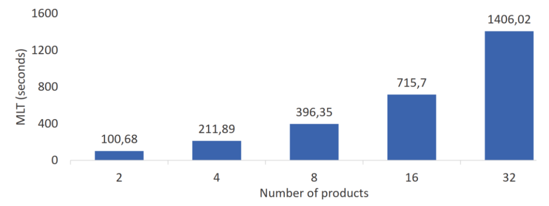

Figure 5. Manufacturing lead time according to the number of products.

Furthermore, a behavior was implemented in the RAs to monitor the health condition of the assets, aiming to detect trends and abnormal situations during the process. This behavior is based on a process control method following the Nelson Rules to determine if a measured variable is under control or not. In this example, the analyzed variable is the processing time of each product (i.e., the period it takes from when a product starts a process on a machine and ends). For instance, rule 1 detects an outlier in the evolution of the measured variable over time and rule 2 identifies a trend in the measured variable. Figure 6 illustrates the achieved results when some deviations in the processing time (i.e., including some delays in the process) are added in one of the machines.

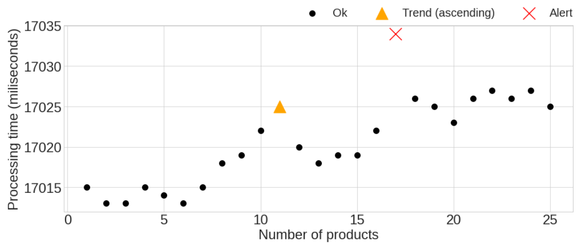

Figure 6. Monitoring behavior to detect unexpected events.

V. Conclusions and Future Work

This paper presented a methodology positioned with respect to the RAMI4.0 layers, which supports the development of proactive AASs, combining AASs and MAS. The proposed methodology defines the main aspects to be considered for integrating AASs and MAS, and its applicability was demonstrated in a small-scale CPS demonstrator.

As a proof of concept, this paper has not focused on providing an extensive formal specification for the agents, presenting just a few examples of how they can be developed. In this sense, future work will be devoted to extend the proposed methodology, particularly providing generic behavior models and collaborative interaction protocols for the agents, considering different scenarios and interaction strategies.

Acknowledgment

The authors are grateful to the Foundation for Science and Technology (FCT), Portugal, for financial support through national funds FCT/MCTES (PIDDAC) to CeDRI (UIDB/05757/2020 and UIDP/05757/2020) and SusTEC (LA/P/0007/2021). The author Lucas Sakurada thanks the FCT for the PhD Grant 2020.09234.BD.

References

[1] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 6–16, 2017.
[2] ZVEI and VDI/VDE, "Reference Architecture Model Industrie 4.0 (RAMI4.0)," 2015.
[3] Plattform Industrie 4.0, "Details of the Asset Administration Shell - from idea to implementation," 2019.
[4] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.
[5] M. V. Dignum, "A Model for Organizational Interaction: based on Agents, founded in Logic," 2004.
[6] A. Lopez, L. Sakurada, P. Leitao, O. Casquero, E. Estevez, F. De la Prieta, and M. Marcos, "Technology-Independent Demonstrator for Testing Industry 4.0 Solutions," in *proceedings of the IEEE International Conference on Industrial Informatics (INDIN'2022)*, 2022, pp. 21–26.
[7] Plattform Industrie 4.0 and ZVEI, "Details of the Asset Administration Shell - Part 2 - Interoperability at Runtime - Exchanging Information via Application Programming Interfaces (Version 1.0RC01)," 2020.
[8] ——, "Details of the Asset Administration Shell - Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01)," 2020.
[9] S. Karnouskos, L. Ribeiro, P. Leitão, A. Lüder, and B. Vogel-Heuser, "Key Directions for Industrial Agent Based Cyber-Physical Production Systems," in *Proceedings of the IEEE International Conference on Industrial Cyber Physical Systems (ICPS'2019)*, 2019, pp. 17–22.
[10] B. Vogel-Heuser, M. Seitz, L. A. C. Salazar, F. Gehlhoff, A. Dogan, and A. Fay, "Multi-agent Systems to Enable Industry 4.0," *at - Automatisierungstechnik*, vol. 68, no. 6, pp. 445–458, 2020.
[11] A. López, O. Casquero, and M. Marcos, "Design patterns for the implementation of Industrial Agent-based AASs," in *Proceedings of the IEEE Conference on Industrial Cyberphysical Systems (ICPS'2021)*, 2021, pp. 213–218.
[12] A. López, O. Casquero, E. Estévez, P. Leitão, and M. Marcos, "Towards the generic integration of agent-based AASs and Physical Assets: a four-layered architecture approach," in *Proc. of the IEEE 19th International Conference on Industrial Informatics (INDIN'2021)*, 2021, pp. 1–6.
[13] B. Vogel-Heuser, F. Ocker, and T. Scheuer, "An approach for leveraging Digital Twins in agent-based production systems," *at - Automatisierungstechnik*, vol. 69, no. 12, pp. 1026–1039, 2021.
[14] S. Jungbluth, J. Hermann, W. Motsch, M. Pourjafarian, A. Sidorenko, M. Volkmann, K. Zoltner, C. Plociennik, and M. Ruskowski, "Dynamic Replanning using Multi-Agent Systems and Asset Administration Shells," in *Proc. of the IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA'2022)*, 2022, pp. 1–8.
[15] L. Sakurada, P. Leitão, and F. De la Prieta, "Engineering a Multi-agent Systems Approach for Realizing Collaborative Asset Administration Shells," in *Proceedings of the IEEE Conference on Industrial Cyber-physical Systems (ICIT'2022)*, 2022, pp. 1–6.
[16] P. Davidsson, "Categories of artificial societies," in *Engineering Societies in the Agents World II*. Springer Berlin Heidelberg, 2001, pp. 1–9.
[17] M. Jacoby, F. Volz, C. Weißenbacher, and J. Müller, "FA3ST Service – An Open Source Implementation of the Reactive Asset Administration Shell," in *Proc. of the IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA'2022)*, 2022, pp. 1–8.
[18] IEEE, "IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions," *IEEE Std 2660.1-2020*, pp. 1–43, 2021.
[19] L. A. Cruz Salazar and H. Li, *Proportional Reliability of Agent-Oriented Software Engineering for the Application of Cyber Physical Production Systems*. Cham: Springer International Publishing, 2018, pp. 139–156.