Applications

Luis Alberto Cruz Salazar* and Birgit Vogel-Heuser

A CPPS-architecture and workflow for bringing agent-based technologies as a form of artificial intelligence into practice

Eine CPPS-Architektur mit Umsetzungsleitfaden um agenten-basierte Technologien als Form Künstlicher Intelligenz in die Anwendung zu bringen

https://doi.org/10.1515/auto-2022-0008
Received January 31, 2022; accepted April 22, 2022

Abstract: Due to the increase in Artificial Intelligence in the production systems domain, Industry 4.0 (I4.0) experts must collaborate with autonomous systems. Industrial AI raises several concerns about existing standards, which provide guidelines and design patterns. One way to realize I4.0 systems are Industrial Agents (IAs) due to their inherent autonomy and collaboration. Multi-Agent Systems (MASs) are well suited for realizing distributed AI in I4.0 components. Considering the properties of IAs and existing standards, an MAS architecture is presented for flexible and intelligent Cyber-Physical Production Systems. The article compares I4.0 standardization efforts relevant to adapt AI in the form of IAs, illustrates how IA design patterns can be used, and introduces the *Multi-Agent aRchitecture for Industrial Automation applying desigN patterNs practicEs* "MARIANNE". An implementation guideline is presented to put this CPPS into practice.

Keywords: Artificial Intelligence, Cyber-Physical Production Systems, Industrial Agents, Multi-Agent Systems

Zusammenfassung: Aufgrund der Zunahme künstlicher Intelligenz im Produktionssystembereich müssen Industrie 4.0 (I4.0) Experten mit autonomen Systemen zusammenarbeiten. Industrielle KI wirft Fragen zu bestehenden Standards auf, die Richtlinien und Entwurfsmuster bereitstellen. Eine Möglichkeit, KI in I4.0-Systemen zu realisieren, sind aufgrund ihrer inhärenten Autonomie und Zusammenarbeit industrielle Agenten (IAs). Multi-Agenten-Systeme (MASs) sind gut geeignet, um verteilte I4.0-Komponenten zu realisieren. Unter Berücksichtigung der Eigenschaften von IAs und bestehender Standards wird eine MAS-Architektur für flexible und intelligente Cyber-Physical Production Systems (CPPS) vorgestellt. Der Artikel vergleicht I4.0-Standardisierungsbestrebungen, die für die Adaption von KI in Form von IAs relevant sind, zeigt auf, wie KI-Entwurfsmuster verwendet werden können und stellt die *Multi-Agent aRchitecture for Industrial Automation applying desigN patterNs practicEs* „MARIANNE" vor. Es wird ein Implementierungsleitfaden vorgestellt, um dieses CPPS in die Praxis umzusetzen.

Schlagwörter: Cyber-physische Produktionssysteme, Industrielle Agenten, Künstliche Intelligenz, Multi-Agenten Systeme

1 Motivation

Artificial Intelligence (*AI*), in the context of *Industry 4.0 (I4.0)*, opens up the possibility to solve machine tasks previously considered to be only performable by humans: interpreting natural language or visual data, identifying design patterns, and making autonomous decisions [19, 21]. In I4.0, interconnections between machines, smart sensors, actuators, are becoming more common. The networked entities, also known as *Cyber-Physical Production Systems* (*CPPSs*), or industrial *Cyber-Physical Systems* (*CPSs*) [24], initially commenced as *automated Production Systems* (aPSs) in various manufacturing domains. The CPPSs consist of CPSs applied in aPS domains to link physical and virtual objects (real world and information-processing) through constantly, and oftentimes globally,

***Corresponding author: Luis Alberto Cruz Salazar,** Institute of Automation and Information Systems, Department of Mechanical Engineering, TUM School of Engineering and Design, Technical University of Munich, Munich, Germany, e-mail: luis.cruz@tum.de, ORCID: https://orcid.org/0000-0001-8386-5568
Birgit Vogel-Heuser, Institute of Automation and Information Systems, Department of Mechanical Engineering, TUM School of Engineering and Design, Core Member of MDSI and Member of MIRMI, Technical University of Munich, Munich, Germany, e-mail: vogel-heuser@tum.de, ORCID: https://orcid.org/0000-0003-2785-8819

interconnected information networks [24]. Typical intelligence concepts enabling CPPSs are "agent" entities that are often related to AI, referring to a smart, self-contained software program [14]. Agent-based definitions, typologies, methodologies, technologies, standards, platforms, design patterns, and programming language approaches, such as *Agent-Oriented Software Engineering*, have all evolved throughout time [6]. *Multi-Agent Systems* (*MASs*) consist out of *Industrial Agents* (*IAs*) that have been touted as a viable and feasible answer for a series of new industrial challenges over the years [6, 14, 17]. However, there is no deep analysis of IA's levels of intelligence, nor their direct correspondence to AI applied in I4.0. Additionally, combining AI through *Machine Learning* (*ML*) into IAs has made it possible to achieve CPPS' learnability and reconfigurability [21], which are necessary properties to deal with I4.0 issues. Furthermore, the deployment of autonomous and collaborative manufacturing entities with enhanced self-capabilities, such as self-optimization, self-awareness, and self-monitoring, is a priority for CPPS [21]. *Industrial AI* via IAs is viewed as an essential technology to accomplish these capabilities and disrupt the way aPS processes and business models are structured as part of the I4.0 paradigm [6, 14, 17]. AI is a sub-discipline of software engineering, capable of implementing IA characteristics traditionally associated with human intelligence, such as autonomy, reactiveness, reasoning, predictiveness (learning), and self-improvement [26]. Despite this, there is no widely acknowledged, precise, and standardized definition of Industrial AI [21].

Notwithstanding the ostensible benefits of these Industrial AI systems – CPPS implemented by IAs – the cost of factory transformation, insufficiently qualified people in essential AI technologies, a lack of design processes, and reusable MAS applications continue to make it difficult for industries to implement I4.0 concepts. For this reason, in recent years, IA working groups, TC-IA[1] by the IEEE P2660.1 and the German IFAC NMO GMA FA 5.15[2] VDI/VDE, have addressed these challenges by establishing design patterns and best practices. Two relevant standards, the *"IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions"* [11] and the *"2653 Sheet 4: Multi-agent systems in industrial automation – Selected patterns for field level control and energy systems"* [30], suggest methods for developing IAs. The combination of these standardization efforts with models that reflect IA design concepts [1, 3–5, 8, 9], and also with established notions such as the *Product, Process, Resource* (*PPR*) *concept*, and I4.0 standardization efforts, specifically RAMI4.0, is crucial though and requires an integrated architecture. Hence, this article makes three contributions. First, it examines how an agent-based CPPS can be combined with relevant *Reference Architecture Model I4.0 "RAMI4.0"* [7] and the PPR model (Con1). Second, an MAS architecture for CPPS derived from IA patterns is presented (Con2). Third, to improve industrial applicability, a guideline is provided in order to facilitate the IAs and AASs implementation into hybrid CPPS platforms (Con3).

This manuscript is structured as follows: Section 2 explains IAs' requirements and introduces the state of the art regarding MASs in I4.0. Section 3 contains the main contribution of this work and presents an agent-based CPPS and its definitions. Section 4 describes the MAS' implementation by applying an I4.0 scenario and Section 5 discusses findings from the evaluation. The paper concludes with a summary and an outlook.

2 State of the art

This section introduces related work regarding Industrial Agents, their standardization, and approaches for combining them in an MAS for I4.0. Different viewpoints are decided by current I4.0 experts, leading to multiple models and meaning various descriptions of the target system [3]. Regarding agent-based CPPS, the two IAs standards are related here, showing diverse IA pattern types that the MAS community analyzed from several functionality points of view.

2.1 Industrial agents for I4.0: categorization, modeling, and standardization

Industry 4.0 and CPPS often refer to the MAS approach [6, 14, 17, 24], and to the *Asset Administration Shell* (*AAS*), which is one of the main specifications of the RAMI4.0 [7]. The AAS, together with an IA, allows smart access to asset resource information, as well as connectivity with other I4.0 components [6]. Applying *Information Technology* (*IT*) for I4.0 is notable and able to deploy the *Digital Twin*

[1] TC-IA refers to the IEEE-IES Technical Committee on Industrial Agents.

[2] FA 5.15 "Agent systems" is a German working group (GMA). English: Society of German Engineers VDI, and German Electrical Engineers VDE. VDI/VDE is known as a National Member Organization (NMO) of IFAC.

(*DT*) concept [22]. Leveraging DTs' technologies, specifically, the AAS to realize an MAS, increases the flexibility and adaptability of aPS [31]. In another hand, an IA is an intelligent entity used for distributed problem-solving in automation, typically characterized as autonomous, collaborative, and communicative [11]. Implementing IA technology within multiple automation fields (e. g., planning, scheduling) has been studied for several years. For example, many international projects foster research on future factories that use IAs, e. g., in smart production, smart logistics, smart grids [17]. In contrast, using IAs in the field at the process level (supervision and control) is comparatively novel research considering hard/soft real-time needs [6, 11, 30]. A specific requirement is that an IA must be autonomous [10, 11, 30]. It may work in an organized way with other external agents, even humans [11]. IAs could be applied to apply the *Human-in-the-loop* concept in I4.0 [14], where the plant floor operators can act and interacts as agents in the CPPS. All instances should consider different amounts of data and ensure a timely response in order to react to work with agents' decisions and actions. The above characteristics divide IAs into multiple categories, as shown in the following subsections.

2.1.1 Traditional types of IAs by response time and behavior

For most agent-based automation developers, it is well known that agent features are mainly based on classifications. These grouping methods – often called design patterns – generate relationships and approximate common functionalities at different automation levels [6]. An IA also provides the intelligence for the sensors/actuators to have *Low-Level Control* "LLC" (with soft or hard real-time) or provides the support that helps foster a desirable collaboration with *Manufacturing Execution System* (*MES*) and the *Enterprise Resource Planning* (*ERP*) levels. Both ERP and MES are part of *High-Level Control* "HLC" and usually do not require real-time capabilities. Therefore, there are initially numerous categories, including the *Reactive Agent* and the *Deliberative Agent* definitions [17].

Unland [29] defines a Reactive Agent as a "simple" agent because this type does not deal with a representative world (modeling), nor does it apply complex reasoning. The Deliberative Agent is often semantically on a higher level than "reactive" and "proactive" [29], since this type is synonymous with "Strategy & Goals" and can involve functions based on (but not limited to) probabilities, logical deduction, knowledge-based reasoning, among other inference mechanisms [32]. The Deliberative Agent's behavior and common architectures are reasonably more sophisticated than the ones of Reactive Agents. This IA type is most prevalent, even if the internal processes of deliberative software are more complicated, which increases to their time and resource consumption. However, in contrast to a human operator, a Deliberative/Proactive agent "understands," only a small part of the entire world, i. e., data acquisition is restricted by non-biological sensors. Nevertheless, it always has wide-ranging, real-world knowledge. In the industry, Reactive Agents are implemented in various ways, including mapping between situations and actions. Their connection ways can be [6]: first direct with the same network domain, i. e., synchronous connection web service or OPC UA [34]; second, indirect across different network domains., i. e., asynchronous FIPA (see *ACL Message Structure Specification* [10]). From those definitions, Deliberative Agents are moderately flexible when immediately acting upon their environment. They can, on the other hand, become substantially more complicated and slower in their reactions. Instead, the Reactive Agent's behavior includes a faster response to relevant stimulations from its environment, as input produces output by simple situation-action associations that are frequently implemented, whilst ignoring the rest of the perceived history (also namely simple *Reflex Agents* [26]). Hence, the Reactive Agent requires fewer resources than the Deliberative Agent, and it reacts more quickly.

Nevertheless, on the negative side, the Reactive Agent is not as dynamic and flexible as the Deliberative Agent that can usually behave proactively. In other research, Russell and Norvig [26] consider the behavior of Reactive Agents to be generally not (much) worse than the one of Deliberative Agents (also namely *Rational Agent* [26]). Under certain conditions, proactiveness would imply agent reactiveness, so IAs react to a state change to achieve a goal [5].

New advances in IA's classification considering multiple smartness dimensions should be an interesting topic for distributed AI researchers, but up to now, it has been avoided. Two reasons for improving IA typologies are: firstly, to prevent the *AI effect*, meaning the IA technologies that were once thought to be intelligent will become outdated as systems are becoming increasingly capable. One example would be providing adaptability to predictability in CPPS architectures that need to be scaled up [24]. Secondly, this IA categorization depends on the existence or not of the normalized IAs. For instance, if a CPPS architecture includes reactive or proactive IAs, this is a traditional MAS [29]. In contrast, IA classification based on its capabilities is more precise, since an IA may handle several functions borrowed from advanced AI characteristics, e. g., learnability [14]. Therefore, Section 2.1.2 proposes a new IA categorization regarding specific requirements in the same section and complements it with the traditional agent types.

2.1.2 Modern classes of IAs (by AI characteristic and IA capability)

This section proposes combining the traditional types of IAs with means of modern categorization by Industrial AI definitions related to I4.0 prerequisites. Summarizing Section 2.1.1, the traditional typology refers to *response time*[3] and main behavior (or feature) by three types of IAs, as follows (those are adapted from [26, 29]):

- Reactive IA, that reacts to perception [29].
- Proactive IA, that performs initiative actions [29].
- Deliberative IA (henceforth "Predictive IA"), that anticipates by learning tasks. Here, we refer to *predictive learning* to specify the *learning agent* that can be formed or not formed from a traditional IA (reactive or proactive ones) [26].

One prerequisite for the I4.0 is the formal specification of capabilities and skills [34]. The *I4.0 Platform* defines a *Capability* as an *"implementation independent potential of an Industrie 4.0 component to achieve an effect within a domain"* [23]. Also, they describe that a *Skill "can be made executable via services"* [23]. On the other hand, the *Competence* of a system is the *"ability to apply knowledge and skills to achieve intended results"* (this taxonomy is standardized by the ISO/IEC/IEEE 24765 [12]). Skills are also adopted from the IAs community to denote one of the MAS self-contained software functionalities [11]. Therefore, in this paper, capabilities state competencies, just as skills state functionalities (set of functions to provide IA services).

Typically, systems capable of Industrial AI implement minimal AI characteristics like autonomy (C1) and reactiveness (C2). In various I4.0 use cases, the system autonomy is provided by auto-adjusting aPS. A more detailed description can be found in [19]. As a result, in Industrial AI, the degree of autonomy of equipment or processes is higher or lower according to the I4.0 scenarios [32]. In the case of reactiveness, for most AI techniques, reactive control is sensor-driven, and it is the most appropriate for low-level actions [26], i. e., hard and soft real-time. Moreover, Industrial AI frequently requires proactive (C3) and predictive (C4) capabilities [21], both are reasoning characteristics, but the last one is the most complex Industrial AI characteristic since it requires learning from the past (as discussed in Section 2.1.1). On one side, reasoning generates global solutions to complex tasks using planning [26], i. e., models for decision making (C3) or models learning from experience/predicted data (C4). On the other side, proactiveness (C3) logically implies reactiveness (C2) [5]. Consequently, Industrial AI often uses reactive methods for LLC and deliberative/reasoning techniques for HLC [26] (see IAs' definitions in Section 2.1.1). Finally, the human cooperativeness characteristic (C5) increasingly consider human-machine integration as a fundamental design principle of CPPSs [14]. However, IA is still far from an entirely symbiotic human and AI interaction, meaning there are a poor relationship, co-existence, and collaboration among humans (C4) and IAs [13]. Therefore, concepts like predictability (C4), as well as the involvement of the Human-in-the-loop (C5), are the most critical capabilities to be achieved in Industrial AI systems [21]. Predictability (C4), applying ML, is one of the AI characteristics through which IAs provide CPPS to achieve learnability [21]. Predictive learning is a term used to describe an unsupervised ML system that can anticipate characteristics of its changing states [26].

A summary of the Industrial AI characteristics discussed above is enlisted in Table 1.

2.2 Standardizing industrial agents

Derived from the Industrial AI characteristics (see Section 2.1.2. Table 1), the authors determined four IA classes with specific capabilities potentially interesting for the development of MASs. As listed in Table 2, Class I refers to *Physical access agent*, Class II to *Organizational agent*, Class III to *Interface agent*, and Class IV to *Human agent*. Each Industrial AI characteristics supports the IA classes by implementing their capabilities. This means that different Industrial AI characteristics implementations can be mapped to each IA class's capability (at least one skill to each class). Moreover, skills and capabilities differ in the level of implementation [23]: while skills offer details of asset-dependent descriptions [22] (e. g., Common Data Dictionary/ECLASS/IEC 61360, OPC UA/IEC 62541 methods), capabilities are independent formal abstractions of

[3] *Response time*, or *Time response* refers here to the how long is the time taken by the Industrial Agent to respond to a certain task (adapted from the IEEE 2660.1 guideline [11]).

Table 1: Definitions of Industrial AI characteristics.

Industrial AI characteristic	Characteristic definition
C1. Autonomy [19]	Degree to which an industrial system can independently master uncertain conditions in a delimited and automated manner achieving its objectives systematically, i. e., without external or human intervention
C2. Reactiveness [26]	Degree to which an industrial system can respond to a request for the processing of its environment information (observation and communication responsiveness in real-time)
C3. Proactiveness [5]	Degree to which an industrial system takes the initiative for deciding and processing information whilst pursuing a goal (reasoning for deliberative tasks).
C4. Predictability[4] [24]	Degree to which an industrial system can predict (*predictive capability* [25]) the next outcomes of actions given the actions in the previous tasks and the self-learning (from past information).
C5. Human cooperativeness [14]	Degree to which an industrial system can apply the Human-in-the-loop concept

Table 2: Industrial Agents, their main competencies and examples.

IA class	IA's competence/capability (capable of)	Instantiation (a particular example)
I. Physical access agent	Abstracting and connecting heterogeneous production equipment with the MAS	This IA acts as a digital representation of a physical object ranging from a single product (or a service) to an enterprise network at the hierarchy axis [2]. This IA class also has access to assets' main functionalities and is building on the normalized *Resource Agent* (see VDI/VDE 2653-4 guideline [30])
II. Organizational agent	Offering various services into an integrated and united execution model that can support managing and organizing the operation of the MAS and its IAs (see *FIPA Agent Management Specification* [10])	This IA type is often concerned with non-physical entities, e. g., orders, production plans, production schedules, among others [29]. The typical instances of this IA class are the normalized *Agent Management System* and the *Process Agent* (see VDI/VDE 2653-4 guideline [30])
III. Interface agent	Providing effective communication between the IAs converting property interfaces into multiple protocols	An IA class' instantiation is the normalized *Communication Agent* (see VDI/VDE 2653-4 guideline [30]), and this may, for example, interconnect IAs and LLC automation functions based on the *IEEE 2660.1 interface practice* [11]
IV. Human agent	Allowing humans to act as agents in the MAS interacting with others agents/systems among the automation levels	This IA type should be able to achieve the concept for Human-in-the-loop in I4.0 [14]

the asset application functionalities and that can be expressed in different ways, e. g., Knowledge Base (KB) formalisms by *Web Ontology Language*, *Unified Modeling Language* models UML/SysML/IEC 19514 [1].

Classes I and II (physical and organizational agents) cover most traditional IA types also normalized by the VDI/VDE 2653-4 guideline [30]. Those agents are named *Resource Agent* (*RA*), *Process Agent* (*PA*), and *Agent Management System* (*AMS*). The *Physical access agent* is derived from the RA to access the capabilities of physical resources, i. e., abstracting and connecting heterogeneous production equipment with the MAS [30]. The AMS is part of the *Foundation for Physical Agents "FIPA"* (see *Agent Management Specification* [10]), while the typical Class III (interface agent), as the *Communication Agent* (*CA*), is addressed by the IA interfacing patterns of the IEEE 2660.1 guideline [11]. The main definitions from both IA standards and FIPA elements concerning this work are described in Section 3. The IEEE 2660.1 interface practices – related to Class III agent – are clustered by location (the place where the HLC/LLC are hosted), i. e., on-device or hybrid; and the interaction mode dimensions (the way the HLC/LLC interact), i. e., tightly coupled or loosely coupled [11]. Thus, the CA accounts for the wide range of IA's interfacing techniques, divided into those two levels of abstraction. In contrast to the other IA classes (I-III), and due to

[4] There are many definitions to Predictability referring to CPS, as discussed by Sun et al. [28]; however, we adapted this IA characteristic based on "the ability to anticipate the behavior of a system" definition presented by Lee [16]. Additionally, in our approach, the agent-based CPPS needs to be predictable (able to be predicted) to be learnable.

its complexity, Class IV (human agent) is not standardized yet. One reason is probably the different approaches from MAS developers to describing intelligence resulting from a Human-in-the-loop between processes and a human being.

As CPPSs are complex, modeling them from different viewpoints helps cut the overall complexity. Taking this into account, CPPS developers demonstrated that integration with legacy IT systems (e. g., ERP, MES, PLM applications) must be addressed proactively [21]. Thus, agent-based CPPSs typically encompass multiple data sources, which are able to get reusable information to successfully deploy distributed AI applications at a large scale, i. e., System of a System concept [14]. A relevant modeling requirement for I4.0 is the RAMI4.0 capability, specifically in the AAS concept. Thus, here RAMI4.0 capability refers to that MAS architectures should accomplish the developed I4.0 reports with various models, providing the basis for expanding new I4.0 components, as the *Details of the Asset Administration Shell* report version 3.0 [22]. Therefore, in order to improve I4.0 semantics, a CPPS should consider RAMI4.0 as a design principle rather than I4.0 conceptual standard. This means that CPPSs need an integral understanding of the AAS context, where the details of the I4.0 component enables binding semantics, clearly identifying its assets, sub-models and properties in a constantly readable directory [22]. Interestingly, MAS authors using the AAS and OPC UA [20, 34] added flexibility by the *Plug & produce* concept (similar to *Plug & play* and *Plug & work* terms [23]) in various I4.0 scenarios. MASs enable I4.0 scenarios such as *Adaptable Factory*, *Order Controller Product*, and *Self-organizing Adaptive Logistics* extended by the authors in [32].

Summarizing, the variety and heterogeneity of available standardization efforts hinders the efficient and interoperable design of agent-based CPPSs, i. e., applying the details of the AAS, RAMI4.0 capability, I4.0 components and I4.0 scenarios concepts. To address these issues, the interconnections between the AAS and the respective models need to be identified, which allows the creation of an MAS architecture compatible with current I4.0 approaches.

2.3 Selected MASs for Industry 4.0

This section analyzes existing MAS from different I4.0 research groups in order to have a wide range of application domains and points of view. The selected researches are named with acronyms or the prominent authors' last name. The I4.0 architectures are selected based on the representative aspects of the aPS domain and the IA classes identified, as follows: Class I for field-level control, PROPHESY-CPS [20] and Zimmermann *et al.* [34]; Class I-II for discrete manufacturing, H-Entity [5] and SemAnz40 [9]; Class I-III for pattern-based CPPS, Cruz *et al.* [6], FAPS [8] and MOSAIK [4]; and finally, covering Class I-IV for Industrial CPS, Ribeiro *et al.* [24].

When comparing the selected architectures (cp. Table 3), it becomes apparent that aPS domains focus on reactiveness (C1) and proactiveness (C2). Regarding the manufacturing domain, the SemAnz40 [9] defines the KBs to support semantic modeling of a reactive aPS (C2). From the relevant designs for I4.0, Ribeiro *et al.* [24] propose a CPPS with strongly human cooperativeness (C5), according to five scale levels of requirements, including adaptability, convertibility, integrability and other requirements; each requirement is described with an obligation grade from three options: shall (must), should (optional), and will (may). Its industrial CPS focuses on local autonomy (C1) and basic protocols, changing its structure dynamically to cover, among others, predictability (C4) [24]. Although no reusable patterns are considered in most selected works, by contrast, MOSAIK [4] determined selected patterns focusing on the role played by the *Object Management Group* (e. g., UML/SysML) and AutomationML as exchange standards for CPPS engineering.

Finally, the creators of promising MASs for CPPS focus on their natural autonomy, reactiveness, and proactiveness, but their different objectives affect the level of abstraction of the model, even in the same application domain. For instance, MOSAIK [4] is a self-organized MAS consisting of different agents or "artifacts" within the manufacturing domain – particularly architectures based on the cloud, *Web of Things*, and *Industrial Internet of Things* technologies. MASs, by their very nature, have often high autonomy (C1) and reactiveness characteristics (C2), as demonstrated by IA researchers [6, 14, 17, 24]. However, MAS architectures have not advanced in the learnability of the agents (C4), and few works consider the design patterns practices [4, 34], which use human analyses (C5) to improve reusability among other benefits [11, 30]. In general, most of the representative CPPS approaches shown in Table 3 are missing predictability characteristics (C4) and the RAMI4.0 capability. Therefore, in order to fulfill those requirements, this paper proposes the *Multi-Agent aRchitecture for Industrial Automation applying desigN patterNs practicEs* (*MARIANNE*) following the IA's normalized guidelines [11, 30], and addressed by standardized definitions of its classes (see Section 2.2).

Exploring the state-of-the-art, the authors considered exemplary MASs extended from [3], as given in Table 3.

Table 3: Selected MAS architectures for Industry 4.0. Extended from [3].

	Cruz *et al.* [6]	**FAPS [8]**	**H-Entity [5]**	**PROPHESY-CPS [20]**	**Ribeiro *et al.* [24]**	**SemAnz40 [9]**	**MOSAIK [4]**	**Zimmermann *et al.* [34]**
Industrial AI characteristic (C1–C5)	C1 Auto. C2 React. C3 Proact. C5 Human coop.	C1 Auto. C2 React. C3 Proact.	C1 Auto. C2 React. C3 Proact. C4 Predict.	C2 React. C3 Proact. C4 Predict.	C1 Auto. C2 React. C3 Proact. C4 Predict. C5 Hum. coop.	C1 Auto. C2 React. C5 Hum. coop.	C1 Auto. C2 React. C5 Hum. coop.	C1 Auto. C2 React. C5 Hum. coop.
IA's classes (patterns)	I-III (RA, PA, CA, AMS)	I, III (RA, PA, CA, AMS)	I-II (RA, AMS)	I (RA)	I-IV (RA, PA)	I, II (RA, PA, AMS)	I-III (RA, PA, CA)	I (RA)
RAMI4.0 capability	Partially	Yes	No	Yes	Partially	Partially	No	Partially
PPR structure	Resource	Resource	Process Resource	Resource	Process Resource	Process Resource	Product Resource	Resource

They are categorized by the Industrial AI characteristic achieved, the IAs applied, RAMI4.0 capability, and the PPR (from the VDI/VDE 3682 guideline) structure correspondence (see details in Section 2.1 and Section 2.2).

Regarding the combination of RAMI4.0, PPR, and IAs suitability for applying CPPSs (Con1), the authors of this study intend to extend preliminary work [6]. One significant differentiation is the development of the DT by AAS together with IAs for a CPPS (see Sec. 4). Another critical factor from this paper is integrating and evaluating an MAS architecture using the existing IA pattern standards (see Sec. 5). However, to the best of their knowledge, DTs and IA design patterns, specifically the AAS, have not yet been combined into an agent-based CPPS architecture. Hence, there is a need for an architecture that supports developers in explaining (Con2) and implementing MASs (Con3). An MAS architecture and its implementation guideline in the CPPSs context shall be developed here.

3 Architecture and implementation workflow for agent-based CPPSs

This section describes a newly developed architecture for an MAS, improving semantic consistency by combining standardized entities. Each component definition and the code implementation described here is freely available on the *GitHub Agent 4.0*[5] project under the GPL v3.0 license. Meta-elements follow the UML class diagram (MOF 2.0), and similar to other IA authors [4], the word "entity" is used as a synonym for "UML object" to avoid misunderstanding with a real object participating in an action. In general, many details such as the unique identifier or ID, name, and description of each entity are not considered to make the MAS architecture easily comprensible.

3.1 Comparing models for I4.0/CPPS

MARIANNE is an agent-based architecture proposed for the manufacturing domain. This MAS is based on various notions, which are partially standardized in I4.0 works. The architecture proposed focuses not only on describing the IA patterns introduced in the VDI/VDE guidelines (2653, 3682) but also on relationships with RAMI4.0 [7], i. e., I4.0 concepts and the AAS concept. MARIANNE associates relevant and traditional aPS domain concepts, i. e., ISA-88 (IEC 61512-1) scenarios. For an overview of MARIANNE's key concepts and how they relate to models developed in the context of I4.0, such as RAMI4.0, but also the PPR concept, see Table 4.

Preliminarily, detailed analyses in Table 4 about existing models' classes for I4.0 should be executed regarding various aspects, such as function hierarchy levels, information classes, level of detail, specific application domain, among others, defined by [3]. In essence, a core model for I4.0 would allow for the creation of a modeling language with standardized concepts and terminologies, specifically based on the RAMI4.0/AAS and the PPR models. For instance, functional hierarchy levels can be realized via the I4.0 component in RAMI4.0/AAS, and via Resources in the PPR model. Using the MARIANNE classes related to the standards such as those mentioned, partic-

[5] MARIANNE codes into the Agent 4.0 project: https://github.com/siulzurc/agent4.0/tree/main/src/MARIANNE

Table 4: Relationship and comparison between models' classes for Industry 4.0.

	How can the (1–3) model realize or define the (a–i)?	Metamodel criteria*								
		a. Functional hierarchy levels	b. Engineer. Process steps	c. Technical flow sorts	d. Material	e. Information classes	f. Discipline range	g. Level of detail	h. aPS type	i. Specific application domain
Vias of the implementation	1. RAMI4.0/ AAS	I4.0-component		AAS: sub-model element collection	Asset	AAS: sub-model element	AAS: property or range	AAS: sub-models	I4.0-system	I4.0-component
	2. PPR model	Resource	Process	Product Process	Product				Process	
	3. MARIANNE (this work)	Physical access agent, Interface agent	Organizational agent	Process energy	Organizational agent	Human agent, Cognitive modeling	Knowledge base	Module: Unit, Equipment, Control	Application	Operation Maintenance Planning Scheduling

*Source: metamodeling aPS criteria from [3].

ularly ISA-88 modules (Unit, Equipment, and Control), a core model might be efficiently linked, mapped, or even utilized to generate new views merging aspects of existing ones [3]. Through reviewing the criteria of the models in Table 4, CPPS developers could work out the properties of the target I4.0 model, employ the existing ones, or extend them.

3.2 MARIANNE architecture

The following sections describe the notions used in MARIANNE that are also used in existing standardization efforts. Second, an implementation guideline for the MAS is provided to integrate the agent-based patterns that one can develop to instantiate the architecture.

3.2.1 Concepts used in MARIANNE that are related to standardization efforts

The main decisional elements from MARIANNE are explained in this section. This MAS architecture is composed of four IA classes that cover the main I4.0 concepts (see Table 4): I4.0 component (can be the Class I or III), Asset (managed by the Class II), and AAS (generated by the Class IV). Each IA is a virtual decision-making entity that can sense, process, store, or act on any CPPS shop floor. The IA structure used was proposed by Wannagat *et al.* [33], and it was employed in MARIANNE IAs. Figure 1 illustrates an overview of the MARIANNE's architecture in the GitHub project (see. Section 3) that is implemented in Python (.py), AASXexplorer (.aasx), Node-RED (.json), and TwinCAT (.tnzip) files, available online. Design pattern identification by [6] organizes the MARIANNE control through their IAs (cp. Figure 1, left). For instance, the entity *Status information function* provides current IA state representations. This entity relates to other IA modules such as the *Unit*, *Equipment* and *Control* from the ISA-88 model (cp. Figure 1, right).

The normalized definitions (classes) refer to the general overview based on the primary static information of the models for I4.0 (see Section 2.3). Consequently, a further (sub-)class defined by DIN 40912 is contained in MARIANNE to cover the main RAMI4.0 aspects, e. g., for implementing the AAS [22] report, version 3. Hence, systems implemented according to MARIANNE can be applied to achieve *I4.0 systems* (usually connected to cloud service providers). However, this class can also contain elements that do not achieve I4.0 requirements and are therefore not I4.0 components [7]. I4.0 system description contains an *I4.0 component* and its primary dependent representation of the RAMI4.0, i. e., *Asset* and *AAS* entities. Like the DT concept, an AAS is a digital representation of a resource that refers to assets [8]. Here, MARIANNE has IAs (as a type of distributed AI) that can encapsulate an *Asset* as a value for an organization [7]. MARIANNE reaches the Industrial AI characteristics (C1-C5, see Section 2.1.2) through IA competencies (cp. Table 2). A *Competence* entity refers to skills that depend on at least one software *Module*–MAS software can be divided into a set of skills [34]. An IA's module refers to common functionalities presented as patterns [6] and is extended by the IA's level of intelligence and AI characteristics. Then, to implement IA skills, external or internal modules that reference mathematical equa-

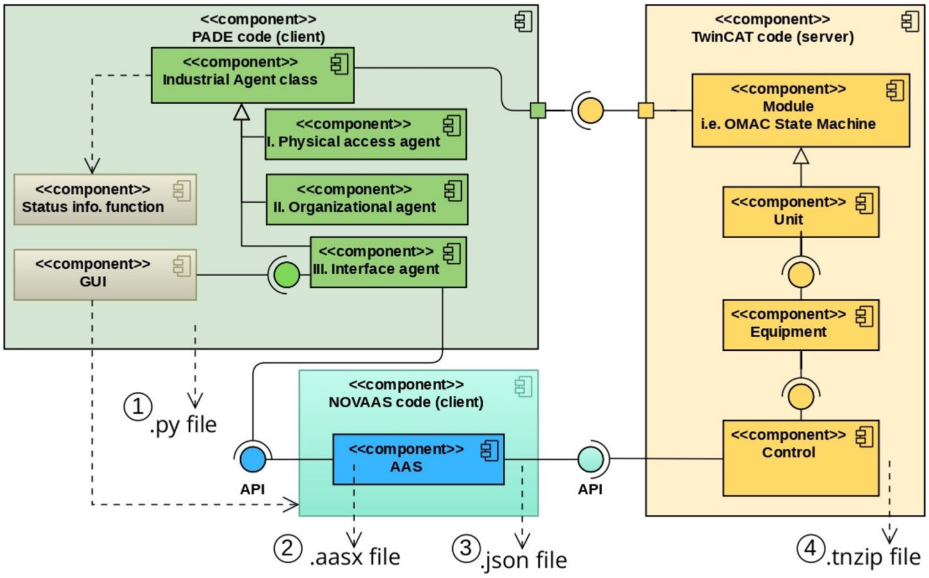


Figure 1: MARIANNE UML Component diagram. Codes for 1) PADE; 2) AASX Package Explorer; 3) Node-RED; and 4) TwinCAT.

tions (e. g., *Control*, *Reasoning*, or *Learning*) or logical descriptions (e. g., ISA-88 physical model: *Unit*, *Equipment*, *Module*) are integrated. Another function pattern in MARIANNE is the *Knowledge Base*, a type of *Database* that enables AIs of the consistent technical component descriptions as local knowledge [6]. Applying KBs enables smart manufacturing in a formalized way; however, there are no standardized MAS ways for generating them [31]. For instance, module entities implement logical production functionalities on different field-level devices, e. g., Programmable Logic Controllers (PLCs), Raspberry Pi, industrial computers. The control or KB entities can model further data to describe the hardware, e. g., the platform information, or define the information models. Here, IA's modules for logical purposes are written in various industrial programming languages, e. g., the IEC 61131-3, IEC 61499, Structured Text, C++/C#. Respective variables in PLCopen XML store local input and output devices' information in different levels of granularity, as given in [34].

Further building blocks to be reused from existing standardization efforts are essential for application in the CPPS domain. Here, an *Application* entity is a software functional unit [12] but refers to a specific solution of an agent-based CPPS to communicate efficiently, intelligently, collaboratively, and conform to a goal-oriented approach [11]. According to various application types, MAS developers consider that IAs are interacting with physical types of equipment to perform control functions in the CPPS domain. Typical aPS application types are *Operation*, *Maintenance*, *Planning*, and *Scheduling*. However, the PPR is also contained in MARIANNE to describe the fundamental domain of the CPPS, e. g., the type of process (Continuous, Discrete, or Batch).

Process, *Product*, and *Resource* from the VDI/VDE 3682 guideline are essential for the MAS architecture. Like an I4.0 component (consisting of an AAS and an asset [7]), a product is processed by a resource within a process. Here, a process is responsive to IA functionalities to make specific executions, deliberating which strategies will apply and which products or services they will offer. Resource entities generate a lot of data and specify the functions required to obtain products or services. The resource features whether the desired process steps can be executed (procedures to transform/transport/store the material/energy/information). Besides products, MARIANNE also covers a *Service* entity, considered as ITIL[6] 4, as *"a means of enabling value co-creation by facilitating outcomes that customers want to achieve, without the customer having to manage specific costs and risks"*.

3.2.2 IA types and reusable IA patterns

This work focuses on using design patterns in Model-Driven Engineering (MDE) for MASs, e. g., using UML/SysML in CPPS [1]. Recently, agent-based design pat-

[6] ITIL, is formerly an *Information Technology Infrastructure Library*. The 4th edition in 2019, focuses on fostering digital transformation, AI, cloud computing, and DevOps detailed practices (source: www.ibm.com).

terns for the industry have been discussed and approved by VDI/VDE-GMA FA 5.15 and IEEE TC-IA members, promoting standardized guidelines [30] and [11], respectively. The first guideline is integrated into the MARIANNE architecture by introducing classes of IA patterns. For the second guideline, MARIANNE's requirements for the types of IEEE 2660.1 interface practices are discussed, and qualitative evaluation from TC-IA guidelines are provided [11]. Software IA must recognize and efficiently handle the interface and functionality of industrial devices (LLC/HLC) [11]. Therefore, the MARIANNE architecture integrates the four IA classes proposed (cp. Table 2): human-, interface-, physical access-, and organizational agents. Those classes' definitions and their capabilities (see Section 2.1.2) are not fully standardized yet in the context of industrial MASs for I4.0. Instead, to be able to abstract in different levels, agent classes of the MARIANNE architecture contain existing IA agent patterns (inheritance relation): RA is an instance of the Class I; PA and AMS are instances of the Class II; and CA is an instance of the Class III (see Section 2.2). The *Physical access agent*, through RAs, access the capabilities of physical resources connecting shop floor equipment with the MAS [30]. The *Organizational agent* can support the general management of the MARIANNE and its *Scheduling*, i. e., it is a PA or an AMS. The *Interface agent* handles communication entities such as the *communication adapter* to provide requirements for different intercommunication systems with the MAS. The CA is an instance of this IA class, which considers the categorization based on the agent patterns interfaces, i. e., interaction mode and location (see Section 2.2, cp. Table 2). Consequently, a CA can derive four communication interface practices [11]: i) Tightly Coupled Hybrid, ii) Tightly Coupled On-device; iii) Loosely Coupled Hybrid; and iv) Loosely Coupled On-device. These interfaces vary depending on the location of the CA control (i. e., LLC/HLC), as well as from its IEEE 2660.1 interface practice [11]. Finally, the *Human agent* entity is able to apply the Human-in-the-loop concept [14], e. g., through the *human factor* or *cognitive modeling* entities.

3.3 MARIANNE's implementation guideline

For an asset or the whole MAS, the CA provides the communication adapters and cohesions to the outside world. The CA enables different communication means, e. g., among plants, between AMSs, or provides the Human Machine Interface (HMI). For the latter, the CA can be implemented in Node-RED, while PADE [18] is used for other IA patterns with an interactive interface. PADE[7] has a similar structure to JADE but uses Python, making IA's implementation more versatile [18]. Regarding the abstract DT concept, according to the online glossary of *Platform Industrie 4.0*,[8] the AAS concretes its implementation [8]. Other options to implement DTs are the DTDL and Web of Things [4]. As a guideline, MARIANNE's implementation flowchart is shown in Figure 2, focusing in the AAS development.

For systems, where real-time capabilities are critical, the IA classes which control the CPPS, are generated programming the LLC, i. e., by the IEC 61131-3/C++ languages (cp. Figure 2, Case 1). For higher-level applications, where real-time capabilities are not as critical, IAs can be implemented using a high-level programming language such as Python (cp. Figure 2, Case 2). Here, additional steps are required to match the HLC to the CPPS. For managing the AAS used for this approach, web flow-based programming, e. g., Node-RED, can be applied to manage the AAS depending on the AAS tools available. In the authors' comprehension, hybrid DT application is acceptable since different DT approaches represent the complexity of asset behavior [8]. A hybrid DT application refers here to the combination of equipment used in direct connection with simulation technologies and with sub-models integrated into the forms of AAS. Therefore, in this work, the applied AAS and simulation are symbiotically united. The structure of AASs is defined and aimed at developing interoperable DTs [22]. Following this reference, MARIANNE uses the AAS, which has the two main parts, *Manifest* and *Component manager*, together with their *Header* and *Body* [6, 22]. Here the body has various sub-models for each CPPS' AAS. DT developers can use various techniques to help them create DTs. Since IAs cover advanced skills (typically an AAS's property/operation [22]) passive AAS are sufficient. Thus, the AASX Package Explorer for the AAS creation and external management is used to create the DT. The Python package PyI40AAS allows editing the AAS file and moving skills [31]. REST utilities and dynamic flow programming practices are the foundations and embrace another technical direction where browser-based interfaces are the foundations. As a result, MARIANNE architecture complies with current web technology developments as part of IT software applications using Node-RED, i. e., based on Node.js. In this case, NOVAAS is used as a runtime for the AAS [20].

[7] PADE project: https://github.com/grei-ufc/pade

[8] Plattform Industrie 4.0: https://www.plattform-i40.de

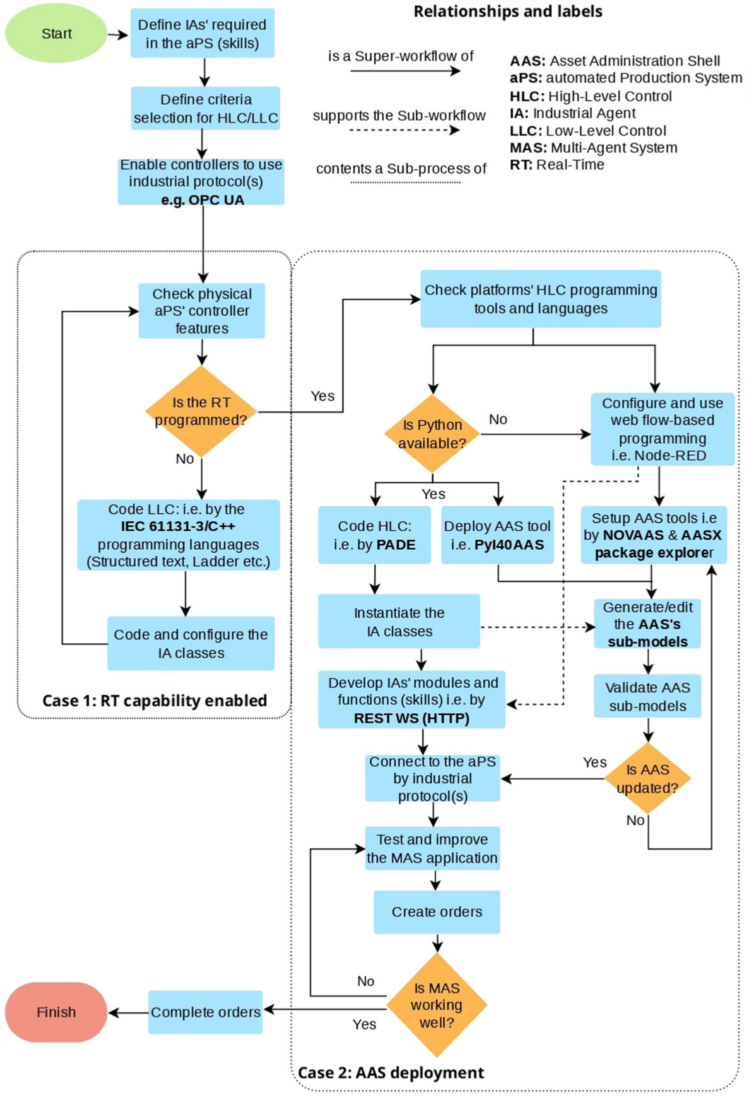

Relationships and labels
is a Super-workflow of
supports the Sub-workflow
contents a Sub-process of
AAS: Asset Administration Shell
aPS: automated Production System
HLC: High-Level Control
IA: Industrial Agent
LLC: Low-Level Control
MAS: Multi-Agent System
RT: Real-Time
Start
Define IAs' required in the aPS (skills)
Define criteria selection for HLC/LLC
Enable controllers to use industrial protocol(s) e.g. OPC UA
Check physical aPS' controller features
Is the RT programmed?
Yes
No
Code LLC: i.e. by the IEC 61131-3/C++ programming languages (Structured text, Ladder etc.)
Code and configure the IA classes
Case 1: RT capability enabled
Check platforms' HLC programming tools and languages
Is Python available?
No
Yes
Configure and use web flow-based programming i.e. Node-RED
Code HLC: i.e. by PADE
Deploy AAS tool i.e. PyI40AAS
Setup AAS tools i.e by NOVAAS & AASX package explorer
Instantiate the IA classes
Generate/edit the AAS's sub-models
Develop IAs' modules and functions (skills) i.e. by REST WS (HTTP)
Validate AAS sub-models
Connect to the aPS by industrial protocol(s)
Is AAS updated?
Yes
No
Test and improve the MAS application
Create orders
Is MAS working well?
No
Yes
Complete orders
Finish
Case 2: AAS deployment

Figure 2: MARIANNE's guideline implementation flowchart and its relationships.

4 Exemplary implementation for the demonstrator plant xPPU

This section presents an MAS implementation based on MARIANNE using the *eXtended Pick&Place Unit* (*xPPU*) [1]. As shown in Figure 3 (bottom right), in this context, the CA control can have a part of the same computational platform (on-device, i. e., PLC) or another type of Operation Technology "OT" (hybrid, i. e., Raspberry Pi, or PC). Typical communication protocols accepted for the I4.0 paradigm, such as Ethernet/EtherCAT, OPC UA, or Profinet [11], are implemented (cp. Figure 3, center and top).

The xPPU control application is contained in three primary devices: a PLC, a Raspberry Pi, and a PC with their KB (cp. Figure 3, top) to provide multiple IA pattern interfaces and multiple communication protocols. Thus, all IAs within the MAS as introduced in [31] are associated with the corresponding asset, including a KB. For coding the IA interfaces (on-device, i. e., PLC) in LLC, the IEC 61131-3 standard was implemented. To program similar IAs and interfaces in HLC (hybrid, PC, and Raspberry Pi), Node-RED/NOVAAS and Python/PADE were applied (cp. Figure 3, bottom). The HLC directly applies control on the LLC (tightly coupled), or brokers can intermediate the interface (loosely coupled). When LLC/HLC within a CA compiles and is deployed as a single set of binaries, it creates a tightly coupled and on-device design scenario. For instance, with *in/output device* entity, RAs (cp. Figure 3, bottom right) connect the sensors and actuators of assets

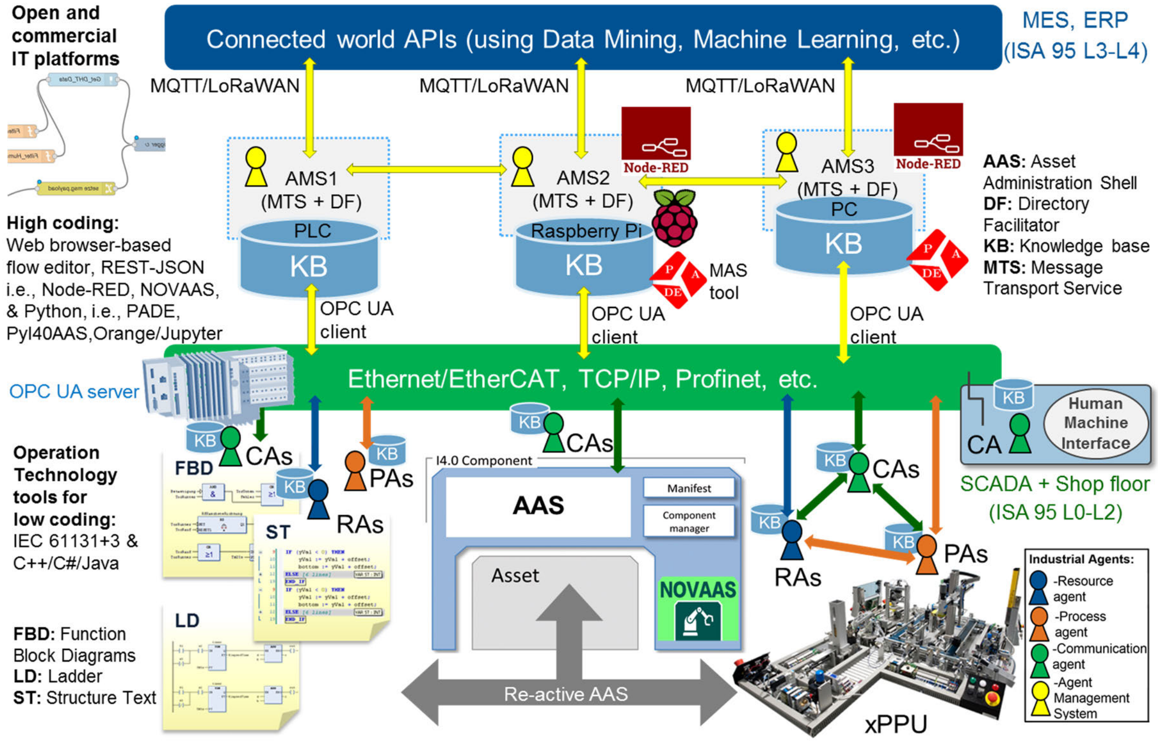


Figure 3: General landscape of the I4.0 scenario proposed with their I4.0 components and their IT/OT technologies.

[6] and connect the CPPS to offer services and thus create products.

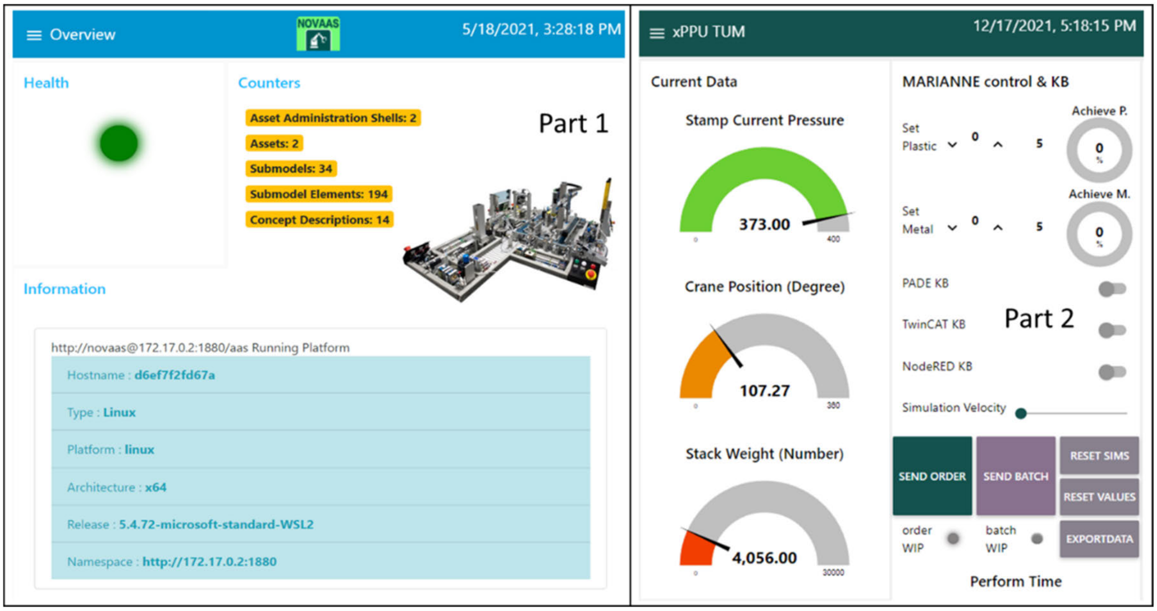
In general, MARIANNE addresses the main I4.0 concepts, and the IAs' skills can be implemented in the form of OPC UA methods, function blocks, and PLCopen XML as given in [34]. The PA (cp. Figure 4, bottom right) can store customers' orders and their execution plan [15], while information of all active IAs is managed by the AMS [10] (cp. Figure 4, top left). The structural software design in an I4.0 scenario must also be modular to comply with the adaptability requirements of hardware modules. Connecting IAs, messages, or upgrades can also be intuitively sent to all MAS using a shared communication network. A communication channel via the OPC UA protocol is chosen to connect to other systems unifying the data exchanged. Additionally, the data live dashboard and GUI function work as the HMI by a set of web interfaces in Node-RED, as shown in Figure 4 (Part 1 and Part 2).

Additionally, IT nodes using technologies like *Long Range Wide Area Network* (*LoRaWAN*) can be subsequently integrated into the MARIANNE through other compatible communication methods, such as the HTTP (REST) or Message Queuing Telemetry Transport (MQTT), and mapping the data into objects for model-based structures, i. e., it is as applied in OPC UA. All communication protocols options for this architecture can be related to the OSI model.

4.1 Implementing MARIANNE for an intelligent light barrier

Figure 5 shows the division of the xPPU in modules using the SysML block definition diagram (bdd, cp. Figure 5, center). The agent-based CPPS architecture can be hierarchically structured through OMAC State Machines (for HLC) and the lowest three levels of the ISA-88 physical model (for LLC): unit module, equipment module, and control module [1]. The xPPU unit module consists of two equipment modules, the stamping part and the sorting part. The stamping part is composed of a stack, a crane, and stamp control modules. The sorting part consists of a conveyor control module (cp. Figure 5, right).

For each ordered WP a PA is created, and then this WP is transported through the stamping plant to the sorting plant. The crane, which is equipped with a vacuum suction cup, picks up the WPs from the xPPU's warehouse and transports them either to the sorting plant or to the stamping plant, representing the processing station of the CPPS. On the stamping plant, a shifting table (crane) transports the WP under the stamp, where it is then imprinted with



Asset Administration Shells: 2
Assets: 2
Submodels: 34
Submodel Elements: 194
Concept Descriptions: 14

http://novaas@172.17.0.2:1880/aas Running Platform

Hostname : d6ef7f2fd67a

Type : Linux

Platform : linux

Architecture : x64

Release : 5.4.72-microsoft-standard-WSL2

Namespace : http://172.17.0.2:1880

Set Plastic 0 5 — 0 %

Set Metal 0 5 — 0 %

Figure 4: Dashboard and interfaces (GUI) of the implementation: 1) xPPU's NOVAAS dashboard, 2) CPPS' HMI in Node-RED.

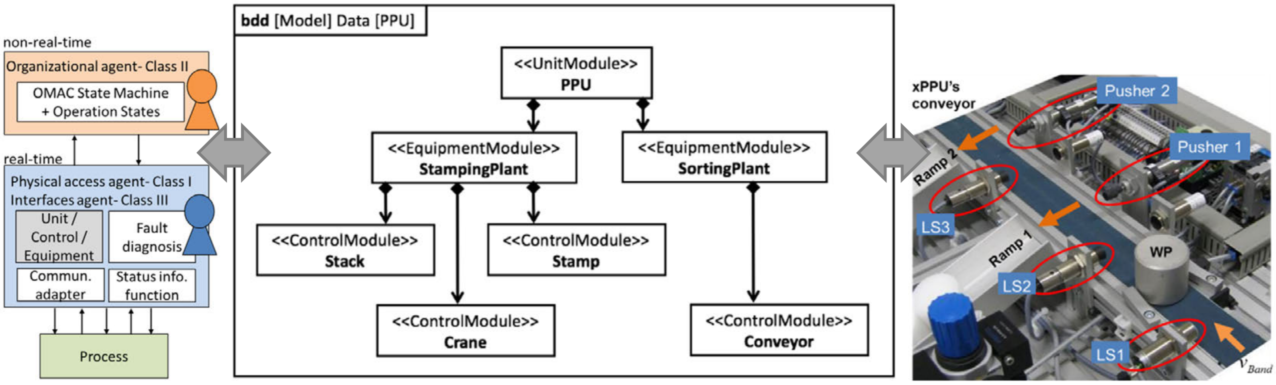


Figure 5: IAs and the SysML block definition diagram of separating the xPPU's HLC/LLC into modules. Adapted from [1].

an adjustable pressure (stack). The crane afterward transports the WP to the sorting plant. From here, the WP can be sorted into one of the three ramps that form its final process (cp. Figure 5, right). The first two ramps are equipped with pushers and sensors for material detection. The third ramp is positioned at the end of the conveyor belt and receives the WPs that have not been separated beforehand. Combining three light (binary) sensors LS1-LS3 (cp. Figure 5, right) makes it possible to determine the condition of the three different material cylinders (WPs).

In this context, physical access- and interface- agent classes are assigned to the individual CPPS modules (cp. Figure 5, left), with a distinction being made between RAs, PAs, and the AMSs, as is also the case in [6]. The WPs are initially assigned to an organizational agent class using PAs. As a result, these PAs define required services to produce various WPs (metallic, plastic, etc.). If the present RA cannot provide the required service, the PA's offer is sent to the next connected RA, who proceeds in the same way. For instance, some RAs include the crane and the conveyor belt for WP's transportation service. If the service is unavailable at the present transport RA, the request is sent to all connected transport RAs until the required service is found. In this case, for each offer request, a response will be sent. The possible processing time (to produce a WP) is adjusted based on the response time (IA real-time) of the available transport RAs, managed by PAs and AMSs. It means the fewer failures in the transport RAs (minor

IA time responses), the better the processing time performance.

The KB of the RA within the conveyor module includes analytical dependencies between the installed actuators and sensors. RAs can learn individual parameters at runtime to substitute missing actual values through mathematical estimations. An example is an agent-based sensor for a light barrier which is located in Ramp 1. The conveyors' drive is provided with an initial speed value (set value) in the primary state. Then, the WPs traveled distance and time are used to estimate an achieved speed (actual value). As a result, the initial speed is compared to the estimated actual speed, providing an error margin (%). If the error value is reasonable, the estimated speed is accurate enough and can be accepted. The most accurate estimated value decides the amplification factor for Ramp 1. However, a unique feature among the decision-making RA is the position-WP function block, which continuously calculates the WP's location on the conveyor based on the estimated speed and defined distance, e. g., LS1 to Ramp 1. In case of an erroneous position value of the limit switch sensor, the WP location will be replaced by the estimated position in the RA function block. To estimate an accurate final position, at least one of the positioning sensors of the entire sorting plant must be functioning.

The PA includes – but is not limited to – information directly and permanently associated with the WP, such as the material type, the processing time, or even the absolute conveyor position during the transfer; all these variables can be estimated by RA's function block. Corresponding ISA-88 modules were previously implemented by Bareiss *et al.* [1]. The present light barrier sensor is based on Wannagat *et al.* [33] and the level of abstraction that is part of this work; however, it uses a much more complex laboratory model that results in two main contributions. The first contribution implements the PA's call for proposal (CFP), using *Contract Net Protocol* – by PADE – that was implemented according to FIPA (see *Contract Net Interaction Protocol Specification* [10]). The second contribution is the implementation of xPPU sub-models embedded into a single AAS to increase interoperability.

4.2 IA patterns

Summarizing, as seen in Figure 6, the manufacturing process is often defined by order generation and execution (typical RAs and PAs interactions). Addressed by the VDI/VDE 2653-4 guideline of IA patterns, the RAs represent physical access (in/output devices) and keep its status information function synchronized with the input of the appropriate device (sensor data). An AMS is responsible for providing a single interface accessible for any IAs, using the same protocol, despite the CA provider. The AMS pattern, in most situations, keeps track of all involved and related IAs and their messaging addresses, as described in the preliminary research about IA patterns [6]. The AMS typically supervises a white pages service, maintaining a directory of IA references, and containing the two typical FIPA management components (see *FIPA Agent Management Specification* [10]): *Directory Facilitator* (*DF*), and *Message Transport Service* (*MTS*). An AMS, together with DF/MTS, often communicates with RAs and PAs to accomplish general MAS goals [6, 30]. Unlike the AMS, the PA is responsible for the manufacturing recipe rather than the technological structure since it naturally includes non-real-time capability [6]. Some MAS architectures replace the PA with a Product IA type, as Kovalenko *et al.* [15].

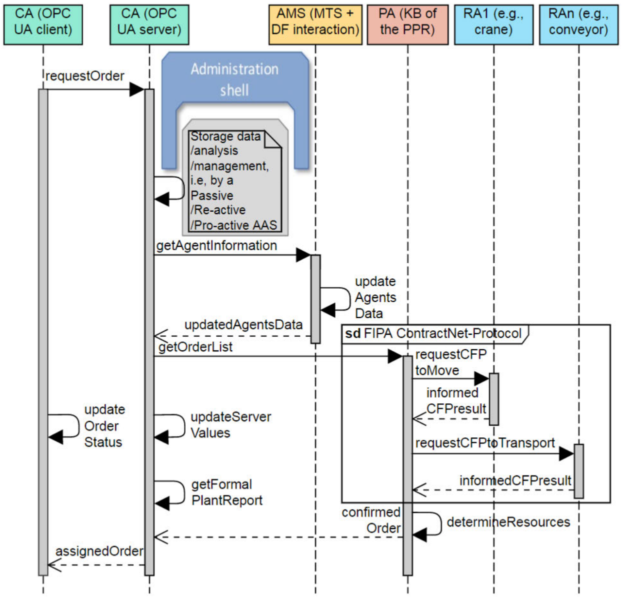


Figure 6: Sequence diagram to detail the IA patterns interactions in the CPPS network. CFP means "Call For Proposal" and refers to FIPA (see *FIPA Iterated Contract Net Interaction Protocol Specification* [10]).

5 Evaluation of the MARIANNE architecture and its agents' AI capabilites

This section gives an overview of the MARIANNE evaluation, providing details about the contributions and the Industrial AI characteristics covered through the IAs in the xPPU demonstrator results.

5.1 MARIANNE IAs and their Industrial AI characteristics

For a qualitative evaluation, we relied on IAs applied in the implementation described in Section 4.1. Those IAs are evaluated using four words to indicate a degree for a scale: *shall*, *should*, *may*, and *can*, as presented in the IEEE Recommended Practice for IAs [11]. That degree of obligation is an enumeration with five possible levels of Industrial AI characteristics related to IAs, analyzed in Section 2.1.2 (see Table 1), i. e., level indicates the number of AI characteristics required to apply a function or skills. The degree values of each IA are proposed by the authors' and justified by literature, with the following words' semantics:

Level 5. *Shall* indicates "mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted" [11].

Level 4. *Should* indicates "that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required" [11].

Level 3. *May* indicates "a course of action permissible within the limits of the standard" [11].

Level 2. *Can* indicates "statements of possibility and capability, whether material, physical, or causal" [11].

Level 1. *Usually not* (authors' semantic) indicates the minimum level of an Industrial AI characteristics' achievement.

According to the analyses from this study, current MAS implementation reaches different Industrial AI levels (C1-C4), while IAs -can apply various functions with a specific description, as given in Table 5. Each of the first IA function descriptions (cp. Table 5, items 1.1, 2.1, 3.1, and 4.1) is drawn from the authors' evaluation of the actual implementation (cp. Section 4); the other skills come from the authors' analyses of the IA concepts and their cited sources.

In the current implementation, the MAS is initiated by the AMS, and it perceives the skills of other IAs. The AMS can restart IAs and update their environment models autonomously (C1). A faster reaction is achieved for field-level control (C2), as the RAs can implement several resources, i. e., conveyor, crane, etc. Proactiveness is supported by PAs that apply CFPs to determine and recalculate necessary RAs in case of broken resources (C2), i. e., the agent-based soft sensors to increase availability. However, the physical resources of the RAs cannot be changed by the MAS itself, this can only be achieved by human actions (C5). IAs make decisions based on their environment models (C2-C3) created from the AAS and update if the xPPU models change. MARIANNE is not built to support the collaboration of RAs into the same order (only overall production process) because PAs usually request single processes. A general overview of the IAs evaluation concerning Industrial AI characteristics is given in Figure 7.

5.2 The MAS evaluation

The MARIANNE architecture comprises design patterns that are structured by four IA classes (Con2). The IAs applied for the xPPU are proposed and evaluated addressing VDI/VDE 2653-4 and IEEE 2660.1 standards [11, 30]. MARIANNE does not focus only on IAs but also RAMI4.0 (Con1), which should be robust, comprehensive, extendible, and meet I4.0 modeling requirements accurately, i. e., AAS concept (see Section 2.3). Our industry experts and IAs focus group members confirmed the utility of the agent-based design patterns concerning the IA classes [30] (Con2). In addition, MAS models show changing numbers of different semantics for CPPS entities and variable levels of abstraction, i. e., hierarchical structure by ISA-88 physical model (see Section 4.1). The authors of this work confirm that, to the best of their knowledge, all identified MARIANNE entities fall within the scope of standard taxonomies (see Section 3.2), ensuring comprehensiveness and consistency. Besides, MARIANNE supports the development of appropriate RAMI4.0 modeling approaches, i. e., AAS compatible (see Section 3.3). This work materializes the levels of abstraction of our IAs into a final implementation (see Section 4), following international standardizations (see Sections 2.2 and Section 3.2). Lastly, summarizing the MAS architecture guideline (Con3), the application shows how RAMI4.0 – which recommends OPC UA as the bridge between IT/OT [20, 22] – enables vertical and horizontal communication within the xPPU demonstrator for its HLC/LLC (see Section 3.3). Moreover, everything wrapped by the AAS concept is not limited to OPC-UA but encourages standard web technologies and IT, particularly by REST/JSON standards within Node-RED (NOVAAS application). This adaptation facilitates the integration of OT into IT while taking advantage of the Industrial AI maturity and steadiness of IA solutions, tools, and applications within IT areas, i. e., PADE plus NOVAAS.

Table 5: IA functions related to Industrial AI characteristics.

Item No.	**IA's function (skill) description**	**Industrial AI characteristic* C1–C2 (see Section 2.1.2, Table 1) evaluation**				
		Autonomy	**Reactiveness**	**Proactiveness**	**Predictability**	**Human cooperativeness**
1. RA (Reactive, Class I), standardized						
1.1	RA may be able to replace sensors data with soft sensors in order to increase reliability/availability of the MAS. See Section 4.1		●		●	●
1.2	RA can represent and control technical plant components such as equipment (often hard/soft real-time capability) [6, 30], as well as the resource allocation and its capabilities as services, e. g., the *ProductionService's* action in [8]		●	●		
1.3	RA can define its actions in a particular context at runtime utilizing its KB, e. g., for controlling and reconfiguring material flow systems [6]		●			●
1.4	RA can be able to carry out real-time execution in the plant floor like planning process, transport, processing workpiece, machining, among others [15]		●	●		
1.5	RA usually not have full autonomy due to the submissive heterarchy (it is often located in the lowest MAS's hierarchy) [6, 30], i. e., instead of negotiating IAs, a more hierarchical structure with dominant and submissive IAs might be more suited at the field-level [31]	●				
2. PA (Proactive, Class II), standardized						
2.1	PA may supervise the execution of a production recipe/plan the collaboration and negotiation of other IAs, e. g., RA, AMS, in order to complete its goals (often non-real-time capability). See Section 4.1		●	●		●
2.2	PA may represent the products that need to be processed [15]	●		●		●
2.3	PA can use graph-search and interaction with the underlying MAS as KB to run a discrete reasoning process to produce optimal production plans [6, 30]			●	●	
2.4	PA can apply a systematic, model-based optimization method during the decision-making process [15]	●			●	
2.5	PA usually are not responsible for the technical system but for the production recipe since it usually requires non-real-time capabilities [6, 30]			●		
3. CA (Reactive, Class III), standardized						
3.1	CA may coordinate the message-based communication among other IAs, e. g., on single or multiple platforms (PLCs, PCs, Raspberry Pis) across the field bus, including people interfaces (HMI). See Section 4	●	●		●	
3.2	CA can convert proprietary interfaces into multiple protocols, e. g., communication interface by TCP/IP (often real-time capability) [6, 30]		●	●		
3.3	CA usually is not limited to direct communication but also by patterns interfaces [11]					●
3.4	CA usually does not have a deterministic behavior communication because message stacks inside CA possibly will overflow. More details of this experiment are described in [27]		●			
4. AMS (Proactive, Class II), standardized						
4.1	AMS shall assume essential functions to coordination, control, and supervision for the IAs by maintaining a table (white pages) that contains their proper identifiers (often non-real-time capability). See Section 4.2	●	●	●	●	●
4.2	AMS can manage the operation of the MAS [10, 30], e. g., the creation, deletion, migration of IAs to and from the MAS [6, 18]	●				●
4.3	AMS can try to restart agents when they fail [18]	●			●	
4.4	AMS usually is not outside the IA's network because of its authority, as only one exists in a single MAS [10, 18]			●		

*Industrial AI characteristics that support the IA main task; ●: needed

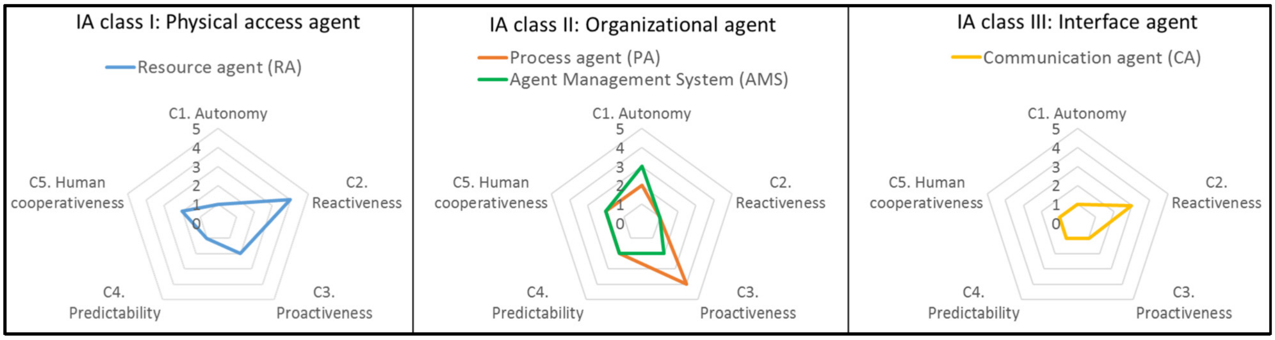


Figure 7: Industrial agents applied in MARIANNE architecture and their level of Industrial AI characteristics (see Table 5).

6 Summary and outlook

AI is an area of study aimed at understanding and creating intelligent systems that fall into the criteria of thinking or behaving logically or humanly [26]. In the I4.0 context, Industrial AI is a technological means of attaining a certain level of autonomy and other AI characteristics like reactiveness, proactiveness, predictability, and human cooperativeness [19, 21]. This paper presented various technologies for improving aPS to complement CPPS services, e. g., using the IA design patterns' potential. MDE for I4.0 and applications have received a lot of research and development attention. MDE can simplify the comprehension of the CPPSs and consequently enable access to an I4.0 scenario. This work examined various IT/OT-technologies and introduced an agent-based CPPS with IA topologies and development platforms. Thus, the MARIANNE architecture is proposed, which combines specific research efforts on how the RAMI4.0 concept might be used to address the agent-based CPPS with the IA classes. The PA generates a high-level production plan comprised of executable skills for each RA. AMS contains (potentially numerous) production process sequences for a specific product. The CA allows multiple types of communication among agents, systems, plants, and users of the CPPS by developing GUIs and HMIs. MARIANNE provides a broad overview of how recent advances in these IA design patterns can be linked with other components such as in/output devices, modules, KBs, applications, and other I4.0 components. Those IT/OT integration technologies have motivated affordable Industrial AI gadgets and linked CPPS services to expand the potential of IT/OT-based services. These developments could provide deeper insights into best IA design patterns practices and enable I4.0 technologies further. This study is the first MAS research conducted on a CPPS by IA design patterns aligned with the VDI/VDE 2653-4 and IEEE 2660.1 standards, to the best of our knowledge.

Definitions and classifications of MAS models characteristics currently lack reusability, semantic interoperability, and require more attention in other application domains and I4.0 standardization (see Section 2.2). Therefore, future IA researchers can face those requirements applying MARIANNE to perform a deep analysis of agent-based CPPS features in the next steps. Furthermore, standardized taxonomies and IA design patterns can relate to MARIANNE and migrate aPS to multiple domains, improving semantics and a shared understanding of CPPS (see Section 3.2). Evaluating further aspects of the MARIANNE approach is subject to upcoming works and publications. MARIANNE can also be applied to the smart grid domain by the IA patterns, as shown in the VDI/VDE 2653-4. For example, using more libraries on PADE capable of the MOSAIK [4, 18], and IT/OT platforms available for energy systems [30]. Additionally, to achieve full interoperability, a normalized way of information exchange between HLC and LLC was necessary, as it is a formalized way of invoking the LLC services into PLC and functions from the HLC by the IAs. Here the DT, through a pro-active AAS, provided the standardized way to support information and structure communication between the IAs and thus interoperability. Combining the ECLASS standard could ensure semantic interoperability between IAs (see Section 2.2) [31]. However, the effort for creating AASs manually would increase, even though there are various open tools available, e. g., the AASX Package Explorer, PyI40AAS.

In the future, it can be expected that new IAs will be much more potent than reactive and deliberative ones. For example, it adds learning strategies from analytics, data mining, and ML as a potential benefit of advanced AI [14]. Additionally, the incorporation of modern ML technologies in a new type of IA should be researched to increase the Overall Equipment Effectiveness of a CPPS. Learning methods for IAs have the advantage of generating Predictive Agents and Learnability Agents, which are initially enabled to operate in unknown environments. This type of agent becomes more capable than its fundamental knowledge using the mathematical analysis of ML.

Funding: The authors acknowledge the financial support by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) for the Lighthouse Initiative KI.FABRIK (Phase 1: Infrastructure as well as the research and development program under grant no. DIK0249).

References

1. Bareiss, P., D. Schutz, R. Priego, M. Marcos and B. Vogel-Heuser. 2016. A model-based failure recovery approach for automated production systems combining SysML and industrial standards. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–7, doi: 10.1109/ETFA.2016.7733720.
2. Baumgartel, H. and R. Verbeet. 2020. Service and Agent based System Architectures for Industrie 4.0 Systems. In: *NOMS 2020–2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, doi: 10.1109/NOMS47738.2020.9110406.
3. Cha, S., B. Vogel-Heuser and J. Fischer. 2020. Analysis of metamodels for model-based production automation system engineering. *IET Collab. Intell. Manuf.* 2(2): 45–55, doi: 10.1049/iet-cim.2020.0013.
4. Charpenay, V. et al.2021. MOSAIK: A Formal Model for Self-Organizing Manufacturing Systems. *IEEE Pervasive Comput.* 20(1): 9–18, doi: 10.1109/MPRV.2020.3035837.
5. Cossentino, M., S. Lopes, G. Renda, L. Sabatucci and F. Zaffora. 2019. A metamodel of a multi-paradigm approach to smart cyber-physical systems development. *CEUR Workshop Proc.* 2404: 35–41.
6. Cruz S., L. A., D. Ryashentseva, A. Lüder and B. Vogel-Heuser. 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.* 105(9): 4005–4034, doi: 10.1007/s00170-019-03800-4.
7. DIN SPEC. 2016. *91345:2016-04 Reference Architecture Model Industrie 4.0 (RAMI4.0)*. Berlin, Germany, doi: 10.31030/2436156.
8. Gangoiti, U., A. López, A. Armentia, E. Estévez and M. Marcos. 2021. Model-Driven Design and Development of Flexible Automated Production Control Configurations for Industry 4.0. *Appl. Sci.* 11(5: 2319, doi: 10.3390/app11052319.
9. Hildebrandt, C. et al.2017. Semantic modeling for collaboration and cooperation of systems in the production domain. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, doi: 10.1109/ETFA.2017.8247585.
10. IEEE. 2005. *Foundation for Intelligent Physical Agents FIPA – Specifications*. Retrieved 15 Mar. 2022, from: http://www.fipa.org/repository/standardspecs.html.
11. IEEE. 2021. IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions. *IEEE Std 2660.1–2020*, 1–43, doi: 10.1109/IEEESTD.2021.9340089.
12. ISO/IEC/IEEE International Standard – Systems and software engineering–Vocabulary. 2017. *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, doi: 10.1109/IEEESTD.2017.8016712.
13. Karnouskos, S. 2021. Symbiosis with artificial intelligence via the prism of law, robots, and society. *Artif. Intell. Law* doi: 10.1007/s10506-021-09289-1.
14. Karnouskos, S., P. Leitão, L. Ribeiro and A. W. Colombo. 2020. Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems: Multiagent Systems Entering Industry 4.0. *IEEE Ind. Electron. Mag.* 14(3): 18–32, doi: 10.1109/MIE.2019.2962225.
15. Kovalenko, I., D. Ryashentseva, B. Vogel-Heuser, D. Tilburyand K. Barton. 2019 Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems. *IEEE Robot. Autom. Lett.* 4(3): 2854–2861, doi: 10.1109/LRA.2019.2921947.
16. Lee, E. A.. 2010. Predictability, repeatability, and models for Cyber–Physical systems. In: *Invited talk, Workshop on Foundations of Component Based Design (WFCD) at ESWeek*.
17. Leitão, P., S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser and A. W. Colombo. 2016. Smart Agents in Industrial Cyber–Physical Systems. *Proc. IEEE* 104(5): 1086–1101, doi: 10.1109/JPROC.2016.2521931.
18. Melo, L. S., R. F. Sampaio, R. P. S. Leão, G. C. Barroso and J. R. Bezerra. 2019. Python-based multi-agent platform for application on power grids. *Int. Trans. Electr. Energy Syst.* 29(6): doi: 10.1002/2050-7038.12012.
19. Müller, M., T. Müller, B. Ashtari Talkhestani, P. Marks, N. Jazdi and M. Weyrich. 2021. Industrial autonomous systems: a survey on definitions, characteristics and abilities. *Autom.* 69(1): 3–13, doi: 10.1515/auto-2020-0131.
20. di Orio, G., P. Malo and J. Barata. 2019. NOVAAS: A Reference Implementation of Industrie4.0 Asset Administration Shell with best-of-breed practices from IT engineering. In: *IECON 2019 – 45th Annual Conference of the IEEE Industrial Electronics Society*, pp. 5505–5512, doi: 10.1109/IECON.2019.8927081.
21. Peres, R. S., X. Jia, J. Lee, K. Sun, A. W. Colombo and J. Barata. 2020. Industrial Artificial Intelligence in Industry 4.0 – Systematic Review, Challenges and Outlook. *IEEE Access* 8: 220121–220139, doi: 10.1109/ACCESS.2020.3042874.
22. Platform Industrie 4.0. 2020, *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)*. Berlin, Germany, [Online]. Available from: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html.
23. Plattform Industrie 4.0. 2022. *Platform Industrie 4.0 Glossary*. Retrieved 10 Jan. 2022, from https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/Glossary/glossary.html.
24. Ribeiro, L. and M. Hochwallner. 2018. On the Design Complexity of Cyberphysical Production Systems. *Complexity* 2018: 1–13, doi: 10.1155/2018/4632195.

25. Robinson, A. R., P. J. Haley, P. F. J. Lermusiaux and W. G. Leslie. 2002. Predictive skill, predictive capability and predictability in ocean forecasting. In: *Oceans'02 MTS/IEEE*, vol. 2, pp. 787–794, doi: 10.1109/OCEANS.2002.1192070.
26. Russell, S. and P. Norvig 2021. *Artificial Intelligence A Modern Approach*, 4th ed. Pearson.
27. Schutz, D., M. Schraufstetter, J. Folmer, B. Vogel-Heuser, T. Gmeiner and K. Shea. 2011. Highly reconfigurable production systems controlled by real-time agents. In: *ETFA2011*, pp. 1–8, doi: 10.1109/ETFA.2011.6058991.
28. Sun, B., X. Li, B. Wan, C. Wang, X. Zhou and X. Chen. 2016. Definitions of predictability for Cyber Physical Systems. *J. Syst. Archit.* 63: 48–60, doi: 10.1016/j.sysarc.2016.01.007.
29. Unland, R. 2015. Industrial Agents. In: *Industrial Agents: Emerging Applications of Software Agents in Industry*, Elsevier, New York, pp. 23–44.
30. VDI/VDE. 2021. *2653 Sheet 4: Multi-agent systems in industrial automation – Selected patterns for field level control and energy systems*, [Online]. Available from: https://www.vdi.de/richtlinien/details/vdivde-2653-blatt-4-multi-agent-systems-in-industrial-automation-selected-patterns-for-field-level-control-and-energy-systems.
31. Vogel-Heuser, B., F. Ocker and T. Scheuer, 2021. An approach for leveraging Digital Twins in agent-based production systems. *Autom.* 69(12): 1026–1039, doi: 10.1515/auto-2021-0081.
32. Vogel-Heuser, B., M. Seitz, L. A. Cruz S., F. Gehlhoff, A. Dogan and A. Fay. 2020. Multi-agent systems to enable Industry 4.0. *Autom.* 68(6): 445–458, doi: 10.1515/auto-2020-0004.
33. Wannagat, A. and B. Vogel-Heuser. 2008. Increasing Flexibility and Availability of Manufacturing Systems – Dynamic Reconfiguration of Automation Software at Runtime on Sensor Faults. *IFAC Proc. Vol.*, doi: 10.3182/20081205-2-cl-4009.00049.
34. Zimmermann, P., E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski and P. Zanini. 2019. Skill-based Engineering and Control on Field-Device-Level with OPC UA. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1101–1108, doi: 10.1109/ETFA.2019.8869473.

Bionotes

Luis Alberto Cruz Salazar
Institute of Automation and Information Systems, Department of Mechanical Engineering, TUM School of Engineering and Design, Technical University of Munich, Munich, Germany
luis.cruz@tum.de

Luis Alberto Cruz Salazar, M.Sc., is graduated in Electronic Engineering from the Universidad Antonio Nariño in 2011 and received a master in Electronic Engineering from Universidad del Cauca (2017). He is a Ph.D. candidate at the Institute of Automation and Information Systems at the Technical University of Munich. His main research interests are the design patterns for holons' and agents' development, as well as the Industry 4.0 by intelligent control software in *Cyber-Physical Production Systems*

Birgit Vogel-Heuser
Institute of Automation and Information Systems, Department of Mechanical Engineering, TUM School of Engineering and Design, Core Member of MDSI and Member of MIRMI, Technical University of Munich, Munich, Germany
vogel-heuser@tum.de

Birgit Vogel-Heuser, Prof. Dr.-Ing., is a full professor and director of the Institute of Automation and Information Systems at the Technical University of Munich. Her main research interests are systems engineering, software engineering, and modeling of distributed and reliable embedded systems. She is core member of TUM's MDSI (Munich Data Science Institute), member of TUM's MIRMI (Munich Institute of Robotics and Machine Intelligence), member of the German Academy of Science and Engineering, chair of the VDI/VDE working group on industrial agents, vice chair of the IFAC TC 3.1 computers in control, and was coordinator of the Collaborative Research Centre (CRC) 768: Managing cycles in innovation processes – integrated development of product-service systems based on technical products.