Technische Universität München
TUM School of Engineering and Design

Software Agents with Design Patterns for Industrial Automation Control based on Cyber-Physical Production Systems

Luis Alberto Cruz Salazar

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. rer. nat. Thomas Hamacher

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Birgit Vogel-Heuser
2. apl. Prof. Dr.-Ing. habil. Arndt Lüder
3. Assoc. Prof. Dr. Elisabet Estévez Estévez

Die Dissertation wurde am 19.03.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 17.07.2024 angenommen.

„Die ihn aber aufnahmen und an ihn glaubten, denen gab er das Recht,
Kinder Gottes zu werden...“
Johannes 1:12

“But as many as received him, to them gave he power to become the sons of God,
even to them that believe on his name…”
John 1:12

Main peer-reviewed publications

In the following, the selected peer-reviewed journal and conference papers, which primarily contributed to the context of this dissertation, are listed.

I. **Cruz S. LA**, Vogel-Heuser B (2017) Comparison of agent oriented software methodologies to apply in cyber physical production systems. In: 15th International Conference on Industrial Informatics (INDIN). IEEE, Emden, Germany, pp 65–71. https://doi.org/10.1109/INDIN.2017.8104748

II. **Cruz S. LA**, Mayer F, Schütz D, Vogel-Heuser B (2018) Platform Independent Multi-Agent System for Robust Networks of Production Systems. IFAC-PapersOnLine 51:1261–1268. https://doi.org/10.1016/j.ifacol.2018.08.359

III. **Cruz S. LA**, Ryashentseva D, Lüder A, Vogel-Heuser B (2019) Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. Int J Adv Manuf Technol 105:4005–4034. https://doi.org/10.1007/s00170-019-03800-4

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

V. **Cruz S. LA**, Vogel-Heuser B (2022) Industrial Artificial Intelligence: A Predictive Agent Concept for Industry 4.0. In: 20th International Conference on Industrial Informatics (INDIN). pp 1–6. https://doi.org/10.1109/INDIN51773.2022.9976159

Complementary peer-reviewed publications

In the following, the other peer-reviewed journal and conference papers are listed, which contributed in a complementary manner to the context of this dissertation.

VI. Vogel-Heuser B, Ryashentseva D, **Cruz S. LA**, et al (2018) Agentenmuster für flexible und rekonfigurierbare Industrie 4.0/CPS- Automatisierungsbzw. Energiesysteme. In: VDI-Berichte (ed) Automation 2018, 1st ed. VDI Verlag, Düsseldorf, pp 1119–1130. https://doi.org/ https://doi.org/10.51202/9783181023303-1119

VII. Lüder A, Zawisza J, **Cruz S. LA**, et al (2018) Identifying Design Pattern for Agent Based Production System Control. In: 44th Annual Conference of the IEEE Industrial Electronics Society, IECON. IEEE, Washington D.C., USA, pp 2896–2901. https://doi.org/10.1109/IECON.2018.8591336

VIII. Ryashentseva D, **Cruz S. LA**, Vogel-Heuser B, Lüder A (2018) Development and evaluation of a unified agents- and supervisory control theory based manufacturing control system. In: 14th International Conference on Automation Science and Engineering (CASE). IEEE, Munich, Germany, pp 187–192. https://doi.org/10.1109/COASE.2018.8560539

IX. Vogel-Heuser B, Seitz M, **Cruz S. LA**, et al (2020) Multi-agent systems to enable Industry 4.0. at - Automatisierungstechnik 68:445–458. https://doi.org/10.1515/auto-2020-0004

X. Haben F, Vogel-Heuser B, Najjari H, Seitz M. Trunzer E, **Cruz S. LA** (2021) Low-entry Barrier Multi-Agent System for Small- and Middle-sized Enterprises in the Sector of Automated Production Systems. In: IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, pp 1351–1357. https://doi.org/10.1109/IEEM50564.2021.9672973

XI. Seitz M, Gehlhoff F, **Cruz S. LA**, et al (2021) Automation platform independent multi-agent system for robust networks of production resources in industry 4.0. Journal of Intelligent Manufacturing 32:2023–2041. https://doi.org/10.1007/s10845-021-01759-2

Standard publication

In the following, a standard made a particularly important contribution to this dissertation.

XII. VDI/VDE (2021) 2653 Sheet 4: Multi-agent systems in industrial automation - Selected patterns for field level control and energy systems. Available in: https://www.vdi.de/en/home/vdi-standards/details/vdivde-2653-blatt-4-multi-agent-systems-in-industrial-automation-selected-patterns-for-field-level-control-and-energy-systems

Acknowledgments

First of all, I dedicate this project and my entire university career to the *LORD* for being the one who prospers and blesses me at every moment of my life, making it a source of success and filling it with the best experiences.

I am deeply grateful to *Prof. Birgit Vogel-Heuser*, who allowed me to follow the PhD under her supervision. Thanks to her excellence and dedication, I am allowed today to culminate this professional dream, and it has definitely been a great benefit her support in the whole process, without her help this dream was not possible. I want to also thank *Prof. Luis Ribeiro* for his accurate feedback on my contribution.

To the Technical University of Munich (TUM) and the Universidad Antonio Nariño (specifically PFAN scholarship) for providing me with the tools, academic support, and economic resources necessary to train as a Doctor-Engineering in the wonderful country of Germany, facing the challenge of transforming my environment with dedication and perseverance. Thanks to both institutions and the Ministry of Science and Technology of Colombia Minciencias (call 756 Doctorates abroad) for their trust and financial support.

Throughout my time at the Institute of Information Systems AIS-TUM, thank my colleagues *Dr.-Ing. Juliane Fischer* and *Dr.-Ing. Suhyun Cha*, for their constant support. They patiently and wisely guided me in the content of most of my publications. Also, thank *Dr.-Ing. Emanuel Trunzer* for the support and wishes in this thesis. I take with me the best memories of the TUM staff for welcoming me and giving me all the required support to reach this work.

To all those people who are important in my life and to whom I owe my time for my professional sacrifices, especially my daughters *Mariana Cruz* and *Paloma Cruz*. To all my family and friends, I thank you from the bottom of my heart for your support since I know that, in one way or another, you participated and paved the way for achieving my training. Now, I project myself as a capable and suitable person to teach in my personal and professional environment, following the German high-quality education.

Thank you to each of you for your teachings, family, friends and colleagues, the list would be endless, but rest assured that each one of you is part of this achievement, your time and appreciation in my life means that it has also been your work!

List of Abbreviations

AI	Artificial Intelligence
AMS	Agent Management System
AOSE	Agent Oriented Software Engineering
CA	Communication Agent
CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
AIS	Institute of Automation and Information Systems
AAS	Asset Administration Shell
aPS	automated Production System
I4.0	Industry 4.0
IA	Industrial Agent
IIoT	Industrial Internet of Things
IT	Information Technology
KB	Knowledge Base
DF	Directory Facilitator
DT	Digital Twin
MAS	Multi-Agent System
ML	Machine Learning
OT	Operational Technology
PLC	Programmable Logic Controller
POU	Program Organization Unit
PA	Process Agent
PPR	Product, Process, Resource
RA	Research Agent
RAMI4.0	Reference Architecture Model Industry 4.0
Req	Requirement
RQ	Research Question
TUM	Technical University of Munich

Table of Contents

Contents

List of Figures

List of Tables

1. Introduction and motivation

Within the framework of *Industrie 4.0 (I4.0)*, interconnections, smart sensors, actuators, and other equipment are becoming more common in an *automated Production System "aPS"*. Multiple aPSs encompass networked entities, also known as a *Cyber-Physical Production System "CPPS"* (Vogel-Heuser et al. 2015a), or an industrial *Cyber-Physical System "CPS"* (Ribeiro and Hochwallner 2018). In this sense, industry and academia discuss CPPS's benefits and common ground with agents as a way to develop CPPSs in various aPS domains, e.g., manufacturing and smart grids (Leitão et al. 2016). Nevertheless, what exactly is an agent for the I4.0? Is it possible to categorize Industrial Agents "IAs" as a type of Artificial Intelligence "AI"? Or does it remain just a software program, as discussed in (Franklin and Graesser 1997)?

1.1 Motivation for automated Production Systems with Multi-Agent Systems

Russell and Norvig define that the agent is an entity that "just acts", because it comes from the Latin *agree* (meaning *to do*). Those authors also introduce the *Rational Agent* concept as part of their AI categories, as *"one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome"* (Russell and Norvig 2010). They use a taxonomy for the following AI system' categorization: i) *thinking humanly*, e.g., artificial neural networks and other cognitive methods; ii) *acting humanly*, e.g., humanoid robots with natural language processing; iii) *thinking rationally*, e.g., expert systems or rules of inference and optimization; and iv) *acting rationally*, e.g., intelligent software agents that are expected to achieve goals. Figure 1 represents this AI categorization, adding industrial application contribution and its complexity addressed by typical IAs (e.g., Resource agent, Process agent).

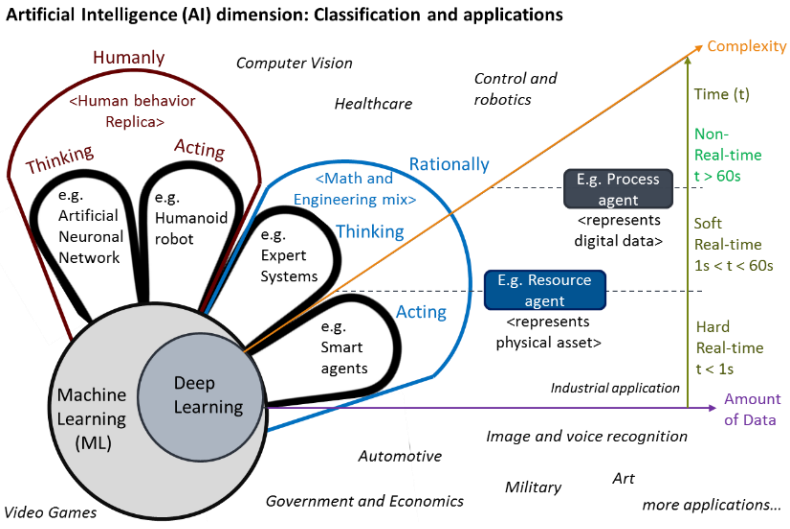

Figure 1: Artificial Intelligence categories and the Industrial Agent application. Source: Based on (Russell & Norvig, 2010).

As a consequence of agent seminal definitions, it is possible to say that in a control system, a human and an agent's behavior are defined by a goal-orientated approach (Telang et al. 2019; Heylighen 2023). An agent's behavior represents a specific combination of tasks, but these tasks are not unique. For example, a human can select any route to resolve math calculations (e.g., multiplications are a sums compound). Agents, therefore, are distinct entities with the ability to take a goal-oriented approach. Agents are able to complete goals with autonomy, similar to humans, but (as far as recent experts demonstrate), they cannot reach the full AI autonomy level yet (Plattform Industrie 4.0 2019). The authors in (Leitão et al. 2016), add modularity, flexibility, robustness, and responsiveness to IA features, which are not entirely part of human behavior.

Along these lines "Artificial Intelligence in Industrie 4.0" is a technical report published by the working groups on "Technological and Application Scenarios" and on AI of the I4.0 platform that presents an Industrial AI concept level yet (Plattform Industrie 4.0 2019). The most relevant conclusion is that I4.0 experts and scientists must become accustomed to the behavior of autonomous AI-controlled systems, collaborate with them, and even comply with their requirements. In this way, initiatives related to IAs instantly raise many concerns about existing norms and new standardization. These regulations often provide guidelines and, in some cases, offer procedures driven by IA design patterns (Ribeiro et al. 2018; Leitão et al. 2021). An IA is one way of achieving I4.0 systems due to natural autonomy and additional intelligent features, e.g., reactiveness, proactiveness, and human collaboration. Thus, collaborative and grouped IAs (named sub-agents) are defined as a Multi-Agent System (MAS), which is particularly well suited for representing distributable AI able to develop industrial CPSs (Karnouskos et al. 2020a) and to use in I4.0 scenarios (Vogel-Heuser et al. 2020; Seitz et al. 2021).

1.2 Why Industrial Agents and Cyber-Physical Production Systems?

A CPPS consists of *"intelligent entities that collaborate and exchange information globally, and they are proclaimed as the basis of Industry 4.0. A CPPS enables characteristics of Cyber-Physical Systems in the production automation domain"* (Vogel-Heuser et al., 2015). Hence, I4.0/CPPS usually refers to the *Fourth Industrial Revolution.* In the industrial context, according to German FA 3.35[1] VDI/VDE experts' standardization, an IA is *"an encapsulated (hardware/software) entity with specified objectives. An agent endeavors to reach these objectives through its autonomous behavior, in interacting with its environment and with other*

[1] Previously FA 5.15, "Agent systems" is a working group of the Society of German Engineers (VDI) and German Electrical Engineers (VDE).

agents" (VDI/VDE, 2012). At the same time, TC-IA[2] by the IEEE P2660.1 working group normalized the IA definition as an *"agile and robust software entity that intelligently represents and manages the functionalities and capabilities of an industrial unit"* (IEEE, 2021). Concerning AI, German experts define it as *"supplements technical systems with the ability to process tasks independently and efficiently"* (Plattform Industrie 4.0, 2019). Nonetheless, those definitions are limited; they do not answer how IA/AI acquire their intelligence. Then, there are multiple and generally accepted definitions of both terms IA/AI that are ambiguous and far from the same between their communities. To bridge these discrepancies and provide a cohesive understanding by research questions (RQs), reference can be made to the pivotal Fig. 2.

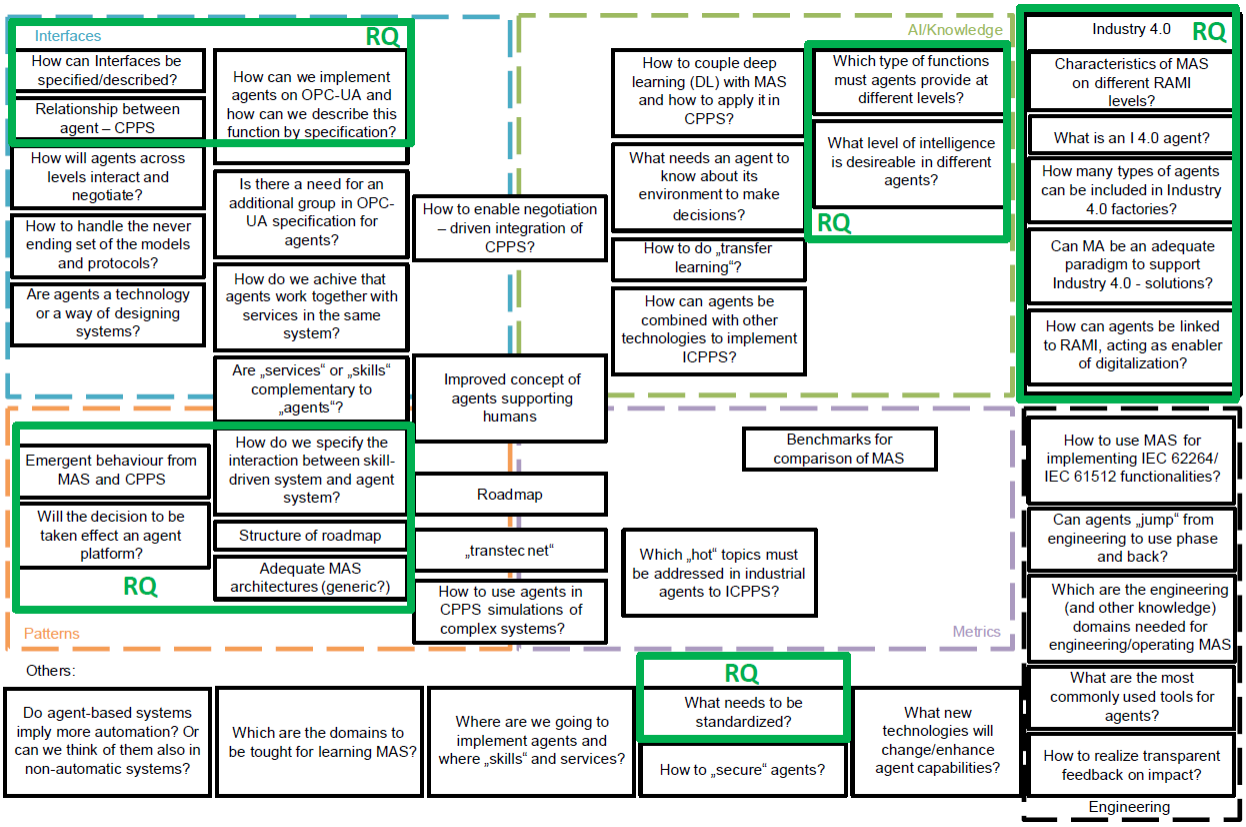

Figure 2: Challenges and gaps of Industrial Agents.
Highlighted green RQs refer to the focus of this dissertation, which started in 2016, and based on those, are created the thesis' RQs; IA experts subsequently proposed the other RQs. Source: Adapted from the presentation of the workshop[3] "Agents in agile manufacturing (CPPS) - Status of Last Meeting", AIS-TUM, 2019.

[2] TC-IA refers to the international IEEE-IES Technical Committee on Industrial Agents.
[3] Following are the participants of the agent-based CPPS workshop and their affiliations: Andrei Lobov (Norwegian University of Science and Technology), Armando Colombo (University of Applied Sciences Emden/Leer), Arndt Lüder (Otto von Guericke University Magdeburg), Birgit Vogel-Heuser (Technical University of Munich), Christoph Hanisch (FESTO AG), Elfahaam Haitham (RWTH Aachen University), Friedrich Durand (afag Automation AG), Kira Barton (University of Michigan), Luis Ribeiro (Linköping University), Marga Marcos (University of the Basque Country), Paulo Leitão (Polytechnic Institute of Bragança), Peer Stritzinger (Erlang Ecosystem Foundation), Peter Göhner (University of Stuttgart), Stamatis Karnouskos (SAP), Ulrich Epple (RWTH Aachen University), and Valeriy Vyatkin (Aalto University).

Figure 2 recompiles a comprehensive list of questions, highlighting the numerous challenges and gaps associated with agents: interfaces, AI/Knowledge, engineering, CPPS, I4.0, patterns, standardization, metrics, and other interlaced IA concepts. Also, this figure illustrates specific RQs that are central to this dissertation, serving as both an informative backdrop and a compass guiding the formulation of the thesis' issues statements and the RQs. The origins of this figure were a presentation from the Workshop titled "Agents in agile manufacturing (CPPS)" held at AIS-TUM in 2019. It is imperative to mention that these discussions and subsequent findings in the workshop were the collaborative endeavors of experts from both IFAC 3.35 NMO GMA VDI/VDE and IEEE IES TC-IA by the IEEE P2660.1 working groups. This collaboration signifies the synthesis of knowledge, experiences, and expertise from renowned IA experts trying to shape the future landscape of agent-based CPPS.

1.3 Delimitation and key differentiation of this thesis

Considering the properties of IAs and their relevant standards, this cumulative thesis presents a MAS architecture to understand the aspects of the flexible and intelligent CPPS. For this thesis, a CPPS often refers to I4.0 and the MAS approach as well as in (Colombo et al., 2021; Gangoiti et al., 2021; Karnouskos et al., 2020; Tang et al., 2018; Vogel-Heuser et al., 2015). Regarding I4.0, the thesis refers to precisely the *Asset Administration Shell (AAS)*, which is one of the main specifications of the *Reference Architecture Model for I4.0* or *RAMI4.0* (DIN SPEC 91345 norm (DIN SPEC, 2016)). AAS is the Digital Twin (DT) of assets that form the I4.0 components, and together with IA, AAS allows smart access to resource information, as well as connectivity with other I4.0 components (Cruz S. et al., 2019). As a result, relevant IA pattern standards and their AI challenges for the I4.0 show how MAS can be overcome with the help of identified IA skills, capable with the Product, Process, Resource (PPR) model (Cruz S. & Vogel-Heuser, 2022a).

This dissertation focuses on IAs design patterns and their capability for the standardized I4.0 concepts (e.g., RAMI4.0, AASs, PPR), proposing an agent-based CPPS architecture to achieve smart production. As introduced by Fig. 3, the MAS proposed is delimited to CPPSs that are able to provide both soft and hard real-time responsiveness, as well as to facilitate vertical integration across various levels of industrial control automation in the specific domain of discrete manufacturing. The innovative differentiation of this thesis are the IA design patterns embedded within an agent-based CPPS that are specifically designed to synergize with core components of industrial automation systems, including Supervisory Control and Data Acquisition (SCADA),

Manufacturing Execution Systems (MES), and Enterprise Resource Planning (ERP) platforms, which are aligned to RAMI4.0 and its AAS concept as demonstrated by (Cruz S. et al., 2019; Cruz S. & Vogel-Heuser, 2022a).

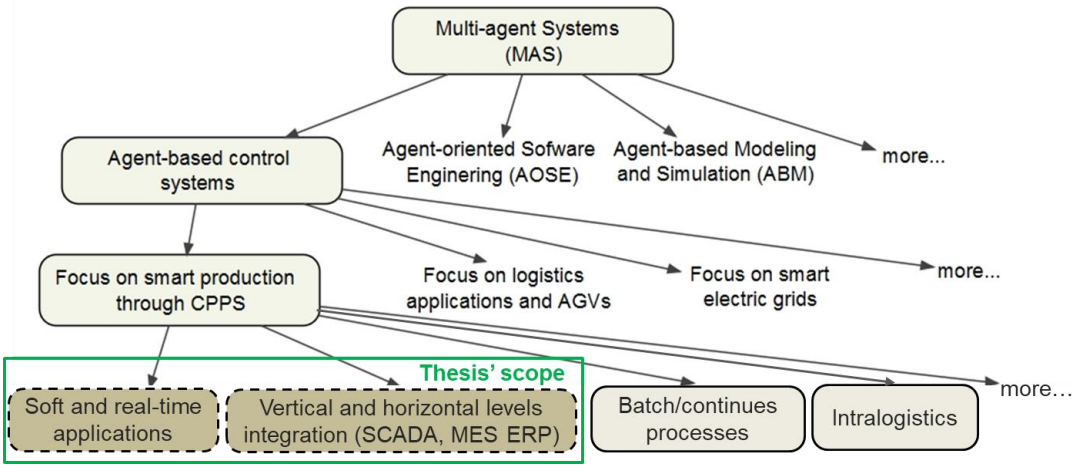

Figure 3: Thesis' scope of the agent-based CPPS proposed for industrial control automation within the manufacturing field.

1.4 Scientific problem: issue statements

The research presented in this dissertation was conducted with the collaborative German FA 3.35 VDI/VDE experts. Based on experiences and feedback made in this working group that involves researchers, industry representatives, and engineers from many scientific disciplines focused on IAs, the recent standard *"2653 Sheet 4: Multi-agent systems in industrial automation - Selected patterns for field level control and energy systems"* (VDI/VDE, 2021), was developed. Altogether, these result in four significant issues that influence traceability in the context of agent-based CPPS based on design patterns, which are briefly discussed in the following:

Issue 1: Lack of a comprehensive overview and classification of MAS patterns in CPPS

Despite the increasing relevance of MAS patterns in CPPS, there remains a gap in the literature offering a consolidated overview and classification of these patterns, involving their depiction, criteria, domains of applicability, and reusability. This lack of information impedes the extensive development of MAS in the industrial field, e.g., logistic, smart manufacturing, smart grids.

Issue 2: Challenge of reusability and extendibility of MAS design patterns for I4.0

As the demand for adaptable and scalable CPPS increases, the need for reusable and extendible MAS design patterns becomes more relevant. Yet, there remains uncertainty regarding which MAS design patterns are universally reusable and how they can be further extended to supply emerging I4.0 scenarios.

Issue 3: Integration of MAS design patterns with existing CPPS models and standards

There's a need to ensure that MAS design patterns are consistent with established CPPS models and standards, like RAMI4.0 and PPR. However, it's unclear how these patterns can be effectively integrated and developed to align with such models.

Issue 4: Implementing sub-agent patterns and AASs into hybrid CPPS platforms.

While sub-agent patterns and AASs offer promise in enhancing CPPS functionalities, there is a lack of clear guidelines on how to seamlessly implement these into hybrid CPPS.

1.5 Research questions, and main contributions

This work addresses four research questions (RQ1-RQ4) that recompile RQs from Fig. 2 and aim to solve the issues above:

RQ1. *How are the MAS patterns for CPPS depicted and what criteria are used to describe them?*

RQ2. *For which domains of CPPS are the MAS patterns designed and applicable?*

RQ3. *What are the reusable MAS design patterns for CPPS?*

RQ4. *How can MAS patterns be used for a CPPS aligned with the RAMI4.0 and PPR model?*

The potential benefits of this thesis' contributions (Con1-Con5) are the following:

Con1. Systematic and well-discussed criteria (or abstraction for the summary) compiled at least in the IA working group of the German IFAC FA 3.35 is presented.

Con2. A mapping of analyzed MAS functional requirements to sub-agent patterns will be provided, considering their capabilities and skills.

Con3. Proposed sub-agent patterns for MAS technology in I4.0 demonstrators are applicable; further extended agent-based CPPS designs and applications are possible for more use cases based on selected I4.0 scenarios.

Con4. The identified design patterns are the basis for the development of agent-based CPPS and for their structural representation (CPPS requirements). The contribution considers an explicit MAS architecture (with final requirements) for the application of an individual CPPS in process industry domain.

Con5. In order to improve industrial applicability, a VDI/VDE norm is used as proof of evaluation for the impact of IA patterns and AASs implementation into hybrid CPPS platforms.

1.6 Structure of this dissertation

The remainder of this cumulative thesis is structured as follows. Chapter 2 presents the research methods and conceptual background, including the related work. Chapter 3 describes the main contributions to CPPS. Chapter 4 summarizes the findings of the included publications. Chapter 5 outlines the contribution of the research, presents some inferences from the findings, and suggests future research directions. The included manuscripts are listed in the Appendix A.

2. Research method and conceptual background

In this thesis, the paradigm of the holistic research method is adopted. It means there are multiple views to understanding a system, called holism (Hurtado 2012), using tools for observing, learning, and depicting what is perceived qualitatively and quantitatively of concepts.

2.1 Research method and strategy stages

Holism could indicate different thoughts, but they must be considered complementary. Therefore, the objectives of the holistic paradigm are classified according to their complexity levels and those have a common goal hierarchy, as depicted in Fig. 4.

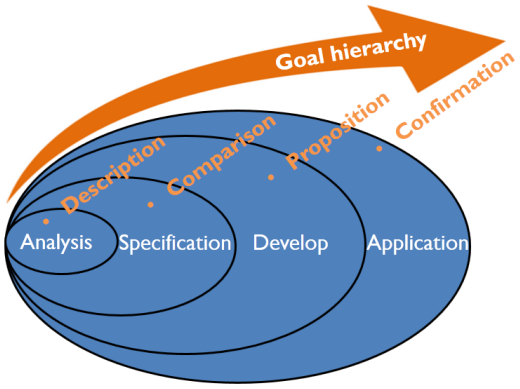

Figure 4: Levels of research objectives in a holistic research method. Source: Adapted from (Hurtado 2012).

In a holistic research method, objectives are organized hierarchically, from the lowest to the highest relevance. The hierarchy of the research objectives levels tracks the following order: perceptual (description), apprehensive (comparison), comprehensive (proposition), and integrative (confirmation). Figure 4 illustrates the general method to obtain this thesis' research goals based on the holistic paradigm (Hurtado, 2012). This proposal will reach the Integrative (confirmative) holistic paradigm level since the objectives of developing agent-based CPPS driven by design patterns involve the researcher's amendment of the event. Therefore, to fulfill

all the objectives of this research project, the following four sequential macro stages and their descriptions are established by inductive[4] reasoning:

I. State of the art and theoretical framework: documenting, analyzing, and characterizing the outstanding models of IA design patterns for CPPS based on the state-of-the-art review.

II. Analysis and design of the model: establishing the requirements and identifying IA design patterns in order to propose an agent-based CPPS.

III. Implementation and validation: applying a MAS architecture to evaluate the agent-based CPPS driven by design patterns, validating the effectiveness of the design.

IV. Publications and dissertation: realizing the monograph and the other publications requirements during the validation of the proposed agent-based CPPS.

2.2 IA design pattern definition

In this thesis, the IA design pattern definition is based on the IA standardization (from FA 3.35 VDI/VDE, see section 1.2) and the "design pattern" term, which provides a means of identification of broader success aspects in particular problems. Design pattern definition has been adapted for various other disciplines, particularly in software engineering (Gamma et al., 1994). Moreover, the original idea of patterns was introduced by the architect Christopher Alexander *et al.* as a reusable form of a solution to a design problem (Alexander et al., 1977):

> *"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."*
>
> *—Alexander et al., (1977).*

Design patterns usually aim to improve the flexibility of object-oriented systems (Gamma et al., 1994), as well as MAS in nature, IA patterns are a research and development field considered to enable flexibility, robustness, and responsiveness to industrial automation systems (Leitão et al., 2018). Within this field, several MAS architectures have been developed over the last 25 years with the intention of providing distinct manufacturing system capabilities (Lüder et al., 2017). Simultaneously in Information Technology (IT) domain, IA experts realized that it is time to consider similarities and differences between the emerged and independent MAS approaches (Ribeiro et al., 2018). MAS patterns need to be also investigated for large-scale systems in CPPS (Colombo et al., 2013). As a result, a more recent description of MAS design patterns was given:

[4] *Inductive* reasoning, or *induction*, involves forming general theories from specific observations. For example, observing something happen repeatedly and concluding that it will happen again in the same way. Source: https://www.dictionary.com/e/inductive-vs-deductive

> *"For each design pattern it is assumed, that it will be described by a descriptive and unique name, a description of goals and reasons of the design pattern and its use, a description of the problem intended to be used by the design pattern, a description of the usage restrictions of the design pattern, a description of the solution the pattern provides including a naming of all relevant entities and their interaction (possibly accompanied by a graphical representation of the reach structure), a description of the impact reached by using the described solution, if possible known applications of the design pattern, and if required other design pattern related to the described design pattern."*
>
> *—Lüder et al., (2017)*

However, the IA expert's community has realized that there is not a formal agreement about IA patterns definition and incurred unintentionally different types of terms, such as: common practices (domain templates) for software agents in low-level automation (IEEE, 2021); or blueprints for the design and realization of MAS (VDI/VDE, 2021). Therefore, there is no widely accepted IA design pattern definition currently. Based on this consideration, an IA design pattern definition for agent-based CPPS control might be derived, enabling control engineers to select a MAS approach within the development of well-proven criteria. This is the primary intention of this thesis section. Consequently, the IA design pattern is defined as follows:

Definition of this thesis: Industrial Agent pattern

An IA design pattern is characterized as a structured approach[5] that delineates the core of a solution to a recurrent issue in industrial systems, adapted in a manner that allows —but not limited to— agent-based CPPS applications. This approach is not merely a template but a comprehensive method that includes a distinctive and descriptive MAS name, an explication of the pattern's objectives and the rationale behind its use, and a detailed description of the problem it aims to solve. It also includes the constraints under which the design pattern operates, a thorough description of the solution provided —including the relevant sub-agent entities and their interactions, often accompanied by a graphical illustration of the structure— and the anticipated impact of employing the proposed solution. Where applicable, known applications of MASs and any associated or complementary design patterns are also described.

IA patterns here refer to the MAS for the industrial automation system domain, mainly agent-based CPPS. Hence, this definition is intended to bridge the gap between various terminologies, offering control engineers a coherent set of MAS-proven structures. Additionally, the IA design

[5] In this thesis, the term "approach" is conceptualized as a collection of architectures, methodologies, and/or standards that adhere to a common scheme, as introduced in (Cruz & Vogel-Heuser, 2017; VDI/VDE, 2021). Concerning architectures, they are exclusively acknowledged as configurations for static system modeling. Frequently, these configurations are proposed by their authors and may lack detailed procedural guidance for implementation. A methodology is defined as a prescribed sequence of actions designed to enhance efficiency in development and to elevate the quality of systems, commonly within the context of software engineering. It further delineates how processes should be systematically, predictably, and reliably executed. Both architectures and methodologies may receive accreditation from global standardizing institutions. A manufacturing standard might be of the private or open variety, depending on the nature of the standard development organization.

pattern definition aims to mitigate the possible *apophenia*[6], which means the potential for MAS developers to perceive false patterns or assign unnecessary significance to unrelated events and entities within complex systems. By providing a clear, well-established set of criteria for the identification and utilization of IA design patterns, this thesis endeavors to ground the development process of MAS design patterns. As a proof of the concepts, empirical evidence has been validated by IA experts and their recent standardization (IEEE, 2021; VDI/VDE, 2021).

2.3 Requirements for MAS architectures for I4.0/CPPS

Hence, the formulated requirements listed in Table 1 need to be fulfilled to assist in analyzing, categorizing, implementing, and evaluating IA patterns in agent-based CPPS.

Table 1: Rating scheme of requirements related to issues (Section 1.4) to evaluate relevant work.

Req1-Classification – Criteria of MAS classification Related to Issue 1: Lack of comprehensive MAS overview	
●	Detailed classification criteria for MAS approaches delivers valid and decidable information for their evaluation
◑	Limited MAS approaches classification for CPPS or not classifying/identifying with similar design pattern's terms, e.g., names, functionalities, etc.
○	No classifications or criteria defined.
Req2-Domain – CPPS application field Related to Issue 1: Lack of comprehensive MAS overview	
●	Support of MAS approaches have application in diverse domains with different goals and benefits e.g., flexibility, adaptability, etc.
◑	Limited CPPS are applicable in every domain in appliance with the real-time requirements of MAS approaches
○	No consideration of a CPPS domain.
Req3-Reusability – MAS design patterns for I4.0 Related to Issue 2: MAS extendibility	
●	There are reusable MAS patterns and sub-agents with functional and non-functional requirements for CPPS design
◑	Limited MAS components follow specific sub-agents, which have particular aims and are reusable for CPPS design
○	No consideration of IA patterns or sub-agent patterns.
Req4-Modelling – Support of models associated to I4.0 Related to Issue 3: CPPS models, and Issue 4: AAS into CPPS	
●	It is possible to harmonize different MAS approaches to obtain a simple CPPS architecture aligned with RAMI4.0/PPR
◑	Limited MAS patterns provide I4.0 component's properties and specific information to its AAS or PPR model
○	No consideration of RAMI4.0 or PPR modelling

2.4 Related work: how do IAs contribute to the CPPSs?

Concurrently, I4.0 is founded on design concepts that are not fully satisfied by the automation languages widely used today. The new design concepts, such as Industrial CPS or CPPS and the DT, aim to provide more customized goods and optimization techniques while addressing the urgent requirement to increase sustainability (Ribeiro & Gomes, 2021). The IEC 61131-3 and

[6] *Apophenia* is the tendency to find patterns or meanings where other people do not, perceiving meaningful connections between unrelated things. In 1958, psychiatrist Klaus Conrad (Berlin, 27.02.1986) introduced the German term, *Apophänie*, from the Greek verb *ἀποφαίνειν* (apophaínein).

IEC 61499 standards and their subsequent enhancements, which now include an object-oriented programming language, are the dominant programming languages in the control of aPS, especially at the plant level (Cruz S. & Vogel-Heuser, 2020). At this industry field level, Programmable Logic Controllers (PLCs) are primarily utilized to control whole aPSs, networking them to create CPPS (Karnouskos et al., 2019). Research in CPPS and DTs, driven by AASs, has significantly enhanced the development of metamodels (López-García et al., 2023). These model-based approaches detail the structure of interconnected machines and systems, supporting the optimization and reusability of PLC pattern codes (Fischer, Vogel-Heuser, Berscheit, et al., 2021).

In the context previously described, IAs contribute to the CPPS optimization by addressing the challenges and needs of modern aPS (Karnouskos et al., 2019; Vogel-Heuser et al., 2020). Based on a systematic keyword search using the tool Scopus AI[7], three keyways were derived in which IAs can optimize CPPS, as follows:

Design patterns and interfaces: IAs help to design and implement CPPS by providing design patterns and interfaces between agents and the systems (Leitão et al., 2021; VDI/VDE, 2021). These design patterns and interfaces enable continuous and collaborative communication between the IAs and modern aPSs.

Metrics for evaluation: IAs contribute to the optimization of CPPS by providing metrics to evaluate the quality of agent-based syetems (Karnouskos et al., 2018; Ribeiro et al., 2018). These metrics assess the performance and effectiveness of the aPS, leading to improvement and optimization.

Distributed intelligence (smartness): IAs implement distributed AI within CPPS (Karnouskos et al., 2020), allowing, for instance, decentralized decision-making and flexibility in task allocation, leading to improved efficiency and adaptability of the systems (Land et al., 2023).

Overall, IAs play a crucial role in optimizing CPPS by providing design patterns, interfaces, metrics for evaluation, and distributed intelligence. As Fig. 5 depicts, IAs are able to address CPPS challenges and improve the industrial automation control of modern aPS (Leitão et al., 2023; Ribeiro & Gomes, 2021).

[7] *Scopus AI* is an intuitive and intelligent search tool powered by generative AI (GenAI) that delivers insights with unprecedented speed and clarity. Scopus AI uses natural language processing, which means that it goes beyond matching specific keywords or Boolean operators; instead, it is possible to type questions, statements, or hypotheticals using everyday language. More info online: https://www.elsevier.com/products/scopus/scopus-ai

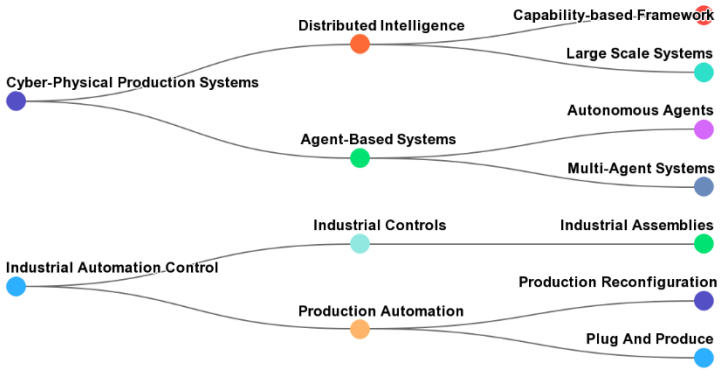

Figure 5: Comprehensive concept maps from Scopus AI, after the question (Dec 2023): How do industrial agents contribute to the optimization of Cyber-Physical Production Systems?

Considering those challenges for agent-based CPPS, Table 2 to 4 summarizes the requirement fulfillment of all requirements presented (Req1-Req4, see Table 1) for CPPS, MAS and patterns approaches, respectively. IA patterns classification is like the Product-Resource-Order-Staff Architecture (PROSA) in which holons are defined for resources, products, orders, and so-called staff services (Valckenaers, 2020). These IA types have been well-researched and are suitable for various MAS architectures (Gehlhoff, 2023; Ribeiro & Gomes, 2021).

Table 2: Evaluation of relevant work (sorted into group/leader) based on rating scheme in Table 1. CPPS approaches

Group/ leader	***Requirement*** / **Work**	***Req1- Classification***	***Req2- Domain***	***Req3- Reusability***	***Req4- Modelling***
Barata	(Barata et al., 2022)	○	◑ MM farm	◑ PL	○
	(Peres, 2019)	◑ AS	◑ generic aPS	◑ PL	◑ RG
	(Rocha, 2018)	◑ AS	◑ predictive aPS	◑ PL	◑ RG
Leitão & Ribeiro	(Colombo et al., 2021)	○	● aPS/smart grid	◑ PL	◑ ISA 95/88
	(Leitão, Colombo, et al., 2016)	◑ AS	● aPS projects	○	○
	(Ribeiro & Hochwallner, 2018)	○	● generic CPPS	○	◑ RG
Cardin & Trentesaux	(Barbosa, 2016)	◑ AS	◑ MM ADACOR2	◑ holons	○
	(Cardin, 2019)	◑ AS	◑ aPS	○	○
	(Jimenez et al., 2017)	◑ AS	◑ hybrid control	◑ entity	○ Pollux
	(Nouiri et al., 2019)	○	● smart grid	○	○
Lüder	(Calà, 2019)	○	● aPS	◑ PL	○
	(Lüder et al., 2020)	○	● aPS	○	◑ AAS
	(Zawisza, 2019)	◑ AS	● aPS	◑ PL	○
Other CPPS developers	(Case, 2015)	○	● smart grid	◑	○
	(E. A. Lee, 2015)	○	● generic CPS	○	○
	(J. Lee et al., 2015)	○	● generic CPPS	○	○
	(Panetto et al., 2019)	○	◑ discrete aPS	○	◑ RG
	(Váncza & Monostori, 2017)	○	● bioinspired aPS	○ Ueda's legacy	○

Standard	RAMI4.0 (DIN SPEC, 2016)	○	● aPS	○	◑ AAS
TUM-AIS	(Schütz et al., 2017)	○	● aPS	○	○
	(Trunzer, 2020)	○	● aPS	○	◑ RG
	(Vogel-Heuser et al., 2022)	○	● aPS	◑ PL	○
Zoitl	(Shakil & Zoitl, 2020)	○	● aPS	◑ IEC 61499	◑ RG

aPS: automated Production System;
CPS: Cyber-Physical System;
AS: limited agent survey (literature review without very well-proven criteria classification);
MM/MS: MAS methodology / MAS software for the industrial domain;
PL: Patterns has limited consideration (only patterns mentioned, no method proposed);
RG: Reference general assessment (RAMI4.0 in general, not focusing on specific AAS/PPR concepts)

Table 2 systematically rates the extent to which current CPPS approaches meet a series of predefined criteria. It identifies the efforts of Colombo et al (2021), Ribeiro & Hochwallner (2019), E. Lee (2015), and Lee et al. (2014) as particularly impactful in domain-specific applications, highlighting need to achieve CPPS flexibility and responsiveness. The table, however, also shows notable variance in the IA patterns reusability and modeling support based on RAMI4.0/PPR models of these approaches (Req3), suggesting a crucial need for a standardized, interoperable framework that enhances CPPS implementation efficiency, as analyzed by (Leitão & Strasser, 2016).

Table 3: Evaluation of relevant work (sorted into group/leader) based on rating scheme in Table 1. MAS approaches.

Group/ leader	***Requirement*** **Work**	***Req1- Classification***	***Req2- Domain***	***Req3-Reusability***	***Req4- Modelling***
Agent platform	(Kruger & Basson, 2019)	○	◑MS Java/JADE	◑ FIPA patterns	○
	(Melo et al., 2019)	◑ platform criteria	◑MS Python/PADE	◑ FIPA patterns	○
Agent project	(Cruz & Vogel-Heuser, 2017)	◑ AOSE criteria	◑ aPS	○	◑ ISA 95 levels
	(Wright, 2001)	○	◑ MM SAAM	◑ IT pattern	○
Agent/IA standard	(IEEE, 2005)	○	◑ MM FIPA	◑ IA norm	○
	(VDI/VDE, 2012)	○	● aPS	◑ IA norm	○
AOSE	(Cardoso & Ferrando, 2021)	◑ AS	◑ MM	○	○
	(Mendonça et al., 2021)	● AOSE criteria	◑ MM	◑ BDI model	○
Barton	(Kovalenko et al., 2019)	◑ AS	◑ aPS	◑ PA pattern	○
	(Kovalenko, 2020)	◑ AS	◑ aPS	● PA pattern	○
Leitão & Ribeiro	(Karnouskos et al., 2018)	● quality criteria	◑ MM ISO/IEC 25010	◑ PL	○
	(Leitão et al., 2021)	◑ AS	● industrial CPS	◑ PL	◑ RG
	(Ribeiro & Gomes, 2021)	◑ AS	● industrial CPS	◑ metamodel	○
	(Sakurada & Leitao, 2020)	◑ AS	● industrial CPS	◑ PL	◑RAMI layers
Casquero,	(Priego, 2017)	◑ AS	◑ MM FAPS	◑ PL	○

Estévez & Marcos	(López-García, 2023)	◑ AS	◑ MS IAMS	● IA interface	◑ AAS
	(Gangoiti et al., 2022)	◑ AS	◑ MM RECON	● IA patterns	○
Fay	(Fay et al., 2019)	○	◑ aPS	◑ PL	◑ RG
	(Gehlhoff, 2023)	◑ AS	● aPS	◑ PL	◑ RG
	(Gehlhoff & Fay, 2020)	◑ AS	◑ aPS	◑ PL	○
	(Reinpold et al., 2024)	◑ AS	● aPS DT	◑ PL	◑ AAS
Göhner	(Badr, 2011)	○	◑ aPS/FMS	◑ PL	○
Lüder	(Ryashentseva, 2016)	◑ AS	◑ aPS	◑ PL	◑ RG
TUM-AIS	(Fischer et al., 2020)	◑ AS	● aPS/MFS	◑ metamodel	○
	(Land et al., 2023)	◑ IA criteria	◑ aPS	○	◑ RG
	(Hoffmann et al., 2017)	◑ AS	◑ aPS	◑ FIPA patterns	○
	(Rehberger, 2020)	◑ AS	◑ aPS	● PA pattern	○
	(Schütz, 2015)	◑ AS	◑ aPS	◑ metamodel	○
	(Vogel-Heuser et al. 2015a)	◑ AS	● industrial CPS	◑ PL	○
	(Wannagat, 2010)	◑ AS	● aPS	● IA resource	○
Other MAS authors	(Bendjelloul et al., 2022)	◑ AS	◑ MS	◑ PL	◑ RG
	(Fast-Berglund et al., 2020)	○	◑ aPS	○	○
	(Hoffmann, 2017)	◑ AS	◑ aPS	● IA interface	○
	(Marschall, Ochsenkuehn, et al., 2022)	○	◑ drink aPS	◑ PL	◑ RG
	(Theiss, 2015)	◑ AS	◑ MS Java	◑ IA interface	○
	(Theiss & Kabitzsch, 2017)	◑ AS	◑ MS Java	● IA interface	○
	(Villavicencio et al., 2019)	◑ AS	◑ MM MAGReS	○	○

Note: see 'footnotes' from Table 2.

Table 4 focuses on evaluating pattern approaches for I4.0, drawing attention to the contributions of López-García & Marcos (2021) and Leitão et al. (2021), who have advanced the field through their work on IA pattern interfaces (Req3) and the MAS application in industrial CPS. Despite these advances, the table indicates that patterns are not fully reusable (only focus on IA interface), demanding a more detailed set of IA patterns (Req3), suitable for various agent-based CPPS aligned to RAMI4.0, as introduced by (Leitão et al., 2023).

Table 4: Evaluation of relevant work (sorted into group/leader) based on rating scheme in Table 1. Pattern approaches.

Group/ leader	Requirement / Work	Req1- Classification	Req2- Domain	Req3- Reusability	Req4- Modelling
IA standard	(IEEE Std 2660.1-2020)	○	◑ aPS	● IA norm	◑ RG
	(VDI/VDE 2653-4 2021)	● structure criteria	● aPS/smart grid	● IA norm	◑ RG
Casquero, Estévez & Marcos	(López-García et al., 2021)	○	◑ aPS	● IA interface	◑ AAS
	(López-García et al., 2023)	○	◑ MS IAMS	● IA interface	◑ AAS
Leitão & Ribeiro	(Karnouskos et al., 2020)	○	● industrial CPS	● IA interface	◑ RG
	(Leitão & Strasser, 2016)	○	● aPS/smart grid	● IA interface	○
	(Leitão et al., 2023)	○	● aPS/smart grid	● IA interface	◑ RAMI layers
	(Ribeiro et al., 2018)	◑ AS	● industrial CPS	● IA interface	○
	(Sharma et al., 2019)	○	◑ MS IASelect	● IA interface	○
Vyatkin	(Patil et al., 2018)	◑ code criteria	● industrial CPS	◑ IEC 61499 FBs	○
	(Sorouri et al., 2012)	○	◑ aPS	◑ IEC 61499 FBs	○
	(Vyatkin, 2016)	○	● industrial CPS	◑ IEC 61499	○
Lüder	(Lüder et al., 2017)	◑ AS	● aPS	● IA resource	○
TUM-AIS	(Fay et al., 2015)	● structure criteria	◑ MM FAVA	◑ metamodel	○
	(Fischer, Vogel-Heuser, Berscheit, et al., 2021)	◑ code criteria	● aPS	◑ IEC 61131-3	○
	(Fuchs et al., 2014)	○	◑ aPS	◑ IEC 61131-3	○
	(Neumann et al., 2020)	◑ structure criteria	◑ aPS	◑ IEC 61131-3	○
Other pattern authors	(Albrecht et al., 2024)	◑ AS	◑ MM MARL	◑ learning IA pattern	○
	(Bloom et al., 2018)	○	◑ MS	◑ IIoT patterns	○
	(Charpenay et al., 2021)	◑ AS	◑ MM MOSAIK	◑ artifacts	○
	(Chitchyan et al., 2007)	○	◑ aPS	◑ metamodel	○
	(Gamma et al., 1994)	○	◑ MM	● IA types	○
	(Papoudakis et al., 2021)	◑ code criteria	◑ MM MARL	◑ learning IA pattern	○
	(Roher & Richardson, 2013)	○	◑ MM	◑ IT patterns	○
	(Schulte et al., 2016)	○	◑ MM	◑ human pattern	○
Valckenaers & Weyns	(Holvoet et al., 2009)	○	◑ MM D-MAS	● IA patterns	○
	(Juziuk et al., 2014)	◑ SRL criteria	◑ aPS	◑ pattern dimension	○
	(Valckenaers, 2020)	◑ AS	◑ MM ARTI/PROSA	● IA patterns	○
	(Weyns, 2012)	● SRL criteria	○	◑ pattern dimension	○
Zoitl	(Sonnleithner et al., 2021)	○	◑ aPS	◑ IEC 61499 skill	○
	(Zoitl & Prahofer, 2013)	○	◑ aPS	◑ IEC 61499 FBs	○

Note: see 'Footnotes' from Table 2.

These standards have substantially shaped the structural criteria and domain applications that are essential to effectively integrate MAS within CPPS. Yet, the table also identifies a gap in the reusability of IAs (Req3), signifying an opportunity for evolving research to address the adaptability of MAS approaches based on patterns, as introduced in (Karnouskos et al. 2020).

The approaches reviewed and their respective contribution fields (CPPS, MAS or design patterns) are summarized in Fig. 6.

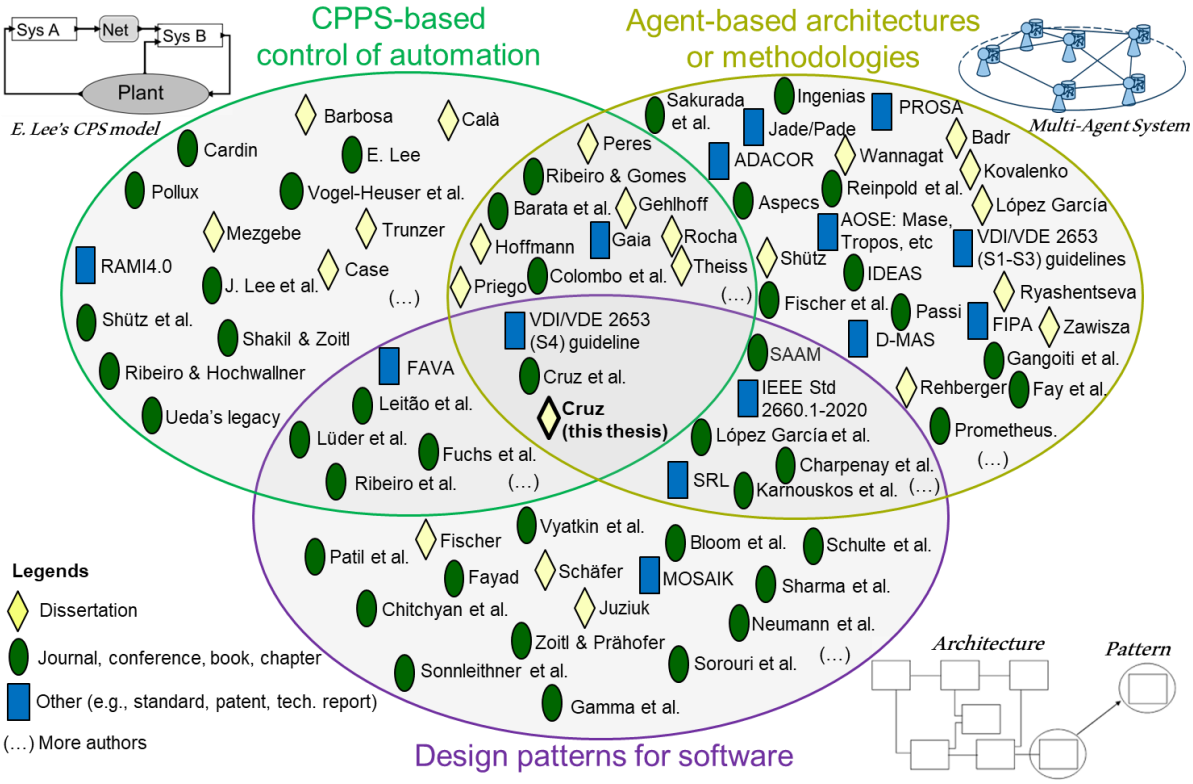

Figure 6: Overview of relevant state-of-the-art contributions, their field of contribution, and identified research gap that can be positioned at the junction where CPPS, MAS, and design patterns intersect.

As can be seen in Fig. 6, many CPPS and MAS approaches that consider aPSs exist. On the other hand, several distinct design patterns for industrial systems were identified. Nevertheless, only five approaches exist that fully encompass a well-proven classification criteria (Req1) for MAS: (D'Avila Mendonça et al., 2022; Fay et al., 2015; Fischer, Vogel-Heuser, Schneider, et al., 2021; Juziuk et al., 2014; VDI/VDE, 2021). In fact, only two publications previous to this thesis have shared points about CPPS, MAS and design patterns of their requirements' contribution (Cruz S. et al., 2019; VDI/VDE, 2021).

Based on the preliminary tables' classification, a synthesis across previous work is conducted to present the state-of-the-art agent-based CPPS, as depicted in Fig. 7. So far, the previous studies reported (cp. Fig. 6) are still mainly collected from some phases of the MASs and CPPSs

developments (e.g., architecture, metamodel, IEC 61131-3/IEC 61499 code); thus, further study is necessary to explore IAs patterns in other phases of agent-based CPPS domains.

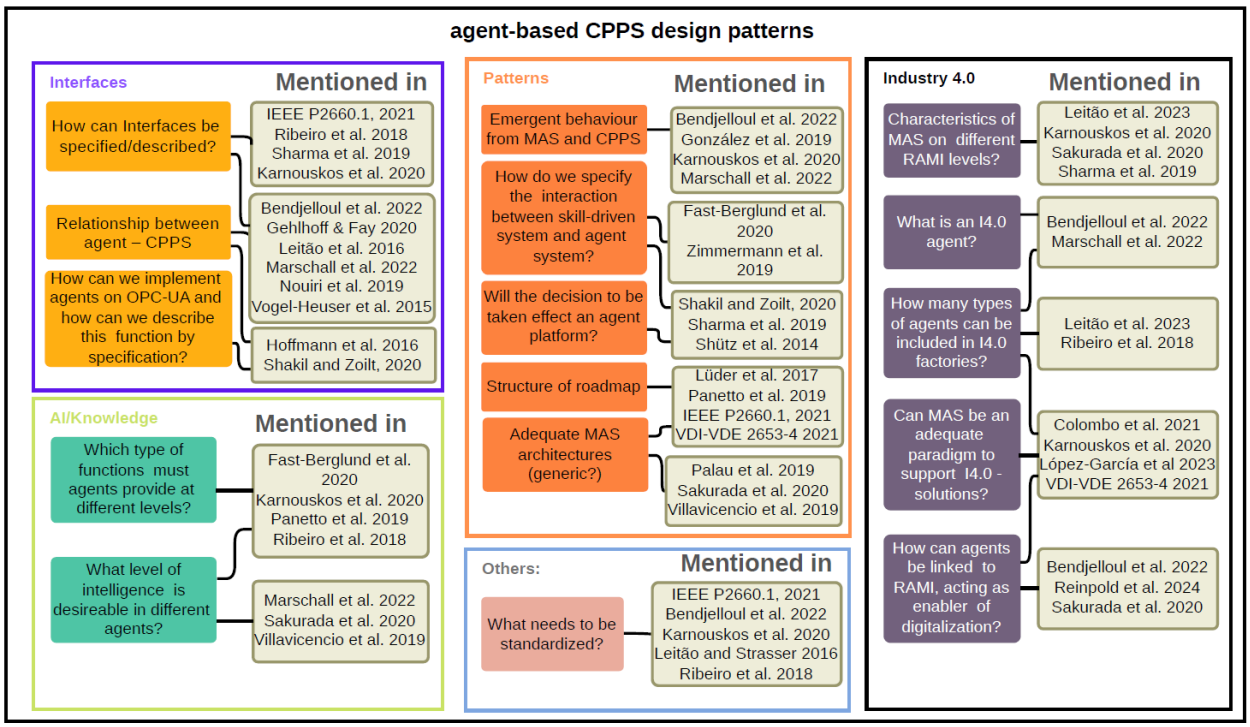

Figure 7: State-of-the-art agent-based CPPS patterns based on their introduced challenges (selected RQs from Fig. 2).

Finally, VOSviewer® is used as a key visualization tool to gain a comprehensive, delimited, understanding of the academic landscape of agent-based CPPS and DTs driven by AASs. Thus, the co-citation map, shown in Fig. 8, identifies the key contributors to these fields (CPPS and DT) and explains the interconnected nature of their work. This map provides insights of IAs driven by DTs research, showcasing the most productive authors and the top 20 most co-cited authors.

Figure 6 conceptualizes the gaps and intersections of CPPS, MAS, and design patterns, exposing the areas ripe for more research. Then, Fig. 7 serves as a visual confirmation of the interconnected research domains, while Fig. 8 (a citation network) provides a graphic insight into the authorial contributions and their interrelationships, indicating the potential for future collaboration and knowledge exchange.

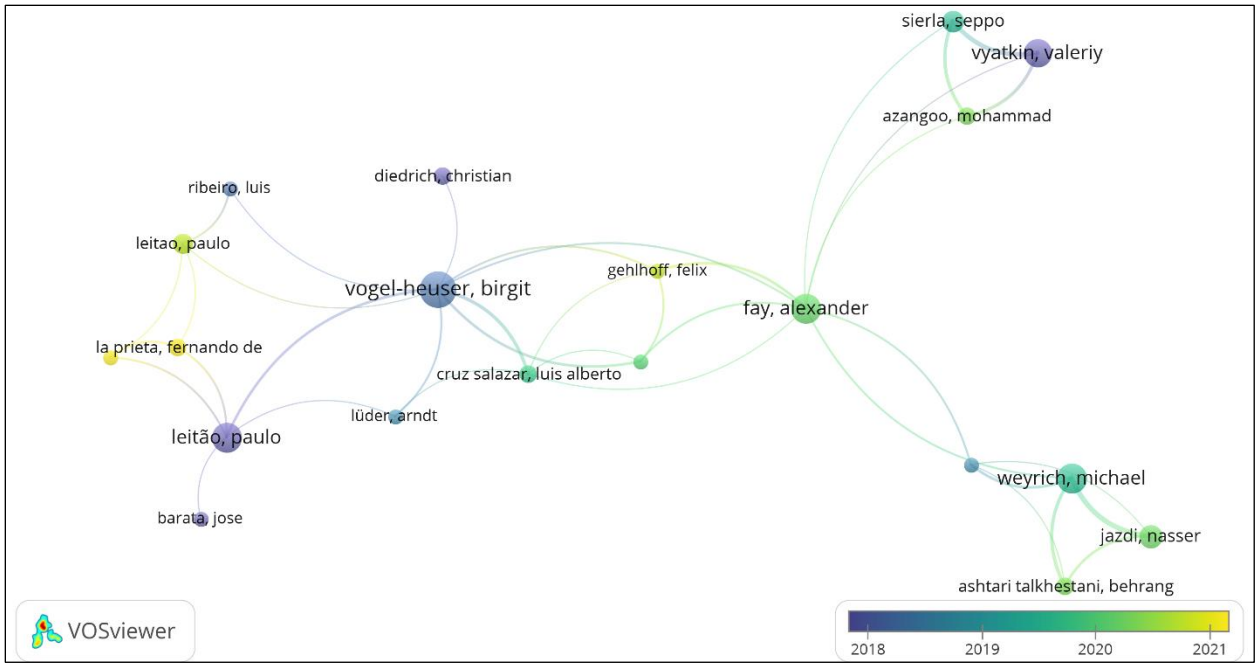

Figure 8: Citation network of selected paper (overlay view in VOSviewer®).
The authors are from a Web of Science search, and as introduced in (Reinpold et al., 2024)*, the selected keywords were: (agent* OR mas OR digital-twin OR "digital twin" OR twin OR "administration shell" OR "aas"; from last 5 year (2018 to 2023) in the field TOPIC (Title-Abs-Key). A number of 145 results and 20 keywords (from 125) with a frequency equal to or superior to 5, were processed. Each node in the figure represents an author, with the size of each node corresponding to the author's productivity or number of works. The links between nodes indicate co-citations, and the thickness of these links represents the frequency of co-citation.*

In conclusion, the related work discussed in this section substantiates the critical role of IAs in optimizing CPPS, as articulated by design patterns, evaluation metrics, and distributed intelligence. The systematic review of MAS approaches, informed by a robust assessment scheme, opens the way to understand how IA patterns contribute to the field of smart manufacturing and CPPS optimization. However, none of the approaches surveyed prove agent-based CPPS implementation efforts using IA patterns capable of both RAMI4.0 and PPR (Req4) and suitable criteria classification (Req1), as shown in gray areas of tables 2 to 4. Therefore, the research gap that is addressed within this thesis is identified as:

Research gap:
There is a deficiency in the comprehensive integration of IA design patterns within agent-based CPPS that aligns with established I4.0 models, i.e., RAMI4.0/PPR/ISA95/88. Existing literature and approaches fail to provide a unified methodology that encapsulates the full characteristics of IA functionalities, particularly regarding scalable patterns, cross-domain applicability, and this standardization conformity. Moreover, there is no universally accepted IAs categorization that sufficiently addresses the heterogeneity of MAS architectures, which is crucial for developing robust, interoperable, and adaptive industrial systems. Additionally, current frameworks do not adequately support the AAS interface generation for diverse communication protocols, which is essential for the efficient and flexible exchange of information within the smart manufacturing domain.

3. Main contributions of IA design patterns to CPPS

The work program is structured into five contributions (***Con1-Con5***): (***Con1***) MAS criteria, (***Con2***) Pattern needs, (***Con3***) I4.0 scenario, (***Con4***) MAS architecture, and (***Con5***) MAS guideline. The relationships between research issues, research contributions and their publications are presented in Fig. 9. In the following, the research contributions are described.

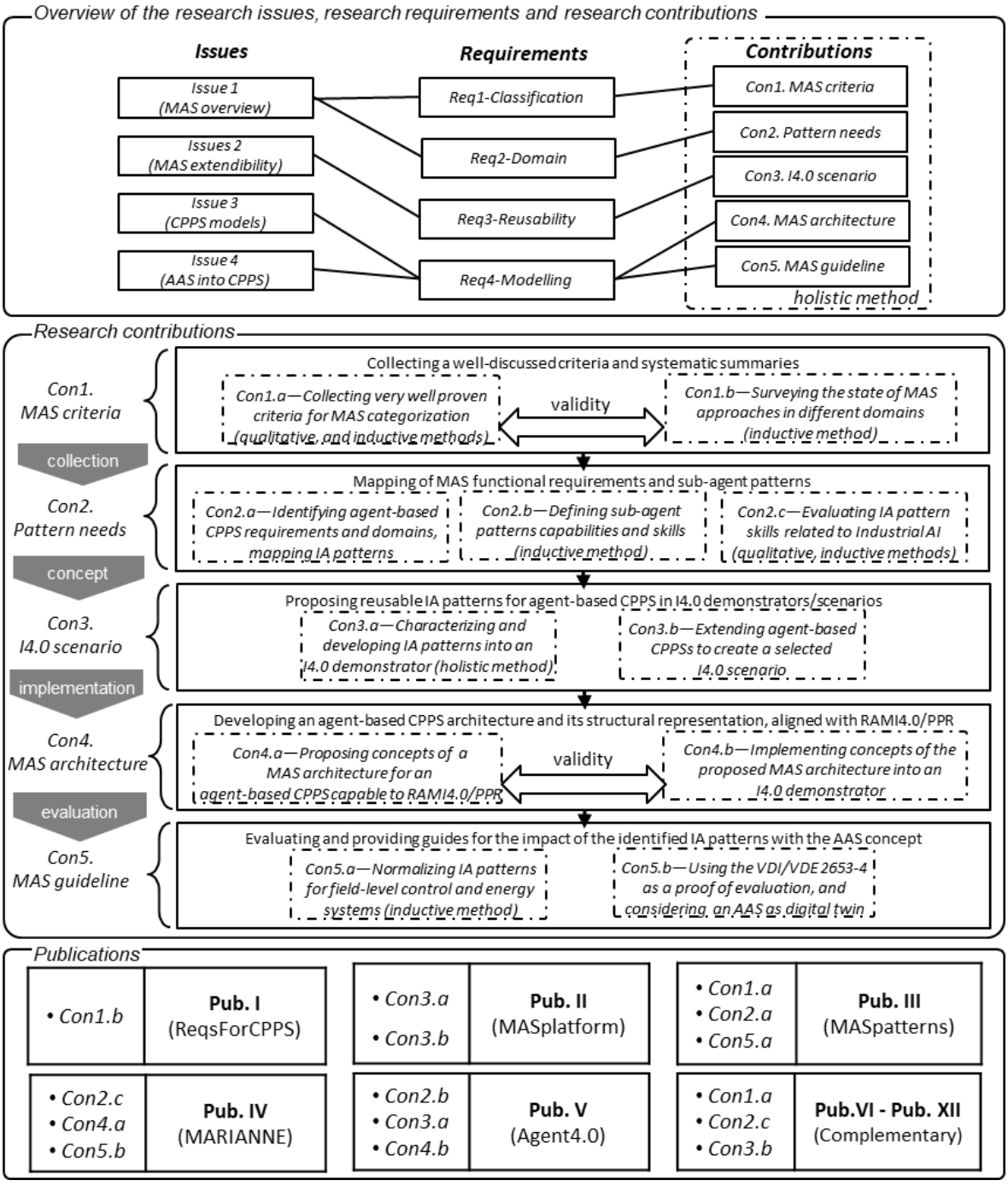

Figure 9: Research issues, contributions, and publications (namely paper numbers Pub.X as occur).

3.1 Contribution 1 (*Con1*): MAS criteria categorization

CPPS refer to mechatronic systems coupled with software entities and digital information, enabling the smart factory concept for I4.0 (Karnouskos et al., 2020; Monostori et al., 2016). The migration of existing control systems to CPPS is still a challenge due to the complexity involved (Calà et al., 2017; Ribeiro & Gomes, 2021). Design patterns are proposed to help developers build software with common solutions derived from experiences (Leitão et al., 2018).

Con1 aims to provide a description and comparison of existing MAS design patterns. Two classification criteria are introduced to support MAS developers in implementing CPPS. Through an evaluation of four selected patterns (named here sub-agents), a well-discussed survey/summary of at least twenty MAS by the German IFAC NMO GMA FA 3.35 finds that IA patterns greatly benefit CPPS design. They also conclude that manufacturing based on MAS is an effective way to address the complex requirements of CPPS development. As example of this contribution, Table 5 introduces the compilation of the criteria for the MAS design pattern template (with examples), including pattern category, pattern type, pattern name, pattern description, context, solution, implementation, MAS-architecture, knowledge base and processing, real-time properties, dependability, learning, MAS-autonomy, and others. Those are valid classification criteria in correspondence to **RQ1** (how describe MAS patterns?).

Table 5: Criteria to classify MAS architectures / patterns (Cruz S. et al., 2019).

Criteria	Descriptions	Examples options
Pattern category	Favorable function patterns: System properties that can be realized by employing MAS, i.e. increased flexibility and adaptability	Flexibility pattern, adaptability pattern, reliability pattern, reconfigurability pattern
Pattern type	Name of the pattern type: Technology-independent task of the MAS (categorized)	Fault-tolerant sensors
Pattern name	Name of the MAS pattern	Soft sensor
Pattern description	Description of the logic structure (which components/agents does the pattern contain?)	MAS with 4 sub-agents, which enable identifying faulty sensors and automatically replacing them with soft sensors based on models
Context / Area of application	Application context of the pattern	Various domains, e.g. logistics, process engineering
MAS-Architecture	Approach for realization of the agents' behavior	Reactive / cognitive / hybrid
Solution	Graphical depiction of the MAS-Architecture	Depiction of the MAS' components (notation class diagram)
Knowledge base and processing	How is the knowledge stored? Models, rules. How is the knowledge processed? With which methods?	Model from engineering, ontology, meta model data structure. Inference mechanisms for ontologies
Learning / Knowledge acquisition	Methods and techniques for learning abilities / knowledge base	Machine learning, neuronal networks

Implementation	Technological realization of the MAS (platform, languages)	Model: SysML, programming language IEC 61131-3
Real-time Properties	Timeliness and concurrency requirements	Usage replacement sensor < 2 PLC-cycles < 40 ms
Dependability	Requirements towards reliability, availability, maintainability, security, or safety	Soft sensor can replace sensor with a reliability of x%
MAS-Autonomy	Autonomy/independence in decision making	Replacement of sensor not autonomously, since number of replaceable sensors is limited
Others	Additional author's comments (remarks, clarifications, etc.)	

Con1 also provides a first conceptualization to agent-based CPPSs that meets the requirements of smart factories based on MAS survey authors into RAMI4.0 layers and Agent Oriented Software Engineering (AOSE) methodologies. Design patterns based on MAS offer properties such as flexibility/changeability, reliability, reconfigurability, adaptability/agility and dependability, which are essential for CPPS (also related and confirmed by ***Con2***). ***Con1*** aims to provide engineers and programmers with existing patterns based on MAS structures to improve design efforts and increase efficiency in manufacturing control. It also explores the applicability and benefits of MAS and design patterns in the industrial sectors in multiple production systems domain: discrete manufacturing, continuous process, hybrid production. The results from ***Con1*** are the necessary inputs for ***Con4*** and ***Con5***. In conclusion, the aimed delivery of ***Con1*** is:

- Providing two classification criteria for comprehensive analysis of IA patterns for CPPS and highlighting their benefits in addressing the requirements of CPPS development. The proposed patterns come from a discussed survey of twenty agent-based approaches which offer a variety of "ready-made' solutions for MAS developers and enable the rapid application of MAS in industry, including selected AOSE methodologies.

3.2 Contribution 2 *(Con2):* IA pattern needs for I4.0

IA patterns are crucial in the landscape of I4.0, marking a significant challenge in how modern aPS are designed, monitored, and controlled (Leitão et al., 2021; Lüder et al., 2017; Ribeiro et al., 2018). IA patterns are able to encapsulate sophisticated capabilities and skills, enabling autonomous decision-making, adaptive behavior, and adding plug & produce features within CPPS (Baumgartel & Verbeet, 2020; Shakil & Zoitl, 2020; Zimmermann et al., 2019). Karnouskos et al. highlighted the importance of IAs having clear propositions on the functionalities, services, and value-added aspects they offer, which supports the notion of IA patterns encapsulating sophisticated capabilities and skills within (Karnouskos et al., 2020).

Provided examples of IA applications in various domains, such as factory automation, power & energy systems, and building automation, demonstrating the wide range of applications of IA patterns aligned to I4.0 standards (IEEE, 2021; Leitão et al., 2023; VDI/VDE, 2021).

The findings from ***Con2*** provide insightful information about IA patterns, skills, and capabilities essential for I4.0 applications, as introduced in Table 6. ***Con2*** aims to describe the importance of standardized IA design patterns and discuss the characteristics of Industrial AI vital for agent-based CPPS. In this sense, ***Con2*** emphasizes the significance of the Industrial AI characteristics (C1-C5), including autonomy, reactiveness, proactiveness, predictability, and human cooperativeness, and their relevance in achieving the goals of I4.0 (Cruz S. & Vogel-Heuser, 2022a). In general, most of the representative CPPS approaches are missing predictability characteristics and the RAMI4.0 capability (Cruz S. & Vogel-Heuser, 2022b). Therefore, in order to fulfill those requirements, ***Con2*** delivers the *Multi-Agent aRchitecture for Industrial Automation applying desigN patterNs practicEs (MARIANNE)*, following the normalized guidelines, and addressed by IA classes and capabilities, as introduced in Table 6.

Table 6: Industrial Agents, their main competencies, and examples. Source: (Cruz S. & Vogel-Heuser, 2022a).

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 (Baumgartel & Verbeet, 2020). This IA class also has access to assets' main functionalities and is building on the normalized *Resource Agent* (see VDI/VDE 2653-4 guideline (VDI/VDE, 2021))
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* (IEEE, 2005))	This IA type is often concerned with non-physical entities, e.g., orders, production plans, production schedules, among others (Unland, 2015). The typical instances of this IA class are the normalized *Agent Management System* and the *Process Agent* (see VDI/VDE 2653-4 guideline (VDI/VDE, 2021))
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 (VDI/VDE, 2021)), and this may, for example, interconnect IAs and LLC automation functions based on the *IEEE 2660.1 interface practice* (IEEE, 2021)
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 (Karnouskos et al., 2020)

Con2 identifies four IA classes (see Table 6), along with their specific capabilities and alignment of Industrial AI characteristics, further highlighting the critical role of IA patterns in MAS architectures for CPPS. The IA patterns, such as the Reactive IA, Proactive IA, and Predictive IA, are categorized based on their response time and main behaviors, explaining the diverse capabilities (C1-C5) necessary for achieving autonomy, reactiveness, proactiveness,

predictability, and human cooperativeness in I4.0 scenarios (Cruz S. & Vogel-Heuser, 2022b). The main importance of these IA patterns, skills, and capabilities allows adaptable CPPS within the context of RAMI4.0 and the AAS concept, related to ***Con4***. These IA patterns and characteristics enable Industrial AI to effectively manage uncertain conditions, respond to environmental information, take the initiative for decision-making, predict outcomes, and facilitate human-in-the-loop interactions, essential for the successful implementation of I4.0 systems (Cruz S. & Vogel-Heuser, 2022b, 2022a). Results from ***Con2*** have necessary inputs for ***Con5,*** defining the IA capabilities and skill for the IA patterns standardization.

By aligning IA design patterns with Industrial AI characteristics and developing predictive agents, e.g., Agent4.0 from Table 7 (Cruz S. & Vogel-Heuser, 2022b). ***Con2*** contributes to the advancement of agent-based CPPS design patterns, ultimately fostering the realization of intelligent and efficient I4.0 systems, for aPS but also able to energy systems (VDI/VDE, 2021). Those are agent-based CPPS goals and benefits in different fields that correspondence to RQ2 (for which CPPS domains?). In conclusion, the aimed delivery of ***Con2*** is:

- Introducing two significant definitions in the standardization of IA patterns: Industrial IA characteristics and IA types together its competencies or skills. Also, the integration of a predictive agent type, like the Agent4.0, with MARIANNE to develop intelligent, efficient, and adaptable CPPS, aligning with RAMI4.0/AAS. This contribution not only supplies smart production but also extends the IA patterns normalization to smart grids, reflecting the versatile applicability of agent-based CPPS in various domains.

Table 7: Agent4.0's Industrial AI characteristics evaluation (Cruz S. & Vogel-Heuser, 2022b).

Industrial AI characteristics (C1-C5) evaluation / **Agent4.0 function (skill) description**	***Autonomy***	***Reactiveness***	***Proactiveness***	***Predictability***	***Human Copera.***
Agent4.0 should increase its initial Knowledge Base competence because of the "learning element" (often non-real-time). Sec.IV.A*	●		●	●	●
Agent4.0 may operate in a time-predictable way, i.e., enabling short/medium/long-term production tasks. Sec.IV.A	●	●		●	
Agent4.0 can predict data valuable to other IAs, by a central learning module. Sec.IV.A	●				●
Agent4.0 can apply a supervised learning method, e.g., a Linear Regression algorithm, to achieve its goals. Sec.V.B				●	●
Agent4.0 usually does not fulfil hard/soft real-time requirements because predictability implies learning from the past and being located at the heterarchy top. Sec.II.C4				●	

•: needed. *Sec.X (section label) refers to the publication reference in (Cruz S. & Vogel-Heuser, 2022b).

3.3 Contribution 3 (*Con3*): agent-based CPPS scenarios

The I4.0 demonstrators play a crucial role in validating the implementation of agent-based CPPS. These demonstrators are essential for showcasing the integration of MAS in building CPPS compliant with the RAMI4.0 (Bendjelloul et al., 2022). The use of IAs has facilitated the realization of more CPPS, emphasizing the significance of these demonstrators in validating the implementation of agent-based CPPS (Marschall, Schleicher, et al., 2022). As proof of concepts, ***Con3*** includes the thesis' evaluations in three I4.0 demonstrators, at the Institute of Automation and Information Systems (AIS), from the Technical University of Munich. The first I4.0 demonstrator is the *Robot Integrated Agent Network (RIAN)* that enables the connection of heterogeneous plants at the control level and the usage of this competence for enabling the interconnection of a cooperating production line. According to the I4.0 concept development, the information of the plants is transmitted via the internet while the physical connection is performed via mobile transportation robots, as demonstrated at the Fair Trade Automatica 2014.

The second I4.0 demonstrator is the *Hybrid Process-Model* (*often named myYoghurt plant)* that is a major laboratory production system for implementing and evaluating the concepts and approaches that are investigated at the AIS-TUM chair, including for platform independent MAS for robust networks of CPPS (Cruz S. et al., 2018; Seitz et al., 2021). Therefore, myYogurth plant consists of multiple plant sections that emulate different industrial domains, i.e. material flow, discrete manufacturing and continuous (chemical) process.

The third I4.0 demonstrator is the *Extended Pick and Place Unit (xPPU)* that handles and manipulates work pieces of different material. The xPPU sets 16 variants evolution scenarios that are characterized by a variety of different changes in platform, context, and software. It perfectly keeps the balance between representing the reality and limiting the complexity and is therefore a useful evaluation plant for IA patterns research (Cruz S. & Vogel-Heuser, 2022a, 2022b). As the Fig. 10 depicts, ***Con3*** uses the adaptation of IA patterns AMS, CA, RA, and PA for standardization (***Con5***) into MARIANNE architecture for the xPPU I4.0 scenarios (Cruz S. et al., 2019; Cruz S. & Vogel-Heuser, 2022b). Those IA patterns have particular aims and can be reused for several CPPSs in correspondence to **RQ3** (what MAS patterns are reusable?).

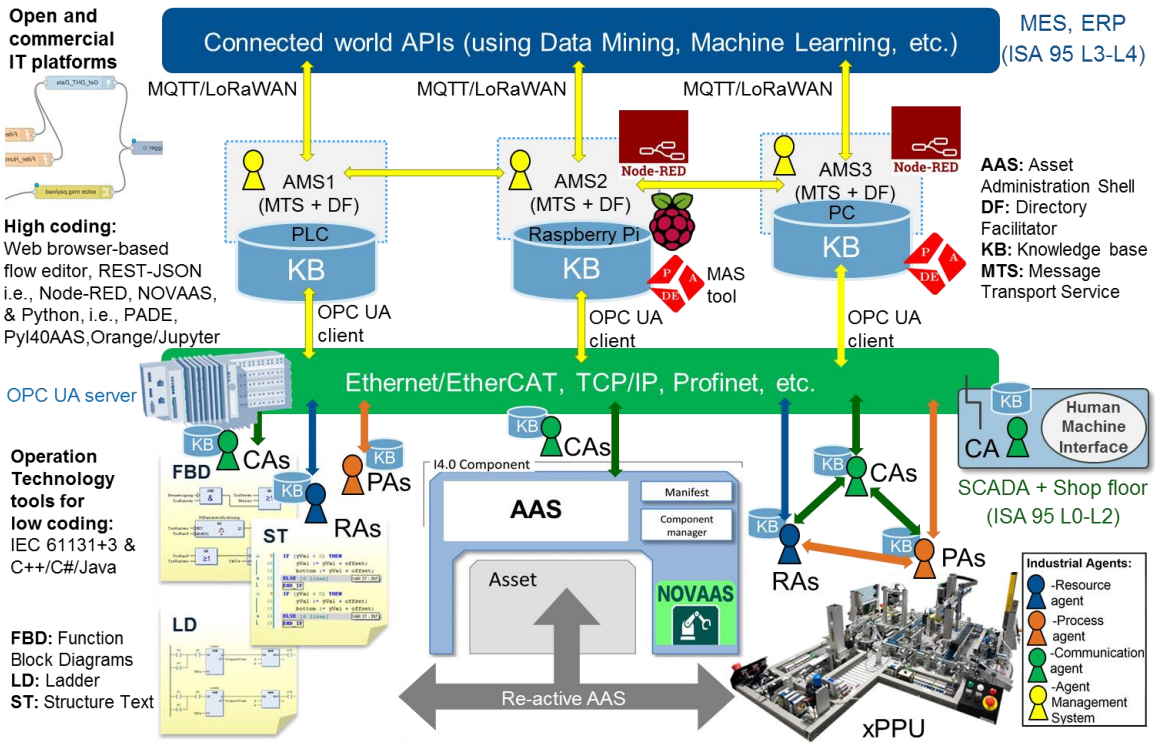

Figure 10: General landscape of the I4.0 scenario proposed with their I4.0 components and their IT/OT technologies.

Con3 is valuable due to the variability of the selected I4.0 demonstrators that differ in many aspects, like accessibility by different protocols, modularity, and independent platform requirements, enabling their application into I4.0 scenarios (Seitz et al., 2021). For example, Table 8 results from testing those CPPSs and the applicable classification criteria based on IA pattern interfaces (IEEE, 2021). Results from ***Con3*** have necessary inputs for ***Con4,*** enabling a common MAS architecture.

Table 8: Qualitative Assessment of IAs interfaces of the I4.0 demonstrators (Cruz S. & Vogel-Heuser, 2022b).

Pattern criteria*	**myYoghurt**	**xPPU demonstrator**
Location	On-device	Hybrid
Interaction mode	Loosely coupled	Tightly coupled
API client	C++/C#, Java (JADE)	REST/JSON, Python (PADE)
Channel	FIPA-ACL, OPC UA	HTTP, FIPA-ACL, OPC UA
*Score**	2.56	3.20

*Criteria recommendation come from (IEEE, 2021). The score value is according to our expertise, providing a qualitative assessment of the IEEE 2660.1 interface practice into the CPPSs.

Con5 shows a proof of concept of the I4.0 scenarios[8] such as the Adaptable Factory (AF), Order-Controlled Production (OCP), and Self-organizing Adaptive Logistics (SAL), reducing the

[8] See detailed information about AF/OCP/SAL scenarios in the documentation online available:

time and cost efforts, as the proposed IA patterns are "ready-made" solutions (Vogel-Heuser et al., 2020). As proof of evaluation, ***Con5*** considers the collaboration of the AIS-TUM and the Institute of Automation Technology with Helmut Schmidt University (see Pub. XI). The xPPU and myYoghurt in combination with other I4.0 demonstrators at Helmut Schmidt University collaborate in the application of the OCP and AF scenarios (Seitz et al., 2021). In conclusion, the aimed delivery of ***Con3*** is:

- Providing insights of the deployment of three I4.0 demonstrators —RIAN, myYoghurt plant, and xPPU— which validate the integration and effectiveness of IA design patterns within CPPS. This is further evidenced by qualitative assessments and collaborative research, providing crucial data for advancing the dissertation's aims and reinforcing the practicality of IA patterns in at least two extended I4.0 scenarios i.e., OCP and AF.

3.4 Contribution 4 (*Con4*): MAS architecture with DTs

The integration of IA patterns and the concept of the AAS from RAMI4.0 significantly influences the development of agent-based CPPS within I4.0 (López-García et al., 2021; Reinpold et al., 2024). AASs considered here as a digital twin, allows the interoperability of agent-based CPPSs, providing a standardized digital representation of assets, enabling unified communication and integration of physical assets with their digital equivalents (Gangoiti et al., 2021; Sakurada & Leitao, 2020). ***Con4*** provides valuable insights into the integration of IAs and the AAS concept for the development of agent-based CPPS. Specifically, ***Con4*** describes a generic MAS architecture derived from IA patterns capable of an AAS and illustrates the relationships between IA patterns and their roles in a CPPS (Cruz S. & Vogel-Heuser, 2022b), as depicted in Fig. 11. This MAS ensures CPPS platform independence and interoperability (Cruz S. et al., 2018), addressing the autonomy, reactiveness, proactiveness, predictability, and human cooperativeness required for agent-based CPPS (Cruz S. & Vogel-Heuser, 2022b), related to ***Con2***. The AAS concept in ***Con2*** serves as a digital representation of physical resources linked to RAs, providing the basis for merging new I4.0 components and enabling whole integration with IT/OT systems. Additionally, the deployment of autonomous and collaborative manufacturing entities with enhanced self-capabilities, such as self-optimization, self-awareness, and self-monitoring, is highlighted as a priority for CPPS (Cruz S. & Vogel-Heuser, 2022b).

https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/aspects-of-the-research-roadmap.html

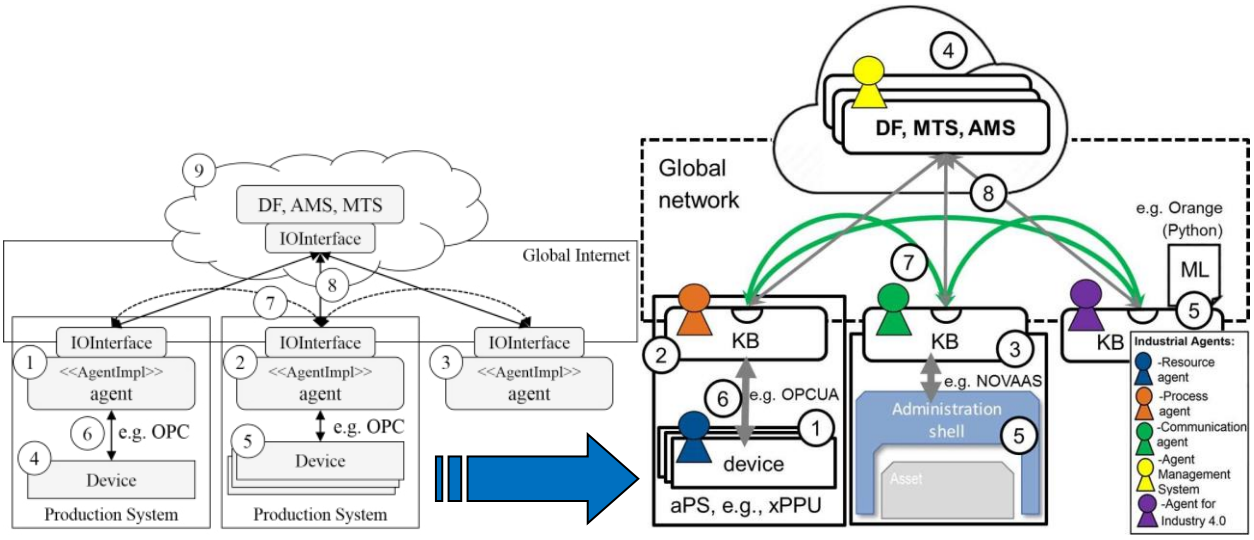

Figure 11: Logical architecture of the MAS (right) extended to an AAS-based MAS version (left).
aPS, automated Production Systems; AMS, Agent Management System; DF, Directory Facilitator; KB, Knowledge Base; ML, Machine Learning; MTS, Message Transport System.

The main importance of integrating IA patterns and the AAS concept lies in achieving a high level of interoperability, but also performance, learnability, and reconfigurability necessary for addressing I4.0 scenarios (***Con3***). ***Con4*** aims to integrate the use of ML models for Agent4.0 to facilitate the identification of critical CPPS situations in an unsupervised training environment. Agent4.0 with its AAS is enabling automatically performed cost-opportunity analyses to decide on incorporating additional agent-based (soft) sensors to increase availability to the resources (Cruz S. & Vogel-Heuser, 2022a). Table 9 introduces how MAS architecture based on metamodel criteria, is aligned with established RAMI4.0/AAS and PPR models, in correspondence to **RQ4 (**how to align CPPS to RAMI4.0/PPR?).

Table 9: Relationship and comparison between I4.0 models (Cruz S. & Vogel-Heuser, 2022a).

		Metamodel criteria*								
	How can the (1-3) model realize or define the (a-i)?	***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 (Cha et al., 2020).

Key findings from ***Con4*** reveal the pivotal role of IA patterns in enabling autonomy and collaboration within the framework of I4.0. The digital representation of an IA by an AAS encapsulates information and services, orchestrating intelligent IA patterns and enabling

cooperation aligned to FIPA protocols and based on their IA patterns skill and capabilities (***Con2***). AASs are essential for providing a comprehensive digital representation of all available information about and from an object, whether it is a hardware system or an independent software platform (Cruz S. et al., 2018). In conclusion, the aimed delivery of ***Con4*** is:

- Supporting a comprehensive and robust MAS architecture for the integration of IA patterns and the AAS concept, laying the groundwork for the realization of intelligent, autonomous, and adaptable CPPS aligned with the principles not only of RAMI4.0 but also the PPR model. The contributions made in ***Con4*** are pivotal in advancing agent-based CPPS design patterns, thereby facilitating the achievement of smart, interconnected, and predictive I4.0 systems by the concept of Agent4.0.

3.5 Contribution 5 (*Con5*). IA patterns standardization

The IA experts highlight the necessity to analyze standardization needs for deploying agent-based technology, addressing industrial requirements imposed by different application fields (Leitão & Strasser, 2016). Furthermore, the IA community discusses the identification of patterns derived from existing implementations of IAs and the assessment of their characteristics, using the ISO/IEC 25010 standard as a starting point (Karnouskos et al., 2018). Additionally, TC IA members, led by Professor Paulo Leitão in the IEEE P2660.1, highlight the use of agent technologies for higher-level decision-making and lower-level automation and control functions in industrial systems, emphasizing the need for IA interface patterns for agent-based CPPS (Ribeiro et al., 2018). Additionally, the VDI/VDE 2653 guidelines, led by Professor Birgit Vogel-Heuser, have been surveyed by the TC 3.35 experts on MAS, indicating their relevance and importance of IA patterns standardization in the industrial context (Vogel-Heuser et al., 2018). In this sense, the VDI/VDE 2653 Sheet 4 is relevant (see Pub. XII) proof of ***Con5*** meaning that the thesis' contribution is valuable and valid due to the fact of the classification criteria and the IAs pattern standardized. ***Con5*** considers the main aspects derived from the classification criteria (***Con2***), but also the evaluation by the German IA experts for each IA pattern, including the integration of MAS approaches in the traditional automation pyramid e.g., based on ISA 95 standard (see Pub. III). The IA pattern categories were derived based on thirteen proposed criteria that classify MAS architectures' patterns, such as sub-agent name, main functionality, automation level, real-time capability, communication base, key properties, and related work (Cruz S. et al., 2019). Accordingly, to these criteria, Fig. 12 introduces the

comparison of two MAS heterarchies and its common IA patterns already standardized (VDI/VDE, 2021): RAs and PAs.

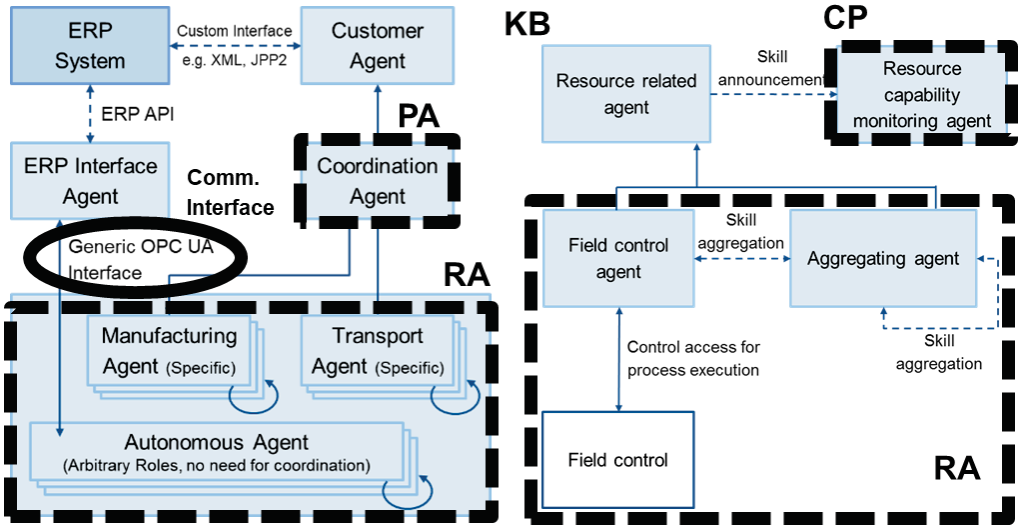

Figure 12: MAS from Hoffmann (Hoffmann, 2017) (left) and Lüder et al. (Lüder et al., 2017) (right). Knowledge Base (KB), Resource Agent (RA), Communication Process (CP) and Process agent (PA). Source: (VDI/VDE, 2021).

The classification criteria demonstrate that similar MAS authors often use common IA patterns with various names, i.e., sub-agents, bust in different domains. Additionally, ***Con5*** aims to integrate IAs design patterns into a CPPS aligned with RAMI4.0's layers, that is crucial for ensuring effective communication, real-time optimization, and decision-making processes within I4.0 systems. Those are agent-based CPPS values in different RAMI4.0's layers that correspondence to **RQ4 (**how to align CPPS to RAMI4.0?). Table 10 provides a list of sub-agent patterns for selected MAS architectures ***(Con1)***. ***Con5*** demonstrates that the sub-agents identified can be classified without consideration of names such as coordination agents, schedule agents, supervision agent, supervisor agents, rescheduler agents, and resource capability monitoring agents among others (Cruz S. et al., 2019). Negotiations and collaborations among those different IAs create a MAS heterarchy for industry. The MAS scope of the architectures discussed include, smart manufacturing, energy systems, material flow systems, image processing and information processing. Instead of considering the names, ***Con5*** involves criteria based on IA common functionalities (i.e., resource access, coordination process, communication interface, or the KB), traditional type of agent (i.e., reactive, or proactive), ISA 95 level of application (L0-L3) and other characteristics of the MAS. ***Con5*** by the classification of sub-agents helps to differentiate the architecture of each MAS and can be defined as modular or integral, according to (Ribeiro, 2017).

Table 10: List of sub-agents patterns for MAS architectures extended from (Cruz S. et al., 2019; Vogel-Heuser et al., 2018).

Autor / Author	**Ressourcenagent** (RA) / **Resource agent** (RA)**:** **L0–L2**	**Prozessagent** (PA) / **Process agent** (PA)**:** **L2–L3**	**Agenten-Management-System** (AMS) / **Agent management system** (AMS)**: L1–L2**	**Kommunikationsagent** (CA) / **Communication agent** (CA)**: L1–L3**
Intelligente Fertigung / Smart manufacturing				
Badr [56]	– RA	± job@	± service@	–
Brehm et al. [57]	++ (RA || field related@)	–	++ gateway@	++ broker@
Cruz et al. [22]	++ RA	++ (product@ & diagnosis@)	++ AMS	–
Rehberger [32]	++ RA	++ product management@	–	+ @interaction
Ryashentseva [28]	++ (executive@ & rescheduler@ & dispatcher@)	++ supervisor@	–	–
Schütz [31]	++ RA	++ PA	++ (control strategy@ & system@)	++ (CA || @interaction)
Theiss [14]	+ plant@	++ (test coordination@ & monitoring@)	± analysis@	+ test@
Ulewicz [33]	++ (hardware@ & system@)	–	++ AMS	++ (CA & system@)
Vogel-Heuser et al. [58]	++ plant@	++ (coordination@ & customer@)	++ AMS	–
Wannagat [29]	++ (RA || control@)	++ (PA & system@)	++ AMS	++ (CA || @interaction)
Folmer [30]	++ control@	+ process@	+ system@	++ CA
Kovalenko et al. [55]	++ RA	++ product@	–	–
Legat [59]	++ execution@	++ (supervision@ & reconfiguration@)	++ AMS	–
Lüder et al. [60]	++ (RA || field related@-RRA)	++ decision support@-DSA	++ (order@ & product type info related@)	–
Hoffmann [34]	+ (autonomous@ || transport@-specific)	++ (coordination@ || manufacturing, specific@)	++ OPC UA Address-Spaceas blackboard	+ customer@
Materialflusssystem / Material flow system				
Fischer [35]	++ (control@ & order@ & system@)	++ coordinator@	++ AMS	++ CA

The comparison of the selected MAS approaches enables mapping four IA patterns: the RA, AMS, CA, and PA (Cruz S. et al., 2019). The PA supervises the execution of a production recipe/plan and interacts with RAs and AMSs. The CA converts proprietary interfaces into multiple communication protocols. The AMS allows bidirectional mapping between IP-addresses and sub-agents' identifications. The RA consists of modules such as Control Module and Diagnosis Module, as a part of its KB. The PA pattern coordinates the execution of process steps and interacts with RAs, CAs, and AMSs. In conclusion, the aimed delivery of ***Con5*** is:

- Leveraging the VDI/VDE 2653 Sheet 4 as a benchmark, affirming the thesis' value and validity by aligning with established guidelines for evaluating, impacting, and extending IA patterns in agent-based CPPS. This alignment ensures that the contribution not only adheres to other recognized standards —e.g., RAMI4.0 (DIN SPEC91345), IEEE Std 2660.1-2020— but also advances the field by providing a systematic classification and application of MAS patterns for field-level control and energy systems.

4. Summary of publications

This chapter highlights the research results obtained in main five papers (publications I to V). Additional publications (publications VI to XII) are mentioned and related as complementary contributions of this dissertation. Table 11 summarizes the percentage value of the candidate contribution in each author publication. Table 12 specifies the details of each publication in accordance with the RQs and contributions. The included papers (namely paper numbers as they occur) are attached in Appendix A.

Table 11: Overview of the author contribution (for each activity of each paper, the contribution of all authors is 100%).

Contribution (%) / **Publication**	**Conceptualization**	**Data**	**Investigation**	**Writing**
Pub.I - ReqsForCPPS (Cruz S. and Vogel-Heuser 2017)	60%	70%	80%	80%
Pub.II - MASplatform (Cruz S. et al. 2018)	60%	40%	70%	60%
Pub.III - MASpatterns (Cruz S. et al. 2019)	60%	50%	75%	55%
Pub.IV - MARIANNE (Cruz S. and Vogel-Heuser 2022a)	70%	65%	70%	50%
Pub.V - Agent4.0 (Cruz S. and Vogel-Heuser 2022b)	75%	90%	80%	70%
Notes about contribution parts **Conceptualization:** Development; conceptual design of the research project **Data:** Data curation or software (acquisition, creating, organizing) **Investigation:** Formal analysis; methodology **Writing:** Visualization; writing - original draft; writing - review & editing				

Table 12: Summary of the primary results from the main publications.

Publication No., year (name)	*Main results*	*Thesis' RQs and contributions*								
		RQ1. How describe MAS patterns?	***RQ2. Which CPPS domains/requirements?***	***RQ3. Which MAS patterns are reusable?***	***RQ4. How use MAS to RAMI4.0 and PPR?***	***Con1. MAS criteria***	***Con2. Patterns needs***	***Con3. I4.0 scenario***	***Con4. MAS architecture***	***Con5. MAS guideline***
Pub. I, 2017 *(ReqsForCPPS)*	•Requirements for agents-based CPPS were identified •Contrast of main AOSE: Gaia, Prometheus, etc.	○	●	○	◑	●	○	◑	○	◑
Pub. II, 2018 *(MASplatform)*	•MAS requirements defined and an agent-based CPPS architecture applied to a I4.0 scenario •Agent-based CPPS FIPA compliant is proposed	○	●	◑	○	◑	○	◑	●	◑
Pub. III, 2019, *(MASpatterns)*	•The main four agent design patterns were defined: AMS, CA, PA, RA •Around twenty MAS analyzed, and patterns were standardized driven classification criteria •Support to create the VDI/VDE 2653-4 norm	●	◑	●	◑	●	●	◑	◑	●
Pub. IV, 2022, *(MARIANNE)*	•Agent-based CPPS architecture derived from MAS patterns (MARIANNE) •Guideline to develop an agent-based CPPS aligned RAMI4.0/PPR	●	○	●	●	◑	◑	●	●	●
Pub. V, 2022, *(Agent4.0)*	•Reusable Agent4.0 concept and its skills •ML-based agent into a MAS architecture by a I4.0 demonstrator	◑	○	●	●	◑	●	●	●	◑

Full covered ●; Partially covered ◑; No covered ○

4.1 Publication I: "Comparison of agent oriented software methodologies to apply in cyber physical production systems" (Cruz & Vogel-Heuser, 2017)

Luis Alberto Cruz Salazar and Birgit Vogel-Heuser

Summary of Pub. I (ReqsForCPPS)

This paper aims to classify existing agent-based approaches as a basis for realizing CPPS and identify the benefits of Agent-Oriented Software Engineering (AOSE) and its development. Despite the several benefits provided by methodologies, such as Gaia, TROPOS, Prometheus, and INGENIAS, their adoption in agent-based CPPS solutions is still uncommon. Then, the achievements of this study include identifying the importance of MAS requirements for industry and highlighting AOSE methodologies as a valuable strategy for application in I4.0/CPPS. A summary of current CPPS projects and their application field, as well as a comparison of hybrid MAS methodologies, are introduced. CPPS projects are categorized based on their focus (i.e., demonstrators, smart manufacturing approaches, electric grid, applications, or architectures) and their alignment with the ISA 95 standard levels. The projects cover a range of applications,

including intelligent manufacturing, autonomous cars, robotic surgery, and smart grids. The significance of this study lies in its contribution to improving manufacturing systems by applying advanced information technologies (ITs) and its integration of operational technologies (OTs) through industrial networking and knowledge of physical things by CPSs. Ten requirements for applying MAS in CPPS are introduced, such flexibility, reconfigurability, operational efficiency, scalability, fault tolerance, interoperability, and adaptability. According to the contribution, those requirements can be achieved by applying new advanced IT/OT technologies for manufacturing systems. The paper also provides insights into the future of automation through the Industrial Internet of Things (IIoT). The authors provide a general overview of the existing AOSE methodologies and their benefits without going into specific details or practical applications. The study concludes that AOSE has significant potential in I4.0/CPPS implementation and can contribute to developing the smart factory concept. However, it emphasizes the need for further research and exploration to leverage the benefits of agent-based CPPS.

Author's contributions on Pub. I

The main contribution to this paper (Cruz & Vogel-Heuser, 2017)**,** was the introduction of ten relevant requirements for agents-based CPPS. Additionally, the principal author contrasted typical AOSE methodologies, such as Gaia, Prometheus, etc., to identify their applicability to industrial applications, highlighting that the selected the majority of the selected AOSE show at least 50% coverage of the requirements on average. Nevertheless, some requirements need urgent attention since most of the AOSE methodologies do not consider them. Last but not least, the thesis' author explains the evolution of systems from embedded systems (i.e., based on Machine-to-machine communication) to the IIoT. The initial version of the manuscript (Cruz & Vogel-Heuser, 2017)**,** was written by the thesis' author.

This section was published as ***Pub. I (ReqsForCPPS)***:

> Cruz, S. L. A., & Vogel-Heuser, B. (2017). Comparison of agent oriented software methodologies to apply in cyber physical production systems. *15th International Conference on Industrial Informatics, INDIN*, 65–71. https://doi.org/10.1109/INDIN.2017.8104748

4.2 Publication II: "Platform Independent Multi-Agent System for Robust Networks of Production Systems" (Cruz S. et al., 2018)

Luis Alberto Cruz Salazar, Felix Mayer, Daniel Schütz and Birgit Vogel-Heuser

Summary of Pub. II (MASplatform)

This paper focuses on the problem of information control in manufacturing, specifically in the context of CPPS. The manuscript emphasizes that customized products require a flexible production process, and CPPS is a potential solution. However, implementing CPPS in heterogeneous production systems requires a platform-independent and robust software solution. The study proposes a MAS software solution for creating application-independent CPPS. The study aims to assess the efficiency of the design and implementation of a MAS, implementation in ANSI C and JAVA to support a variety of hardware platforms. The concept was evaluated through different use cases and experiments, providing robust and distributed software, and implementing in heterogeneous CPPS. Protocols and messages facilitate the communication and collaboration between IAs in the CPPS. The IAs can represent either physical systems or organizational entities. Physical systems agents manage access to the production system, dynamically regulating this access based on company policies. Organizational entity agents perform tasks such as diagnosis services and introducing production requests. IAs can dynamically reconfigure production systems to achieve load balancing within the CPPS network i.e., real-time response. The study also discusses using MAS to decentralize automation, enhance flexibility, and enable advanced functionality in production plants. The organizational entities periodically check all IAs for availability to update directories, which are distributed in a cloud among multiple nodes. This distributed directory, i.e., the DF, and the AMS pattern minimize search request time. The MAS software architecture is designed to represent different abstraction layers of communication, including hardware, protocol, and messages from FIPA standard. A hierarchical approach is adopted, with an IA interface responsible for handling connections and directory services. Each IA can have multiple communication interfaces to support different hardware platforms. In general, the study provides an overview of a MAS architecture and the functionality of the communication platform for CPPS, highlighting the potential benefits of heterarchical and isoarchical architectures. However, due to the lack of quantitative evaluation, the study primarily presents qualitative experiments and discussions.

Author's contributions on Pub. II

The main contribution to this paper (Cruz S. et al., 2018), was the description of the MAS requirements defined and an agent-based CPPS applied to a I4.0 scenario. Also, the author

proposed a MAS architecture for CPPS networks that is FIPA compliant (extended in Pub. XI). The partial manuscript (Cruz S. et al., 2018) was written and improved by the thesis' author.

This section was published as ***Pub. II (MASplatform):***

Cruz S., L. A., Mayer, F., Schütz, D., & Vogel-Heuser, B. (2018). Platform Independent Multi-Agent System for Robust Networks of Production Systems. *IFAC-PapersOnLine*, *51*(11), 1261–1268. https://doi.org/10.1016/j.ifacol.2018.08.359

4.3 Publication III: "Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns" (Cruz S. et al., 2019)

Luis Alberto Cruz Salazar, Daria Ryashentseva, Arndt Lüder and Birgit Vogel-Heuser

Summary of Pub. III (MASpatterns)

This article explores how design patterns focused on MAS can be applied in advanced manufacturing settings, specifically in CPPS. It offers a systematic classification and examination of MAS patterns, setting out criteria to quickly adapt them across different industrial sectors. The key contribution of this paper is in detailing existing MAS patterns that assist engineers and programmers in developing and enhancing the efficiency of manufacturing control. It compares various MAS solutions, providing a detailed survey and insights into their architectural IA patterns.

The paper also explores into the creation of agent-based CPPS and its structural design (functional requirements), supporting the integration and development of MAS at various domains (discrete manufacturing, continuous process, hybrid production), in line with traditional industry standards such as ISA 95. The MAS design pattern requirements provide a framework for developing and implementing CPPS architectures. These align with the RAMI4.0 standard and the AAS concept. The RAMI4.0's Asset Layer contains the physical elements managed by a MAS patterns, with Resource Agent (RA) patterns being key to asset management. However, the paper does not specify the direct relationship between RAs and assets. Communication is facilitated through the RAMI4.0's Communication Layer, utilizing CA patterns compatible with field bus and industrial communication protocols like OPC UA.

Not all IAs in the MAS directly control the assets—mostly RAs do—while MAS may also offer additional services like the (Agent Management System) AMS and the Directory Facilitator

(DF), following the FIPA standard. Another accepted IA pattern is the Communication Agent (CA) that converts proprietary interfaces into multiple industrial protocols. The topmost organization layer of RAMI4.0 can include the organizational structure's representation (Information Layer). Here, the Process Agent (PA) pattern enables the operation of organizations within diverse MAS. The Information Layer of RAMI4.0 holds semantic information about assets and the organization of PAs, which can represent a product in this layer. As PAs have their own information on procedures and plans, they coordinate their own production.

The article also examines four general research questions linked to eight hypotheses, five of which are fully- and three partially-true. RQs and true hypotheses address various aspects, including the depiction of MAS patterns for CPPS, the domains to which MAS patterns are applicable, and the reusability of MAS design patterns for CPPS; additionally, there are missing concerns, such as how MAS design patterns align with RAMI4.0 and functional and non-functional requirements, whose hypotheses are partially true.

As practical examples, two I4.0 demonstrators —myYoghurt and the Robot Integrated Agent Network "RIAN"— showcase the application of these patterns in agent-based CPPS. These case studies analyze and implement functional requirements like Resource Access, Knowledge Base, Coordination Process, and Communication Interface. However, the paper suggests further exploration of additional IA patterns and non-functional requirements is needed.

Author's contributions on Pub. III

The main contribution to this paper (Cruz S. et al., 2019), was the identification and categorization of the four IA patterns introduced: AMS, CA, PA, and RA. Those were preliminary works discussed in several international events (see publications VI, VII and VIII). Twenty MAS developed by the leading German FA 3.35 group in agent research (including personalized MAS figures), and patterns with their classification criteria (***Con1***). The members of the working group have been asked to report their IA patterns in its architecture using the criteria templates. Subsequently, as proof of evaluation (***Con5***), four IA patterns were evaluated by German FA 3.35 group and published in the VDI/VDE 2653 Sheet 4 guideline (see Pub. XII). The initial version of the manuscript (Cruz S. et al., 2019), was written by the thesis' author.

This section was published as ***Pub. III (MASpatterns)***:

> Cruz S., L. A., Ryashentseva, D., Lüder, A., & Vogel-Heuser, B. (2019). Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of

selected approaches mapping four agent patterns. *International Journal of Advanced Manufacturing Technology*, *105*(9), 4005–4034. https://doi.org/10.1007/s00170-019-03800-4

4.4 Publication IV: "CPPS-architecture and workflow for bringing agent-based technologies as a form of artificial intelligence into practice" (Cruz S. & Vogel-Heuser, 2022a)

Luis Alberto Cruz Salazar and Birgit Vogel-Heuser

Summary of Pub. IV (MARIANNE)

This paper discusses the classification of IAs based on their behavior. The motivation for this study is the increasing use of AI in production systems and the need for collaboration between I4.0 experts and autonomous systems. The study highlights the importance of existing standards and design patterns in realizing I4.0 systems and discusses the properties and capabilities of IAs in the context of AI. It distinguishes between Reactive Agents and Deliberative Agents, in which Reactive Agents have a faster response to their environment by simple situation-action associations. In contrast, Deliberative Agents have more sophisticated behavior and can behave proactively. The paper highlights the advantages and disadvantages of each type of agent and proposes the need for improved IA typologies to address the increasing capabilities of AI systems. Additionally, the study discusses the use of IA design patterns in Model-Driven Engineering (MDE) for MAS in the context of I4.0/CPPS, including integrating function patterns and knowledge bases and achieving autonomous and collaborative manufacturing systems. The study compares different I4.0 standardization efforts relevant to the implementation of AI using IAs and introduces a MAS architecture called "MARIANNE" for flexible and intelligent CPPS. The manuscript also provides an implementation guideline for implementing this agent-based CPPS. Overall, the study aims to provide a guide for implementing AI in the form of IAs in CPPS. MARIANNE includes physical access (in/output devices) represented by RAs but also uses the AMS, PA, and CA patterns, relating to the PPR model. The CPPS uses Python, AASXexplorer, Node-RED, and TwinCAT files. IA patterns in MARIANNE include types such as Control, Reasoning, Learning, and logical descriptions like Unit, Equipment, and Module from the ISA-88 model, but also apply the AAS concept for RAMI4.0 to achieve I4.0 systems.

Author's contributions on Pub. IV

The main contribution to this paper (Cruz S. & Vogel-Heuser, 2022a), was the development of an agent-based CPPS architecture derived from MAS patterns, named MARIANNE, and driven by the most recently IA standardization (IEEE, 2021; VDI/VDE, 2021). From the best of the thesis' author knowledge, there is no similar work that integrates both norms in a unique MAS architecture. Additionally, the main author presented a guideline to develop an agent-based CPPS aligned to RAMI4.0/PPR and extendable to I4.0 scenarios (see Pub. XI). The initial version of the manuscript (Cruz S. & Vogel-Heuser, 2022a), was written by the thesis' author.

This section was published as ***Pub. IV (MARIANNE):***

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

4.5 Publication V: "Industrial Artificial Intelligence: A Predictive Agent Concept for Industry 4.0" (Cruz S. & Vogel-Heuser, 2022b)

Luis Alberto Cruz Salazar and Birgit Vogel-Heuser

Summary of Pub. V (Agent4.0)

This paper focuses on the concept of Industrial AI in the context of I4.0. The innovative part is the authors introduce a predictive IA concept called Agent4.0 that applies supervised learning to increase the predictability of an agent-based CPPS. They suggest that IA patterns can be used to represent distributable AI in various I4.0 scenarios. The study evaluates the five Industrial AI characteristics for the Agent4.0: autonomy, reactiveness, proactiveness, human cooperativeness, and the ability to learn from online/offline operations. The study contributes to the IA patterns field by introducing a "learning element" into the Agent4.0, which collaborates with the AAS in an I4.0 demonstrator. This distinguishes it from existing IA design patterns that are primarily reactive and proactive, with limited predictive capabilities. The authors highlight that probability is not predictability, indicating the need for a distinction between probabilistic and predictive IA models. They emphasize the importance of training the Agent4.0 through ML methods to achieve predictive systems. Then, acquisition of data and other ML methods such as artificial neural networks, fuzzy logic, and linear regression are proposed as ways to support decision-making in intelligent systems. As limitation, definitions of predictive IA and AI used in this

study may not be universally accepted, leading to potential ambiguity and lack of consensus within the AI and I4.0 communities. The acquisition of complex and comprehensive data required for training IA's Knowledge Base models may pose challenges, as obtaining such data can be difficult in industrial settings, i.e., hard real-time capable. The study focuses on a specific ML concept (linear regression) into Agent4.0, which may limit the generalizability of the findings to other AI applications and MAS architectures. Overall, the study contributes to the advancement of agent-based technologies and AI in practice.

Author's contributions on Pub. V

The main contribution to this paper (Cruz S. & Vogel-Heuser, 2022b), was the introduction of the reusable Agent4.0 concept and its skills description. This is an ML-based agent into a MAS architecture by an I4.0 demonstrator. The tools implemented were preliminary used in collaboration with the author on a Small- and Middle-sized Enterprises (see Pub. X). The initial version of the manuscript (Cruz S. & Vogel-Heuser, 2022b), was written by the thesis' author.

This section was published as ***Pub. V (Agent4.0):***

> Cruz S., L. A., & Vogel-Heuser, B. (2022b). Industrial Artificial Intelligence: A Predictive Agent Concept for Industry 4.0. *IEEE 20th International Conference on Industrial Informatics (INDIN)*, 27–32. https://doi.org/10.1109/INDIN51773.2022.9976159

5. Discussion and outlook

A brief overview of the research contribution is discussed to summarize the thesis. It connects the results to the issues raised in the Introduction. After that, if the requirements have been fulfilled, it is reviewed, and a future work as outlook is provided.

5.1 Main publications results related to the issues

This section outlines the main contribution of the thesis throughout the Table. 13, which summarizes the main results from the contributions that supported the concept design and evaluation of this dissertation, related to the introduced issues and RQs (see Section 1).

Table 13: Overview of the thesis' storyline contributions.

Issues	Research Questions	Contributions
Lack of a comprehensive overview and classification of MAS patterns in CPPS	**RQ1.** How are MAS patterns for CPPS depicted and what criteria are used to describe them?	***Con1.*** A well-discussed criteria and systematic summaries in the industrial agents working group of the German IFAC NMO GMA FA 3.35 is presented
	RQ2. For which domains of CPPS are the MAS patterns designed and applicable?	***Con2.*** Mapping of MAS functional requirements to sub-agent patterns was provided considering their capabilities and skills

Challenge of reusability and extendibility of MAS patterns for I4.0	**RQ3.** What are the reusable MAS patterns for CPPS?	***Con3.*** Proposed IA patterns for agent-based CPPS in I4.0 demonstrators and selected I4.0 scenarios are applicable
Integration of MAS patterns with existing CPPS models and standards	**RQ4.** How can MAS patterns be used into a CPPS aligned with the RAMI4.0 and PPR model?	***Con4.*** The identified design patterns are the basis for the development of an agent-based CPPS architecture and for its structural representation, aligned with established models e.g. PPR
Implementing sub-agent patterns and AASs into hybrid CPPS platforms		***Con5.*** *An VDI/VDE g*uideline is used as proof of evaluation for the impact of the IA patterns and AASs implementation into hybrid CPPS platforms

The thesis enriches the literature on IA patterns research in the CPPS domains, which is scarce since there were only few publications on the topic primarily targeting IA patterns standardization. Furthermore, the stepwise research for challenges presented in Section 1.2 can be employed in future work on the agent-based CPPS. Research contributions related to the four issues that are initially mentioned in the Introduction (see Section 1.4), is covered as follows:

Issue 1: *Lack of a comprehensive overview and classification of MAS patterns in CPPS.*

MAS approaches are classified in order to facilitate the migration from the conventional automation systems to the CPPS in ***Con1***. Therefore, authors use a template that consists of a list of classification criteria validated by experts in the German community FA 5.15 (see Section 3.1). Because all MAS approaches were created to be used in different domains and different layers of the automation pyramid, a notable part of the IA pattern functional requirements are concentrated to provide autonomy, reactiveness, proactiveness, predictability, and human cooperativeness. Their capabilities and skills of these IA capabilities and skills are enlisted and described in ***Con2***.

Issue 2: *Challenge of reusability and extendibility of MAS design patterns for I4.0.*

The reusability and extendibility of MAS design patterns, particularly in the scope of I4.0, are strongly exemplified through ***Con3***. This contribution showcases I4.0 demonstrators, such as RIAN, the myYoghurt plant, and the xPPU, which highlight the adaptability and scalability of MAS design patterns across diverse industrial domains (see Section 3.3). By implementing these patterns in various contexts, from discrete hybrid manufacturing, ***Con3*** not only proves their effectiveness but also their capacity for reuse and extension in response to the evolving requirements of I4.0/CPPS. The qualitative assessment of IA interfaces within these demonstrators, as mentioned in ***Con3***, provides critical insights into the integration and effectiveness of IA design patterns, further supporting the argument for their broad applicability and potential for innovation in CPPS. This directly confronts Issue 2 by demonstrating practical

applications and reinforcing the need for IA patterns that are both reusable and extendable, ensuring they can evolve alongside I4.0 demonstrators and by driven by standards.

Issue 3: *Integration of MAS design patterns with existing CPPS models and standards.*
Con4 and ***Con5*** address this issue integrating MAS patterns with established I4.0 models and standards. ***Con4*** outlines a comprehensive MAS architecture that leverages the concept of the AAS from RAMI4.0, illustrating a strategic alignment that enhances CPPS's interoperability and digital twin capabilities. This architecture ensures that IA patterns are not only compatible with I4.0 standards but are fundamentally designed to enhance CPPS functionality through improved data integration, communication, and operational efficiency. Furthermore, ***Con5*** emphasis on the standardization of IA patterns builds upon this foundation by offering a methodical approach to embedding IA patterns within CPPS. This includes the development of predictive agents, such as Agent4.0, which aligns with the MARIANNE framework, demonstrating a forward-thinking approach to CPPS design that integrates IA capabilities and skills. By collectively focusing on harmonization with standards like RAMI4.0 and the PPR model, these contributions address the critical need for a unified framework that not only respects existing standards but also drives CPPS towards better implementation.

Issue 4: *Implementing sub-agent patterns and AASs into hybrid CPPS platforms.*
The dual contributions of ***Con4*** and ***Con5*** again come to the face in addressing Issue 4, focusing on the challenges associated with implementing sub-agent patterns and AASs within complex, hybrid CPPS platforms. ***Con4***'s detailed exposition on MAS architecture and its integration with AASs highlights the necessity of a robust and adaptable framework to CPPS. This architecture not only supports the integration of digital twins through AASs but also ensures CPPS can fully benefit from the autonomy, interoperability, and adaptability promised by I4.0. Meanwhile, ***Con5*** extends this integration by standardizing IA patterns, thus facilitating the deployment of sub-agent patterns in a manner that is both effective and in alignment with industry standards such as RAMI4.0 and the PPR model. This concerted effort is key in scaling the implementation challenges, enhancing CPPS functionalities, and ensuring a unified integration process. Through the strategic alignment of IA patterns with digital twin capabilities and the standardization of IA patterns, contributions 4 and 5 provide a clear roadmap for overcoming the inherent complexities of implementing advanced MAS structures within hybrid CPPS environments.

5.2 Fulfillment of the requirements and the covered CPPS challenges

Table 14 presents the results of a self-evaluation of requirements fulfillment introduced before (see Section 2.). Since a requirement is still only partially met, significant concerns may be covered. Nevertheless, the three conditions that were met demonstrate the success of this thesis.

Regarding MAS classification (Req1), **Pub. I** and **Pub. III** detail classification methods for MAS approaches by describing the criteria used to depict agent-based CPPS architectures, including those based on AOSE. Those publications state that the classification criteria deliver valid and decidable information for the evaluation of MAS approaches, detailed in Section 3.1 (***Con1***). **Pub. III** also mentions that MAS approaches for CPPS can be classified and identified based on similar and reusable design pattern terms (Req3).

Table 14: Summary of the rating of requirement fulfilment.

Requirement	Rating*	Detailed rating and reference to publication
Req1-Classification	●	Fulfilled – An enlargement of the collected classification criteria (cp. Table 5 and Table 6) is reported in **Pub. III** and detailed in Section 3.1 (***Con1***). Additionally, the **Pub. I** and **Pub. III** provide surveys of MAS practices based on main AOSE methodologies, differentiating them by CPPS requirement' classification
Req2-Domain	●	Fulfilled – **Pub. III** presents agent-based CPPS requirements and MAS architectures, mapping four IA patterns and their domains of application, detailed in Section 3.2 (***Con2***). **Pub. IV** and **Pub. V** define and evaluate IA pattern capabilities skills, relating Industrial AI characteristics and providing applicability in a I40 demonstrator with an extendible domain
Req3-Reusability	●	Fulfilled – **Pub II** introduces the characterization of reusable IA patterns applicable into I4.0 demonstrators, detailed in Section 3.3 (***Con3***). Proof of its reusability is the normalized patterns which are Resource Agents, Process Agents, Communication Agents, and Agent Management Systems (cp. Table 10). Consequently, **Pub. IV** and **Pub. V** develop a proposed MAS architecture based on these reusable patterns and named MARIANNE. **Pub. II** to **Pub. V** introduce various I4.0 demonstrators where IA patterns were applied and a selected I4.0 scenario was derived.
Req4-Modelling	◑	Partially fulfilled – **Pub. III** combines RAMI4.0, and IA patterns suitability for CPPS, considering the AAS model, detailed in Section 3.4 (***Con4***). Additionally in Section 3.5 (***Con5***), **Pub. IV** and **Pub. V** make three contributions regarding modelling and standardization: examining the combination of VDI/VDE 2653-4 and IEEE 2660.1 standards, presenting an MAS architecture for CPPS derived from IA patterns (MARIANNE), and providing guidelines for implementing IAs and AASs into hybrid CPPS platforms. In addition, **Pub. VII** and **Pub. VIII** only reported early findings and initial I4.0 scenarios with their PPR modeling. Therefore, those results can be seen as complementary results of this thesis. Thus, further work is needed to incorporate more aspects of RAMI4.0 and PPR to build up a major comprehensive agent-based CPPS model for I4.0.

*Rating means ● fulfilled, and ◑ partially fulfilled.

Additionally, **Pub. I** discusses how AOSE can be developed into a CPPS-aligned framework. Both **Pub. I** and **Pub. III**, have selected MAS architectures, methodologies, or standards for classification, surveying the state of industrial MAS practices (Req1). Primarily, regarding the CPPS requirements, the capability to interface with various application domains (Req2) is ensured through the utilization of an open software MAS architecture, as elaborated in **Pub. II** to **Pub. V.** Independence from specific levels and diverse domains is realized by the incorporation of four types of IA patterns (Req3): Resource Agents (RAs), Process Agents (PAs),

Communication Agents (CAs), and Agent Management Systems (AMSs), detailed in Section 3.2 (***Con2***).

The implementation of TCP/IP as the basis of the communication protocol, exemplified by OPC UA, addresses certain aspects of error handling and recovery, and facilitates the integration of CPPS networks with other application domains (Req2). The distribution of organizational sub-agents within cloud environments, such as PAs and AMSs, leads to a decentralized agent-based CPPS with reusable structure (***Con4***), as depicted in **Pub. II**, **Pub. IV,** and **Pub. V**. However, the Acceptance of I4.0 components and the capability of sub-agents in PPR model (Req4) require additional investigation and more evaluation experiments. Preliminary evaluations of the PPR model and IAs have been conducted and are documented in the complementary references of this thesis by **Pub. VII** and **Pub. VIII**.

To fulfill the RAMI4.0 requirements (Req4), I4.0 demonstrators are assembled out of I4.0 components, accommodating various engineering models and standards, detailed in Section 3.3 (***Con3***). The focus of the MAS architecture on software components, particularly sub-agent patterns, facilitates the association of physical asset connections via AAS (Req3), aligning with the functional requirements of AutomationML, detailed in Section 3.4 (***Con4***). The principles of system boundaries and nestability for I4.0 components introduced in **Pub. III**, are further organized along RAMI4.0 layers and architecture axis within the MAS (Req4), respectively, which is implemented in **Pub. IV** and **Pub. V**. The general AAS model, discussed in **Pub. III** to **Pub. V** achieves the virtual representation –digital twin concept– and the functional characteristics of I4.0 components from RAMI4.0 (Req4). However, the agent-based CPPS architecture has not yet articulated non-functional requirements as a kind of feature of model elements (Req3), as noted in **Pub. III**. This limits the modeling (Req4), such as explicit quality characteristics or evaluation metric attributes, which would detail the extent to which the IA patterns fully fulfill their supported models associated to I4.0.

Pub. IV and **Pub. V** uses the VDI/VDE 2653-4 and IEEE 2660.1 standards (***Con5***), proposing and evaluating the applied IA patterns for the MARIANNE architecture for the xPPU demonstrator, as detailed in Section 3.5 (***Con5***). MARIANNE incorporates IA patterns based on these standards and focuses on the relationships with I4.0 concepts such as RAMI4.0/AAS, and the PPR model (Req4). **Pub. IV** suggests that further analysis should be done to execute these existing models for I4.0, considering various aspects such as function hierarchy levels and

information classes. Above all presented findings, the enlargement of agent-based CPPS based on design pattern and its covered challenges is summarized in Fig. 13 (related to the challenges in Fig. 2), that is considered the main contribution of this thesis.

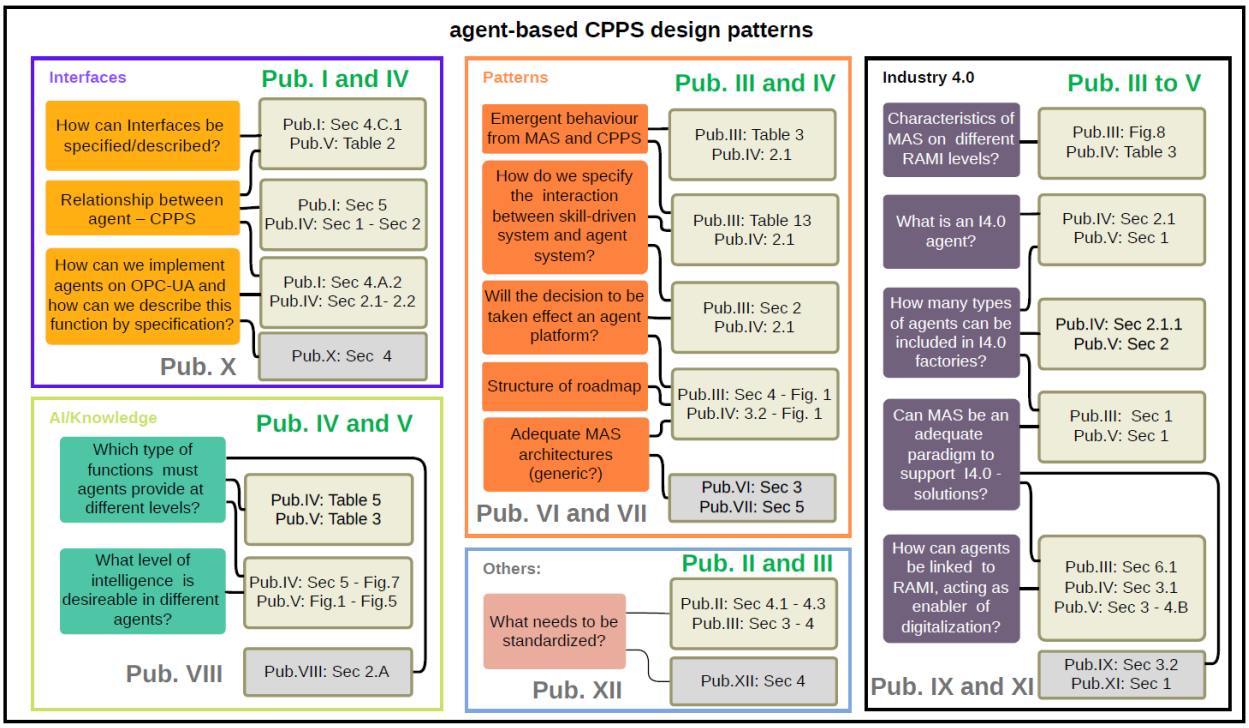

Figure 13: Challenges and gaps of Industrial Agents – related thesis' contribution (selected RQs from Fig. 2).
Publications refer to: Pub. I (Cruz S. & Rojas A., 2013); Pub. II (Cruz S. et al., 2018); Pub. III (Cruz S. et al., 2019); Pub.IV (Cruz S. & Vogel-Heuser, 2022a); Pub.V (Cruz S. & Vogel-Heuser, 2022b); Pub. VI (Vogel-Heuser et al., 2018); Pub. VII (Lüder et al., 2018); Pub. VIII (Ryashentseva et al., 2018); Pub. IX (Vogel-Heuser et al., 2020); Pub. X (Haben et al., 2021); Pub. XI (Seitz et al., 2021); and Pub. XII (VDI/VDE, 2021). Source: Adapted from the presentation of the Workshop "Agents in agile manufacturing (CPPS) - Status of Last Meeting", AIS-TUM, 2019.

5.3 Conclusion and outlook

The advancement of IA patterns within CPPS raises at a fundamental connection, ready for significant evolution in the context of I4.0. The demonstrable success of MAS in addressing complex integration, reusability, and scalability challenges, as evidenced by this thesis contribution, sets a solid foundation for future innovations in agent-based CPPS. The practical applications showcased through I4.0 demonstrators highlight the adaptability of MAS patterns, promising a future where CPPS are increasingly dynamic, interoperable, and efficient. The alignment of MARIANNE architecture with I4.0 standards, particularly through the integration with RAMI4.0 and the AAS, underscores a strategic direction towards enhanced DT capabilities and system interoperability. This alignment not only facilitates a seamless integration of MAS within existing CPPS models and standards but also drives the development of intelligent,

autonomous systems that can effectively respond to the complex demands of modern manufacturing environments. Looking ahead, the standardization of IA patterns and the incorporation of advanced IAs, such as Agent4.0, within the MAS architecture signal a move towards more predictive, self-optimizing CPPS. This evolution will likely emphasize the need for MAS frameworks that are not only adaptable to various industrial domains but also capable of anticipating and reacting to changes in real-time, thereby enhancing the resilience and efficiency of manufacturing processes. Furthermore, the challenges of implementing sub-agent patterns and AASs in hybrid CPPS platforms will drive innovations in MAS design, focusing on the integration of complex IA capabilities and skills. A well-discussed and comprehensive VDI/VDE guideline for effectively deploying IA patterns is crucial in probing agent-based CPPS challenges and ensuring CPPSs can leverage the total target of I4.0. Thus, the issues and RQs stated in the introduction have been successfully answered, and the IA patterns with the proposed MARIANNE architecture address the research gap by fulfilling most requirements.

As a future work, creating and standardizing KPIs specific to agent-based CPPS will enable MAS developers to measure system performance against industry benchmarks, facilitating a more objective assessment of their operational efficacy, scalability, and resilience. As well as the standardized IA patterns, these KPIs must be normalized, including metrics related to system adaptability, integration success rates, efficiency gains, and the effectiveness of predictive and autonomous functionalities within CPPS. By quantifying these aspects, MAS developers can better estimate the return on investment of implementing IA patterns and justify further innovation and adoption in the industry domain.

In conclusion, the future of MAS patterns within CPPS for I4.0 looks promising, with a clear trajectory toward more intelligent, flexible, and interconnected aPS. The ongoing research and development efforts, as highlighted through the contributions discussed, will indeed play a critical role in shaping the next generation of CPPS, ultimately contributing to the realization of the smart factory vision.

6. References

Albrecht, S. V, Christianos, F., & Schäfer, L. (2024). *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press. https://www.marl-book.com

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction* (Vol. 2).

Badr, I. (2011). *Agent-based dynamic scheduling for flexible manufacturing systems*. PhD thesis, Faculty 5: Computer Science, Electrical Engineering and Information Technology, University of Stuttgart.

Barata, J., Jassbi, J., & Nikghadam-Hojjati, S. (2022). *A Framework of Collaborative Multi-actor Approach Based Digital Agriculture as a Solution for the Farm to Fork Strategy* (pp. 503–518). https://doi.org/10.1007/978-3-031-14844-6_40

Barbosa, J. (2016). Self-organized and evolvable holonic architecture for manufacturing control [PhD thesis, Université de Valenciennes et du Hainaut-Cambresis]. In *PHD Theses*. https://tel.archives-ouvertes.fr/tel-01137643v2

Baumgartel, H., & Verbeet, R. (2020). Service and Agent based System Architectures for Industrie 4.0 Systems. *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 1–6. https://doi.org/10.1109/NOMS47738.2020.9110406

Bendjelloul, A., Mihoubi, B., Gaham, M., Moufid, M., & Bouzouia, B. (2022). A framework for an effective virtual commissioning of agent-based cyber-physical production systems integrated into manufacturing facilities. *Concurrent Engineering*, *30*(4), 399–410. https://doi.org/10.1177/1063293X221121819

Bloom, G., Alsulami, B., Nwafor, E., & Bertolotti, I. C. (2018). Design patterns for the industrial Internet of Things. *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 1–10. https://doi.org/10.1109/WFCS.2018.8402353

Calà, A. (2019). *A novel migration approach towards decentralized automation in cyber-physical production systems*. http://dx.doi.org/10.25673/13760

Calà, A., Lüder, A., Cachada, A., Pires, F., Barbosa, J., Leitão, P., & Gepp, M. (2017). Migration from traditional towards cyber-physical production systems. *Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017*, 1147–1152. https://doi.org/10.1109/INDIN.2017.8104935

Cardin, O. (2019). Classification of cyber-physical production systems applications: Proposition of an analysis framework. *Computers in Industry*. https://doi.org/10.1016/j.compind.2018.10.002

Cardoso, R. C., & Ferrando, A. (2021). A Review of Agent-Based Programming for Multi-Agent Systems. *Computers*, *10*(2), 16. https://doi.org/10.3390/computers10020016

Case, D. M. (2015). *Engineering complex systems with multigroup agents* [PhD thesis, Computing and Information Sciences, Kansas State University]. http://hdl.handle.net/2097/19045

Cha, S., Vogel-Heuser, B., & Fischer, J. (2020). Analysis of metamodels for model-based production automation system engineering. *IET Collaborative Intelligent Manufacturing*, *2*(2), 45–55. https://doi.org/10.1049/iet-cim.2020.0013

Charpenay, V., Schraudner, D., Seidelmann, T., Spieldenner, T., Weise, J., Schubotz, R., Mostaghim, S., & Harth, A. (2021). MOSAIK: A Formal Model for Self-Organizing Manufacturing Systems. *IEEE Pervasive Computing*, *20*(1), 9–18. https://doi.org/10.1109/MPRV.2020.3035837

Chitchyan, R., Pinto, M., Rashid, A., & Fuentes, L. (2007). COMPASS: Composition-Centric Mapping of Aspectual Requirements to Architecture. In *Transactions on Aspect-Oriented Software Development IV* (pp. 3–53). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77042-8_2

Colombo, A. W., Bangemann, T., & Karnouskos, S. (2013). A system of systems view on collaborative industrial automation. *Proceedings of the IEEE International Conference on Industrial Technology*. https://doi.org/10.1109/ICIT.2013.6505980

Colombo, A. W., Karnouskos, S., Yu, X., Kaynak, O., Luo, R. C., Shi, Y., Leitão, P., Ribeiro, L., & Haase, J. (2021). A 70-Year Industrial Electronics Society Evolution Through Industrial Revolutions: The Rise and Flourishing of Information and Communication Technologies. *IEEE Industrial Electronics Magazine*, *15*(1), 115–126. https://doi.org/10.1109/MIE.2020.3028058

Cruz S., L. A., Mayer, F., Schütz, D., & Vogel-Heuser, B. (2018). Platform Independent Multi-Agent System for Robust Networks of Production Systems. *IFAC-PapersOnLine*, *51*(11), 1261–1268. https://doi.org/10.1016/j.ifacol.2018.08.359

Cruz S., L. A., & Rojas A., O. A. (2013). Comparación de enfoques de sistemas de control tradicionales y el paradigma de los Sistemas Holónicos de Manufactura. *II International Congress of Engineering Mechatronics and Automation, CIIMA*, 6.

Cruz S., L. A., Ryashentseva, D., Lüder, A., & Vogel-Heuser, B. (2019). Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. *International Journal of Advanced Manufacturing Technology*, *105*(9), 4005–4034. https://doi.org/10.1007/s00170-019-03800-4

Cruz, S. L. A., & Vogel-Heuser, B. (2017). Comparison of agent oriented software methodologies to apply in cyber physical production systems. *15th International Conference on Industrial Informatics, INDIN*, 65–71. https://doi.org/10.1109/INDIN.2017.8104748

Cruz S., L. A., & Vogel-Heuser, B. (2020). Applying Core Features of the Object-Oriented Programming Paradigm by Function Blocks Based on the IEC 61131 and IEC 61499 Industrial Automation Norms. In T. Borangiu, D. Trentesaux, P. Leitao, A. Giret Boggino, & V. Botti Navarro (Eds.), *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future* (1st ed., pp. 273–289). Springer International Publishing. https://doi.org/10.1007/978-3-030-27477-1_21

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

Cruz S., L. A., & Vogel-Heuser, B. (2022b). Industrial Artificial Intelligence: A Predictive Agent Concept for Industry 4.0. *IEEE 20th International Conference on Industrial Informatics (INDIN)*, 27–32. https://doi.org/10.1109/INDIN51773.2022.9976159

D'Avila Mendonça, G., Filho, I. P. de S., & Guedes, G. T. A. (2022). *A Detailed Analysis of a Systematic Review About Requirements Engineering Processes for Multi-agent Systems* (pp. 46–69). https://doi.org/10.1007/978-3-031-10161-8_3

DIN SPEC. (2016). *Reference Architecture Model Industrie 4.0 (RAMI4.0)*.

Fast-Berglund, Å., Romero, D., Åkerman, M., Hodig, B., & Pichler, A. (2020). *Agent- and Skill-Based Process Interoperability for Socio-Technical Production Systems-of-Systems* (pp. 46–54). https://doi.org/10.1007/978-3-030-57997-5_6

Fay, A., Vogel-Heuser, B., Frank, T., Eckert, K., Hadlich, T., & Diedrich, C. (2015). Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns. *Journal of Systems and Software*, *101*, 221–235. https://doi.org/10.1016/j.jss.2014.12.028

Fay, A., Vogel-Heuser, B., Seitz, M., Gehlhoff, F., & VDI/VDE. (2019). *Agents for the realisation of Industrie 4.0*. https://www.vdi.de/ueber-uns/presse/publikationen/details/agents-for-the-realisation-of-industrie-40

Fischer, J., Lieberoth-Leden, C., Fottner, J., & Vogel-Heuser, B. (2020). Design, Application, and Evaluation of a Multiagent System in the Logistics Domain. *IEEE Transactions on Automation Science and Engineering*, 1–14. https://doi.org/10.1109/TASE.2020.2979137

Fischer, J., Vogel-Heuser, B., Berscheit, A., & Parigger, S. (2021). Comparison of Two Concepts for Planned Reuse of Variant-rich IEC 61131-3-based Control Software. *2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 713–720. https://doi.org/10.1109/IEEM50564.2021.9672967

Fischer, J., Vogel-Heuser, B., Schneider, H., Langer, N., Felger, M., & Bengel, M. (2021). Measuring the Overall Complexity of Graphical and Textual IEC 61131-3 Control Software. *IEEE Robotics and Automation Letters*, *6*(3), 5784–5791. https://doi.org/10.1109/LRA.2021.3084886

Franklin, S., & Graesser, A. (1997). Is It an agent, or just a program?: A taxonomy for autonomous agents. In *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages* (pp. 21–35). https://doi.org/10.1007/BFb0013570

Fuchs, J., Feldmann, S., Legat, C., & Vogel-Heuser, B. (2014). Identification of Design Patterns for IEC 61131-3 in Machine and Plant Manufacturing. *IFAC Proceedings Volumes*, *47*(3), 6092–6097. https://doi.org/10.3182/20140824-6-ZA-1003.01595

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education. https://books.google.de/books?id=6oHuKQe3TjQC

Gangoiti, U., López, A., Armentia, A., Estévez, E., & Marcos, M. (2021). Model-Driven Design and Development of Flexible Automated Production Control Configurations for Industry 4.0. *Applied Sciences*, *11*(5), 2319. https://doi.org/10.3390/app11052319

Gangoiti, U., López-García, A., Armentia, A., Estevez, E., Casquero, O., & Marcos, M. (2022). A Customizable Architecture for Application-Centric Management of Context-Aware Applications. *IEEE Access*, *10*, 1603–1625. https://doi.org/10.1109/ACCESS.2021.3138586

Gehlhoff, F. (2023). *Agent-based Decentralised Architecture for Integrated Process Planning and Scheduling of Transport and Production Processes*. PhD thesis, Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg.

Gehlhoff, F., & Fay, A. (2020). Agent-based decentralised architecture for multi-stage and integrated scheduling. *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1443–1446. https://doi.org/10.1109/ETFA46521.2020.9212059

Haben, F., Vogel-Heuser, B., Najjari, H., Seitz, M., Trunzer, E., & Cruz S., L. A. (2021). Low-entry Barrier Multi-Agent System for Small- and Middle-sized Enterprises in the Sector of Automated Production Systems. *International Conference on Industrial Engineering and Engineering Management (IEEM)*, 1351–1357. https://doi.org/10.1109/IEEM50564.2021.9672973

Heylighen, F. (2023). The meaning and origin of goal-directedness: a dynamical systems perspective. *Biological Journal of the Linnean Society*, *139*(4), 370–387. https://doi.org/10.1093/biolinnean/blac060

Hoffmann, M. (2017). *Adaptive and Scalable Information Modeling to Enable Autonomous Decision Making for Real-Time Interoperable Factories*. PhD thesis, Faculty of Mechanical Engineering, RWTH Aachen.

Hoffmann, M., Meisen, T., & Jeschke, S. (2017). OPC UA Based ERP Agents: Enabling Scalable Communication Solutions in Heterogeneous Automation Environments. *Intern. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*.

Holvoet, T., Weyns, D., & Valckenaers, P. (2009). Patterns of Delegate MAS. *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 1–9. https://doi.org/10.1109/SASO.2009.31

Hurtado, J. (2012). Metodología de la investigación guía para la comprensión Holística para la ciencia. In *Quirón Ediciones*. Ediciones Quirón-Sypal.

IEEE. (2005). *Foundation for Intelligent Physical Agents FIPA - Specifications*. http://www.fipa.org/repository/standardspecs.html

IEEE. (2021). IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions. *IEEE Std 2660.1-2020*, 1–43. https://doi.org/10.1109/IEEESTD.2021.9340089

Jimenez, J. F., Bekrar, A., Zambrano-Rey, G., Trentesaux, D., & Leitão, P. (2017). Pollux: a dynamic hybrid control architecture for flexible job shop systems. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2016.1218087

Juziuk, J., Weyns, D., & Holvoet, T. (2014). Design Patterns for Multi-agent Systems: A Systematic Literature Review. In *Agent-Oriented Software Engineering* (pp. 79–99). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-54432-3_5

Karnouskos, S., Leitão, P., Ribeiro, L., & Colombo, A. W. (2020). Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems: Multiagent Systems Entering Industry 4.0. *IEEE Industrial Electronics Magazine*, *14*(3), 18–32. https://doi.org/10.1109/MIE.2019.2962225

Karnouskos, S., Ribeiro, L., Leitão, P., Lüder, A., & Vogel-Heuser, B. (2019). Key Directions for Industrial Agent Based Cyber-Physical Production Systems. *IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, 17–22. https://doi.org/10.1109/icphys.2019.8780360

Karnouskos, S., Sinha, R., Leitão, P., Ribeiro, L., & Strasser, Thomas. I. (2018). Assessing the Integration of Software Agents and Industrial Automation Systems with ISO/IEC 25010. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 61–66. https://doi.org/10.1109/INDIN.2018.8471951

Kovalenko, I. (2020). *Intelligent Product Agents for Multi-Agent Control of Manufacturing Systems*. PhD thesis, University of Michigan, Departament of Mechanical Engineering.

Kovalenko, I., Tilbury, D., & Barton, K. (2019). The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. *Control Engineering Practice*, *86*, 105–117. https://doi.org/10.1016/j.conengprac.2019.03.009

Kruger, K., & Basson, A. H. (2019). Evaluation of JADE multi-agent system and Erlang holonic control implementations for a manufacturing cell. *International Journal of Computer Integrated Manufacturing*, *32*(3), 225–240. https://doi.org/10.1080/0951192X.2019.1571231

Land, K., Nardin, L. G., & Vogel-Heuser, B. (2023). Increasing Robustness of Agents' Decision-Making in Production Automation using Sanctioning. *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 1–6. https://doi.org/10.1109/INDIN51400.2023.10217852

Lee, E. A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors (Switzerland)*, *15*(3), 4837–4869. https://doi.org/10.3390/s150304837

Lee, J., Bagheri, B., & Kao, H.-A. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, *3*, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001

Leitão, P., Colombo, A. W., & Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, *81*, 11–25. https://doi.org/10.1016/j.compind.2015.08.004

Leitão, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., & Colombo, A. W. (2016). Smart Agents in Industrial Cyber–Physical Systems. *Proceedings of the IEEE*, *104*(5), 1086–1101. https://doi.org/10.1109/JPROC.2016.2521931

Leitão, P., Karnouskos, S., Ribeiro, L., Moutis, P., Barbosa, J., & Strasser, Thomas. I. (2018). Integration Patterns for Interfacing Software Agents with Industrial Automation Systems. *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2908–2913. https://doi.org/10.1109/IECON.2018.8591641

Leitão, P., Karnouskos, S., Strasser, T. I., Jia, X., Lee, J., & Colombo, A. W. (2023). Alignment of the IEEE Industrial Agents Recommended Practice Standard With the Reference Architectures RAMI4.0, IIRA, and SGAM. *IEEE Open Journal of the Industrial Electronics Society*, *4*, 98–111. https://doi.org/10.1109/OJIES.2023.3262549

Leitão, P., & Strasser, T. (2016). Analyzing standardization needs for applying agent technology in industrial environments. *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, 754–759. https://doi.org/10.1109/ISIE.2016.7744984

Leitão, P., Strasser, T. I., Karnouskos, S., Ribeiro, L., Barbosa, J., & Huang, V. (2021). Recommendation of Best Practices for Industrial Agent Systems based on the IEEE 2660.1 Standard. *Proceedings of the IEEE International Conference on Industrial Technology*, *2021-March*, 1157–1162. https://doi.org/10.1109/ICIT46573.2021.9453511

López-García, A. (2023). *Industrial agents for resilient manufacturing systems. An I4.0 plantform for the manufacturing domain.*

López-García, A., Casquero, O., Estévez, E., Armentia, A., Orive, D., & Marcos, M. (2023). An industrial agent-based customizable platform for I4.0 manufacturing systems. *Computers in Industry*, *146*, 103859. https://doi.org/10.1016/j.compind.2023.103859

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

Lüder, A., Behnert, A.-K., Rinker, F., & Biffl, S. (2020). Generating Industry 4.0 Asset Administration Shells with Data from Engineering Data Logistics. *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 867–874. https://doi.org/10.1109/ETFA46521.2020.9212149

Lüder, A., Calà, A., Zawisza, J., & Rosendahl, R. (2017). Design pattern for agent based production system control — A survey. *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, 717–722. https://doi.org/10.1109/COASE.2017.8256187

Lüder, A., Zawisza, J., Cruz S., L. A., Seitz, M., & Vogel-Heuser, B. (2018). Identifying Design Pattern for Agent Based Production System Control. *44th Annual Conference of the IEEE Industrial Electronics Society, IECON*, 2896–2901. https://doi.org/10.1109/IECON.2018.8591336

Marschall, B., Ochsenkuehn, D., & Voigt, T. (2022). Design and Implementation of a Smart, Product-Led Production Control Using Industrial Agents. *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, *3*(1), 48–56. https://doi.org/10.1109/JESTIE.2021.3117121

Marschall, B., Schleicher, M., Sollich, A., Becker, T., & Voigt, T. (2022). Design and Installation of an Agent-Controlled Cyber-Physical Production System Using the Example of a Beverage Bottling Plant. *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, *3*(1), 39–47. https://doi.org/10.1109/JESTIE.2021.3097941

Melo, L. S., Sampaio, R. F., Leão, R. P. S., Barroso, G. C., & Bezerra, J. R. (2019). Python-based multi-agent platform for application on power grids. *International Transactions on Electrical Energy Systems*, *29*(6). https://doi.org/10.1002/2050-7038.12012

Mendonça, G., Filho, I., & Guedes, G. (2021). A Systematic Review about Requirements Engineering Processes for Multi-Agent Systems. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, 69–79. https://doi.org/10.5220/0010240500690079

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., & Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, *65*(2), 621–641. https://doi.org/10.1016/j.cirp.2016.06.005

Neumann, E.-M., Vogel-Heuser, B., Fischer, J., Ocker, F., Diehm, S., & Schwarz, M. (2020). Formalization of Design Patterns and Their Automatic Identification in PLC Software for Architecture Assessment. *IFAC-PapersOnLine*, *53*(2), 7819–7826. https://doi.org/10.1016/j.ifacol.2020.12.1881

Nouiri, M., Trentesaux, D., & Bekrar, A. (2019). Towards Energy Efficient Scheduling of Manufacturing Systems through Collaboration between Cyber Physical Production and Energy Systems. *Energies*, *12*(23), 4448. https://doi.org/10.3390/en12234448

Panetto, H., Iung, B., Ivanov, D., Weichhart, G., & Wang, X. (2019). Challenges for the cyber-physical manufacturing enterprises of the future. *Annual Reviews in Control*, *47*, 200–213. https://doi.org/10.1016/j.arcontrol.2019.02.002

Papoudakis, G., Christianos, F., Schäfer, L., & Albrecht, S. V. (2021). Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. *Advances in Neural Information Processing Systems - Datasets and Benchmarks Track*. http://arxiv.org/abs/2006.07869

Patil, S., Drozdov, D., & Vyatkin, V. (2018). Adapting Software Design Patterns To Develop Reusable IEC 61499 Function Block Applications. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 725–732. https://doi.org/10.1109/INDIN.2018.8472071

Peres, R. A. F. da S. (2019). *An Industrial Data Analysis and Supervision Framework for Predictive Manufacturing Systems*. http://hdl.handle.net/10362/91108

Plattform Industrie 4.0. (2019). *Technology Scenario "Artificial Intelligence in Industrie 4.0."*

Priego, R. (2017). *A model-based approach for supporting flexible automation production systems and an agent-based implementaction*.

Rehberger, S. (2020). *Combining Product- and Resource-Related Reasoning for Agent-Based Production Automation*.

Reinpold, L. M., Wagner, L. P., Gehlhoff, F., Ramonat, M., Kilthau, M., Gill, M. S., Reif, J. T., Henkel, V., Scholz, L., & Fay, A. (2024). Systematic comparison of software agents and Digital Twins: differences, similarities, and synergies in industrial production. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-023-02278-y

Ribeiro, L. (2017). Cyber-physical production systems' design challenges. *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 1189–1194. https://doi.org/10.1109/ISIE.2017.8001414

Ribeiro, L., & Gomes, L. (2021). Describing Structure and Complex Interactions in Multi-Agent-Based Industrial Cyber-Physical Systems. *IEEE Access*, *9*, 153126–153141. https://doi.org/10.1109/ACCESS.2021.3127344

Ribeiro, L., & Hochwallner, M. (2018). On the Design Complexity of Cyberphysical Production Systems. *Complexity*, *2018*, 1–13. https://doi.org/10.1155/2018/4632195

Ribeiro, L., Karnouskos, S., Leitão, P., Barbosa, J., & Hochwallner, M. (2018). Performance Assessment Of The Integration Between Industrial Agents And Low-Level Automation Functions. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 121–126. https://doi.org/10.1109/INDIN.2018.8471927

Rocha, A. D. B. da S. P. (2018). *Increase the adoption of Agent-based Cyber-Physical Production Systems through the Design of Minimally Invasive Solutions*. http://hdl.handle.net/10362/58083

Roher, K., & Richardson, D. (2013). Sustainability requirement patterns. *2013 3rd International Workshop on Requirements Patterns (RePa)*, 8–11. https://doi.org/10.1109/RePa.2013.6602665

Russell, S., & Norvig, P. (2010). Artificial Intelligence A Modern Approach Third Edition. In *Pearson*. https://doi.org/10.1017/S0269888900007724

Ryashentseva, D. (2016). *Agents and SCT based self* control architecture for production systems* [PhD thesis]. PhD thesis, Faculty of Mechanical Engineering, Otto-von-Guericke University Magdeburg.

Ryashentseva, D., Cruz S., L. A., Vogel-Heuser, B., & Lüder, A. (2018). Development and evaluation of a unified agents- and supervisory control theory based manufacturing control system. *14th International Conference on Automation Science and Engineering (CASE)*, 187–192. https://doi.org/10.1109/COASE.2018.8560539

Sakurada, L., & Leitao, P. (2020). Multi-Agent Systems to Implement Industry 4.0 Components. *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, 21–26. https://doi.org/10.1109/ICPS48405.2020.9274745

Schulte, A., Donath, D., & Lange, D. S. (2016). *Design Patterns for Human-Cognitive Agent Teaming* (pp. 231–243). https://doi.org/10.1007/978-3-319-40030-3_24

Schütz, D. (2015). *Automatische Generierung von Softwareagenten für die industrielle Automatisierungstechnik der Steuerungsebene des Maschinen- und Anlagenbaus auf Basis der Systems Modeling Language*. Technische Universität München.

Schütz, D., Aicher, T., & Vogel-Heuser, B. (2017). Automatic generation of shop floor gateway configurations from systems modeling language. *2017 IEEE International Systems Engineering Symposium (ISSE)*, 1–8. https://doi.org/10.1109/SysEng.2017.8088288

Seitz, M., Gehlhoff, F., Cruz S., L. A., Fay, A., & Vogel-Heuser, B. (2021). Automation platform independent multi-agent system for robust networks of production resources in industry 4.0. *Journal of Intelligent Manufacturing*, *32*(7), 2023–2041. https://doi.org/10.1007/s10845-021-01759-2

Shakil, M., & Zoitl, A. (2020). OPC UA based IEC 61499 Device Configuration Interface. *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, 162–167. https://doi.org/10.1109/ICPS48405.2020.9274770

Sharma, C., Sinha, R., & Leitao, P. (2019). IASelect: Finding Best-fit Agent Practices in Industrial CPS Using Graph Databases. *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, 1558–1563. https://doi.org/10.1109/INDIN41052.2019.8972272

Sonnleithner, L., Wiesmayr, B., Ashiwal, V., & Zoitl, A. (2021). IEC 61499 Distributed Design Patterns. *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8. https://doi.org/10.1109/ETFA45728.2021.9613569

Sorouri, M., Patil, S., & Vyatkin, V. (2012). Distributed control patterns for intelligent mechatronic systems. *IEEE 10th International Conference on Industrial Informatics*, 259–264. https://doi.org/10.1109/INDIN.2012.6301149

Tang, H., Li, D., Wang, S., & Dong, Z. (2018). CASOA: An Architecture for Agent-Based Manufacturing System in the Context of Industry 4.0. *IEEE Access*, *6*, 12746–12754. https://doi.org/10.1109/ACCESS.2017.2758160

Telang, P. R., Singh, M. P., & Yorke-Smith, N. (2019). A Coupled Operational Semantics for Goals and Commitments. *Journal of Artificial Intelligence Research*, *65*, 31–85. https://doi.org/10.1613/jair.1.11494

Theiss, S. (2015). *Echtzeitfähige Softwareagenten zur Realisierung cyber-physischer Produktionssysteme*.

Theiss, S., & Kabitzsch, K. (2017). A Java software agent framework for hard real-time manufacturing control. *At - Automatisierungstechnik*, *65*(11), 749–765. https://doi.org/10.1515/auto-2017-0036

Trunzer, E. (2020). *Model-driven System Architectures for Data Collection in Automated Production Systems*.

Unland, R. (2015). Industrial Agents. In *Industrial Agents: Emerging Applications of Software Agents in Industry* (pp. 23–44). Elsevier. https://doi.org/10.1016/B978-0-12-800341-1.00002-4

Valckenaers, P. (2020). Perspective on holonic manufacturing systems: PROSA becomes ARTI. *Computers in Industry*, *120*, 103226. https://doi.org/10.1016/j.compind.2020.103226

Váncza, J., & Monostori, L. (2017). Cyber-physical Manufacturing in the Light of Professor Kanji Ueda's Legacy. *Procedia CIRP*. https://doi.org/10.1016/j.procir.2017.04.059

VDI/VDE. (2012). *2653 Sheet 1:2010 Multi-agent systems in industrial automation - Fundamentals*.

VDI/VDE. (2021). *2653 Sheet 4: Multi-agent systems in industrial automation - Selected patterns for field level control and energy systems*. 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

Villavicencio, C., Schiaffino, S., Andres Diaz-Pace, J., & Monteserin, A. (2019). Group recommender systems: A multi-agent solution. *Knowledge-Based Systems*, *164*, 436–458. https://doi.org/10.1016/j.knosys.2018.11.013

Vogel-Heuser, B., Lee, J., & Leitão, P. (2015). Agents enabling cyber-physical production systems. In *At-Automatisierungstechnik* (Vol. 63, Issue 10, pp. 777–789). https://doi.org/10.1515/auto-2014-1153

Vogel-Heuser, B., Neumann, E.-M., Fischer, J., Marcos, M., Estevez, E. E., Barbieri, G., Sonnleithner, L., & Rabiser, R. (2022). Automation Software Architecture in CPPS - Definition, Challenges and Research Potentials. *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 01–08. https://doi.org/10.1109/ICPS51978.2022.9816893

Vogel-Heuser, B., Ryashentseva, D., Salazar Cruz, L., Ocker, F., Hoffmann, M., Brehm, R., Bruce-Boye, C., Redder, M., & Lüder, A. (2018). Agentenmuster für flexible und rekonfigurierbare Industrie 4.0/CPS- Automatisierungsbzw. Energiesysteme. In VDI-Berichte (Ed.), *Automation 2018* (1st ed., pp. 1119–1130). VDI Verlag. https://doi.org/10.51202/9783181023303-1119

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

Vyatkin, V. (2016). *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, Third Edition* (V. Vyatkin, Ed.; Third Edit). International Society of Automation.

Wannagat, A. (2010). *Development and evaluation of agent-based automation systems in order to increase the flexibility and reliability of manufacturing plants* [Dissertation]. PhD thesis, Faculty of Mechanical Engineering, Technical University of Munich.

Weyns, D. (2012). *Design Patterns for Multi-Agent Systems: A Systematic Literature Review*. Https://Homepage.Lnu.Se/Staff/Daweaa/SLR-MASpatterns.Htm#overview.

Wright, T. (2001). *Fault Tolerance in the Server and Agent Based Network Management (SAAM) System*. http://hdl.handle.net/10945/1177

Zawisza, J. (2019). *Entwicklung und Integration interdependenter Agentensysteme zur dezentralen Produktionsplanung und -steuerung*. http://dx.doi.org/10.25673/13728

Zimmermann, P., Axmann, E., Brandenbourger, B., Dorofeev, K., Mankowski, A., & Zanini, P. (2019). Skill-based Engineering and Control on Field-Device-Level with OPC UA. *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1101–1108. https://doi.org/10.1109/ETFA.2019.8869473

Zoitl, A., & Prahofer, H. (2013). Guidelines and Patterns for Building Hierarchical Automation Solutions in the IEC 61499 Modeling Language. *IEEE Transactions on Industrial Informatics*, *9*(4), 2387–2396. https://doi.org/10.1109/TII.2012.2235449

7. Appendix A. Includes main contribution papers (Pub.I-V)

This part consists of five included papers.

[Pub.I] **Cruz S. LA**, Vogel-Heuser B (2017) Comparison of agent oriented software methodologies to apply in cyber physical production systems. In: 15th International Conference on Industrial Informatics (INDIN). IEEE, Emden, Germany, pp 65–71. https://doi.org/10.1109/INDIN.2017.8104748

[Pub.II] **Cruz S. LA**, Mayer F, Schütz D, Vogel-Heuser B (2018) Platform Independent Multi-Agent System for Robust Networks of Production Systems. IFAC-PapersOnLine 51:1261–1268. https://doi.org/10.1016/j.ifacol.2018.08.359

[Pub.III] **Cruz S. LA**, Ryashentseva D, Lüder A, Vogel-Heuser B (2019) Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. Int J Adv Manuf Technol 105:4005–4034. https://doi.org/10.1007/s00170-019-03800-4

[Pub.IV] **Cruz S. LA**, Vogel-Heuser B (2022) A CPPS-architecture and workflow for bringing agent-based technologies as a form of artificial intelligence into practice. at - Automatisierungstechnik 70:580–598. https://doi.org/10.1515/auto-2022-0008

[Pub.V] **Cruz S. LA**, Vogel-Heuser B (2022) Industrial Artificial Intelligence: A Predictive Agent Concept for Industry 4.0. In: 20th International Conference on Industrial Informatics (INDIN). pp 1–6. https://doi.org/10.1109/INDIN51773.2022.9976159

Publication I (ReqsForCPPS)

IEEE 15th International Conference on Industrial Informatics "INDIN" (2017), pp. 65-71.
https://doi.org/10.1109/INDIN.2017.8104748

Comparison of Agent Oriented Software Methodologies to Apply in Cyber Physical Production Systems

Luis A. Cruz S., B. Vogel-Heuser
Institute of Automation and Information System
Technische Universität München
Garching near Munich, Germany
{luis.cruz; vogel-heuser}@tum.de

***Abstract*— Cyber-Physical Systems (CPS) could be the most modern electronic development as yet, thanks to the integration of information and communication technology (ICT). CPS has associated with computer systems (cyber part) which are closely related to the real-world processes (physical part). A CPS is supported by the newest and foreseeable further advances of computer science, data and communication equipment on the one hand, and of manufacturing science and tools, on the other. On the contrary, within the multiple applications, there are CPS for manufacturing systems, called CPPS (Cyber-Physical Production Systems). The fourth industrial revolution regularly distinguished as I4.0 is based on CPPS. Considerable numbers of authors agree that paradigms agent-based as Multi-Agent Systems (MAS) converge or they make up some parts to apply CPPS. In general, this paper emphasizes that there are different important approaches in CPPS implementation which point near, in particular to MAS. The objective of this article is to provide general and specific concepts associated CPPS implementation through agents, considering the current multiples approaches and methods. A key result is that Agent-Oriented Software Engineering (AOSE) methodologies have been a highlight to comparisons leading benefits to apply in CPPS.**

Keywords*— *Agent Oriented Software Engineering (AOSE), Cyber-Physical Production Systems (CPPS), Multiagent Systems, Industrial Automation, Industry 4.0.

I. INTRODUCTION

The flexibility and re-configurability along with the robustness and operational efficiency of manufacturing systems can be substantially improved by applying new and advanced information and communications technology (ICT). Recent trends in IT are regularly associated with of the Internet of Things (IoT). High impact academic institutions, industries, and governments all over the world develop plans and strategies for their long-term evolution around those terms [1].

Under this agreement, Industry, 4.0 (I4.0) was initialized in Germany and extended to Europe, aiming at the expansion of the fourth industrial revolution. Leading prospect of I4.0 and other related concepts are mass customization and flexibility of production systems, based on Cyber-Physical Systems (CPS).

Substantial changes are required in the manufacturing systems for I4.0, which should include the new technologies such as reconfigurable machines and more intelligent robots which are principally different in appearance and functionalities; nevertheless, they follow a similar pattern among them in their communication and interaction behavior.

Based on [2], the evolution of systems could include the industrial type of IoT or IIoT/I²oT, as shown by from Fig. 1. First, the demand to develop specific tasks and real-time computing creates the *Embedded Systems*. Second, systems, products, services are generated and evolved through the application of interconnected networks and systems, or communication from Machine to Machine (M2M).

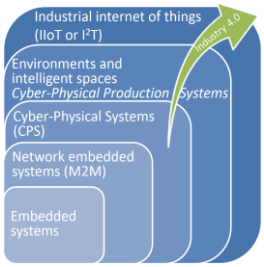

Fig. 1. Evolution from Embedded Systems to the IIoT [2].

Different approaches can be modified or adapted to develop CPS, in particular for the consideration of industrial automation systems for manufacturing or CPPS (Cyber-Physical Production Systems). The main contribution of this paper is to classify existing agent-based approaches as a technological basis to realize CPPS and to identify benefits of Agent Oriented Software Engineering (AOSE) as a useful strategy for CPPS development.

Therefore, first, this paper shows related applications of currently CPPS projects and their approaches (Section II). Second, the next section explains ten requirements relevant for the apply Multi-Agent Systems (MAS) for CPPS (Section III). Compendium and comparison of primary MAS methodologies are given in the last section (Section IV). The paper is concluded with a summary and outlook of future work.

II. IMPACT OF CYBER-PHYSICAL PRODUCTION SYSTEM IN INDUSTRIAL AUTOMATION

Nowadays, a significant amount of manufacturing information is already generated. There is a great need for a new generation of systems to design and realize more than networking, ICT, and knowledge being integrated into physical things [3]. Hence, with the recent advent of Information Technology (IT), it is possible to apply computers to store data, analyze statistics, recover records, transmit signals, and manipulate general information for the production environment. IT evolution generated the Cyber-Physical Systems terminology.

CPS concept consists of a virtual part (software) and a real part (hardware). CPS have a trend towards more flexible cooperative distributed systems, and they will introduce new communication concepts [4]. The primary benefit of the CPS in automation is smooth integration of software components at runtime so that its operability, especially productivity for manufacturing is not hindered as much as possible.

Thanks for primary integration benefit, there are a lot of applications for CPS such as autonomous cars, robotic surgery, smart buildings, smart grid, medical implant devices and smart manufacturing and they are just some of the examples since they are in permanent extension [2].

CPS for manufacturing systems has also generated Cyber-Physical Production Systems concept. CPPS are intended to develop the necessary contributions to obtain the *Smart Factory* [1]. CPPS could be done more practical thanks to the advance of newly available devices for the future of automation through IIoT [5]. A reported landscape of works with CPPS and their proposals are enlisted through Table I.

TABLE I. SELECTED CPPS PROJECTS PROPOSALS

Project	Proposal
myJoghurt [1], [6]	Implementing examples for CPPS through prototypical MAS-based evaluating and using the production scenario from different German chairs
IDAPS [7]	Perceiving as a microgrid is intelligent, having a built-in multi-agent functionality in the context of Intelligent Distributed Autonomous Power Systems
ENIAC JU E2SG [7]	Developing methods for detecting and controlling energy flows in the grid with information transmitted over the grid itself
Socrades [7], [8]	Exploring application of service orientation and web services using formalisms for modeling, analysis, and execution for next generation of industrial automation
GRID4EU & SGAM [7]	Testing innovative concepts and technologies in real-size environments, to highlight and help remove barriers to the deployment of Smart Grids in Europe
IMC-AESOP [7], [8]	Proposing a Service-Oriented Architecture (SOA) for very large-scale distributed systems in batch and process control applications
Grace [9]	Developing a modular, intelligent, and distributed control system that integrates process and quality control using the MAS principles
IDEAS [7], [10]	Enabling fast deployment of mechatronic modules based on eEAS paradigm, advocating the use of process-oriented and associating interacting agents
Pabadis Promise [7]	Distributing manufacturing execution systems and bringing flexibility features to the control systems using software agents and plug-and participate technology
iSiKon [1]	Increasing ding flexibility in heterogeneous material flow systems based on intelligent software in self-configuring modules
SmartFactory^{KL} [11]	Working on new concepts, standards, and solutions to form the basis for highly flexible automation technology and manufacturer-independent *Industrie 4.0* plant
HySociaTea [12]	Establishing the basis for production environment of the future with a team of humans closely collaborating with robots and virtual agents
It's OWL [13]	Focusing is in the fields of self-optimization, human-machine interaction, intelligent networking, energy efficiency and systems engineering
uPlant [14]	Testing facility for methods from the areas of monitoring, modeling, control and optimization of modern and future automation technologies
PhyNetLab [15]	Developing of ultra-low power WSN for decentralized control of materials handling facility

Selected projects from Table I are several large-scale R&D initiatives, which were conducted over the years to research the use of CPS in industrial applications. These projects have demonstrated crucial challenges, such as safety, security, and interoperability and have become a reality especially manufacturing systems [7]. Furthermore, these projects have also been covered by industrial partners as well as academic experts. Therefore, the parallel development of academic and industrial approach has been possible.

There are many different definitions for CPPS. One in [4] says, "Cyber-Physical Production Systems (CPPS) are Cyber-Physical Systems as applied in the domain of manufacturing/production, in Germany the term *Industrie 4.0* is used". Other authors in [16] mentioned that "Cyber-physical systems (CPS) are systems of collaborating computational entities which are in intensive connection with the surrounding physical world and its on-going processes, providing and using, at the same time, data-accessing and data-processing services available on the Internet."

A preview summary could be that CPPS as the previous generation for Intelligent Manufacturing Systems (IMS) and future automation [2], [5]. Thus recent works show that the most significant contributions of CPPS are the following [1], [6], [7], [16]–[18]:

- Vertical and horizontal integration through value and smart networks.
- Manufacturing devices are intelligent to acquiring information from their environs and act autonomously (smartness).
- Cooperation and collaboration will be some inherited skills to use connections to the other system actors (including *human beings).*
- Reaction properties towards internal and external changes or failures (robustness).
- Optimal decision making for energy and resource efficiency.

Along last decade, in the selected projects from Table I, classified in Table II, they have been developed *CPPS demonstrators*, *Smart manufacturing* approaches, *Electric Grid applications* or *Architectures* installed in industrial environments. The Table II also categorizes these selected CPPS projects agreeing to ISA 95 standard levels. As surveyed in [17], the ISA 95 levels could be classified according to *Device Level* (L1); *Supervisory Control And Data Acquisition* or SCADA Level (L2); *Manufacturing Operations Management* or MOM Level (L4); and the *Enterprise* or ERP Level (L4).

TABLE II. CATEGORIZATION OF SELECTED CPPS PROJECTS

	CPPS type				ISA 95 Level			
Project	CPPS demonstrator	Smart Manufacturing	Electric Grid Application	Architecture development	Device	SCADA	MOM	Enterprise
myJoghurt	X	X	-	-	X	X	X	-
IDAPS	-	-	X	X	X	X	-	-
ENIAC JU E2SG	-	-	X		X	X	-	-
Socrades	X	X	-	-	X	X	X	X
GRID4EU & SGAM	X	-	X	X	X	X	X	X
IMC-AESOP	-	X	-	X	X	X	X	X
Grace	-	X	-	-	X	X	-	-
IDEAS	-	X	-	X	X	X	-	-
Pabadis'promise	-	X	-	-	X	X	X	-
iSiKon	-	X	-	-	X	X	-	-
SmartFactoryKL	X	X	-	X	X	X	X	X
HySociaTea	X	X	-	-	X	X	X	-
It's OWL	X	X	-	-	X	X	X	X
uPlant	X	X	-	-	X	X	X	-
PhyNetLab	X	X	-	-	X	X	X	-

III. APPROACHES REVIEW FOR CPPS

After the first classification of existing CPPS projects, in the previous section, manufacturing concepts, CPPS approaches, and its general characteristics are discussed in this chapter.

A. Concepts of Architecture, Methodology, and Standard in Manufacturing Approaches

For this paper, an *approach* is defined as a set of *architectures*, *methodologies* and/or *standards* that follow a common scheme. In the case of *architectures*, they are considered only as structures of static modeling systems. Most of these are frameworks patented by their authors and often do not have the procedural information to carry out their implementation (methodology). A *methodology* determines a series of steps to be taken to improve productivity in development and quality systems (generally for engineering software). It also indicates how it will perform the process in a systematic, predictable and repeatable way. Both an architecture and the methodology can be endorsed by international institutions which generate *standards*. Depending on the nature of the organization, a standard in manufacturing may be a private or open type.

In the ideal case, architecture can be promoted by its methodology to carry out its implementation, and they both can be standardized. However, in the reviewed authors' academic literature (e.g. [7], [10], [17]), architectures, methodologies, and standards are mutually exclusive characteristics. In fact, not all architectures, nor methodologies in manufacturing systems are supported by international standards.

In summary, a manufacturing approach is a collection of architectures, methodologies, or standards that are part of a similar paradigm but do not maintain the same structural, dynamic, and procedural properties.

B. Manufacturing Approaches for CPPS

This section will give an overview of traditional manufacturing approaches and their particularly comprised points. After that, agents based schemes will be introduced with their characteristics.

1) Traditional Hierarchical Approach

Most of the traditional manufacturing systems belong to this classification. These are implemented using centralized and staggered control techniques, and present good responses regarding outputs due to their optimization capability. Such methods typically follow a rigid multilevel structure, which prevents them from reacting agilely to possible variations.

Hierarchical architectures are similar to that pyramid Computer Integrated Manufacturing (CIM). In this, the different levels cannot take the initiative; therefore, the system is vulnerable to disturbances and autonomy, and its reactions to disturbances are weak. This rigidity increases the costs of its development produces a system problematic to maintain [5].

A significant example is a norm applied to batch processes, called ISA-88, which corresponds to hierarchical schemes due to its centralized nature. This standard does not present a solution to the automation system, but it refers to an ordered method for thinking, working and communicating. It has a hierarchy characteristic between control levels of devices and equipment. It also contains models and terminology that allow analyzing the organization.

2) Heterarchical Approach based on Multiagent System

Heterarchical manufacturing techniques introduce a proper response to the requirements of flexibility and agility. These designs provide an excellent performance against changes and can be continuously adapted to their environment. Systems are fragmentations of small and completely autonomous units.

The independent components, called agents, are the main part of this architecture, and they obtain cooperation skill through negotiation protocols structures. Multi-agent system (MAS) approach prohibits all types of hierarchy to give all the power to the necessary modules. By eliminating hierarchical relationships in the system, the modules cooperate as equals, generating a flat architecture rather than assigning subordination and supervisory relationships.

Gaia is a particular example tailored to the analysis and design of MAS [8]. Gaia is a general methodology that supports both levels of the individual agent structure and the agent society in the MAS development process [9]. In this methodology, MAS looks like a system constituted of a conglomeration of autonomous interactive agents that exist in an organized society in which each agent plays one or more specific roles. Gaia structures MAS regarding a role model, based on the roles that agents have to play within the heterarchy and the interacting protocols between such different characters. Functions include the following attributes: responsibilities, permissions, activities, and protocols.

3) Hybrid Approach Based on Hybrid Systems

Another main approach similar to MAS is a holonic (or holon based) manufacturing system (HMS) by P. Leitão, H. Van Brussel, and P. Valckenaers [8], which consists of autonomous, intelligent, flexible, distributed, co-operative holons. Multiagent systems (MAS) and holonic MAS (HMAS) may comprise complex systems. Getting started with such hybrid architectures can be challenging to implement CPS.

The design enables the product cases to drive their production; consequently, coordination through holons can be completely decentralized. In contrast to many decentralized setups, the manufacturing based on holarchies (levels of holons) predicts future behaviors and proactively uses actions to prevent impending difficulties from occurring.

Hence, one of the most hopeful features of HMS is that they symbolize a transition between fully hierarchical and heterarchical systems. Review literature of HMS indicates that the ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) is one of the most remarkable for HMSs. ADACOR architecture identifies four types of basic holons: *Product holon* (PH), *Task holon* (TH), *Operational holon* (OH), and *Supervisor holon* (SH) [8]. It is a holonic design that offers an adaptive manufacturing control approach scales from a stationary state to a transient state, in typical and unexpected conditions, respectively, combining the benefits of hierarchical and heterarchical control structures applying some adaptive elements.

Finally, a classification of the approaches is provided, to identify levels of ISA 95 (y-axis Fig. 2) and on the Z-axis real time requirements for important works in CPPS. Also, Fig. 2 shows the classification into hierarchical, heterarchical and hybrid on the x-axis and arranges a selected *Architectures*, *Methodologies,* and *Standards,* which could be applied in the development of CPPS according to the proposed classification.

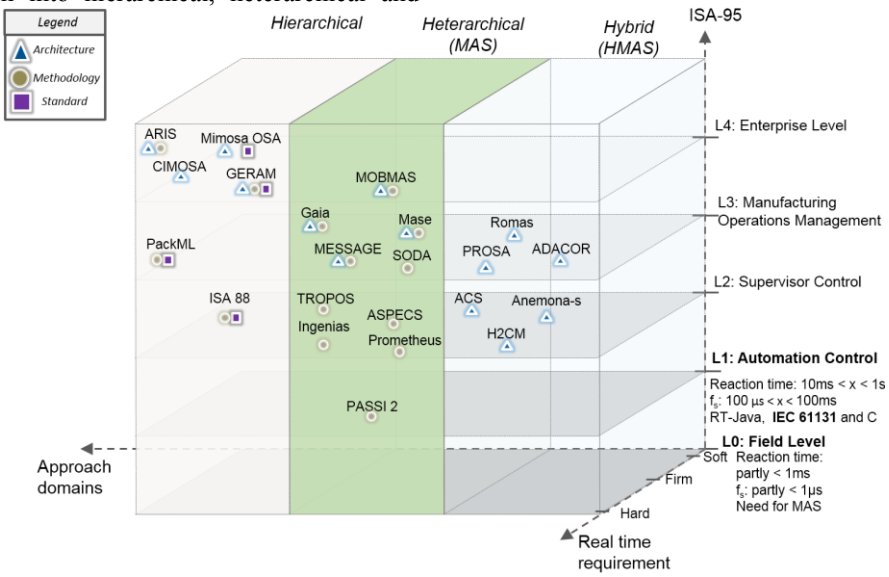

Fig.2. Classification of agent-oriented architectures, methodologies, and standards for CPPS approaches

IV. REQUIREMENTS FOR CPPS

Main requirements need to be satisfied before obtaining the vision of integration and convergence of the IIoT, and in this way, all its benefits can be achieved [4], [6]. In particular, it is important to consider that even in highly developed countries there are asymmetries on the degree of digitalization of manufacturing, and even within the same organization, there are areas which have been highlighted on automation from others [7]. To overcome these challenges and to homogenize a comparison, in this section, requirements for future manufacturing architectures and CPPS necessities are derived.

A. CPPS Minimal Conditions (Requirement 1)

Some studies have been conducted to investigate the conditions to discover the necessary technical characteristics for CPPS, realizing a CPS architecture that could couple various industrial production facilities [6]. Consequently, several fundamental properties of a CPPS were recognized that could be summarized into four main groups of following elements:

1) Independence architecture model (R1.1)

Modules could be simple to integrate with open architecture and platform independent implementation. From the numerous varieties of automation system items that need to be able to play in CPPS, network requirements concerning the computing devices for which a necessary examination arises. For example, Programmable Logic Controllers (PLC) are frequently used in industry. Thus, these devices need to be considered as one platform type to integrate into the agent-based CPPS. Nevertheless, independence of architecture means that CPPS is not limited to this class of devices. However, expanding the different platform should always be possible to maintain data transfer for more multiple kinds of applications.

2) Open communication protocol for IIoT (R1.2)

This requirement is related to standardization and is pointed out by the industry as a major distinctive for the manufacturing acceptance of any technology (open architecture). There could be easy and quick abilities to switch between open protocols for IIoT (e.g. OPC-UA). Due to the importance of the communication in this ubiquitous networking era, many kinds of components, layers, and protocols are required to have OPC-UA standard features in a manufacturing control systems.

3) Levels of automation are enabled from ISA 95(R1.3)

All levels of automation are allowed (ISA 95) depending on the scenario in which the CPPS is applied. Various parts of a manufacturing system may have to be connected to the network. It means, in a simple automation system, only overall production equipment, and their respective information element systems need to be linked to the CPPS network. Then, to be applicable in several different scenarios, a CPPS should not be limited to a particular hierarchy level of a manufacturing system. For this, the connection of random system components should be feasible independently of their locations in a plant hierarchy or global context of ISA level.

4) Easy to adapt the system to future products (R1.4)

It will be necessary to have easy adaptation the system to future products (Smart products). According to that information and the knowledge regarding their techniques, the

production facilities can reason about circumstances for executing the processes and, based on results, return the product with either an offer or rejection.

Behind a pure syntactical correspondence of the received operation report and the models that describe production capabilities, the application of semantic technologies enables a semantic checking and hence enhanced possibilities of uniting the different entities of a CPPS. The communication necessary for switching the request, offers or refusals, is realized by information that is either published by global data inside of the CPPS or sent directly to the other entities of the system in a Machine to Machine (M2M) communication manner.

B. Intelligent Characteristics Attributes (Requirement 2)

1) Autonomy (R2.1)

Autonomy could be achieved by deducing behaviors of the CPPS on agents from its experience, and processes. Agent-based approaches support the success, called Plug and Work production systems, where various elements are joined to a complete production system without hand-operated configuration efforts. The primary goal of these developments is the creation of a basically soft agent platform that presents guidelines and facilitates a fast, platform-neutral implementation of the agent technology.

2) Communication and ontology (R2.2)

Communication is necessary to speak same languages and common agent ontology. In general cases, agents may communicate to achieve goals or due to selected event. Considerations of inter-agent communication include which protocol to use, how to define a domain –in terms an agent from another field can understand– and how competent could be the communication technique.

3) Cooperation (R2.3)

Cooperation is crucial to enable developing mutually acceptable goals. Cooperative skills for CPPS offer necessary elements and subsystems to connect an intelligent network. CPPS networks will be based on the context within and across all levels of production, from processes through machines up to ERP systems.

Manufacturing control systems require autonomous entities to be classified in hierarchical and heterarchical structures for cooperation. Cooperation requirement is related to the kind of behavior that the control unit at factory level should exhibit. Manufacturing control based agent are regularly handling a high number of duplicated events, which are known but random. This flow of events should be processed in an efficient manner with temporal constraints and agent collaboration.

The administration of the agent events can consequently be determined apriori by routines, while the beginning and execution of these routines should be performed in a real-time collaboration technique. The size of the event set and their activity patterns increase over time.

4) Pro-activity (R2.4)

The agent is capable of achieving his assigned goal. It means that MAS must have skills to take the initiative not solely motivated by events, also adapt itself generating "rational" actions to succeed goals. This suggests some degree of Pro-activeness (e.g. it tracks its' own agenda). For CPPS researchers, this is a defining attribute of an agent.

C. Formalized Modeling Terms (Requirement 3)

Innovative approaches to abstractions (formalisms) and architectures are necessary to enable control, communication and computing integration. CPPS implementation implies the rapid design and to be developed. They should admit the combination and interoperability of heterogeneous systems that formed the CPSs in a modular, practical and hardy manner.

1) Using standard language (R3.1)

The models for CPPS require internationals standards which are the base for the expansion of standard lines between SCADA, MES and ERP levels systems. Formalism such as the Unified Modeling Language (UML) helps to structure and comprehend information from manufacturing architectures through understandable models.

Other modeling languages have been proposed to model a CPS in [18], called Systems Modeling Language (SysML). SysML has been established in automation systems based on UML to support Model-Based Systems Engineering (MBSE). A related semantic is Automation Markup Language (AutomationML), and it is one of the imminent upcoming open standard series (IEC 62714) for the description of production plants and their components. For *Plug and Work* concept, AutomationML defines the contents, which is exchanged between the parties and systems complex.

It requirement helps to model plants and plant components with their skills, topology, interfaces, and relations to others, geometry, dynamics and even logic and behavior.

2) Level of abstraction for overview (R3.2)

There could be a different degree of abstraction for applying the model in CPPS. The conventional approaches and methods for manufacturing system modeling, such as CIM, are mainly based on a top-down scheme. The user's requirements and the general conceptual design constitute the whole set of modeling limitations. With these approaches, very rigid hierarchical architectures are built.

Other non-traditional designs were differentiated as being bottom-up structure. Nonetheless, in line with the order of the complexity of the distributed system made up by a network of smart entities, IMS modeling requires several development methods. It includes bottom-up and top-down integration, depending on the level being formed. It is not mandatory to define the whole set of constraints at the origin. A mixed construction process allows the generation of reconfigurable and scalable structures.

3) IDE coverage and complexity (R3.3)

The model must provide details to facilitate the implementation in Integrated Development Environment (IDE) and platforms. In traditional automation systems, there are only a few languages including the languages defined in IEC 61131-3. These languages were developed for IDE with a focus on automation systems. Depending on a proper runtime, tools could be often programmed in C or assembly language. With the increase of mobile devices, such as smartphones or tablet PCs playing an important part in CPPS, the range of different languages and platforms gets even wider. Every platform uses its runtime, and again even the various programming languages.

For example, instance applications for Android applications use a Java framework; hence, they have to be written in Java language. Another example is Firefox OS, which is in progress and it uses JavaScript language in combination with Hyper Text Markup Language (HTML) for mobile applications. could be required to find devices with this IDE to apply CPPS there soon.

D. Systems and Human Integration Needs (Requirement 4)

1) Open systems to different systems domain (R4.1)

It is open to different kind of systems area (e.g. energy, manufacturing, or process). Due to the broad diversity of industrial process systems, the development custom or tailored solutions has to be reduced. On the contrary, an architecture for agent-based CPPS should apply to a variety of situations, i.e. different kind of products and processes. For this reason, the MAS architecture, protocols, and messages for CPPS should be independent of a particular application.

2) Hybrid topologies (R4.2)

It is necessary to include hybrid topologies to enlarge and downsize the production system because many different architectures can be present in a CPS. It means that various manufacturers may integrate several designs. A specialized engineering or development tool is established for every element in a CPS. The developers are used to their respective devices and have their skill in its approach. It should be possible to continue developing in the similar languages. Because of this, devices with different runtime systems have to be mixed into a CPS.

3) Social norms considering human factors (R4.3)

CPPS must provide social norms to execute MAS considering human factors. Also, if all data and information available concerning a CPPS and its products, production facilities, and architecture is modeled and semantically represented, the preparation of this knowledge for human personnel or customers remains an essential issue. It must include concepts that support the engineering of CPPS and their system entities (e.g. intelligent products and production facilities) as well as mechanisms to preprocess the relevant process data during production for human operators, support personnel and even for the customers of the produced goods. This information will provide opportunities for individual arrangements (e.g. age or user level distinguished visualization and interaction mechanisms) as well as integrations with regularly used especially mobile devices and humans [6].

V. AOSE STRENGTHS TO IMPLEMENT CPPS

MAS or Agent-based approaches signify a natural method of realizing CPPS [6]–[8], [17]. The important concept of MAS is Agent Oriented Software Engineering, and there are several AOSE methodologies, which are at least ten years old [5]. Indeed, the decision of an AOSE methodology depends on the MAS demands, in this case, CPPS requirements expressed in section III.

Considering requirements from part III, this article will now examine the different methodologies of reported in the dedicated literature of AOSE. The goal of this revision is to determine to what extend these procedures into account the requirements for implementing CPPS. Firstly, this section presents a brief summary of the various methods (more details can be found in [8], [9]). Finally, this chapter will make a comparison discussion based on the requirements it has cited in Chapter III.

A. Main Agent Oriented Software Engineering (AOSE)

Previously selected ones of AOSE for the CPS event-driven multi-agent model, a comparison should be performed based on the following evaluation criteria, grouped into four main categories: CPPS Minimal Conditions, Intelligent Characteristics Attributes, Systems and Human Integration Needs. Table III presents the comparison between AOSE approaches that could be used in a real CPPS implementation.

Table III. MAIN AOSE STRENGTHS ORDERING

	R1- CPPS minimal attributes				R2- Intelligent characteristics				R3- Formalized modeling			R4- Systems and Human Integration		
Main AOSE	R1.1	R1.2	R1.3	R1.4	R2.4	R2.5	R2.6	R2.7	R3.1	R3.2	R3.3	R4.1	R4.2	R4.3
Gaia	+	+	~	+	~	+	~	+	+	~	~	-	+	-
MaSE	+	+	+	+	+	+	+	-	-	-	~	+	-	-
MESSAGE	+	~	-	+	+	+	+	+	+	~	+	+	+	~
TROPOS	+	-	-	+	~	~	~	-	+	-	~	+	+	-
Prometheus	+	~	+	~	~	-	+	-	+	-	~	+	-	-
INGENIAS	+	+	~	+	+	+	~	+	+	~	+	+	+	~
SODA	+	-	-	~	+	+	+	-	+	-	-	+	-	-
PASSI2	+	~	-	~	+	+	~	-	-	~	~	~	-	-
ASPECS	+	-	-	+	~	+	~	-	+	-	+	+	-	+
MOBMAS	+	-	-	~	+	~	+	-	+	-	-	+	-	-

Notation: + High; ~ Medium; - Low

B. Discussion

The analysis of Table III allows extracting some important conclusions related to the adoption of AOSE to develop CPPS. In the *CPPS Minimal Attributes* (R1), the area of interest covered by the Independence architecture model requirement (R1.1) is entirely covered. However, on the same item, there is a little coverage of the *ISA 95 Levels* for vertical integration automation (R1.3). It means that AOSE methodologies must help to increment integration of separate system components regardless of their location in a plant hierarchy (or global context of ISA level).

The second general distinctive is that *Intelligent Characteristics* requirement (R2) has coverage satisfactorily. However, *Pro-activity* requirement (R2.4) is not available for many AOSE yet. It is necessary to consider improving AOSE technologies to have skills with more initiatives achieving their assigned goal.

In the same line, there is good coverage in the *Formalized Modeling Terms* requirement (R3) for AOSE methodologies. Conversely, there is low *Level of Abstraction for an Overview* element (R3.2); then, AOSE requires a different development formal method, bottom-up and top-down depending on the degree being formed.

Another observation is that *Systems and Human Integration Needs* (R4) are the least covered because main parts of this specification are not included yet. In fact, both *Hybrid Topologies* (R4.2) and the *Social Norms Considering Human Factors* needs (R4.3) are weak in the selected AOSE methodologies. That can be given by the complexity of human behavior and its corresponding integration into the manufacturing system in a predictable way.

At last, an essential issue when reviewing the selected AOSE methodologies for CPPS is that the majority group show at least 50% coverage of the requirements (R1-R4), on average. Nevertheless, there are some requirements (R1.3, R3.2, and R4.3) that need urgent attention since they are not considered by most of the AOSE methodologies.

In summary, it is important to note from Table III that we can conclude the adoption of AOSE to apply CPPS combined with other approaches with different architectures, methodologies, and international standards could improve all the coverage of requirements from section III.

VI. Conclusion and Roadmap

The comparison reported in this paper analyzes the combined strength of approaches for implementing CPS in manufacturing. As contained in the context, a CPPS could be considered as a system of multiple agents with a precise technique called MAS or hybrid HMAS. CPPS would have better flexibility, adaptability, and proactivity due to agent-based negotiation and holarchies. In the MAS approach, essential issues to be applied on CPPS are AOSE methodologies. Future work could address the main AOSE benefits and problems for CPPS and extend this to evaluate results through metrics. Metrics are crucial to obtain benefits with clearness and to compare offers from different CPPS providers. For example, flexibility is one of the upper goals of CPPS, and it will require metrics to estimate reliable results.

References

[1] T. U. M. IAS, “Institute of Automation and Information Systems,” 2014. [Online]. Available: http://i40d.ais.mw.tum.de/. [Accessed: 24-Mar-2017].

[2] K. J. Saumeth C., F. Pinilla T., A. Fernández A., D. J. Muñoz A., and L. A. Cruz S., “Sistema Ciber-Físico de una CNC para la producción de circuitos impresos,” in *IV Congreso Internacional de Ingeniería Mecatrónica y Automatización - CIIMA 2015*, 2015, vol. 1a.edición, pp. 154–155.

[3] T. Sanislav and L. Miclea, “An agent-oriented approach for cyber-physical system with dependability features,” *2012 IEEE Int. Conf. Autom. Qual. Testing, Robot. AQTR 2012 - Proc.*, pp. 356–361, 2012.

[4] M. Riedl, H. Zipper, M. Meier, and C. Diedrich, *Automation meets CPS*, vol. 46, no. 7. IFAC, 2013.

[5] L. A. Cruz Salazar and O. A. Rojas Alvarado, “The future of industrial automation and IEC 614993 standard,” in *2014 3rd International Congress of Engineering Mechatronics and Automation, CIIMA 2014 - Conference Proceedings*, 2014, pp. 1–5.

[6] B. Vogel-Heuser, C. Diedrich, D. Pantförder, and P. Göhner, “Coupling heterogeneous production systems by a multi-agent based cyber-physical production system,” *Proc. - 2014 12th IEEE Int. Conf. Ind. Informatics, INDIN 2014*, pp. 713–719, 2014.

[7] P. Leitão, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, “Smart Agents in Industrial Cyber-Physical Systems,” *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, 2016.

[8] J. Debenham and A. Prodan, *Industrial Applications of Holonic and Multi-Agent Systems*, vol. 8062, no. August. 2013.

[9] K. Kravari and N. Bassiliades, “A survey of agent platforms,” *Jasss*, vol. 18, no. 1, pp. 1–17, 2015.

[10] L. Ribeiro, J. Barata, M. Onori, and J. Hoos, “Industrial Agents for the Fast Deployment of Evolvable Assembly Systems,” *Ind. Agents Emerg. Appl. Softw. Agents Ind.*, pp. 301–322, 2015.

[11] D. Gorecky, S. Weyer, A. Hennecke, and D. Zühlke, “Design and Instantiation of a Modular System Architecture for Smart Factories,” *IFAC-PapersOnLine*, vol. 49, no. 31, pp. 79–84, 2016.

[12] T. Schwartz *et al.*, “Hybrid teams of humans, robots, and virtual agents in a production setting,” *Proc. - 12th Int. Conf. Intell. Environ. IE 2016*, pp. 234–237, 2016.

[13] R. Dumitrescu, C. Jürgenhake, and J. Gausemeier, “Intelligent Technical Systems,” pp. 24–27, 2012.

[14] D.- Karlsruhe, “Department of Measurement and Control Universit ¨ at Karlsruhe,” *Measurement And Control*, 2009. [Online]. Available: http://www.uni-kassel.de/maschinenbau/institute/isac/mrt.html. [Accessed: 15-Mar-2017].

[15] A. K. R. Venkatapathy, M. Roidl, A. Riesner, J. Emmerich, and M. ten Hompel, “PhyNetLab: Architecture design of ultra-low power Wireless Sensor Network testbed,” *IEEE 16th Int. Symp. A World Wireless, Mob. Multimed. Networks*, pp. 1–6, 2015.

[16] L. Monostori *et al.*, “Cyber-physical systems in manufacturing,” *CIRP Ann. - Manuf. Technol.*, vol. 65, no. 2, pp. 621–641, 2016.

[17] Y. Lu, K. Morris, and S. Frechette, *Current Standards Landscape for Smart Manufacturing Systems*. 2016.

[18] P. Hehenberger, B. Vogel-Heuser, D. Bradley, B. Eynard, T. Tomiyama, and S. Achiche, “Design, modelling, simulation and integration of cyber physical systems: Methods and applications,” *Comput. Ind.*, vol. 82, pp. 273–289, 2016.

Publication II (MASplatform)

IFAC-PapersOnLine 51/11 (2018), pp. 1261-1268.

https://doi.org/10.1016/j.ifacol.2018.08.359

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-11 (2018) 1261–1268

Platform Independent Multi-Agent System for Robust Networks of Production Systems

Luis Alberto Cruz Salazar*, Felix Mayer, Daniel Schütz**, Birgit Vogel-Heuser***

**Automation and Information Systems (AIS), Technical University of Munich (TUM) (e-mail: luis.cruz@tum.de)*

***GEFASOFT GmbH, Munich, Germany.*

Abstract: The production of customized products requires a flexible production process. Cyber-Physical Production Systems (CPPS) are currently seen as one possibility to achieve this goal, since flexibility for the production processes can be achieved by enabling cross-company collaboration. However, to fully implement CPPS for heterogeneous production systems an application and hardware platform independent, robust, and distributed software solution is required. This paper proposes a novel design and implementation for a Multi-Agent System (MAS), that can be used to create application independent CPPS. To support a variety of hardware platforms, the MAS was designed as a lightweight implementation in the programming language ANSI C. The proposed concept was evaluated by different use cases and experiments, which, in a first step, evaluated the efficiency of design and implementation.

Keywords: Automation, Cyber-Physical Systems (CPS), Multi-Agent Systems, Production control.

1. INTRODUCTION

Individually made products help companies to set themselves apart from competitors and satisfy customers' need for individuality. Consequently, small lot sizes and customization are current trends in production. Both trends demand high flexibility and adaptability of production systems. The coupling of different locally distributed production systems enables, e.g., scenarios for collaborative manufacturing of customized products and exploration of new possible processes, which emerge from the collaboration. The connection of multiple plants' sensors can, e.g., provide opportunities for a cross production system diagnosis, i.e., the identification under which conditions a certain type of sensor is most likely to exhibit failures. One approach to dynamically connect different systems in order to realize different use cases is the utilization of Service-Oriented Architectures (SOA)(Jammes and Smit, 2005). In SOA, every function of a given system is exposed by the system itself as a remotely callable service. The application of software agents and Multi-Agent Systems (MAS) (Leitão, Marik and Vrba, 2012) is another solution for a network of dynamically connected remote systems. MAS can increase flexibility and fault tolerance while retaining simplicity and a higher degree of autonomy for the participating entities.

This paper proposes a generic MAS platform, whose agents can be deployed on a great variety of different computational devices, in order to fulfil different use cases inside a network of locally distributed production facilities. The architecture of this platform is derived from the already existing standard of the Foundation for Physical Agents (FIPA) for MAS. In contrast to other platforms like the Java Agent DEvelopment Environment (JADE) (Bellifemine, Poggi and Rimassa, 2001), the proposed platform is intended to be lightweight and able to connect different heterogeneous production systems as well as their components, e.g., small sensors, in order to realize a network of Cyber-Physical Production System (CPPS) for arbitrary use cases. CPPS constitute a specialization of the concept of Cyber-Physical Systems (CPS), which are often defined as "*integrations of computation with physical processes*" (Lee, 2008). Inside a CPPS network, cyber representations (C), e.g., agents, of controlled physical entities or systems (P), e.g., a plants or sensors, in a production environment (P) connect to other related entities to form a bigger system (S) and realize different industrial related use cases.

The distributed intelligence and decision finding inside MAS renders the management of uncertainties and dynamics inside a CPPS network, as the complexity of a central node would be exceedingly high otherwise. Another challenge is to support the implementation on devices with merely limited computing resources. This requires a suitable and flexible architecture for software agents as well as resource-friendly communication protocols and messages. The approach recognizes five requirements for the realization of a CPPS (sec. 2). Based on these requirements, the paper proposes an approach separated into a concept for a logical architecture, a concept for the software architecture, and a concept for protocols and messages (sec. 4). These three concepts already partially satisfy a subset of the imposed requirements by design. The fulfilment of the other requirements is evaluated using an operating agent-based CPPS network (sec. 5). The paper concludes with a summary and an outlook.

2. REQUIREMENTS OF AN AGENT-BASED CPPS

This section derives the requirements for an agent-based network of locally distributed CPPS before related work is analysed regarding these requirements in sec. 3.

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.08.359

(R1) Application independence: Due to the broad variety of industrial production process classes, the development of multiple custom-tailored solutions has to be considered not feasible. To the contrary, an architecture for agent-based CPPS networks should be applicable for a variety of scenarios, i.e., different production processes. Consequently, the basic MAS architecture, protocols and messages, should be independent of a specific application.

(R2) Level independence: Depending on the scenario, in which the CPPS network is applied, different parts of an automated production system may have to be connected to the network. For example, in a simple production scenario only overall production facilities and their respective automation / IT systems need to be connected to the CPPS network. However, for a diagnosis scenario, that considers multiple plants inside a CPPS network, also field-level devices may be relevant. Consequently, an agent-based CPPS should not be limited to a specific hierarchy level of a production automation system.

(R3) Platform independent implementation: From the great variety of automation systems' components, that need to be able to participate inside a CPPS network (cp. R2), restrictions and requirements regarding the computing devices arise. For the automation of overall production plants Programmable Logic Controllers (PLC) are predominantly used in industry. Consequently, these platforms need to be considered as one important device type. However, due to the required level-independence of the approach (cp. R2), it must not be limited to this class of devices. Especially for small sensors often cost efficient and not very powerful hardware is used, in order to reduce the overall cost of the components. Consequently, basic software and data for the MAS need to be small and lightweight, i.e., resource-friendly concerning permanent and non-permanent memory, CPU, and bandwidth.

(R4) Robustness against errors: Typical problems found in large networks include connection loss, unsteady bandwidth, and load problems. During runtime, unforeseen problems and changes may occur, e.g., failure of machinery, new urgent tasks or maintenance. These may lead to unavailability of the corresponding system. The local problem may also affect the whole network, e.g., by delaying production. Thus, intelligent software and intelligent communication, e.g., broadcast as well as point-to-point messages with dynamic routing, should be used. In some cases, e.g., permanent unavailability of production systems or connected components, the CPPS has to reconfigure itself to ensure overall availability and continuation of the production. The same holds true for orderly joining and leaving systems at runtime. In all cases, the MAS must react to these dynamic conditions in an appropriate way, i.e., it must be robust against (unforeseen) reconfiguration requests, especially against reconfiguration of the network, i.e., participants joining and leaving.

(R5) Decentralization: Not only the CPPS as a whole, but also subsets have to deal with temporary network connection loss. This means, the CPPS may split in smaller parts for a limited amount of time prior to re-connection. During this time, participating remote systems and system components must not be incapable of action. Thus, a central management node with all required knowledge as well as implemented a priori calculations is not suitable. Instead, critical information should be distributed between multiple nodes, so that access is ensured as far as possible and decisions can be made dynamically.

3. RELATED WORKS

To persist in today's global markets, information must flow between all layers of a company and even between collaborating companies. This requires new approaches to communication and production. Some work is done at comparing the HTTP protocol to Modbus (Jestratjew and Kwiecien, 2013) at PLC level. HTTP is considerable lower, mainly due to slow string processing on PLCs. A common way to access the factory floor is using gateways. In (Sauter and Lobashov, 2011) an overview of suitable high-level protocols to access automation data via gateways is presented. One possible implication is a reconfigurable sensor interface (Tao *et al.*, 2014). This work also presents new design method, but only focuses on the perception layer of the IoT architecture (R2). A work concerning groups and grouping of devices is also described in (Vicaire *et al.*, 2012). The work uses a central middleware (R5), programmed in JAVA (R3). An approach for agent-based gateway implementations is presented in (Faul, Jazdi and Weyrich, 2016)but not evaluated regarding its resource efficiency and, thus, its platform independence (R3). In the context of Industry 4.0 recently a number of architectures have been presented based on the paradigms SOA (Moghaddam, Silva and Nof, 2015), MAS (Alexakos and Kalogeras, 2015; Hoffmann, Meisen and Jeschke, 2017), Internet of Things (IoT) (Sauer, Hausten and Hofstedt, 2016), or CPS in general (Bagheri *et al.*, 2015), which did not investigate a specific implementation's resource efficiency in detail (R2, R3). This also holds for the works presented in (Quintanilla *et al.*, 2016), which propose a holonic architecture for CPPS, although they evaluate the need for communication (i.e., number of messages exchanged) in a simplified production scenario. For simulation real-time models, the author in (Aksyonov *et al.*, 2015) describes the integration problem on a basic class of models further extended by the intelligent distributed agents (R5). The main model consists of agent resources conversion process with support for MAS modelling, in combination with discrete-event modelling and the ontology of the system.

In (Girbea *et al.*, 2014) it is focused on designing a SOA additionally capable of real-time operation. This is achieved using a priori algorithms and is thus not suitable for dynamically changing environments (R4). Other approaches use SOA for diagnosis (Calvo *et al.*, 2012) and the concept presents a possible architecture and diagnosis algorithms without implementation. A different possible implementation language for SOA is IEC 61499, e.g., proposed in (Barata, Cândido and Feijao, 2008) in combination with a message broker. A runtime based on a formal mapping between SOA and IEC 61499 is proposed in (Delamer and Lastra, 2006). In contrast to other programming languages, IEC 61499 only runs on special hardware or with dedicated runtimes (R3).

However, this excludes, e.g., small 8-bit micro-controllers. An approach to enable reconfiguration is to use a (central) middleware to orchestrate a SOA. One such approach is the iLand project (Calvo *et al.*, 2012; García-Valls, Rodríguez-López and Fernández-Villar, 2013). A-priori algorithms are used to analyse the system and to calculate possibilities and strategies in case of failure and other reconfiguration triggers. Nevertheless, a priori calculations cannot be done in global networks with unknown behaviour (R4). Additionally, reconfiguration and behavioural intelligence is accumulated inside a central node running on powerful hardware (R2, R3, R5). Other work often utilizes the Holonic Manufacturing System (HMS) paradigm and corresponding reference architectures like PROSA (Brussel *et al.*, 1998). Although in this and other works applying the HMS paradigm, e.g., ADACOR (Leitão and Restivo, 2006), central entities do exist, the decentralization of the architectures decision making can be adjusted by altering the controlling entities' autonomy.

MAS are increasingly investigated in order to decentralize automation, enhance flexibility of production plants and realize advanced functionality (Brennan, 2007; Leitão, Marik and Vrba, 2012). Enabling reconfiguration within a production plants is the most common usage for software agents and can be used e.g., to handle device failures, structural changes or dynamic production planning (Lüder *et al.*, 2005). Also agents for the industry have been applied to achieve other crucial properties of CPPS such as complexity management, intelligence, modularity, robustness, adaptation, and responsiveness (Leitão and Karnouskos, 2015; Cruz S. and Vogel-Heuser, 2017). Thanks to these benefits, increasing adoption of MAS in manufacturing have been demonstrated via important industry initiatives and projects results with agents in smart production, smart electric grids, and scheduling and logistics optimization (R1, R2, R3) (Leitão *et al.*, 2016). There are resource-efficient agent systems available (Theiss *et al.*, 2008) that, however, do not support PLCs and other small devices (R3). Enabling reconfiguration on automation hardware, e.g., in context of a transportation system, is also possible (Vallée *et al.*, 2011). In this work, a MAS is used to reconfigure a transportation system during runtime, in case of failure of one of the conveyor belts. This agent system is also implemented in the programming language IEC 61499, so that additional software or specialized hardware for running the code is needed, thus excluding micro-controllers (R3). Other MAS implemented in IEC 61499 are also aimed at modular logistics systems (Yan and Vyatkin, 2013).

Other research on new approaches by using IEC 61499 as an emerging standard for industrial automation is presented in (Vyatkin, 2011). Other work targets the use of MAS in the field of intelligent energy systems or smart grids (Vrba *et al.*, 2014). Apart from manufacturing industry, MAS are also used in process industry, e.g., to control critical processes (Metzger and Polakow, 2011). Another concept and implementation for reconfiguration is presented in (Barata, Cândido and Feijao, 2008). This research focuses on challenges of the shop floor, especially addition and removal of manufacturing components (modules) during runtime. Communication is based on JADE, using JAVA and FIPA ACL Messages. Thus, the platform independence is limited (no small devices are supported) (R3). The IDEAS project (Onori *et al.*, 2012) uses specifically designed and produced boards to bring agent technology to lower automation levels. These boards are designed to support JAVA (JADE). Thus, to deploy this agent system, specifically designed hardware must be used small microcontrollers or other existing automation hardware are not supported (R1, R2, R3).

4. CONCEPT OF AN AGENT-BASED CPPS

For the development of the proposed architecture, different aspects have been taken into account. The aspect *logical architecture* describes the relations between the agents and how the MAS is set up. The aspect *software architecture* describes measures taken in order to allow a wide variety of hardware platforms, i.e., ensure platform independency. The aspect MAS *protocols and messages* describes the design of the data exchange between the connected agents. These aspects play an essential role in realizing the overall features of the proposed concept (cp. section 2).

4.1 MAS Logical Architecture

Derived from the already existing standard of the Foundation for Physical Agents (FIPA) for MAS, the architecture shown in Fig. 1 was developed for arbitrary use cases. All agents of the CPPS are connected via a common network, e.g., the global internet. The communication and collaboration is realized by protocols and messages described in sec. 4.3. To support a variety of different use cases (cp. R1), an agent of the CPPS can have one of two characteristics: each agent represents either a physical system (Fig. 1; 1, 2) or an organizational entity (Fig. 1; 3), e.g., an agent fulfilling diagnosis services (diagnosis agent) or introducing production requests into the system (product agent). In the first case, the agent manages the access of the CPPS network to the production system (and vice versa) while, due to an agent's inveterate autonomy, it can dynamically regulate this access in accordance with the policies of the company which owns the production system. Furthermore, these agents can dynamically reconfigure their production system, e.g., in a manufacturing-specific scenario, in order to realize a load balancing of production orders inside the CPPS network.

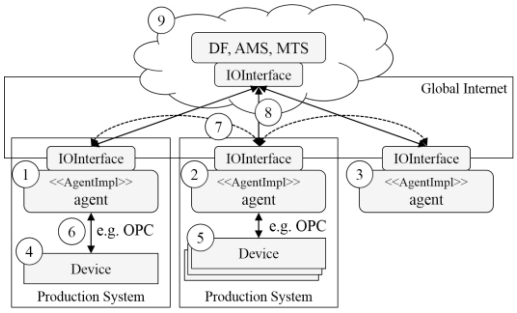

Fig. 1. Logical Architecture of the MAS.

A CPPS network may consist of a varying number of this type of agents. In order to support the easy migration of existing arbitrary systems to network-enabled systems (cp. R1, R2), an agent may be a dedicated part of the automation

software (Fig. 1; 1) as well as separate software on separate hardware (Fig. 1; 2). It may represent a single device (Fig. 1; 4) as well as several devices (Fig. 1; 5). Application, level and platform independence (cp. R1, R2, R3) as well as migration are further increased by not setting a default for the communication with attached physical systems (Fig. 1; 6). Possible implementations are e.g., field bus protocols, OPC or proprietary protocols. To support as many different hardware devices as possible, the code that manages communication with physical systems is released in separate communication modules (cp. R3). This also allows future implementation of further use cases (cp. R1). The operation and reconfiguration (cp. R4) of the CPPS is based on a FIPA compliant platform (Bellifemine, Poggi and Rimassa, 2001), which requires three basic organizational entities, e.g., for discovery purposes: an *agent management system* (AMS), a *message transport system* (MTS) and a *directory facilitator* (DF). Robustness against errors (cp. R4) can be enhanced by using direct connections between agents. The AMS allows the bidirectional mapping between IP-addresses and agents' names. This enables direct communication between agents (Fig. 1; 7) by e.g., requesting the IP-address of an agent (identified by its name) and afterwards using this information to establish a direct connection that does not depend on other entities. Use-Case independence (cp. R1) can be enhanced by keeping protocols and messages flexible. The MTS is a directory of protocols and messages in a format based on eXtensible Markup Language (XML), which are required to participate in a use case. It can be extended and adapted to specific use cases. The DF is a service that stores agents' abilities, i.e., the production capabilities of a system. All agents register themselves with these organizational entities (Fig 1; 8) and are afterwards detectable by other agents by name, ability or address. This also enables new agents or systems to access to the CPPS network. In order to enable high availability and independence from single nodes (cp. R5) as well as easy handling of joining and leaving nodes (cp. R4), the organizational entities also periodically check all agents for availability to keep the directories up to date (cp. R4). Similar to internet name services, the directories are distributed in a cloud (Fig. 1; 9) among multiple nodes. This also minimizes the time required for search requests because of the proximity of the directory relative to the agent.

4.2 MAS Software Architecture

The presented logical architecture serves as a specification for the MAS software architecture. From this and under consideration of the requirements stated in sec. 2, the software architecture for the agent-based CPPS shown in Fig. 2 was developed. To achieve application, level and platform independence (cp. R1, R2, R3), several measures were taken. For the organizational entities, i.e., the AMS, MTS, efficient implementations in the programming language ANSI C were developed according to the specified logical architecture. Therefore, the implementations comprise declarations for the particular directories as well as lookup functions. The implementation of the basic agent is provided as an ANSI C library, upon which an application specific executable implementation can be built. All basic functionality is exposed to the final executable in the form of multiple interfaces that can be adjusted to the user's specific applications and needs. By separating basic, i.e., the pure administration of an entity inside the CPPS, and use case specific behaviours, e.g., scheduling production orders to the shop floor, the implementation can be used for a multitude of use cases (cp. R1). To be able to distinguish between basic functionality and use case specific functions, an appropriate interface was developed. In detail, the agent interface and class is responsible for overall intelligent behaviour, situation awareness and runtime adaption of the single agent itself. The implementation in the executable (*AgentImpl*), i.e., all agents shown in Fig. 1, may be used by the developer of a specific application in order to realize application specific properties and functions, including all intelligence required to handle its application specific tasks (cp. R1). For example, in a manufacturing-specific scenario the *AgentImpl* would contain the implementation to dispatch, monitor and reconfigure running manufacturing tasks to the production units of a plant. In contrast, the generic interface in the library (*Agent*) defines all basic operations needed to participate in the MAS, e.g., joining, registration and heartbeat monitoring. A separate class (*AgentConfig*) handles initial configuration of the agent, as well as its internal dynamic reconfiguration during runtime (R4) inside the MAS network, e.g., when communications need to be changed.

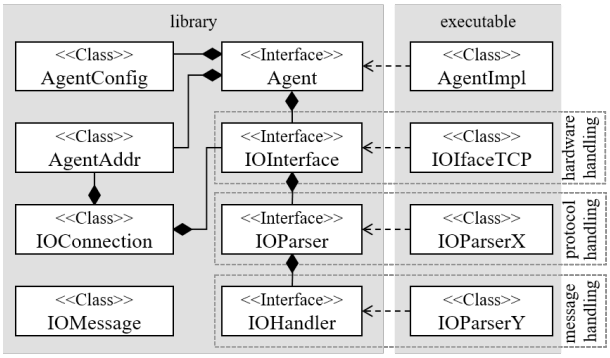

Fig. 2. Software Architecture of the MAS.

In order to represent the different abstraction layers of a communication, i.e., hardware, protocol and message, a hierarchical approach, from general to use case specific, was selected. The interface class (*IOInterface*) and its realization is responsible for handling connections to other agents and the directory services, e.g., the DF (cp. Fig. 1, 7, 8). It abstracts specific behaviours by handling communication hardware and low-level protocol specific tasks, functions and behaviours, e.g., creating and closing connections. Each agent can have multiple communication interfaces simultaneously, in order to support many different hardware platforms (cp. R3). The actual realization inside the executable implements the behaviour for a specific hardware interface, e.g., TCP/IP (*IOIfaceTCP*). Information relevant for an established connection to another agent is saved in a dedicated class (*IOConnection*) one instance per connection. The parser class (*IOParser*) handles the raw data stream received through an interface and extracts messages out of it, e.g., by handling packet fragmentation and finding message boundaries. These messages can be basic messages for managing the overall agent system, as well as use case

specific messages. A specific class inside the executable (*IOParserX*) handles basic and use case specific protocols, e.g., downloaded from the MTS. The handler class (*IOHandler*) manages the incoming use case specific messages and reacts according to the application specific procedures, provided in the executable (*IOHandlerY*). In summary, raw data is received via the *IOInterface* and split into agent messages by the *IOParser*. Afterwards the messages are processed by the *IOHandler*, i.e., appropriate actions are executed.

To support different hardware platforms, the implementation was developed to be platform independent (cp. R3), i.e., the programming language was chosen appropriately. Widely used languages for platform independent programming include C and JAVA. Although there are JAVA runtime environments (JRE) available for small devices, there is none small enough to fit a small sensor's 8-bit micro-controller together with the agent itself. Consequently, platforms based on JAVA are not suitable. Much the same applies to C++. Commonly used industrial automation devices as PLCs and soft-PLCs predominantly support the programming languages standardized by the IEC 61131-3. In addition, with their programming environments a great variety of PLC vendors provides support for programming PLCs in C. Therefore, because of the broad support that also includes a variety of PLCs, C was chosen as the programming language to implement the agent-based CPPS network. Implementations of different application specific systems can be generated for a multitude of hardware platforms using the corresponding ANSI C compilers. Therefore, specific characteristics of the hardware platform, e.g., big-little endian problem, are abstracted by the compiler. However, by using ANSI C as the implementation language of the agents for the CPPS network, the interfaces and classes are all implemented as plain Cstructs. Inheritance is implemented similar to the Linux kernel, i.e., realizations derive from these structs using function pointers and the parent as first member.

4.3 MAS Protocol and Messages

The global internet as an omnipresent communication medium suggests itself as a suitable solution for communication between multiple production facilities. By using the internet, the network layer of the OSI model is fixed (IP). For the transport layer there are two possibilities, namely TCP and UDP. In contrast to UDP, TCP is a connection-oriented protocol that offers high reliability, packet ordering and flow control. When using UDP-based communication error checking, packet ordering etc. needs to be done by the application, thus stressing CPU and memory and increasing application size. Therefore, TCP was chosen as the standard transport layer protocol. On top of TCP, the protocols used on a higher level (process level) have to be specified. This includes protocols for the application independent agent system management, as well as protocols for the application. Following the structure of the internet protocol suite, the application specific protocols are embedded in the platform protocol as the platform protocol is embedded in TCP (transport layer). Likewise, also the message format and message encoding have to be specified. As the framework only provides basic functionality there is also only a basic set of management functions used to identify, search and connect agents. In a production scenario these functions can be used either for (plant) agents to register production capabilities, which their facilities offer or for (product) agents to broadcast inquiries for offered capabilities.

Because of the open software design of the agent system, neither the message format nor protocols are static. Nowadays, with the advent of the IoT, many systems use HTTP as a primary protocol. Likewise, MQTT is popular for small devices, but uses a publish/subscribe architecture that needs a central node, just as OPC UA uses a client-server architecture. To avoid the costs required for processing HTTP headers (cp. R3) and still not needing a message broker (cp. R5) or increasing size, a very small protocol was developed that enables the agents to exchange arbitrary data. It uses a fixed binary header, containing all needed information. This includes e.g., size, so that also binary data is possible, sequence number or conversation id and the actual data. The header can easily be processed because of the fixed size and positions. Since for the communication messages based on XML are used, which are parsed by the corresponding function of the receiving agent (cp. sec. 4.2), specific characteristics of the hardware platforms, e.g., the big-little endian problem mentioned earlier, are abstracted from by the developed communication mechanism. Subsuming this section, by the proposed logical architecture and software architecture as well as the protocols and messages, a generic MAS platform for networked CPPS was developed considering multiple requirements. Due to the requirements regarding platform and level independence, ANSI C was used as the programming language for the platform.

5. USE CASES AND EVALUATION

Some of the requirements stated in sec. 2 are already partly achieved by the design of the proposed architecture: The suitability for different use cases (R1) is ensured by both the logical and the software architecture. Level independence (R2) and platform independence (R3) are partly achieved by using a modular library and ANSI C. Using TCP/IP as underlying communication protocol solves parts of the error handling and recovery (R4) but also enables connections to systems using other implementations. By distributing organizational elements in the cloud, the MAS is decentralized (R5). However, resource-friendliness (R3) and the reconfiguration in case of joining/leaving agents (R4) were further investigated by quantitative experiments. As a first assessment of the platforms suitability, these experiments measure the pure footprint of the platform.

5.1 Application Examples of the Agent-Based CPPS

To investigate the required application independency of the approach (cp. R1), the proposed generic architecture for an agent-based CPPS was applied in two different specific application examples. At first, the agent-based CPPS was used in an academic demonstrator that produces mass-customized products, i.e., yoghurt.

The production facilities were represented by a varying number (with minimum of three) operational laboratory production plants of different German universities (Vogel-Heuser *et al.*, 2014). All the facilities were administered resembled a network of locally distributed production sites, connected by the internet, which offer the execution of sub-processes, e.g., the processing of raw materials (yoghurt), adding different flavours, or finishing or packaging the yoghurt. Therefore, with the first implementation, facilities for (batch) processes as well as discrete manufacturing were controlled and linked by the MAS inside the CPPS network. The second application example additionally considered logistics between production facilities executed by mobile robots. Here, the agent-based CPPS implemented a distributed production environment on a trade show, including multiple companies in different exhibition halls. The manufacturing steps, available at the different facilities contributed to the production of a bottle opener, which could be customized by trade show visitors. The possibilities for an application specific implementation (AgentImpl, cp. sec. 4.2) of the developed generic platform were used by developers of the different participating companies to implement on their own hardware the functionality that was necessary to manage the execution of production steps at the different facilities in the exhibition halls. Thereby, different hardware platforms, e.g., PC and PLC, were used. Due to the agent platform's open protocol that builds on TCP/IP connections to other implementations, e.g., based on C++, were enabled for this application scenario. The transport of the give-away articles between the different companies and exhibition halls was done using small partly autonomous mobile robots that were also represented by agents and were automatically embedded in the overall process. In both use cases, the production facilities were represented by autonomous agents that are connected to the local hardware and accept new orders after registering at the directory services. Since the customers' orders for products need to be decomposed into multiple different manufacturing tasks, to which the facility agents can respond, a coordinator agent was implemented for this purpose. The mobile robots are also represented by agents, thus dynamically responding to transport requests. The agents are running on diverse hardware platforms, e.g., PC and PLC, under different operating systems, i.e., Windows and Linux. Each use case has a dedicated set of messages for communication, based on the lightweight protocol described earlier. New orders are entered by the user via a web interface and are afterwards distributed inside the production network.

The two use cases show, that the MAS is able to physically and logically connect multiple plants (high level) as well as smaller devices via internet. The connected agents run on a multitude of different hardware while the basic functions are the same with all use cases. It is not necessary to provide a central node for management or decision making purposes besides the organizational entities, which mainly enable the communication between the autonomous entities, and the coordinator agent, which decomposes product orders into manufacturing tasks. Consequently, although a high level of autonomy may be implemented for the controlling entities, compared to other works, the proposed architecture does not fully support the implementation of heterarchic, e.g., (Rey *et al.*, 2013), or isoarchic, e.g., (Pujo, Broissin and Ounnar, 2009), architectures yet. Since the specific implementations for the second application example, i.e., the industrial production scenario on the trade show, were developed by the companies themselves, no comprehensive experiments for a quantitative evaluation with these scenarios could be conducted. Consequently, as first step in this direction, qualitative experiments using only the basic platform are presented in the following.

5.2 Experiments for Quantitative Evaluation

To further examine the fulfilment of the requirements stated earlier, different measurements were conducted in a laboratory experiment. This includes the resource requirements of an agent (cp. R3) and the suitability of the approach for large, constantly changing networks (cp. R3, R4). This was measured by obtaining timings during platform runtime. A network consisting of agents running on different platforms was set up. The different platforms included the ARM platform as well as x86 (32bit and 64bit). The first point that was investigated was the resource usage of a single instance of an agent of the proposed approach (cp. Fig. 2, *AgentImpl*) without application specific implementation parts, i.e., the pure footprint of the agent platform. The specific implementation of an agent's *main()* function is strongly related to its role inside an agent system the reconfiguration capabilities that have been implemented, and its developer's skills. Consequently, in order to evaluate the suitability of the proposed MAS platform, in a first step, these parts of an agent's implementation were not considered. Table 1 shows the resource usage (footprint) on an ARM platform, running Arch Linux (3.6.11-12-ARCH+, GCC 4.9.2). The agent, together with the agent library, needs approximately 23k bytes of permanent memory, e.g., flash memory. In comparison, the Matrikon OPC UA Embedded Server needs about 240k bytes of flash memory on an ARM architecture, OPC UA server implementation were proposed that require 116k bytes (Iatrou and Urbas, 2016), and a typical MQTT implementation needs about 44k bytes. Consequently, the resources needed for MAS platforms based on standard issue OPC UA stacks, e.g., (Hoffmann, Meisen and Jeschke, 2017), can be considered higher, although such approaches are promising in terms of already available technology. During execution, the heap usage of an agent is about 3k bytes of memory per connection. This is due to the fixed buffer sizes (2*1600 bytes) currently used and can thus be drastically reduced by using dynamic memory allocation for receiving and sending data. On a Windows platform, the values differ: Disk usage is 52k bytes for the library and 10k bytes for the agent. Values for the AMES framework (disk size is not available due to unavailability of the AMES platform), JADE (Bellifemine, Poggi and Rimassa, 2001) and AKKA (Gupta, 2012) are given as additional references. Both JADE and AKKA are very powerful JAVA libraries, offering a wide variety of functionality, covering many industrial and scientific aspects, thus further increasing size and memory consumption. The comparison done shows the suitability of the proposed MAS for resource-constrained small devices. The values for AMES were taken from (Theiss

et al., 2008), whereas the values for JADE and AKKA were measured on an Intel Core2 Quad with 8GB RAM, running Windows 7 64bit. All values highly depend on the actual platform, due to different instruction sets, library volume and optimization possibilities. Especially the memory footprint differs and can be further enhanced by increasing response time. The table shows, that the executable's size is about 24k (on ARM). Much of this size originates from the C library that is used to e.g., create an entry point for the operating system. On Linux, about half of the executable's size amounts to these operating system functions. However, as a small micro-controller typically, does not have an operating system, this can be discarded in this case (R3).

Table 1. Resource Usage

Object	Disk Size	Heap Memory
BaseAgent	4004 bytes	3k bytes / connection
Library	19896 bytes	-
AMES	-	4k bytes
JADE	2.97M bytes	> 7.2M bytes
AKKA	4.04M bytes	> 19.3M bytes

Response time and typical internet characteristics, e.g., connection loss and joining/leaving nodes, were investigated as part of the experiments as well. A varying number of agents, i.e., connections to the network, joining and leaving at random, were used. The agents ran on different hardware platforms, communicating with each other. For the measurements only a single directory (non-distributed) service was used as a central node, running on an Intel Core 2 Duo 6600, with 4GB of RAM, with Windows 7 (64bit). Fig. 3 (left) shows the average time needed to process an order request inside the CPPS, i.e., until all agents have bid and the winner is selected from a varying number of agents, ranging from 1 to 500. The agents are using a simple text-based protocol with a typical message size of about 40 bytes. Such an exemplary agent uses a dynamically linked list to find a suitable offer for a request. On smaller platforms (ARM), due to less processing power, it is more likely that network packets drop or calculations take longer. On powerful hardware (x86) the increase for the operation is not as significant. The graph also shows that although agents join and leave the network and the overall time increases, the CPPS always reconfigures itself and a suitable solution for production distributing is found (R4). Otherwise the timeout for the order request would be reached at three seconds. Fig. 3 (middle) shows the average workload of the directory service, expressed in time needed to handle a single agent (connection) in relation to the overall number of agents connecting to the network ranging from 1 to 800. The workload rises continuously with the number of agents present in the network from approximately 0.3ms at one connection to approximately 2.5ms at 800 connections. The rise results from additional lookups and interruptions as Fig. 3 (right) shows that the time needed to process the raw data packets without handling the action involved can be assumed nearly as constant for a number of agents ranging from 1 to 500. This also shows that the reconfiguration of the CPPS, in case of joining and leaving nodes, does not affect the time needed to process single requests. Thus, the CPPS stays operational (R4). A comparison with OPC UA shows that typical servers are limited to about 100 connections. A comparison with MQTT message brokers shows, that performance heavily depends on the implementation.

Subsuming, the results of the evaluation experiments imply that the generic MAS platform can be considered suitable for the imposed requirements (cp. section 2). The evaluation of the resource usage (cp. Table 1) shows that the agent implementation is small enough to be deployed on micro-controllers and other small hardware (R2, R3). Also, the response and reconfiguration time of the CPPS is not affected by joining or leaving nodes but only by the total number of connected agents (cp. Fig. 3, left and middle) and, thus, the CPPS is flexible enough to handle dynamic environments (R4). Even with a high number of agents implemented on less powerful hardware, a production order can be processed in less than 2.5 seconds by the basic agent platform (cp. Fig. 3, left), a time that can be considered appropriate for production order dispatching.

6. SUMMARY AND OUTLOOK

Small lot sizes are a common trend in the production industry due to the demand for customized products. Likewise, dynamic global markets and new technologies pose new challenges for companies. Using a CPPS can help with attaining a higher level of flexibility and adaptability, especially when such a system allows for migration of existing facilities. By distributing intelligence and decision-making, individual circumstances of a plant can be respected and fault tolerance increases. MAS offer a way to implement a distributed intelligent system that is adaptable and highly flexible. This paper proposes an approach that is separated into three aspects: the *logical architecture*, the *software architecture*, and the *protocols and messages* of agent-based CPPS network. A subset of the imposed requirements was fulfilled by the particular design of the three aspects. To further evaluate all the requirements and especially the ones not satisfied by design, measurements of a basic operating agent-based CPPS network were conducted. The evaluation showed that the basic MAS can be deployed on small devices and that timely reconfiguration without influences on other participants is possible.

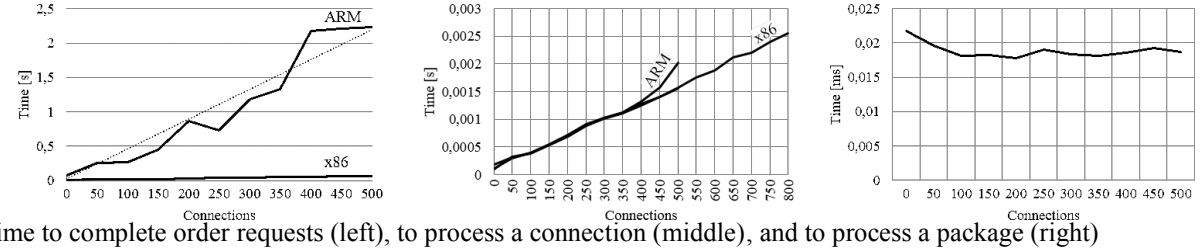

Fig. 3. Time to complete order requests (left), to process a connection (middle), and to process a package (right)

However, since this first experiments only provide a quantitative evaluation for the basic platform, in the next steps further experiments need to be conducted that provide implementations, which are realistic for industrial applications.

In summary, the design and implementation for an agent-based CPPS network proposed in this paper is a possibility to easily connect devices and let them participate in a global intelligent network. The applied MAS is platform independent, resource-friendly and thus suited for a wide variety of devices and use cases. By the proposed implementation in the programming language C, the approach is portable but not limited to Programmable Logic Controllers, as it would be the case with the programming languages of the IEC 61131-3. The use of intelligent, proactive and autonomous agents yields benefits such as inner and outer reconfigurability, robustness against perturbations and high overall flexibility. The MAS is scalable and adaptable to users' needs as well as suitable for high-level and low-level use cases. Future work includes the improvement of the programming of the application specific tasks and intelligence, possibly by using model-based approaches. This includes the evaluation of the MAS design and implementation with PLCs as the targets for the deployment of the agents. Other work will improve the protocols currently used in the communication between agents in order to support open standards such as JSON or HTTP and it will especially improve security aspects such as authentication and encryption that were considered only marginally in the current implementation. In the case of the open source framework AKKA, actors would be on the same level of implementation since is possible to provide basic communication and agent (-like) classes. Since AKKA runs on JAVA8 or later versions this could implement actor driven design attributes and is now extensively used in the industry increasing the fault tolerance and reactiveness of systems.

REFERENCES

Aksyonov, K. *et al.* (2015). Simulation of the real-time simulation systems and its integration with the automated control system of an enterprise. in *2015 IEEE International Symposium on Robotics and Intelligent Sensors*.

Alexakos, C. and Kalogeras, A. P. (2015). Internet of Things integration to a Multi Agent System based manufacturing environment. in *IEEE ETFA*.

Bagheri, B. *et al.* (2015). Cyber-physical Systems Architecture for Self-Aware Machines in Industry 4.0 Environment. *IFAC-PapersOnLine*.

Barata, J., Cândido, G. and Feijao, F. (2008). A multiagent based control system applied to an educational shop floor. *Robotics and Computer-Integrated Manufacturing*, 24(5), pp. 597–605.

Bellifemine, F., Poggi, A. and Rimassa, G. (2001). JADE: A FIPA2000 Compliant Agent Development Environment. in *Proc. Int. Conf. on Autonomous Agents*.

Brennan, R. W. (2007). Toward real-time distributed intelligent control: A survey of research themes and applications. *IEEE Trans. Syst., Man, Cybern. A: Syst., Humans*, 37(5), pp. 744–765.

Brussel, H. Van *et al.* (1998). Reference architecture for holonic manufacturing systems: (PROSA). *Computers in Industry*, 37(3), pp. 255.

Calvo, I. *et al.* (2012). Ubiquitous Computing and Ambient Intelligence. in. Springer Berlin Heidelberg, pp. 282–289.

Cruz S., L. A. and Vogel-Heuser, B. (2017). Comparison of Agent Oriented Software Methodologies to Apply in Cyber Physical Production Systems. in *2017 IEEE 15th International Conference on Industrial Informatics*.

Delamer, I. M. and Lastra, J. L. M. (2006). Service-Oriented Architecture for Distributed Publish/Subscribe Middleware in Electronics Production. *IEEE Trans. Ind. Inf.*, 2(4).

Faul, A., Jazdi, N. and Weyrich, M. (2016). Approach to interconnect existing industrial automation systems with the Industrial Internet. in *IEEE ETFA*.

García-Valls, M., Rodríguez-López, I. and Fernández-Villar, L. (2013). iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems. *IEEE Trans. Ind. Inf.*

Girbea, A. *et al.* (2014). Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications. *IEEE Trans. Ind. Inf.*, 10(1), pp. 185–196.

Gupta, M. (2012) *Akka Essentials*. Packt Publishing Ltd.

Hoffmann, M., Meisen, T. and Jeschke, S. (2017). OPC UA Based ERP Agents: Enabling Scalable Communication Solutions in Heterogeneous Automation Environments. in *Intern. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*.

Iatrou, C. P. and Urbas, L. (2016). Efficient OPC UA binary encoding considerations for embedded devices. in *IEEE INDIN*.

Jammes, F. and Smit, H. (2005). Service-oriented paradigms in industrial automation. *IEEE Trans. Ind. Inf.*, 1(1), pp. 62–70.

Jestratjew, A. and Kwiecien, A. (2013). Performance of HTTP Protocol in Networked Control Systems. *IEEE Trans. Ind. Inf.*, 9(1), pp. 271–276.

Lee, E. A. (2008). Cyber Physical Systems: Design Challenges. *IEEE Int. Symp. on Object Oriented Real-Time Distributed Computing*.

Leitão, P. *et al.* (2016). Smart Agents in Industrial Cyber Physical Systems. *Proceedings of the IEEE*, 104(5), pp. 1086–1101.

Leitão, P. and Karnouskos, S. (2015) *Industrial Agents: Emerging Applications of Software Agents in Industry*.

Leitão, P., Marik, V. and Vrba, P. (2012). Past, present, and future of industrial agent applications. *IEEE Trans. Ind. Inf.*, 9(4), pp. 2360–2372.

Leitão, P. and Restivo, F. (2006). ADACOR: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry*, 57(2).

Lüder, A. *et al.* (2005). Distributed Automation: PABADIS versus HMS. *IEEE Transactions on Industrial Informatics*, 1(1), pp. 31–38.

Metzger, M. and Polakow, G. (2011). A survey on applications of agent technology in industrial process control. *IEEE Trans. Ind. Inf.*, 7(4).

Moghaddam, M., Silva, J. R. and Nof, S. Y. (2015). Manufacturing-as-a-Service—From e-Work and Service-Oriented Architecture to the Cloud Manufacturing Paradigm. *15th IFAC Symposium onInformation Control Problems inManufacturingINCOM 2015*, 48(3), pp. 828–833.

Onori, M. *et al.* (2012). The IDEAS project: plug & produce at shop-floor level. *Assembly Automation*, 32(2), pp. 124–134.

Pujo, P., Broissin, N. and Ounnar, F. (2009). PROSIS: An isoarchic structure for HMS control. *Engineering Applications of Artificial Intelligence*.

Quintanilla, F. G. *et al.* (2016). Implementation framework for cloud-based holonic control of cyber-physical production systems. in *IEEE INDIN*.

Rey, G. Z. *et al.* (2013). The control of myopic behavior in semi-herterarchical production systems: A holonic framework. *Engineering Applications of Artificial Intelligence*, 26(2), pp. 800–817.

Sauer, P., Hausten, T. and Hofstedt, P. (2016). Using internet of things technology to create a really platform independent robotics framework. in *IEEE ISSE*.

Sauter, T. and Lobashov, M. (2011). How to Access Factory Floor Information Using Internet Technologies and Gateways. *IEEE Trans. Ind. Inf.*, 7(4), pp. 699–712.

Tao, F. *et al.* (2014). CCIoT-CMfg: Cloud Computing and Internet of Things-Based Cloud Manufacturing Service System. *IEEE Trans. Ind. Inf.*, 10(2), pp. 1422–1435.

Theiss, S. *et al.* (2008). AMES - A Resource-Efficient Platform for Industrial Agents Department of Computer Science Dresden University of Technology. *IEEE WFCS*, pp. 405–413.

Vallée, M. *et al.* (2011). Decentralized reconfiguration of a flexible transportation system. *IEEE Transactions on Industrial Informatics*, 7(3).

Vicaire, P. A. *et al.* (2012). Bundle: A Group-Based Programming Abstraction for Cyber-Physical Systems. *IEEE Trans. Ind. Inf.*, 8(2).

Vogel-Heuser, B. *et al.* (2014). Coupling heterogeneous production systems by a multi-agent based cyber-physical production system. *IEEE INDIN*.

Vrba, P. *et al.* (2014). A Review of Agent and Service-Oriented Concepts Applied to Intelligent Energy Systems. *IEEE Trans. Ind. Inf.*, 10(3).

Vyatkin, V. (2011). IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review. *IEEE Trans. Ind. Inf.*, 7(4), pp. 768.

Yan, J. and Vyatkin, V. (2013). Distributed Software Architecture Enabling Peer-to-Peer Communicating Controllers. *IEEE Trans. Ind. Inf.*, 9(4).

Publication III (MASpatterns)

International Journal of Advanced Manufacturing Technology 105/9 (2019), pp. 4005-4034.
https://doi.org/10.1007/s00170-019-03800-4

The International Journal of Advanced Manufacturing Technology (2019) 105:4005–4034
https://doi.org/10.1007/s00170-019-03800-4

ORIGINAL ARTICLE

Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns

Luis Alberto Cruz Salazar[1,2] · Daria Ryashentseva[1] · Arndt Lüder[3] · Birgit Vogel-Heuser[1]

Received: 3 September 2018 / Accepted: 17 April 2019 / Published online: 26 July 2019

Abstract
The growing complexity of production systems requires appropriate control architectures that allow flexible adaptation during their runtime. Although cyber-physical production systems (CPPS) provide the means to cope with complexity and flexibility, the migration with existing control systems is still a challenge. The term CPPS denotes a mechatronic system (physical world) coupled with software entities and digital information (cyber part), both enabling the smart factory concept for the Industry 4.0 (I4.0) paradigm. In this regard, design patterns could help developers to build their software with common solutions for manufacturing control derived from experiences. We provide a description and comparison of the already existing multi-agent systems (MAS) design patterns, which were collected and classified by introducing two classification criteria to support MAS developers. The applicability of these criteria is shown in the case of specific example architectures from the lower and higher control levels. The authors, together with experts from the German Agent Systems committee FA 5.15, gathered more than twenty MAS patterns, evaluated, and compared four selected patterns with the presented criteria and terminology. The main contribution is a CPPS architecture that fulfills requirements related to the era of smart factories, as well as the Reference Architectural Model I4.0 (RAMI 4.0). The conclusions indicate that agent-based patterns greatly benefit the CPPS design. In addition, it is shown that manufacturing based on MAS is a good way to address complex requests of the CPPS development.

Keywords Cyber-physical production systems · CPPS · Design patterns · Distributed control systems · Industry 4.0 · MAS · Multi-agent systems · RAMI 4.0

1 Introduction

Commonly, companies widen their product portfolio and attempt to shorten their production time to increase revenue and market presence. These actions may indirectly increase the complexity of the production process. At the same time, the Industry 4.0 (I4.0) paradigm tries to meet the emerging DIN SPEC 91345 norm [1], regarding the Reference Architectural Model I4.0 (RAMI 4.0) inside the factory and migrate from conventional automation systems to cyber-physical production systems (CPPS) [2], and their admitted standards [3, 4]. The term CPPS denotes a mechatronic system coupled to smart entities that enable the smart factory and machines tools of I4.0 concept [5, 6]. The application of the distributed control theory based on the multi-agent systems (MAS) is employed [7, 8] to cope with CPPS challenges. Since it is not always straightforward to create such a system from scratch, ready-made solutions, such as design patterns, based on the experience of specialists from different fields are required. In order to reduce the time, cost, and risk of developing a new design, MAS developers shall understand these prepared solutions in an easy way. Also, as the MAS were not often implemented in the industry due to the limited

All author are members of the technical committee FA 5.15 "Agent systems" of the Society Measurement and Automatic Control (GMA) within the Society of German Engineers (VDI) and German Electrical Engineers (VDE), and National Member Organizations (NMO) of IFAC and the IEEE-IES Technical Committee on Industrial Agents (TC-IA)

✉ Luis Alberto Cruz Salazar
luis.cruz@tum.de

[1] Institute of Automation and Information System, Technical University Munich, Garching, 85748 Munich, Germany

[2] Universidad Antonio Nariño, Bogotá, Colombia

[3] Institute of Ergonomics, Manufacturing Systems and Automation, Otto-von-Guericke University, Magdeburg, Germany

understanding [8, 9], to increase their acceptation, the prepared patterns can help.

Design patterns provide a means of identification and consideration of broader success aspects in particular problems [10]. They are described as an abstraction of the system developing process, since they are based on already working solutions. The abstraction focuses on the essential aspects captured by the pattern [11]. Therefore, design patterns due to the properties of the agents [9] provide increased connectivity between control levels of the automation pyramid, to ease the migration to the CPPS. As the MAS have a flexible programmable and dynamic architecture [8], patterns could provide the support of properties such as reusability, flexibility, adaptability, and modularity, which will satisfy the requirements of CPPS [3, 12], following the future needs of the automation [13, 14].

1.1 Contribution to the industrial automation

The idea of this paper is to provide engineers and programmers with existing patterns based on the MAS structures, which will help improve their design efforts and therefore increasing the efficiency of the manufacturing process control, and decreasing the development design cost (reusability [15]). Based on prior experiences, design patterns are delivered to address MAS into different levels of the automation hierarchy (low/high level) with real-time and non-real-time control systems, mainly industrial controllers' technologies, e.g., Programmable Logic Controller (PLC). Furthermore, design patterns enable MAS developers to easily have the same understanding of the solution system's design [15–17]. Thus, ready-made templates are designed to simplify the comparison of MAS alternative solutions [17]. Other benefits of this contribution are the following:

1. A well-discussed survey/summary at least in agents working group of the German IFAC NMO GMA FA 5.15 is presented.
2. Mapping of analyzed MAS functional requirements to sub-agents' patterns was provided.
3. Proposed sub-agent patterns for MAS technology in industrial environments are extendible; further extended designs are possible for more use cases.
4. The proposed sub-agent patterns will reduce the time and cost efforts, as the proposed pattern is a "ready-made" solution.
5. The identified design patterns are the basis for the development of agent-based CPPS and for their structural representation. However, the contribution does not consider an explicit MAS architecture (with final requirements) for the application of an individual CPPS or system domain.
6. The proposed sub-agent patterns support integration and development of MAS for different automation levels based on ISA 95 and RAM I4.0.

Existing control architectures are frequently based on the developer's experience from every single domain, but MAS developers are often unaware of the benefits of design patterns. Therefore, this manuscript provides 13 criteria (see Section 3), as a key result from a preliminary classification of different MAS and their evaluation in various domain solutions, as shown in [18]. Thereby, this work aims at finding and validating the criteria required for pattern creation enabling the migration to CPPS:

- Using relevant requirements of the CPPS from Ribeiro and Hochwallner [12].
- Aligned with smart agent proposals for industry from Leitao et al. [9].
- Aligned with the RAMI 4.0 model [1].

Based on the essential properties of MAS [8], this paper gives a deep classification and analysis of the collected MAS with the derived criteria, which will help for further pattern development, and suggestion of agent-based CPPS architecture.

1.2 Research questions and hypotheses

Despite the fact that the MAS application has not been popular in industry, nowadays, its admission is acceptable [8, 9] and the MAS applicability is more extensive than over the last years [8, 9, 18]. Moreover, the existence of patterns will ease the perception and comprehension for the MAS developers. Design patterns based on MAS for manufacturing would enable rapid application in industry [17]. Besides, the agent-based architectural solutions usually possess the characteristic of "plug and produce" use and are applicable in many domains after simple parameter adjustment [19]. In addition, they are suitable for different control layers regarding the automation pyramid that will allow the creation of versatile approaches regarding CPPS [9]. It is a big challenge to characterize a universal design that applies to the logical architecture and to software abstractions. Therefore, in this work, the target of design patterns concerns a functional system level as a MAS logical architecture that does not deal with software level abstractions. However, relevant information to logical architecture and software could be addressed by the final design proposal. Consequently, this work addresses four general research questions (RQ1-RQ4) connected to eight research hypotheses (RH1.1-RH4.2), as shown in Table 1.

The manuscript is structured as follows: Section 2 reviews the state-of-the-art of the collected MAS patterns for manufacturing systems. Section 3 introduces the discussion about the classification criteria to compare MAS approaches for the further elaboration of the patterns. Section 4 evaluates the four different MAS approaches applying the 13 classification criteria. In Section 5, common functional requirements of MAS patterns are presented. Finally, Section 6 represents the

Table 1 Research questions and related hypotheses

Research questions	Hypotheses	Proof
RQ1—How are the MAS patterns for CPPS depicted and what criteria are used to describe them?	RH1.1—classification criteria for MAS approaches delivers valid and decidable information for their evaluation	D
	RH1.2—MAS approaches for CPPS can be classified and identified with similar design pattern's terms (e.g., names, functionalities, etc.)	E
RQ2—For which domains of CPPS are the MAS patterns designed and applicable?	RH2.1—MAS approaches have application in diverse domains with different goals and benefits (e.g., flexibility, adaptability, etc.)	D
	RH2.2—CPPS are applicable in every domain in appliance with the real-time requirements of MAS approaches	E
RQ3—Which MAS design patterns for CPPS are reusable?	RH3.1—there are reusable MAS patterns with functional and non-functional requirements for CPPS design	E
	RH3.2—MAS components follow specific sub-agents, which have particular aims and are reusable for CPPS design	E
RQ4—How do the MAS design patterns develop into a CPPS aligned with RAMI 4.0?	RH4.1—it is possible to harmonize different MAS approaches to obtain a simple CPPS architecture aligned with RAMI 4.0	E
	RH4.2—MAS patterns provide Industry 4.0 component's properties and specific information to its administration shell	E

D, insights gained from documents and feedback of MAS patterns' authors; E, insights gained by the validation analysis of this manuscript's authors

agent-based CPPS architecture aligned with RAMI 4.0 model. This work is summarized within the conclusion in Section 7.

2 Related work

As described in the previous section, this paper presents selected MAS ready-made solutions to give support to developers, so they can easily produce new control systems [15, 17]. It helps to avoid the common design mistakes during the system's development phase [8]. The new solution should support among others reusability, flexibility, modularity, as defined in [8, 20]. The MAS approaches are classified in order to facilitate the migration from the conventional automation systems to the CPPS. Therefore, authors use a template that consists of a list of classification criteria validated by experts in the German community FA 5.15 (see Section 3). Because all approaches were created to be used in different domains and different layers of the automation pyramid, a notable part of them are concentrated to provide the flexibility or changeability (FC) of the system. Others concentrate on other features such as reliability (RL), adaptability or agility (AA), reconfigurability (RC), and dependability (DP). All of these characteristics are enlisted and described in Table 2, according to [8, 12, 20, 21].

The comparison of existing systems architectures and their focuses—regarding specific CPPS and RAMI 4.0 requirements—are collected and presented in Table 3; as shown, almost all architectures concentrate on providing flexibility for the automation systems. Table 3 also compares the structure for CPPS approaches regarding the control distribution's classes from Trentesaux [22]. The CPPS structures can be classified between Classes 0 and III according to their control and decision-making mechanism, as the following list:

- Class 0. Centralized control systems (e.g., CIMOSA [23])
- Class I. Fully hierarchical control system (e.g., acquire, recognize, and cluster architecture for SoA or ARC-SoA [24])
- Class II. Semi-heterarchical control system (e.g., ADACOR [25] architecture)
- Class III. Fully heterarchical control system (e.g., D-MAS architecture [26])

Any MAS may be used regardless of its class, since all MAS were proposed for different use cases, e.g., for the industry, smart grids, etc. [9] and be applied for the purpose of satisfying the CPPS and RAMI 4.0 requirements (see Section 2.1). In fact, these approaches apply methods and techniques such as the Industrial Internet of Things (IIoT), decision-making mechanisms, semantic models, process synthesis, and optimization [18].

The German committee FA 5.15 initiated the idea of this work, where 20 different agent-based approaches were collected inside of the group discussions. Additionally, each FA-author had a direct access to the evaluation of the pattern and gave feedback. Therefore, authors do not claim that the collected list of MAS is holistically completed, as it focuses just on German applications for production systems. However, the list of MAS can be further extended to consider other applications. At the moment, it covers the different fields of application from software and manufacturing domains: smart grids, logistics, geography and image process applications, etc.

Wannagat [27] presents a MAS implementation concept for handling a faulty sensor or an actuator for automation systems. This architecture provides the flexibility, reliability, and reconfigurability for the production systems. The work of

Table 2 Description of characteristics for CPPS [8, 12, 20, 21]

Feature	Description
Flexibility/changeability [12, 21] (FC)	It is often the grade to which a product or system can be used with effectiveness, efficiency, freedom from risk, and satisfaction in contexts beyond those initially specified in the requirements
Reliability [21] (RL)	A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time (four attributes: maturity, fault tolerance, recoverability, reliability compliance)
Reconfigurability [20] (RC)	A system designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or regulatory requirements
Adaptability/agility [8, 12] (AA)	The capability of surviving and prospering in a competitive environment of continuous and unpredictable change by reacting quickly and effectively to changing markets, driven by customer-designed products and services
Dependability [20] (DP)	The set of independent production events (ES) that completely defines the available production processes in a production system. Their number could be given by an equation in [20]

Schütz [28], based on Wannagat, proposes a heterarchical approach about a PLC-Agent-System with individual knowledge-based agents. Main features of the approach are flexibility and reconfigurability. The MAS of Ulewicz [7] represents an abstract architecture concept for plants inside industrial automation. Its focus is on providing flexibility and reliability and it has been applied on real industrial context, validated by industrial experts. Legat [15] proposed an agent-based architecture for handling unforeseen failures; the main features of this approach are flexibility and reconfigurability [29].

The approach of Rehberger [19] is designed for achieving both flexibility and availability during run-time (for coping with unknown product recipes and breakdowns of sub

Table 3 Related work focusing on providing benefits and regarding CPPS and I4.0 requirements

Author(s)	Characteristic benefit					Classification		CPPS requirement					RAMI 4.0 requirement				
	FC	RL	RC	AA	DP	Scope	Class	Req. 1.1	Req. 1.2	Req. 1.3	Req. 1.4	Req. 1.5	Req. 2.1	Req. 2.2	Req. 2.3	Req. 2.4	Req. 2.5
ADACOR (Leitão and Restivo, 2006) [25]	●		●	●		HMS architecture	II	+	++	++	+	+	–	–	++	–	–
Andrén *et al.*, 2013 [30]	●					MAS for smart grid	III	+	+	++	+	–	+	–	–	–	–
Cruz S. *et al.*, 2018 [31]	●		●	●		CPPS architecture	II	++	++	++	++	++	–	–	++	–	–
Fischer *et al.*, 2018 [32]	●				●	MFS agent-based	III	++	+	+	+	–	–	–	+	–	–
Karnouskos and De Holanda, 2009 [33]			●			MAS for smart grid	III	+	+	++	++	–	–	–	++	–	–
Leitão *et al.*, 2016 [9]	●	●	●	●		MAS for industry	III	++	++	++	++	+	–	–	–	–	–
Lüder *et al.*, 2017 [16]	●	●				MAS for industry	III	+	+	++	–	–	–	–	++	–	–
Lüder *et al.*, 2017 [34]	●					MAS for industry	III	+	+	++	++	–	+	+	+	+	+
Nieße, A., 2015 [35]	●					MAS for smart grid	III	+	+	++	++	++	+	+	++	+	+
PROSA (Brussel *et al.*, 1998) [36]	●		●	●		HMS architecture	II	+	++	++	+	+	–	–	++	–	–
Regulin *et al.*, 2016 [37]	●					MFS agent-based	III	+	++	++	+	–	–	–	–	–	–
Rehberger *et al.*, 2017 [19]	●				●	MAS for industry	III	+	+	+	++	–	–	–	+	–	–
Ribeiro and Hochwallner, 2018 [12]	●	●	●	●		CPPS architecture	III	++	++	++	++	+	+	+	++	+	–
Ryashentseva, 2016 [38]			●	●		MAS for industry	III	+	+	++	++	+	–	–	++	+	-
Schütz *et al.*, 2011 [28]	●		●			MAS for industry	III	++	++	++	++	+	–	–	+	–	–
Theiss and Kabitzsch, 2017 [39]	●					MAS for industry	III	+	+	++	++	+	–	–	+	–	–
Ulewicz *et al.* [7]	●	●				MAS for industry	III	+	++	++	+	–	–	–	+	–	–
Vogel-Heuser *et al.*, 2014 [40]	●					MAS for industry	III	+	++	++	+	+	–	–	+	–	–
Wannagat, 2010 [27]	●	●	●			CPPS architecture	III	+	+	++	+	–	–	–	+	–	–

Notation: ● Applicable; ++ High; + Medium; – Low

modules) as well as adaptability during engineering (MAS with exchange-/adaptable knowledge base in form of a discrete and continuous plant model). The works of Fischer [32] and Regulin et al. [37] present MAS control approaches enhancing the flexibility and reconfigurability of material flow systems (MFS). All of them focus on the flexibility and dependability features. Hoffmann's approach [41] is proposed to reach the customized products and production configuration providing dynamic reconfiguration, production fault compensation, and predictive maintenance. The next approach of Pech [42] enables flexibility and adaptability for the user interaction and query formulation for information retrieval.

The approach of Ryashentseva [38] presents a supervisor-based and self-adapting architecture with the focus to realize reconfigurability and adaptability of the production system. Lüder et al. [16] propose the resource allocation and the resource access design patterns for manufacturing systems. He focuses on reliability, adaptability, and flexibility properties. Based on this proposal of the ready-made pattern for resource processing, the possibility of pattern elaboration for the CPPS is suggested in this paper [16]. Its literature review was based on a Google Scholar search (exploiting the terms "manufacturing system," "agent," and "control") and have limited the works published the last 10 years after 2006 [8, 16]. Summarizing the MAS for manufacturing, a variety of agent-based methodologies exist for the model-based development of software for manufacturing [32, 43]. Surrounded by their last 100 most recent results, just 19 papers have been selected, since other papers were not in the production system control field with architecture representation. Other surveys about the design patterns for distributed automation have been analyzed in [11, 17].

Meanwhile, diverse agent-based approaches intending to provide manufacturing control for CPPS, based on Agent Oriented Software Engineering (AOSE), have been presented. The most relevant point of views from the authors are listed and compared in [4], such as Gaia, MaSe, and other methodologies. Holonic Manufacturing Systems (HMS) are also considered for distributed control systems, in [25, 36, 44, 45]. Finally, MAS approaches also enable to boost energy efficiency via smart grids, as shown in [30, 33, 46].

This section concludes that control decisions and the overall intended behavior of different MAS approaches listed are described. In addition, their integration in the automation pyramid has been specified, and the different control decisions are outlined. Each industrial agent mapped to a control pyramid layer designated, to which the agent belongs and has a control decision and specific features (e.g., flexibility). Based on this research and preliminary work of Lüder et al. [16], the next chapter presents the development of the classification criteria in order to evaluate the collected MAS patterns. Further, in the paper, it will be used to evaluate four different approaches [16, 27, 38].

2.1 Requirements regarding CPPS and RAMI 4.0

Adapted from Ribeiro and Hochwallner [12] concepts, requirements are understood as explicit conditions which must be represented by system in order to fulfill a specification, or a standard.

For an agent-based CPPS, aligned from [4, 12, 31, 40], there are five key requirements: the application independence (Req1.1), meaning that a MAS and its protocols and messages should be independent of a specific application. The level independence (Req1.2) referring that all levels of automation for ISA 95 (see Section 5.6) are available depending on the scenarios in which the CPPS will be applied. Platform independent implementation (Req1.3) implying that modules are effortlessly integrated with independent implementation (open technologies). Robustness against errors (Req1.4) meaning MAS must react to faults and dynamic conditions in an appropriate way, i.e., it must be robust against unforeseen. Decentralization (Req1.5) means that MAS have to deal with temporary network connection loss and critical data should be distributed between multiple nodes.

Regarding the RAMI 4.0 model (see Section 6), there are other five crucial requirements for the I4.0 components concept [47]. The sub-models (R2.1) shall support various engineering disciplines. The system boundary (R2.2) implies that a sub-model describes the relationships between the RAMI 4.0 layers. The nestability principle (R2.3) for the specific engineering discipline shall have its own organizing principles for the relevant resources (assets in hierarchy dimensions). The virtual representation (R2.4), an administration shell, can denote a digital active with their parts. Finally, the functional properties (R2.5) require that the manifest has an externally accessible set of meta-models describing its functional and non-functional properties.

3 Classification criteria for MAS patterns (RQ1)

In this section, the classification criteria for the MAS design patterns are described. It is based on preliminary considerations that were briefly discussed in the previous section. As specified in the state of the art, there are many different design solutions for the control of the manufacturing systems with MAS architectures. However, the application and thereby the evaluation of the design pattern criteria will be shown in the next section that is based on only four MAS field approaches. Accordingly, the adapted SLR method of outlining and obtaining design patterns [10], usually applied in the area of software for mechatronics systems, is presented here. To extend the work done by Lüder et al. [16], a bottom-up approach is proposed in this paper. The adaptation of the design pattern is based on distributed automation systems [17] and

developed by the classification criteria from Lüder et al. [16] and Leitao et al. [8] and Ribeiro and Hochwallner [12]. Finally, thirteen criteria were proposed (cp. Table 4) to classify MAS architectures' patterns. These patterns were introduced and evaluated by the members of the German FA 5.15 working group.

The industrial automation field should support the developers to create new functionalities based on different common parts and experiences, to fulfill requirements from the previous section (see Section 2.1). Consequently, different kinds of design patterns were to support engineers in solving the respective problems and obtain solutions with common methods.

One of the main contributions of this paper is the compilation of the criteria for the MAS design pattern template (cp. Table 4). First, the template introduces the pattern category, pattern type, pattern name, pattern description, context, solution, and implementation used for the distributed systems pattern in [17]. Second, the template adds MAS-architecture, knowledge base and processing, real-time properties, dependability, learning, MAS-autonomy, and others.

In addition to the criteria of the MAS pattern (Table 4), a classification of each sub-agent pattern of the MAS architecture is developed. These supplementary criteria describe in details the features of each sub-agent in order to better understand MAS architecture and better compare their identifying patterns. Then, Table 5 extends the patterns description criterion from Table 4, according to the following items: sub-agent name, main functionality, ISA 95 level (automation level), real-time capability, source type info, communication base, key properties, and related work.

For this proposal paper, an approach is demarcated as a set of architectures, methodologies, or standards, which follow a common scheme. In the case of architectures, these are considered single structures of static system model. The aspect MAS architecture describes the associations between different types of agents (or sub-agents) and includes the MAS set-up [31]. In addition, most of the MAS are not patented by their authors and usually do not have the practical data to carry out their implementation (i.e., clear methodology). In this case, MAS methodology should determine the best steps to follow in order to improve reusability in development and quality systems (usually used for software engineering as AOSE [4]). A good methodology should indicate how the MAS would satisfy all its process in a systematic, predictable, and repeatable way. In the end,

Table 4 Criteria to classify MAS architectures/patterns

Criteria	Descriptions	Examples options
Pattern category	Favorable function patterns: system properties that can be realized by employing MAS, i.e., increased flexibility and adaptability	Flexibility pattern, adaptability pattern, reliability pattern, reconfigurability pattern
Pattern type	Name of the pattern type: technology-independent task of the MAS (categorized)	Fault-tolerant sensors
Pattern name	Name of the MAS pattern	Soft sensor
Pattern description	Description of the logic structure (which components/agents does the pattern contain?)	MAS with 4 sub-agents, which enable identifying faulty sensors and automatically replacing them with soft sensors based on models
Context/area of application	Application context of the pattern	Various domains, e.g., logistics, process engineering
MAS-architecture	Approach for realization of the agents' behavior	Reactive/cognitive/hybrid
Solution	Graphical depiction of the MAS-Architecture	Depiction of the MAS' components (notation class diagram)
Knowledge base and processing	How is the knowledge stored? Models, rules. How is the knowledge processed? With which methods?	Model from engineering, ontology, meta model data structure. Inference mechanisms for ontologies
Learning/knowledge acquisition	Methods and techniques for learning abilities/knowledge base	Machine learning, neuronal networks
Implementation	Technological realization of the MAS (platform, languages)	Model: SysML, programming language IEC 61131-3
Real-time properties	Timeliness and concurrency requirements	Usage replacement sensor < 2 PLC-cycles < 40 ms
Dependability	Requirements towards reliability, availability, maintainability, security or safety	Soft sensor can replace sensor with a reliability of *x*%
MAS-autonomy	Autonomy/independence in decision making	Replacement of sensor not autonomously, since number of replaceable sensors is limited
Others	Additional author's comments (remarks, clarifications, etc.)	

Table 5 Criteria to classify the patterns description (sub-agents)

Criteria	Descriptions	Examples options
Sub-agent name	The name of the sub-agent (or acronym)	Coordination agent, resource agent
Main functionality	The main functionality of the sub-agent with text descriptions	Communication entity among other sub-agents
ISA 95 level	Action's automation levels	L2, L1–L3. See Section 5.6
Real-time capability	Requires or not hard real-time execution for its functionality	Yes, No
Source type info.	Sub-agent's info. source (data/hardware/both)	Data/hardware/both
Communication base	Communication-based concept/theory/protocol (direct or indirect)	Control net protocol—CNP, ACL, FIPA specification
Key properties	Social primary properties or abilities	Autonomous: control over its behavior
Related work	Has a preliminary design?	Author name, standard

an ideal final stage of the MAS design phase would be its standardization (creation of norm). International institutions such as ISO, ASME, IEC, IEEE, and others might endorse both, MAS architectures and methodologies, as a current standard for smart manufacturing as supported by experts [3, 4, 21].

The evaluation of the criteria proposed in this section is introduced as follows based on MAS approaches for manufacturing.

4 Evaluation of four selected MAS architectures applying criteria classification (RQ2)

This section presents the application of the criteria for MAS design patterns discussed in the previous section (cp. Tables 4 and 5).

The general patterns' analysis starts from the lowest layer of the traditional automation pyramid, applied (see Section 5.6) in the logistics domain, manufacturing execution systems (MES). The first architecture of Wannagat [27] concentrates on the field level control and presents a MAS architecture for hard real-time and dependability, applied to PLC controllers (the most popular in industrial environments). Based on that, many other authors continue to build their MAS architectures on it (Folmer [48], Schütz [28], Rehberger [19], and Ulewicz [7]). Second author Fischer [32] also applied MAS for hard real-time including control level to put adaptability, flexibility, and dependability attributes into MFS. In this case, new components based on metamodeling can be added. The reflection of this idea can be found in other researches such as Priego et al. [49] and Hanisch et al. [50]. A third work by Ryashentseva [38] represents a MAS approach focused on the real-time capabilities for self-reconfigurability of production plants, and has implemented supervisory control theory (SCT) increasing self-adaptability. This approach covers the middle and low levels of the automation pyramid from the cooperation with legacy systems on field control level and communication with MES. Finally, Lüder et al. [16] propose the fourth MAS approach, which includes design patterns considering different levels of manufacturing, even upper-level MES.

In this paper, the authors choose the following three basic terms in order to facilitate the discussion of the following MAS approaches: i) From the VDI standard 2653 sheet 1, a sub-agent is an encapsulated entity (of software, hardware, or both) with specific goals inside the whole MAS architecture. The sub-agent endeavors to reach his goals with autonomy and by interacting with its environment and among other sub-agents [18]. ii) From Ribeiro and Hochwallner [12], a module is "tightly coupled within and loosely connected to the rest of the system." Hence, a module is a MAS software component that does not have dynamic characteristics and intelligent properties like a sub-agent (e.g., autonomy, message interactions, and cooperativeness). However, it can determine specific functions, methods, or routines which are often part of or used by sub-agents (e.g., control module [28]). iii) Adapted from the ISO/IEC 2382-1, a database (DB) is a collection of data ordered giving to a conceptual structure relating the features of the info and the associations among their corresponding entities, supporting one or more request areas and accessible in various ways. Mostly, a DB in MAS architectures is an organized collection of data for the module's interactions. It is stored and accessed electronically as the "yellow pages" for services exposed to other sub-agents [7, 32].

Below, the following abbreviations will be used in the corresponding figures: resource agent (RA), coordination process (CP), knowledge base (KB), and communication interface (CI).

4.1 Design pattern for the resource agent

The RA architecture presented in [27] provides an agent-based interface for technical components in the field control level (see Fig. 1).

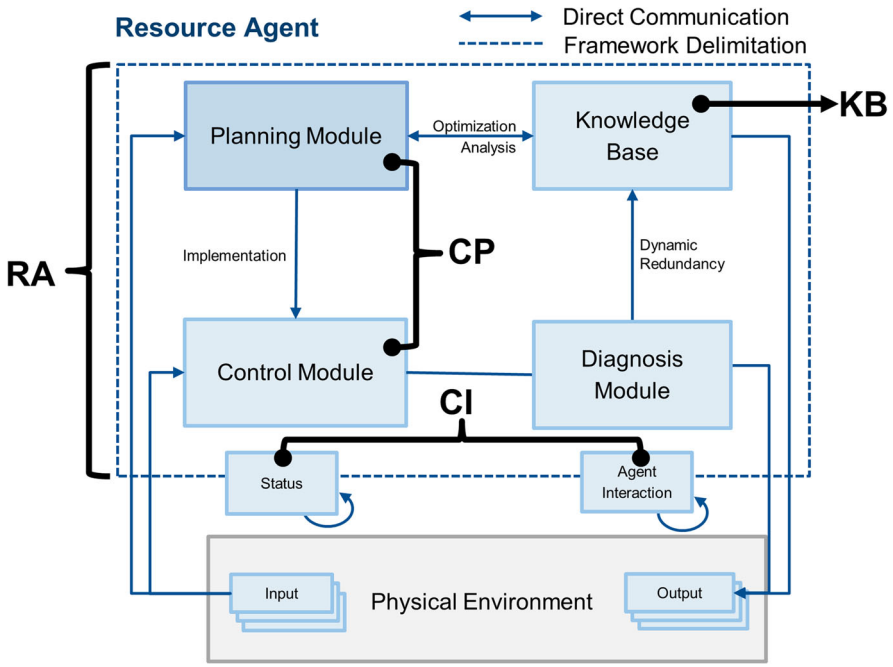

Fig. 1 Identifications of RA pattern in Wannagat's architecture [27]

A RA has four main modules with specific characteristics. One of these is the Control Module that is connected with the I/Os (sensors and actuators signals) of the plant hardware. From this module, the data of the control variables are sent to the actuators and the information from the sensors is measured. A Diagnosis Module detects failures within the sub-agent's status, which identifies the existing situation based on the sensor data (signals measurements or other sub-agents' messages from Agent Interaction). Table 6 shows the design pattern for the RA in production plants [19, 28].

Incoming sensor measurements are processed in order to detect sensor failures. In this case, the Diagnosis module connects to the Knowledge base module, and it specifies the system model to the corresponding technical device. Afterwards, each sub-agent reviews the parameters of the technical specific system from the MAS architecture. It also preserves the processes inside the explicit limits. Redundant sensor measurements are calculated using analytics. These "virtual" data from I/Os provide for the fault state based on the Diagnosis module (a result of the compensation failures). Finally, the Planning module contains local goals and negotiates time schedules for message exchange with other sub-agents. Additional three basic entities from FIPA standard have discovery dedications: a sub-agent called Agent Management System (AMS), the Message Transport System (MTS), and the Directory Facilitator (DF). Robustness against errors should be enhanced by using direct connections between sub-agents. The AMS allows the bidirectional mapping between IP-addresses and sub-agents' identifications. In addition, the RA contains communication interfaces to update error status through message interactions, delivered to sub-agents in higher heterarchy levels (e.g., AMS).

Table 7 shows the list of identified sub-agents for the MAS based on the RA pattern in [27].

4.2 Design pattern for plug and produce of MFS

Fischer presents a MAS architecture in [32] that provides basic entities (sub-agents and modules) for the coordination of an entire MFS. Figure 2 illustrates the general implementation scenario of the approach based on Fischer's static graphic models. Table 8 shows the design pattern for MFS according to Fischer's MAS architecture [32].

This MAS approach is self-motivated in only one of the MFS's modules. It contains the sub-agent called AMS, the DF, and the MTS, all of them from FIPA standard. These are used in the same manner as presented by Wannagat et al. [19, 28] in Section 4.1. Fischer's pattern focuses on the MFS, but both patterns (Fischer-Wannagat) are used for the real-time intelligent conveyors' rerouting. AMS comprised of methods for registering or deregistering a module from/to the MAS. The DF allocates orders to the agreeing module and sub-agents by allocating the equivalent principles in the

Table 6 Reconfiguration of faulty devices MAS (Wannagat [27]), according to the introduced classification

Criteria	Descriptions
Pattern category	Flexibility, reliability, and reconfigurability
Pattern type	MAS implementation concept for faulty sensor or actuator identification in automation systems
Pattern name	Agent@PLC
Pattern description	Main part is the resource agent (RA), and three more sub-agents. See Table 7.
Context/area of application	Solution adapts the actual values with appropriate changes instead of using worst-case values in predefined replacements. The process operation time will be longer under the prerequisite that the process operation is still beneficial with reduced precision, speed, etc. This leads to higher availability in different context and domains
MAS-architecture	Hybrid-pattern replaces faulty sensor value with virtual one, calculated based on other sensor's and model's information (MDE based calculation). The faulty sensor has to be identified. The possible decrease in correctness is identified; the virtual sensor is used until the real one is available again
Solution	See Fig. 1
Knowledge base and processing	Object-oriented and agent-based concepts (OOP) and Systems Modeling Language (SysML)
Learning/knowledge acquisition	Possible, filtering wrong values
Implementation	IEC 61131-3
Real-time properties	Hard real-time capable thanks to the resource agents' behavior into physical plant devices such as PLC
Dependability	Higher degree of dependability of the MAS–failures of plant components detected by virtual sensors
MAS autonomy	It is half-half dependable–individual control agents represent and control technical plant units (e.g., machines) to allocate their services encapsulated [28]
Others	Application uses three different type virtual sensors

Communication Agent. The Order Agent executes the received module's demands. It manages whether the module could fulfill the demanded order and, if that is possible, it supplies the order into a list containing scheduled re-

Table 7 Identification of sub-agents patterns for the resource agent as used by Wannagat [27]

Sub-agent name	Main functionality	ISA 95 level	Real-time	Source type info.	Communication base	Key properties	Rel. work
Resource agent	Deliberates and reasons about the demanding task to answer to the PM with an offer	L0–L2	Yes	Hardware	Fieldbus IEC 61158 Industrial Ethernet, EtherCAT	Autonomy, reactiveness	[7, 19, 28, 32]
Agent interaction	Allows remuneration between software objects at runtime depending on the current situation set	L0–L2	Yes	Data/hardware		Cooperativeness	
Communication agent	Coordinates the message-based communication between the agents on a single PLC or net PLC	L0–L2	No	Data		Cooperativeness, reactiveness	
Agent management system	Contains methods for de/registering module to/from the system	L2	Yes	Data	FIPA specification	Cooperativeness, proactiveness	
Process agent	Supervises and handles global tasks that concern the whole system (e.g., check global errors)	L0–L2	No	Data		Cooperativeness, proactiveness	

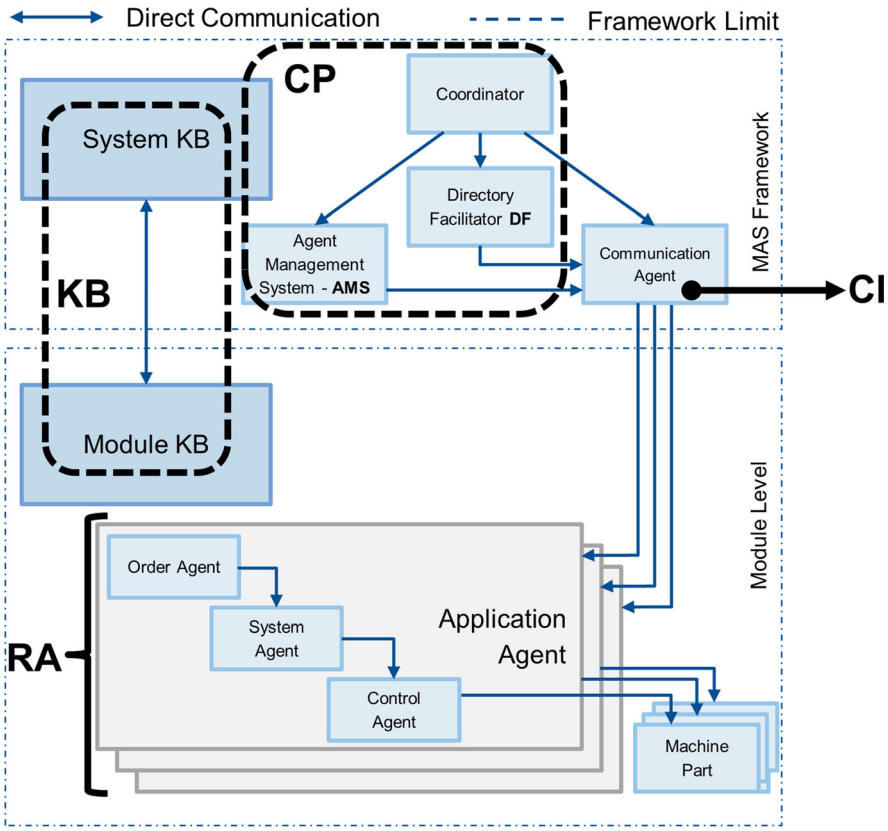

Fig. 2 Identification of MFS patterns in Fischer's architecture [32]

quests. There is also a System Agent with two main responsibilities. First, it provides the module report checked in the module's knowledge base to the Coordinator. Secondly, it processes from Order Agent requests. The Coordinator is the highest authority of the system that initiates the registration or deregistration of modules and

Table 8 Design pattern for MFS (Fischer [32]), according to the introduced classification

Criteria	Descriptions
Pattern category	Reconfigurability and flexibility (changeability) pattern
Pattern type	Agent control approach enhancing the flexibility and reconfigurability of MFS
Pattern name	Plug and produce of MFS
Pattern description	There are five sub-agents with FIPA specifications. See Table 9
Context/area of application	Logistic domain
MAS-architecture	Reactive
Solution	See Fig. 2
Knowledge base and processing	1) MAS system: agent is implemented with the module's control code on an individual PLC. 2) Coordinate system approach: use of two different types the module and global coordinates system
Learning/knowledge acquisition	No
Implementation	Sub-agents: FIPA and ADS (automation device specification) protocols, implemented in IEC 61131-3. Low level: IEC 61131-3, object-oriented extension
Real-time properties	Yes, since PLCs are hard real-time systems they have to ensure constant cycle times to read/write/process all the MFS signals
Dependability	No
MAS autonomy	Half-half autonomy thanks to acting individual agents, which are capable of communicating to give a task, but, a coordinator connects the MAS to superordinate levels
Others	Re-routing considers not only transportation abilities but also manipulations, which need to be performed in order to fulfill an order correctly

Table 9 Identification of sub-agents patterns for MFS as used by Fischer [32]

Sub-agent name	Main functionality	ISA 95 level	Real-time	Source type info.	Communication base	Key properties	Rel. work
Agent management system	Contains methods for de/registering module to/from the system	L2	Yes	Data	FIPA specification	Cooperativeness, proactiveness	FIPA, [7, 37]
Coordinator agent	Initiates add or removal modules (highest authority)	L2	No	Data	Automation device specification (ADS) protocol	Autonomy, proactiveness	
Communication agent	Transfers and receives information of agents via ADS and Ethernet communication	L0–L2	Yes	Data/hardware		Cooperativeness, reactiveness	
Order agent	Manages the incoming module sub-orders	L0–L1	Yes	Hardware		Cooperativeness	
System agent	Module description and processes sub-orders provides	L2	Yes	Hardware		Cooperativeness	
Control agent	Communicates with the control POUs to get data/starts actuators connection to the hardware	L0–L1		Hardware		Reactiveness	

the recalculation of the system KB at startup or when the system configuration changes. Table 9 shows the list of sub-agents for the MAS architectures for MFS design pattern [32].

A Control Agent directly communicates with the field control level through Program Organization Units (POUs), and represents the lowest entity in the MAS heterarchy [15]. A POU gets information or module requirements for founding

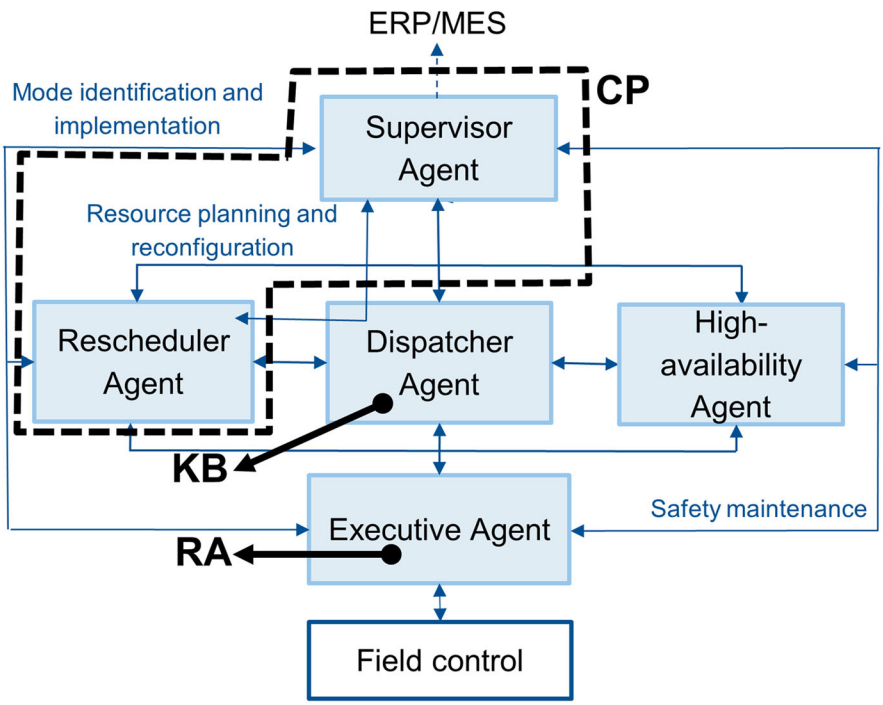

Fig. 3 Patterns for self*-control MAS in Ryashentseva's architecture [38]

Table 10 Agents pattern for self*-control architecture (Ryashentseva [38]), according to the introduced classification

Criteria	Descriptions
Pattern category	Flexibility pattern
Pattern type	Supervisor-based self-adapting architecture
Pattern name	Agents and SCT based self*-control architecture for production systems
Pattern description	Pattern consists of five sub-agents. See Table 11
Context/area of application	Applicable in different context and domains
MAS-architecture	Hybrid (the high-availability agent reacts on the failures of the other sub-agents proactively, while other agents operate reactively)
Solution	See Fig. 3
Knowledge base and processing	Meta-model and ontology/inference machine and meta-model
Learning/knowledge acquisition	Learning is possible: fuzzy-model is located in supervisor agent and can be learned through the experience of rescheduler agent and executive agent
Implementation	Modeled by SysML, implement with FIPA standard
Real-time properties	Hard real-time capabilities since the executive agent exchanges data with sensor, actuator and other hardware agents (e.g., PLC); Other sub-agents in real-time are working on a request, selecting a suitable resources
Dependability	MAS is valid to provide higher reliability, reconfigurability, security and safety
MAS autonomy	Knowledge base is edited autonomously; reconfiguration is not autonomously
Others	Domain specific knowledge and model are editable during run-time

the module's KB. Individually, one of the sub-agents and modules recorded above are implemented as an individual Function Block (FB) into PLC software and a device. The module's capabilities and agents are bounded by techniques given to the equivalent FB following object-oriented extensions with IEC 61131-3 languages [15].

Table 11 Identification of sub-agents patterns for self*-control as used by Ryashentseva [38]

Sub-agent name	Main functionality	ISA 95 level	Real-time	Source type info.	Communication base	Key properties	Rel. work
Executive agent, EA	Exchanges data with sensor, actuators and other hardware agents (e.g., PLC), safety maintenance (with SA)	L0–L1	Yes	Data/hardware	FIPA specification	Cooperativeness	[7, 16, 32]
Supervisor agent, SA	Communicates ERP/MES and peripheral systems, process optimization, decision making, resources' plans (with DA)	L2–L4	Yes	Data/hardware	FIPA specification	Cooperativeness	[32, 51, 52]
Dispatcher agent, DA	Dispatches the system (with HAA); knowledge base (resources, services, modes); provides the control rules, plan services and resources (with the EA and SA)	L1–L2	Yes	Data/hardware	FIPA specification	Cooperativeness, reactiveness	[27, 32]
High availability agent, HAA	Provides safety maintenance (with EA); fault tolerance: back-up controller; security: leakage protection; safety: system work check; dispatcher of the crossed tasks (DA)	L0–L1	Yes	Data/hardware	FIPA specification	Cooperativeness, autonomy	[16]
Rescheduler agent	Implementation resources configuration (with SA/DA); KB tuning (DA); mode identification together with EA and SA	L1–L2	Yes	Data/hardware	FIPA specification	Cooperativeness, autonomy	[16, 27, 51, 53]

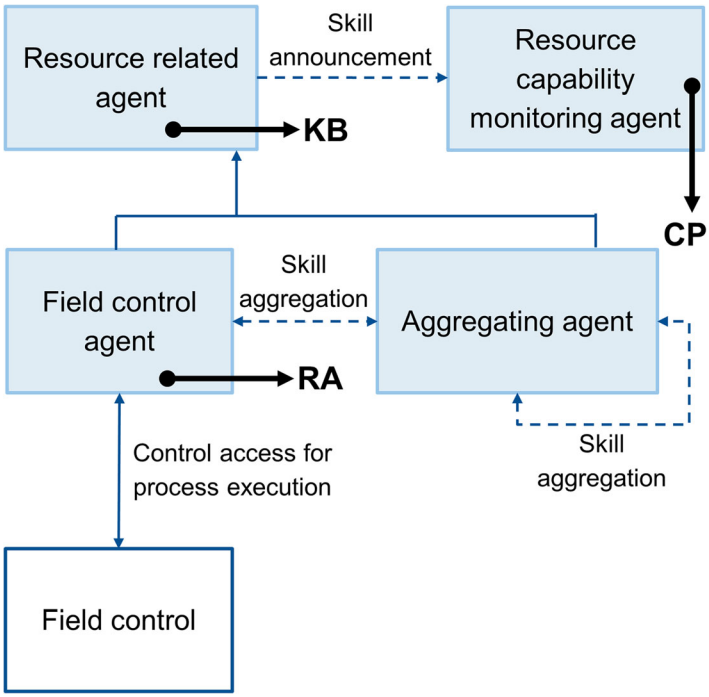

Fig. 4 Resource access design pattern from Lüder et al. [16]

4.3 Design pattern for agents with self*-control

The MAS architecture with self-control from Ryashentseva [38] consists of five logically and physically separated sub-agents cooperatively performing different tasks (see Fig. 3). The tasks of each sub-agent in this pattern are evenly allocated between them, describing all the necessary functions and properties of the cyber physical control system, e.g., from the field level, where

Table 12 Design pattern for resource access (Lüder [16]), according to the introduced classification

Criteria	Descriptions
Pattern category	Flexibility, adaptability, and agility pattern
Pattern type	Resource access pattern enables coordination of resources and decoupling of control layers. A structure of interacting resource related agents is applied enabling processing capability aggregation
Pattern name	Resource access design pattern
Pattern description	Two mandatory types of sub-agents: the resource related agent and order agent. See Table 13
Context/area of application	For any production system control and its architecture representation
MAS-architecture	Hybrid
Solution	See Fig. 4
Knowledge base and processing	Ontology/processing: sub-agents types will execute a negotiation process based on a contract net protocol
Learning/knowledge acquisition	No
Implementation	Implementation of production process related capabilities and their control; FIPA specifications
Real-time properties	Yes, hard real-time capability through to the resource related agent type
Dependability	Medium maintainability since the product type information agent stores detailed info about the product ordered, the maintenance actions and others
MAS Autonomy	By one sub-agent type providing decision support
Others	Representation of the forming of agent coalition and physical resource access, which are required for production process execution

Table 13 Identification of sub-agents patterns for resource access as used by Lüder et al. [16]

Sub-agent name	Main functionality	ISA 95 level	Real-time	Source type info.	Communication base	Key properties	Rel. work
Order agent	Allocates of order related actions leading to an order related schedule	L2–L4	No	Data	FIPA specification	Autonomy, proactiveness	FIPA, [38], works in [16]
Product type information agent	Stores detailed info about the product ordered, the maintenance actions to be taken, data to be collected, etc.	L2–L4	No	Data	FIPA specification	Cooperativeness	
Algorithm processing agent (decision support agent)	Executes of mathematical algorithms required to calculate schedule proposals, costs or other negotiation relevant values	L2–L4	No	Data	FIPA specification	Cooperativeness	
System state monitoring agent (decision support agent)	Provides order and resource agents data about the current state of the overall systems and its parts (resource availability, allocation, etc.)	L2–L4	No	Data	FIPA specification	Cooperativeness	
Resource capability monitoring agent	Collects and distributes resource capability descriptions required to identify resource agents applicable for a certain action for an order	L2–L4	No	Data	FIPA specification	Cooperativeness	
Resource related agent	Takes all resource allocation related decisions leading to a resource related schedule	L1–L3	Yes	Data/hardware	FIPA specification	Cooperativeness, reactiveness	
Aggregating agent (resource related agent)	Coordinates skills of other resource related agents to more complex skills	L1–L3	Yes	Data	FIPA specification	Cooperativeness, autonomy	
Field control agent (resource related agent)	Provides basic skills by directly accessing and interacting with field control devices	L0–L2	Yes	Hardware	FIPA specification	Cooperativeness, autonomy	

the communication with legacy systems is considered, to the highest levels of automation pyramid, where the availability of resources is also considered to produce the highly customized product. For example, the High-Availability Agent is responsible for safety and security functions, whereas the Rescheduler Agent is in charge of data processing inside the process control. This last sub-agent also ensures the availability of all necessary resources of the system. The Supervisor Agent performs supervisory tasks concerning data processing in the MAS and process optimization. Table 10 shows the pattern and Table 11 shows the list of sub-agents of the MAS self*-control architecture [38].

The Dispatcher Agent manages access to the system functions and deals with the knowledge base to ensure sustainable control. The Executive Agent is used to communicate with and control the legacy systems that are used now in the industry. This MAS control architecture contributes to high product customization and quality due to its low development and implementation costs. The universal features of the proposed control system make its operation and adaptability feasible for different uses in the industry.

4.4 Resource access design pattern

The resource access design pattern presented by Lüder et al. [16] is shown in Fig. 4. It consists of a Resource Related Agent that provides process-related capabilities to the global MAS by registering them with the Resource Capability Monitoring Agent. Control devices (e.g., PLC, RNC, CNC, etc.) typically implement the single resource-related parts of the manufacturing process. These execute control modules and software with hard real-time reaction (often < 1 s). One category of the Resource Related Agent is the Field Control Agent that provides fundamental means to access and directly interact with the field control level by applying timing restrictions. The second sub-agent is the Aggregating Agent, which has not a direct access to the field control level, but it is also able to organize actions of other resource related agents by integrating them in a higher-level action. This sub-agent gains more multifaceted abilities by controlling the coordinated application of the underlying abilities.

Table 12 shows the design pattern of the resource access in production plants [16]. Table 13 shows the list of sub-agents used by Lüder et al. in this pattern [16].

The classification criteria for MAS manufacturing control from the Section 3 have been applied. The four patterns used in this classification were selected based on preliminary work of authors in [18], which demonstrated that patterns could be identified despite their different terminology. By applying the classification to 20 different MAS (see Section 5.6), there is a necessity to differentiate the classification of sub-agents of each MAS architecture. In the following section, the patterns

included in MAS approaches will be classified further based on similar function terminologies and automation levels.

5 Common functionalities and automation level patterns (RQ3)

In this part, the application of the introduced approaches in Section 4 and their pattern identification are discussed.

Abstracting a logical composition of CPPS and the scope of its application's environment suggest the practice of compositional architecture [5, 40, 54, 55]. CPPS functions with services can be implemented by recombining the features of the different types of sub-agents or components inside the MAS approach. Regarding the design patterns from the MAS models analyzed, the authors of this manuscript showed in Section 4 that the approaches do not follow the same structure with comparable heterarchy (agent's hierarchy) of the MAS, although these are similar in certain functional respects. The heterarchy refers to the field of application of the sub-agents in the automation levels (e.g., often associated by ISA 95 levels).

Functionalities refer to the sub-agents' services (functional requirements) and the quality of them (non-functional requirements) [21]. The flexibility of manufacturing systems is realized by an agent-based control. To apply the agent-based control in practice, it requires a balance between modular and integral MAS designs. According to Ribeiro in [5], the MAS's structure types regarding the modularity can be defined as modular MAS and integral MAS. The modular architecture is composed of hybrid modules interconnected to respectively well-defined interfaces. The integral architecture contains multiple functions, and interacts with many agent interfaces and often has no discernable modules.

As already mentioned in Section 1.1, the superficial MAS pattern description does not satisfy the aim of this paper to create a ready-made solution. Consequently, it is necessary to identify more specified pattern definitions. Based on the analysis of the collected approaches, this section focuses on the common sub-agents and their action fields, which are usually, applied in the MAS architectures. The next section introduces patterns called resource access, knowledge base, coordination process, and communication interface. According to the analysis, the identified MAS solutions are aligned with these function terminologies—as part of functional requirements—although sometimes with different names.

5.1 Resource access common function

Resource Access (RA*) is a common function closely related to the hard real-time capabilities of the MAS. For example, Wannagat's Resource Agent (Section 4.1) is a type of a modular architecture. Its four modules contain limited application cases and the RA modules interact over specific interfaces (e.g., Agent Interaction interface or Communication agent). RA is very similar to the MFS architecture from Fischer (see Figs. 1 and 2), and also to another work of MFS in [37]. A sub-agent module from Fischer, called Application agent, encompasses the system behavior. However, a MAS architecture includes other three sub-agents with additional modules: order agent, system agent, and control agent. Fischer's architecture can be considered as a modular architecture based on agents' entities and modules. Furthermore, this MAS has crucial similarities with the RA of Wannagat [19, 28]. Instead, Ryashentseva (see Fig. 3) and Lüder et al. (see Fig. 4) approaches are integral architectures where the MAS have similar sub-agent types: the executive agent and the field control agent. Both provide basic abilities and interact with all components and real-time devices in the field control level, respectively.

All specified behaviors of the MAS architectures and their interactions across the defined interfaces are identical. Besides, the goals of resource agent, application agent, executive agent, and field control agent include direct connectivity with the field control level to get data from sensors and actuators in order to manage the incoming module orders.

5.2 Coordination process common function

The MAS architectures from Ryashentseva and Lüder et al. present production processes with related capabilities to coordinate an overall system by managing the internal components, such as a supervisor agent, rescheduler agent, and resource capability monitoring agent. These types of sub-agents are generally located higher in the MAS heterarchy and are parts of the Coordination Process (CP) pattern function. CP defines the boundaries for the sub-agents' operations and reconfigurations in order for them to stay within the adequate limits (e.g., restrictions of RA). The description of Fischer's MAS architecture has three main entities: the AMS, DF, and the specific Coordinator Agent (see Section 4.2). RA from Wannagat has an exclusive sub-agent for the coordination process and its functionality is covered by the Diagnosis module and Planning module (see Section 4.1). All the sub-agents shown in this section could be grouped into the CP function, since these are based on FIPA standard and contain methods for de/registering modules from/to any MAS approach [7, 19, 38].

5.3 Knowledge base common function

Another crucial MAS pattern is the Knowledge Base (KB). For Wannagat, there is a KB module, which includes an explicit system model of the consistent technical component as local knowledge. Fischer uses the same pattern divided in the System KB and the Module KB. In the case of the integral architectures, there are clear examples with Ryashentseva (see Fig. 3) and Lüder et al. (see Fig. 4) approaches, since both are just based on agents' entities with specific tasks. Another

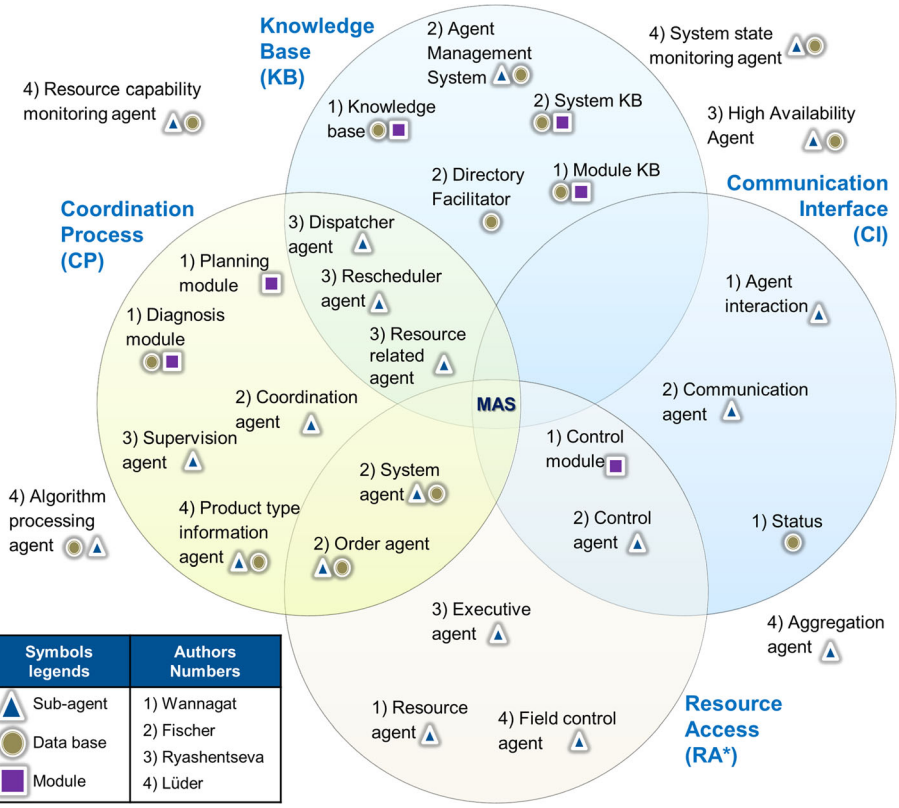

Fig. 5 Summary of the comparison of Wannagat, Fischer, Ryashentseva, and Lüder et al. MAS approaches to map common functional requirements

similarity of the self-control architecture is the presence of the Dispatcher Agent. This sub-agent is comparable to the KB characteristics of the Resource Related Agent. In general, the KA should contain an explicit system model of the corresponding technical component as local knowledge. The components (sub-agents, modules, and databases) composing the KB function are able to check whether the values of the parameters of the technical systems and processes do not violate the predetermined constraints [7, 28, 38].

5.4 Communication interface common function

The Communication Interface (CI) function enables and abstracts communication in between all components (sub-agents, modules, databases, etc.) of all ISA 95 levels. Different platforms via open communications interfaces (e.g., industrial Ethernet, Profibus) and appropriate communication protocols (e.g., based on JADE, applying JAVA and FIPA ACL Messages [31, 39, 56]) have to be accepted by the CI function. These communication interfaces are also used for sending errors and state messages to the sub-agents ranked higher [15, 19, 28, 32, 38]. For example, the Communication Agent, from Fischer, transfers and receives information of sub-agents via ADS protocol (ADS, automation device specification) and Ethernet communication [7, 32]. Tasks of the CI function are compared with the specific goals of the interfaces from RA, which are called Agent Interaction and Status. From Ryashentseva (see Fig. 3) and Lüder et al. (see Fig. 4) approaches, there are additional sub-agents called High Availability Agent and Aggregation Agent with special predictive abilities for maintenance purposes. However, in comparison with the Wannagat and Fischer design patterns, Ryashentseva and Lüder et al. do not clearly define the sub-agent designated to the CI function.

5.5 Summary of the common functionalities

Comparing the designed patterns in Wannagat, Fischer, Ryashentseva, and Lüder et al., Fig. 5 shows a summary of them regarding the common functional requirements discussed in this section. The four circles represent the KP, CP, CI, and RA* as well as based on the internal MAS components (sub-agents, databases, and modules). In Fig. 5, it is shown that RA*, CP, and KB are often implemented by the authors (more MAS components elements are inside the circles) than CI, which was considered only by Wannagat and

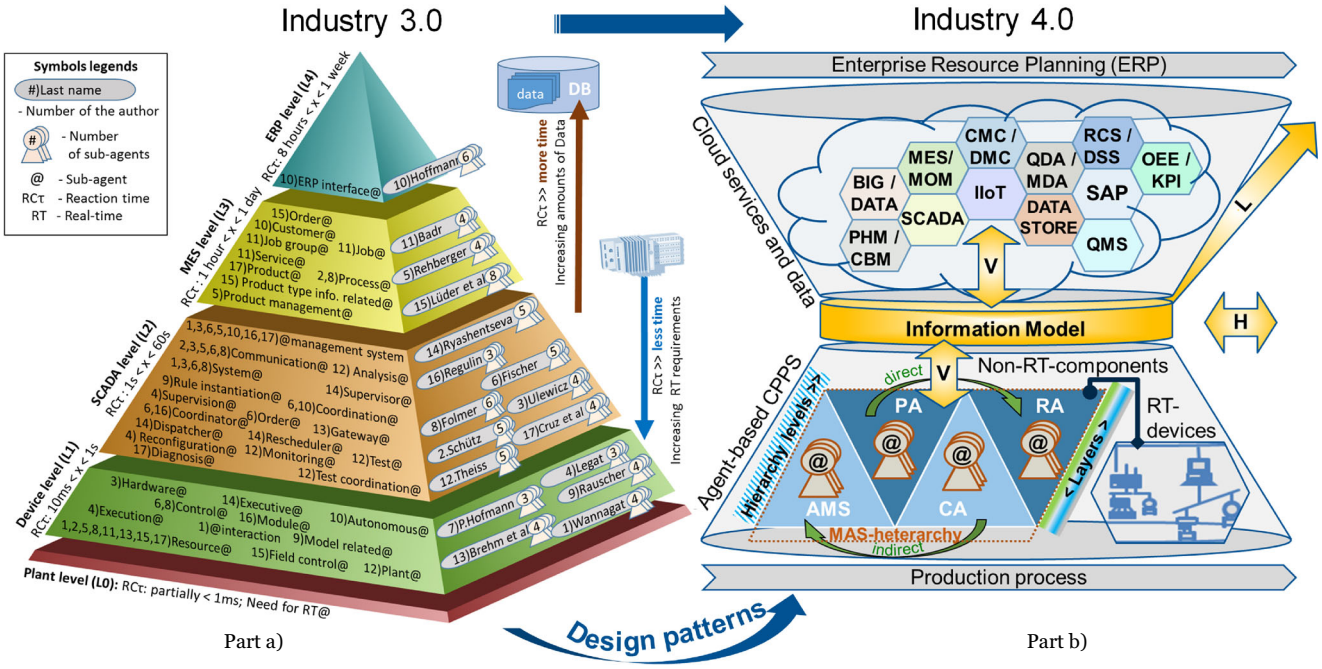

Fig. 6 Migration from the traditional levels of the automation pyramid (part a) to the "diabolo" [57] topology (part b). @: Sub-agent pattern; AMS, agent management system; CA, coordination agent; CBM, condition based monitoring; CMC, collaborative manufacturing community; CPPS, cyber physical production system; DMC, decentralized manufacturing community; DSS, decision support system; H, horizontal integration; IIoT, industrial internet of things; KPI, key performance indicator; L, life-cycle integration; MAS, multi-agent system; MES, manufacturing execution systems; MOM, manufacturing operations management; OEE, overall equipment effectiveness; PA, process agent; PHM, prognostics and health management; QMS, quality management system; RA, resource agent; RCS, resilient control system; RT, real-time; SAP, systems applications products; SCADA, supervisory control and data acquisition; and **V**, vertical integration

Fischer. Additionally, some MAS components (e.g., high availability agent) do not assign a functional pattern, but these can apply appropriate quality controls (non-functional requirements).

5.6 Automation levels and features of sub-agents patterns

This part explains the most important patterns and their features. Sub-agents have been specified and applied in the automation levels mapped into the ISA 95/IEC 62264 standard, with open software and technologies application. The standard follows the traditional automation pyramid (five levels: L0-L4) where the Plant Level (L0) is the lowest level. The identified sub-agent patterns show some random elements with proprietary interfaces that are often used in the industrial control (e.g., mostly PLCs implementing IEC 61131-3 languages programs). Next, Device Level (L1) includes the most popular sub-agent called RA. The components of this level have typically control devices' Reaction time (10 ms < RCτ < 1 s). From the functional point of view, RA covers the components of a manufacturing system in the real world (L0), with the lowest RCτ (partially < 1 ms). RA is also a part of the SCADA Level (L2), with both hard and soft real-time capabilities (1 s < RCτ < 60 s). The Process Agent is the most popular sub-agent for the MES Level (L3), with medium reaction time (1 h < RCτ < 1 day). PA sub-agent pattern usually supervises the execution of a production recipe/plan, and interacts with RAs and AMSs to achieve this goal. In contrast to AMS, PA is not responsible for the technical system but for the production recipe, since it usually requires non real-time capabilities. The MOM/MES functionalities are often results of negotiations/collaborations among different RAs, AMSs, and PAs. In this manner, human operators can revise production orders and rescheduling decisions that result in those negotiations. Another popular sub-agent here is the Communication Agent (often in L1-L3) that converts proprietary interfaces into multiple protocols. If for example some of the RAs request has to be linked to upper automation levels, they usually communicate via CA in protocols such as ADS, OPC UA, and FIPA specifications. Figure 6 shows the organization of the sub-agents in the automation pyramid for the Industry 3.0 and its migration to the adapted "diabolo" architecture [57] for Industry 4.0.

The left part of Fig. 6 shows traditional automation levels with all identified sub-agents. The vertical integration of this

Table 14 List of sub-agents patterns for MAS architectures extended from [18]

Pattern	Sub-agent name	++Resource Agent (RA)	++Process Agent (PA)	++Agent Management System (AMS)	+Communication Agent (CA)	Others (no pattern)	
	Common functionality	KB, RA*	KB, CP	KB, CI, CP	KB, CI	KB	
	ISA 95 level	0–2	2–3	1–2	1–3	0–4	
	Type of agent [8] (reactive or proactive)	Often reactive	Often proactive	Often proactive	Often reactive		MAS scope
Main author last name	Badr	–RA	±Job@	±Service@	–	Job group@	Smart manufacturing
	Brehm *et al.*	++(RA || field related@)	–	++Gateway@	++Broker@	Operator@ (HMI)	Energy systems
	Cruz *et al.*	++RA	++(Product@ & diagnosis@)	++AMS	–	–	Smart manufacturing
	Fischer	++(Control@ & order@ & system@)	++Coordinator@	++AMS	++CA	–	MFS
	Folmer	++Control@	+Process@	+System@	++CA	–	Smart manufacturing
	Legat	++Execution@	++(Supervision@ & reconfiguration@)	++AMS	–	–	Smart manufacturing
	Lüder *et al.*	++(RA || field related@-RRA)	++Decision support@-DSA	++(Order@ & product type info related@)	–	Resource capability monitoring@, (type of DSA)	Smart manufacturing
	M. Hoffmann	+(Autonomous@ || transport@-specific)	++ (Coordination@ || manufacturing, specific@)	–	+Customer@	ERP Interface@	Smart manufacturing
	Nieße A.	+Control@	+Planning@	–	–	–	Energy systems
	P. Hofmann	+Control@	–	+Rule set adaptation@	–	Image object@	Image processing
	Pech	±User@	±Query management@	±Query@	±Ontology@	Information retrieval@	Information processing
	Rauscher	–	±(Coordination@ & rule instantiation@)	±Model related@	–	Rule@	Information processing
	Regulin *et al.*	+Module@	++Coordinator@	++AMS	–	–	MFS
	Rehberger	++RA	++Product management@	–	+@interaction	–	Smart manufacturing
	Ryashentseva	++(Executive@ & rescheduler@ & dispatcher@)	++Supervisor@	–	–	High availability@, HAA	Smart manufacturing
	Schütz	++RA	++PA	++(Control strategy@ & system@)	++(CA || @interaction)	–	Smart manufacturing
	Theiss	+Plant@	++(Test coordination@ & monitoring@)	±Analysis@	+Test@	–	Communication agent
	Ulewicz	++(Hardware@ & system@)	–	++AMS	++(CA & system@)	–	Smart manufacturing
	Vogel-Heuser *et al.*	++Plant@	++(Coordination@ & customer@)	++AMS	–	–	Smart manufacturing
	Wannagat	++(RA || control@)	++(PA & system@)	++AMS	++(CA || @interaction)	CPPS plant@	Smart manufacturing

Notations: same colors mean these are following a similar pattern with these degrees of "likeness": ++High; +Medium; ±Low; −Very low or nothing. Symbols for logical representations are & (and) sub-agent are complementary; || (or) sub-agent are similar. The names are reduced replacing "agent" word by the "at" sign (@). References of the works are: Badr [58]; Brehm et al. [59]; Cruz et al. [31]; Fischer [32]; Folmer [48]; Legat [53]; Lüder et al. [16]; M. Hoffmann [41]; Nieße A. [35]; P. Hofmann [52]; Pech [42]; Rauscher [51]; Regulin et al. [37]; Rehberger [19]; Ryashentseva [38]; Schütz [28]; Theiss [39]; Ulewicz [7]; Vogel-Heuser et al. [40] and Wannagat [27]

pyramid is one of the essential challenges for the dynamic evolution of Industry 4.0 [57]. Therefore, the right part introduces the adapted Distributed Architecture to Bolster Lifecycle Optimization or "diabolo" from [41]. This part of Fig. 6 shows the crucial functions of an MES within the top cone and device level processes (real and non-real-time) on

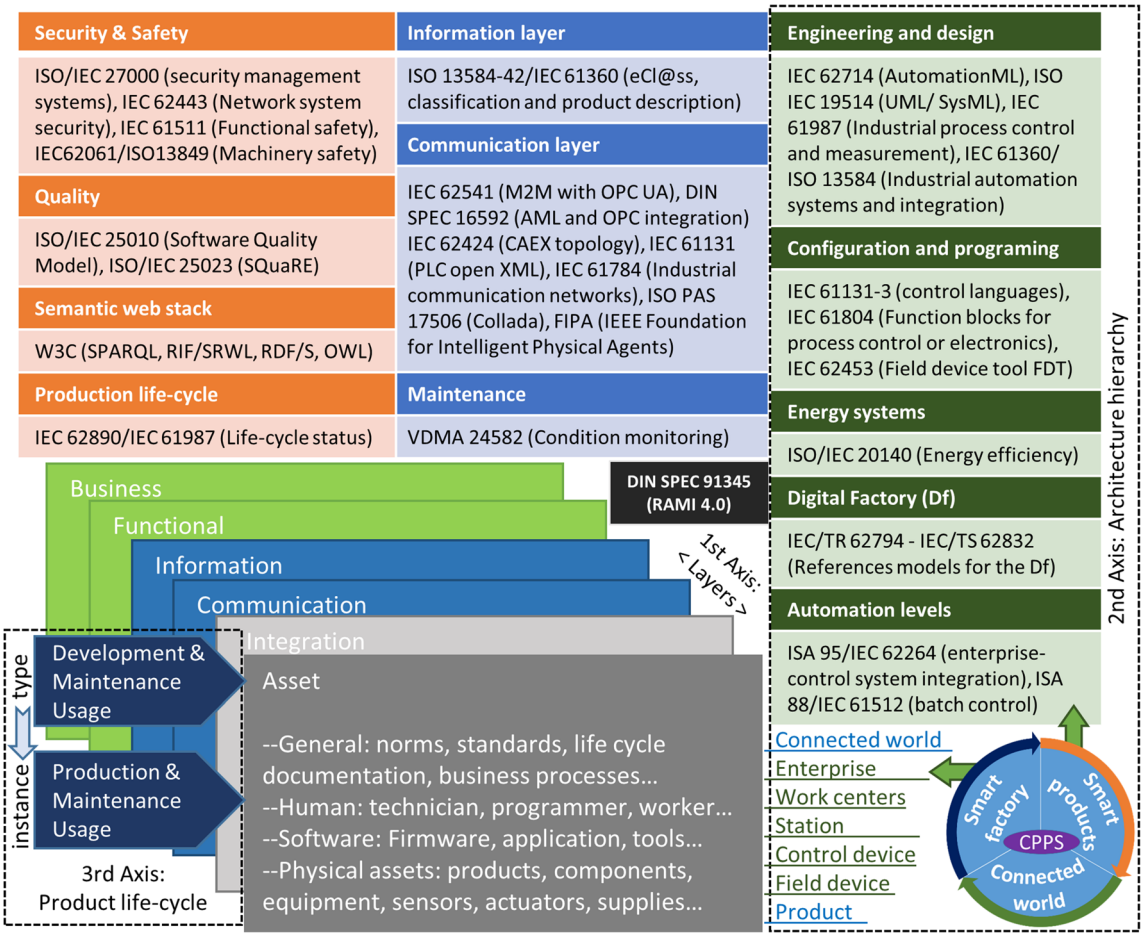

Fig. 7 The landscape of the RAM I4.0's axes and their optional norms

the bottom of the diabolo. Direct and indirect communication ways are enabled in the bottom of the cone of the diabolo. The agent-based CPPS architecture with the four patterns (RA, PA, CA, and AMS) attempts to harmonize the data exchange between these two cones (e.g., using modeling language for technical specifications and evaluation of the processes and resources by Overall Equipment Effectiveness).

The list of all identified sub-agents is shown in Table 14 and organized in the automation pyramid of Fig. 6 (left side). In the last case, the AMS is also the main pattern of the L1-L2 levels, since this sub-agent can be mapped in a bidirectional way between sub-agents' identifications. The AMS provides a unified interface that makes it possible for every component of the same type, regardless of the provider, to be reached with the same protocol. A major requirement for the sub-agents introduced in the L0-L2 levels is that they should be executable in a hard real-time operating system and should follow the hardware settings. The top-level (L4) of ISA 95 has the longest reaction time for the ERP system with long-term schedules (8 h < RCτ: < 1 week). L4 components should provide a human interface and interface with eventually cloud services. Regarding the sub-agents identified, there are not many sub-agent types, which mainly provide patterns to create orders or get status information about this level. An example out of the patterns in the L4 is the ERP Interface Agent from Hoffmann [41] that establishes via OPC an internal information exchange with the ERP. More extended specifications and pattern descriptions of these main sub-agents are scoped in the next section of this paper.

6 Agent-based CPPS architecture for I4.0 component evaluation (RQ4)

The I4.0 focuses on key aspects of smart manufacturing that can be explained as interactions between the following features [47, 60]: i) horizontal and vertical integration through value networks and within a factory or production shop; ii) life cycle management that refers the end-to-end engineering; iii) the human beings coordinating the stream value; and iv) the security to achieve the confidentiality, integrity, and availability of assuring data (transfer and storage). Likewise, the mass personalization known as

the Additive Manufacturing can combine the smart manufacturing to a paradigm move for the I4.0 [61]. In order to facilitate and promote the smart manufacturing aspects mentioned above, RAMI 4.0 provides a flexible architecture based on functions and information levels within 3D dimensions. As illustrated in Fig. 7, there are different applicable standards [47], to follow the guidelines in the RAMI 4.0 model.

The RAMI 4.0 model provides a structured view of the multiple levels (even a specific Asset level) using an architecture consisting of three axes (see Fig. 7). The aim of the model is to create manageable segments (sub-models) by combining the different axes at each point in the asset's phases, to represent each relevant characteristic. The following items describe the RAMI 4.0 axes distribution [47, 60]:

- The first axis is named the "Architecture hierarchy". It is based on the traditional IEC 62264-1 (ISA 95) and IEC 61512-1 (ISA 88) standards and their levels' hierarchies. The goal of this first axis is to define assets and their combinations with the necessary precision, since the description of RAMI 4.0 is a purely logical one.
- The second axis is named the "Layers." This one uses six layers to represent the relevant information for the multiple assets' roles: Business, Functional, Information, Communication, Integration and Asset.
- The third axis is named the "Product life-cycle." Based on IEC 62890, it represents the lifetime of an asset and the value-added process.

In the next section, the authors of this paper address the alignment of sub-agent patterns with the RAMI 4.0 model by comparing only two dimensions. Many similarities can be found between agent-based architecture for CPPS and the RAMI 4.0; however, the Product Life-cycle axis is out of this paper's scope.

6.1 MAS architecture based on RAMI 4.0 model

Regarding the Layers axis, the MAS proposed based on patterns should describe the I4.0 components in terms of properties, system structures, specific data and functions, and their external behavior. Since the present layers do not conform to the ISO-OSI guidelines, it is not mandatory for a RAMI 4.0 layers to provide the corresponding information. As a result, some layers can also be ignored in specific domain systems that are not applicable. A layer just characterizes parts of asset's behaviors and their connection between adjacent layers. A possible definition for the agent-based CPPS architecture is proposed in the upcoming paragraphs.

An Asset is a physical/logical item having actual value to the organization [34, 60] (e.g., products, equipment, software, human resource, standards, and documentation). In case of a physical asset, according to the IEC TS 62443-1-1, for industrial control automation, the device under control can contain the largest directly quantifiable value. The Asset Layer is the lowest level of the RAMI 4.0 model because it reflects the physical components, administrated by other upper layers, which are in the cyber world.

The Integration Layer is a type of adaptation for the transition among physical and cyber worlds. The main aim of this layer is to convert a physical variable into a digital one. The resulting data is converted according to specific formats. Therefore, the Resource Agent is the most significant entity in this layer. For example, RAs have the adequate subroutines to send and receive data for a controller to regulate the speed in a conveyor (sub function). Additionally, operator interactions could take place in this layer, e.g., via Human Machine Interfaces (HMIs).

The Communication Layer covers connection lines according to the guidelines of I4.0. This layer distributes information to other I4.0 components and receives data back from them. The base of the Communication Layer absolutely follows the seventh ISO/OSI layer's guidelines [47]. A Communication Agent, using a uniform data presentation contained by the CPPS, can standardize the communication methods. In addition, CAs provide services for control in the route of the adjacent Information Layer. The definition of communication technologies within I4.0, such as Machine-to-Machine (M2M) and Machine-to-Business (M2B), e.g., the OPC UA (IEC 62541), clearly applies to direct communication [34]. Other communication protocols, such as FIPA specifications or MQTT, are much more flexible; therefore, they enable indirect communication [31, 34]. For this layer, the patterns AMS, DF, and MTS from FIPA can support the Communication Layer and will be further explained in the Functional Layer. For an agent-based CPPS architecture, the scalability and the interoperability of industrial communication

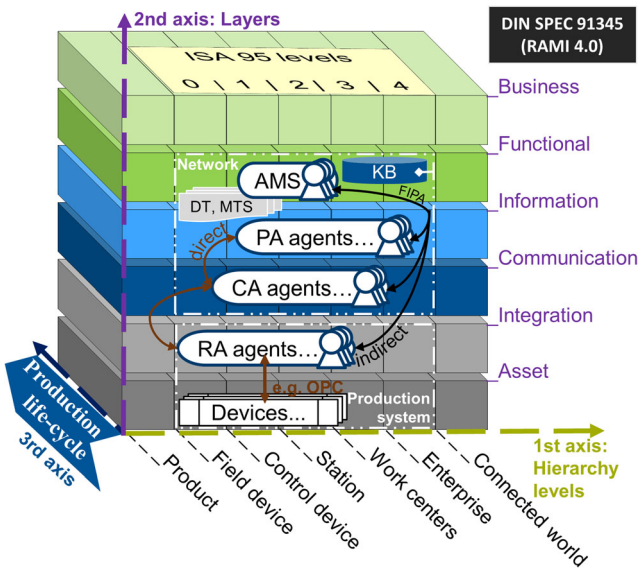

Fig. 8 Agent-based CPPS architecture aligned into two axes of RAMI 4.0

networks (IEC 61784) are decisive for numerous smart components and functions as well (e.g., RFID sensors).

The Information Layer defines the information for significant functions and data storage sites of a particular asset (i.e., the Cloud) [47]. The PA could be a logic abstraction for a product in this layer. Since the PA holds its own information of procedures and plans, it is responsible for coordinating its own production. The PA pattern is a special entity of the agent-based CPPS, which orchestrates the execution of processes steps (with cooperative skills to interact with RAs, CAs, and AMSs sub-agents). The Information Layer is important to understand the different partial models of all the sub-agents, including existing data exchange formats for each specific case. To fulfill the contents of this layer, its 4.0 implementation should be based on models integrating different fields with reliable and standard methods. Here, RAMI 4.0 suggests Automation Markup Language (AutomationML) specifications, which trace a modular document structure with the aim to join the diverse and modern engineering tools in their heterogeneous disciplines (e.g., mechanical engineering and electrical designs). Thus, for information models, the AutomationML can enhance or adapt the existing XML-based data formats, integrating AML and OPC (DIN SPEC 16592), or using IEC 62424 (CAEX topology), ISO PAS 17506 (Collada), IEC 61131 (PLC open XML). It is also possible to use engineering and designing tools, e.g., ISO/IEC 19514 (UML/SysML) [34]. For the semantics of AutomationML, the standard's properties from ISO 13584-42/IEC 61360 (eCl@ss: classification and product description) with the Common Data Dictionary (IEC 61360 CDD) can be applied. All of the above comply with the standard for the Digital Factory (IEC 62832).

The Functional Layer follows the rules of I4.0 by assigning all logical functions and services of the assets. These technical functionalities get data from the Information Layer and deposit them back in the same layer, as methods and decision-making logic (e.g., mathematical functions) [34, 47]. According to the use case, these methods can also be executed in other lower layers such as the Integration, Communication Layer or Asset Layer. Therefore, AMSs have a fundamental role in this layer, since it contains methods for registering and deregistering modules to and from the system. Other leading patterns are the knowledge base modules because they allow the RAs or PAs to check whether the parameters of the technical systems and processes are kept within the predetermined functional limits.

The Business Layer is the highest level that defines the pertinent business procedures with their structure requirements and the business-related features of the assets (e.g., regulations, legislative requests, contracting, and licensing) [60]. This layer does not refer to any concrete systems such as the ERP, since the Functional Layer (in the factory plant context) usually sets the ERP's functions. Since no pattern was found that could fulfill the Business Layer's requirements, this layer will not be further researched in this paper.

Each sub-agent located in the agent-based CPPS architecture, as Fig. 8 shows, is not required to have a fixed location in the RAMI 4.0 layers.

Given the model associates multiple layers in two dimensions, each MAS component pattern is a primary part with a specific role in its respective layer, but they are possibly joined with the other adjacent layers, as described above.

There is another important second axis from RAMI 4.0, the Architecture hierarchy that represents the hierarchy position of functionalities and responsibilities within the factories/plants. This functional hierarchy is not only the equipment classes or the automation levels of the classical pyramid. As mentioned above, this axis follows the ISA 95 and ISA 88 standards to realize the classification within the plant (see Fig. 8) [47]. However, RAM I4.0 considers other levels to cover as many areas as possible from traditional industry to the new factory automation. New terms based on ISA 95 levels (see Section 5.6) are established such as the Enterprise Level (L4), Work Unit Level (L3), Station Level (L2), and Control Device Level (L1). I4.0 considers other multiple equipment or systems within the factory because not only the controllers are decisive for this one. Therefore, the Field Device Level (L0) has been added below the Control Device Level, and it is a practical level of a smart field device (e.g., an MAS RFID intelligent sensor [62]). Moreover, not only the plant and its equipment are essential in I4.0, but also the products to be factory-made itself. Then, RAMI 4.0 adds the Product Level as the lowest level that allows standardized consideration of the product to be mass-produced and the manufacturing capability (with their relationships).

An addition has also been made at the highest end of the Architecture hierarchy axis. The two ISA/IEC standards cited only define the levels within a plant (see Section 5.6). However, I4.0 goes further by describing group corporations, interdependencies, and net of factories (e.g., alliance with outer engineering companies and component suppliers/customers). Consequently, the Connected World Level has been added to observe above and outside the Enterprise Level. Regarding the heterarchy sub-gent's patterns in this axis, their locations are based on the first (Lo) to last (L4), as ISA 95 automation levels mention. Consequently, the Connected World and the Product levels are out of the agent-based CPPS architecture proposed in this paper.

6.2 Design pattern for the administration shell

For this section, from [1, 34], there are specific terms as important definitions for I4.0:

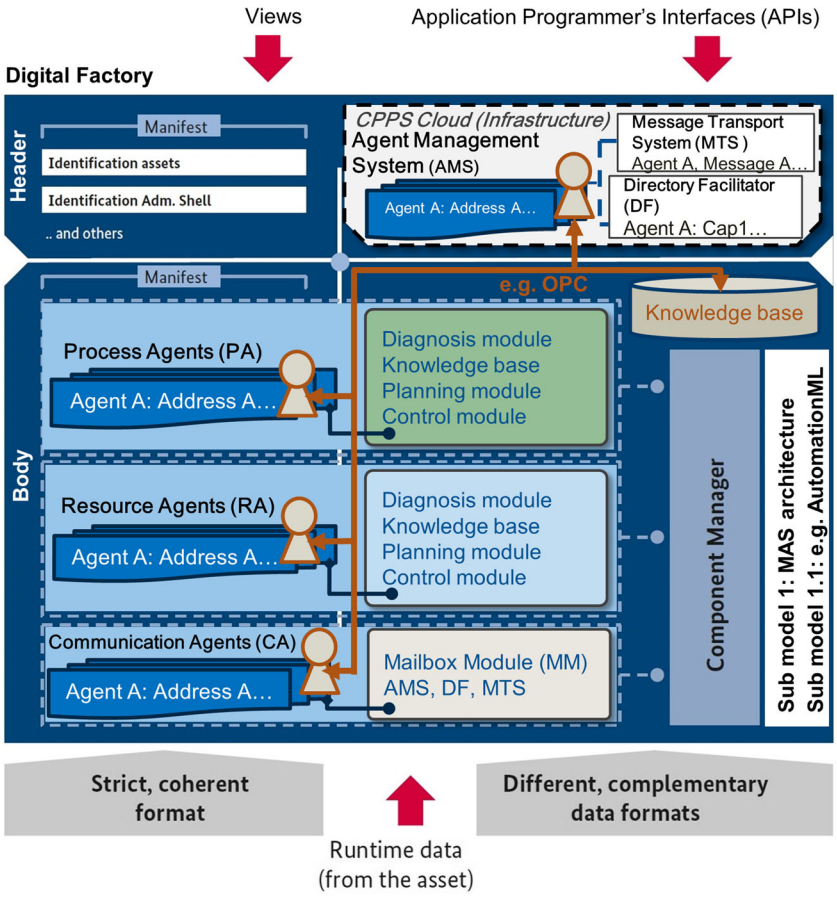

Fig. 9 CPPS's administration shell for I4.0 components (adapted from [60])

- The Industry 4.0 component (henceforth, I4.0 component) is a "globally uniquely identifiable participant with communication capability consisting of administration shell and asset within an I4.0 system which there offers services with defined QoS (Quality of Service) characteristics."
- The administration shell is the "virtual digital and active representation of an I4.0 component in the I4.0 system and contains the manifest and the component manager."
- The Manifest is an "externally accessible defined set of meta-information, which provides information about the functional and non-functional properties of the I4.0 component."
- The component manager is "the organizer of self-management and of access to the resources of the I4.0 component…"

Looking to fulfill the requirements for the CPPS and the RAMI 4.0 (see Section 2.1), a general organization for the administration shell based on the MAS can be developed. For this, the I4.0 components for the CPPS proposed display an abstract form that defines real objects. For example, these could be a valve as a control element, a pipe as the controlled process, a sensor as a measuring element, and a PLC's algorithm as a controller, etc. MAS architecture should be based on design patterns described above, in both ways, physical (assets) to the cyber (digital data). Moreover, Information and Communication Technology (ICT) needs to increase additionally regarding the appropriate smart manufacturing aspects such as the horizontal/vertical integrations, product life-cycle, human's interaction and others, as mentioned at the beginning of this chapter. As shown in Fig. 9, the administration shell contains the "Header" and the "Body" parts. Both in order to provide better identification via asset(s) designations [34, 47, 60].

The intention of this section is to describe a general implementation of I4.0 components using a sub model design with the identified MAS patterns. A basic application of an I4.0 component is based on the suggested international standard of AutomationML, as a method for the Information Layer, and OPC UA for the Communication Layer. Both together could realize the Body of the administration shell of the I4.0 components with an agent-based CPPS architecture. For the Header part, the CPPS provides an adequate unique identification of the I4.0 component by a server. Also, the data representation and its function access should be integrated. According to RAMI 4.0 suggestions [47, 60], a unique identification of the object could be using a UUID (Universally Unique Identifier) or URIs (Unique Resource Identifiers, e.g., for RDF). The AutomationML concept specifies every object with a UUID that could be kept as long as the object exists.

For communication, the I4.0 component provides access to technical functions pre-realized in the AMS sub-agent (with their respective DFs and MTSs), in order to enable the access to the representation of any asset's information.

The Body part of the administration shell contains structured sub-models which might denote information and functions [34]. A standardized format eCl@ss, which is based on IEC 61360, is suggested to describe the data and functions in a diverse and harmonize format. The features of all sub-models, in consequence will always develop a comprehensible table of contents (each I4.0 component and its respective associated manifest and administration shell). As a prerequisite for required semantics, the Header shall individually recognize administration shells, Assets, Sub models and their properties globally.

This paper's approach assumes that a physical asset type of interest is controlled by open controller architecture (e.g., PLC) that implements lower level programing codes (e.g., IEC 61131-3). As the MAS design patterns are shown (see Section 5), the sub-agents of the different assets type can be located on all ISA 95 automation levels. Hence, for the operation of an I4.0 component, it has to be clearly specified, which technical functions are provided by the component and their configurations limits. For the PLC implementation example, adequate variables for the code should be accessible via functions with multiple OPC UA servers interlinked and following a service-oriented architecture (SoA), proposed in [34]. As a result, the administration shell of a I4.0 component consists of multiple sub models (first is the MAS architecture) and a nonempty set of interlinked designs (e.g., AutomationML projects, mechanical computer-aided designs or CADs, interconnecting FBs/POUs models, UML classes diagrams, and others [47]).

Other standards can be applicable to MAS patterns and aligned to RAMI 4.0 with multiple aims (see Fig. 7): The VDMA 24582 (condition monitoring) for maintenance purposes into the Asset Layer and Integration Layer. The ISO/IEC 27000 for the security of management systems. The IEC 62443 used for the network system security and IEC 62351 for secure authentication [63]. The IEC 61511 applies the functional safety and the IEC 62061/ISO13849 relates the machinery safety. Software quality can be valued by ISO/IEC 25010 and the ISO/IEC 25023 (SQuaRE method) [21]. Semantic web stack can follow the W3C consortium definitions such as SPARQL, RIF/SRWL, RDF/S, and OWL. Energy efficiency can refer to the ISO/IEC 20140. Finally, configuration and programing typical tools are based on C++ plugins for control languages (e.g., mostly in IEC 61131-3 or IEC 61499 [13, 14], IEC 61804 (FBs for process control or electronics), and the IEC 62453 (Field device tool, FDT).

6.3 Discussion of the CPPS and RAMI 4.0 requirements evaluation

The majority of the requirements specified in Section 2.1 are already partly completed by the design pattern of the proposed MAS architecture. First, for the CPPS requirements (Req1.1-Req1.5), the compatibility to different applications (Req1.1) is warranted by the open software MAS architecture (see Section 5). Level independence (Req1.2) and platform independence (Req1.3) are partly achieved by applying four types of sub-agents: the RAs, PAs, CAs, and AMSs (see Section 5.6). Using the TCP/IP as fundamental communication protocols, (e.g., OPC UA) can solve parts of handling and recovery errors (Req1.4), and allows the CPPSs networks to be accessed by other applications. By distributing organizational sub-agents in the cloud (e.g., the PAs, and AMs), the agent-based CPPS is decentralized (Req1.5), as shown in Section 5. However, the multiple platform acceptation (Req1.3) and the reconfiguration of sub-agents (Req1.4) should be further examined by quantitative experimentations. As a first assessment of the platforms suitability, some experiments with these CPPS requirements were measured into multiple platforms in [31].

Table 15 Research questions, hypotheses (see Table 1), and their evaluation

Research question	Hypothesis	Status result	Proof section related
RQ1 (how describe MAS patterns?)	RH1.1 (valid classification criteria)	True	3
	RH1.2 (similar design MAS pattern's terms)	True	4, 5
RQ2 (for which CPPS domains?)	RH2.1 (different goals and benefits)	True	2, 4
	RH2.2 (only real-time requirements)	Partially true	4, 5
RQ3 (which MAS patterns are reusable?)	RH3.1 (functional–and non–requirements)	Partially true	5
	RH3.2 (specific sub-agents)	True	5
RQ4 (how to aligned CPPS to RAMI 4.0?)	RH4.1 (simple CPPS aligned to RAMI 4.0)	Partially true	6
	RH4.2 (administration shell capable)	True	6

Regarding the RAMI 4.0 requirements, the proposed I4.0 components support multiple engineering disciplines and norms (Req2.1). For example, the MAS architecture is focus on the software components (sub-agents' patterns); it is also possible to associate the physical connections of assets via CAD diagrams, according to functional considerations of AutomationML, as Section 6 mentioned. The MAS systems boundaries (Req2.2) and nestability (Req2.3) principles for the I4.0 components are aligned by the MAS's organization in the axis of layers and Architecture axis, respectively (see Section 6.1). The general administration shell model of Section 6.2 partially gets the virtual representation of I40 components (Req2.4) and its functional properties (Req2.5), as shown in Section 6.2. However, the agent-based CPPS architecture did not specify non-functional requirements (Req2.5) yet, such as precise quality characteristics (non-functional requirements) or evaluation metrics attributes (e.g., degree to which the sub-agents cover all their tasks and objectives). The summary of the hypotheses evaluation according to the fourth research questions (RH1.1-RH1.4) is shown in Table 15.

From the eight hypotheses (see Table 1), five are true, and three are partially true, considering the evaluation of this manuscript's authors, and the FA 5.15 expert discussions. These results are extended by fulfilling preliminary requirements (see Section 2.1) and represent the related sections of this manuscript, as shown in Table 15.

6.4 Comparison of the agent-based CPPS architecture to other approaches

Considering the two essential architecture types from Trentesaux [22] (hierarchical and heterarchical interaction entities), CPPS can be described through different designs with advantages and disadvantages of the distributing control decisions (see Section 2). Explicitly, hierarchy could be seen as a type of "vertical control distribution" while heterarchy is a type of "horizontal control distribution" [64]. The type of architecture will define the quality characteristics of the production system. Traditional approaches are included into the Class 0 and Class I types of architectures, respectively centralized and fully hierarchical. What is common in these two architecture types is that they both have a main decision node, where the planning and information processing are concentrated [65]. These classes show better optimization qualities, but a slow response and low tolerance to faults and expansibility [65]. Thus, it is possible to construct a CPPS architecture typology that is inspired by Computer Integrating Manufacturing or CIM (e.g., [23]). The CPPS of

Table 16 Different classes of CPPS approaches

Name of the architecture/author	CPPS approach	Sub-agent pattern			
		Resource agent (RA)	Process agent (PA)	Agent management system (AMS)	Communication agent (CA)
Class 0: Centralized control systems					
CIMOSA [23]	Based on CIM	+Resource	−Capability set	−Organization unit	–
Class I: Fully hierarchical control system					
ARC-SoA [24]	SoA and CPS	+Data adaptor	−Data client agent	–	+Shared variable engine
iLand [66]	SoA	+Service manager	+Control manager	−Application manager	+Communication middleware
Lee et al. [54]	Industrial CPS	+Snapshot collection	−Similarity identification	−Synthesis optimized future steps	–
Class II: Semi-heterarchical control system					
ADACOR [25]	HMS	+Operational holon	++Product holon	+Supervisor holon	–
IDEAS [8]	MAS	++Machine resource agent	++Product agent	+AMS	+Transportation system agent
Pollux [67]	Hybrid control	++Resource decisional entity	++Local decisional entity	++Global decisional entity	–
PROSA [36]	HMS	+Resource holon	++Product holon	+Order holon	–
This paper's authors	MAS	++Resource agent	++Process agent	++AMS	+Communication agent
Class III: Fully heterarchical control system					
D-MAS [26]	MAS	++Delegate ant MAS	++Delegate MAS	–	++Smart messages entity
Ueda legacy [68]	Bio-inspired	+Service	+Service engineering	–	–

Notation of degrees of "equivalence": ++High, +Medium, −Low/nothing

Table 17 Advantages and disadvantages of CPPS classes [22, 64, 65]

Main features of CPPS approaches	Classes	
	0-I	II-III
Have short reaction delays (reducing long-term instability (e.g., bullwhip effect in supply chains)	–	++
Make easier the procedures to initialize and reconfigure (plug and produce systems capability) and breakdowns recovery	–	++
Increase product traceability and allow "smart" products (more active life cycle, e.g., distribution, logistics, inventory, generation, design, effectiveness, and agility)	–	++
Permit robustness with external/internal unexpected changes to return on long-term investments (opposite to CIM scheme)	–	++
Can include the lack of predictability, analytical solutions, and poor ability to define optimal loadings (e.g., cause deadlocks)	–	++
Facilitate supply chain collaboration mechanisms (business agility). Systems can co-exist with several hierarchies	++	+
Optimize resources utilization (system extensions and unforeseen modifications are facilitated).	–	++
Enable flexibility and reactivity to disturbances (Fault tolerant)	–	++
Address a global optimization of the decision-making	++	–
Allow a limit complexity and facilitate system implementation	++	+
Get poor ability to extend the system, and make unforeseen modifications (additions are difficult to make)	++	–
Have poor reliability (paralysis of the levels below a point of failure) and poor fault tolerance	++	–

Notation of applicability: ++High, +Medium, −Low

Class 0 and Class I (one-level heterarchy) are applicable for CIM, since these are based on pure hierarchical interactions (e.g., [24]). On the opposite side, Class III uses full heterarchical interactions to lead mostly distributed architectures (e.g., [26]).

Class II CPPS architectures, being semi-heterarchical, can be positioned in between because they can integrate both hierarchical or heterarchical interactions (e.g., [25])—assimilating both advantages and certain disadvantages. The main advantages of the hierarchical type are the robustness, predictability, and efficiency. Then, in the CPPS approaches of Class II, local decisions are made taking into account global criteria and these are distributed to different controllers. Despite their advantages, traditional methods do not show the capability of adaptation due to the rigidity of the control architecture that as a result weakly responds to changes. Such types of production systems will not show the capabilities of responsiveness, flexibility, and reconfigurability [65]. Therefore, an advantage of

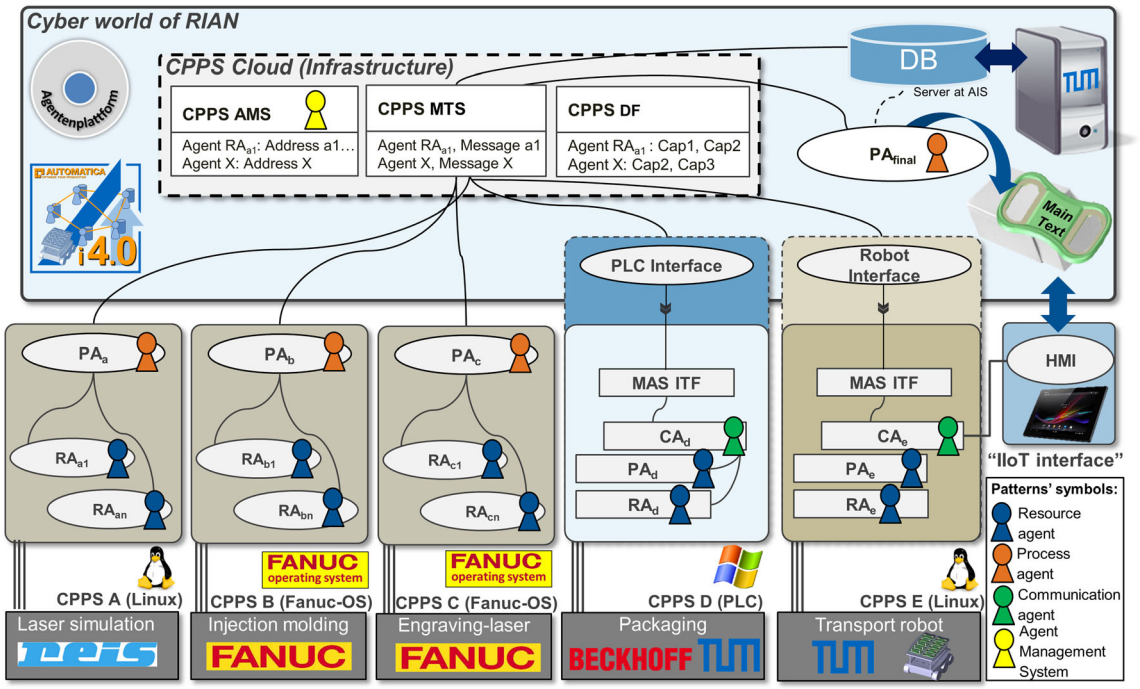

Fig. 10 The robot integrated agent network "RIAN" (adapted from [69])

the proposed agent-based CPPS architecture can be found according to the classification of [64]. The proposed architecture is classified as Class II type, since sub-agent interconnection is not strongly associated (not Class III), while there is at least a strong sub-agent connection (not Class I) [64]. For example, a unique RAs' network (Class III) of the CPPS could be a Class II control system with supervisory level sub-agents (with PA or AMS). Other CPPS approaches [22, 64, 65] can also support advantages of Class II as well as MAS, bionic, bio-inspired (e.g., [68]), and holonic. Among the last named CPPS approaches, the PROSA [36] and ADACOR [25] are the most relevant architectures. In principle, each holon shall represent a logical unit of the manufacturing system, while the sub-agent patterns could help its actual implementation [8, 65].

In heterarchical control systems (Classes II or III), long-term optimization could be hard to get and to validate, while with traditional classes (Classes 0 or I), short-term optimization is easier to obtain [64]. In the Class III type, as long as all the entities (e.g., agents) get the equal level of autonomy, an adequate level of performance can be attained, but there is no global view of the system [65]. As these features are disadvantages of Class III—even for MAS approaches of Class II—the proposed agent-based CPPS architecture cannot claim to be exempt from this problem and only an adequate AMS's global response to it could address the issue.

Table 16 summarizes different CPPS approaches examples which allocate control decisions from centralized control systems (Class 0 and Class I) aiming to design non-centralized control systems (Class II and Class III). This table compares components of different CPPS approaches with patterns of this paper's proposed CPPS architecture.

Table 17 compares advantages and disadvantages of hierarchy (Classes 0/I) and heterarchy (Classes II/III) of CPPS approaches (based on [22, 64, 65]).

6.5 Use case evaluation with an I4.0 demonstrator

The "Robot Integrated Agent Network" (RIAN) completes the evaluation of the proposed agent-based CPPS architecture. The RIAN demonstrator was presented at Automatica fair in an industrial environment [69]. The purpose of the demonstrator was to crosslink heterogeneous production equipment and robots in a network for common customized production. RIAN was the result of a collaboration of the academy (Technical University of Munich "TUM" and Brandenburg Technical University of Cottbus "BTU") with different industry partners [69], which applies the reference architecture of the MyJoghurt I4.0 demonstrator [40]. With RIAN I4.0 demonstrator, the users could customize the bottle opener (with freely definable lettering) online and choose a delivery time depending on available production capacity (IIoT-HMI interface). A chain of production stations composed of autonomous and operator-controlled mobile transport robots defines RAIN. These stations cover the production line for the individualized bottle opener consisting of the following: cutting

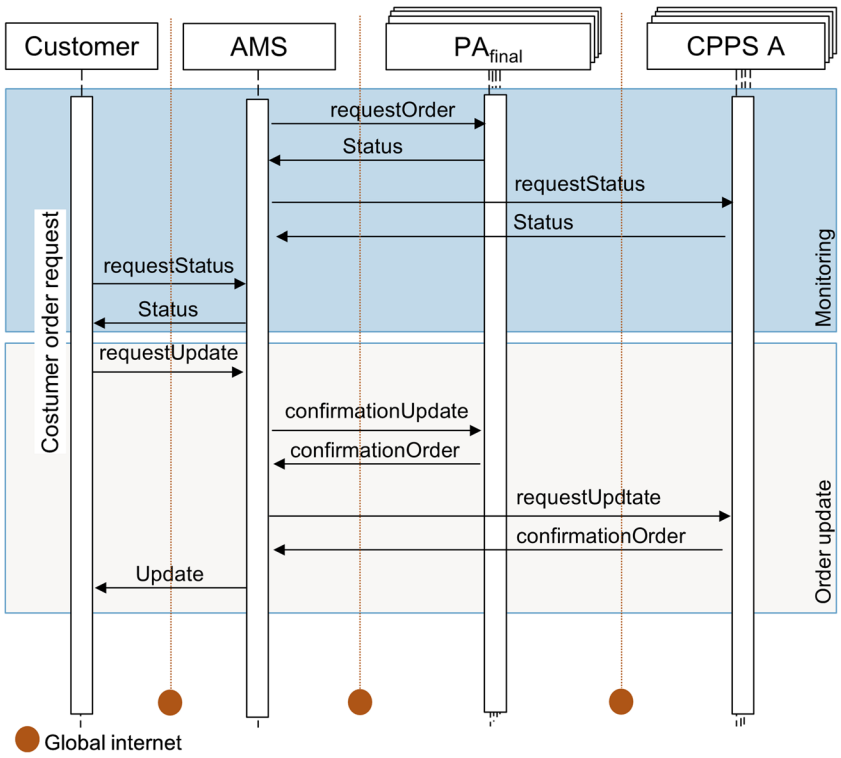

Fig. 11 Interaction in the agent-based CPPS network (adapted from [69])

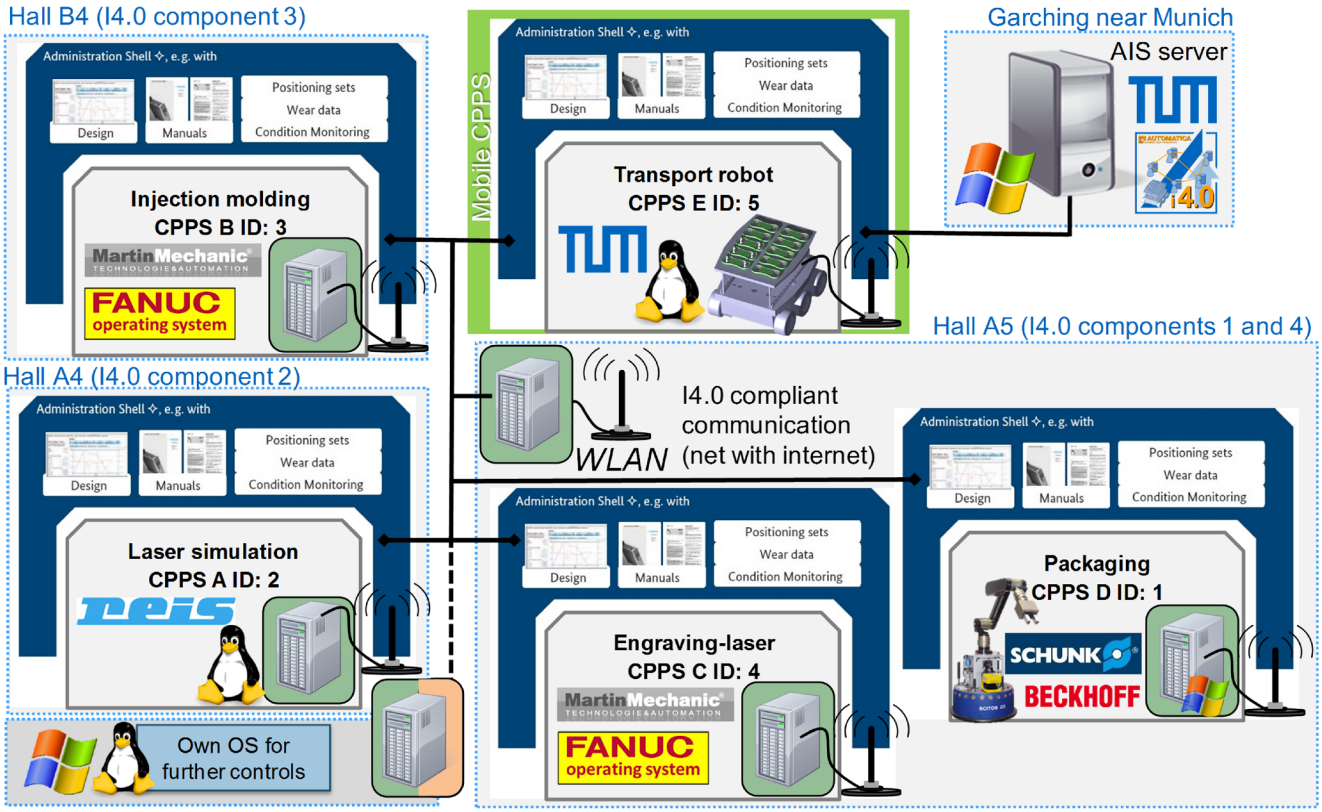

Fig. 12 The I4.0 components of the RIAN demonstrator

by laser simulation (CPPS A), injection molding (CPPS B), engraving-laser (CPPS C), packaging (CPPS D), and customer delivery by a Mobile transport robot (CPPS E), depicted in Fig. 10 (adapted from [69]).

Starting from the warehouse, mobile robots transport pieces between stations of the production process. Intermediate robot stations have hardware interfaces, which ensure the exact positioning of pieces or their detection by vision sensors. All CPPS communicate via an agent-based CPPS network in order to exchange necessary processing steps as well as clearances for manipulation. The current production progress is traceable for the client, for the maintenance and operating personnel. This is possible due to the aggregated reports of the individual sub-agent patterns (RAs, PAs, CAs, and AMS) into an external server (at TUM Garching near Munich), as shown in Fig. 10.

All CPPS A-E include RAs with goal-orientation algorithms (even with artificial intelligent) to achieve PAs orders. Inside the Mobile transport robots (CPPS E) and the Packing station (CPPS D) exists a hardware MAS interface (MAS ITF) and agents (CAs) which ensure by computer vision systems the exact positioning of the products (PAs) in the plant (see Fig. 10). An agent (PA) assigns initial orders to the transport robot (RA) from the storage and links the information about the process steps and the corresponding features of the products. RIAN defines a distribution of production phases for multiple participating companies and technologies (Req1.1-Req1.3).

All CPPS interactions are connected to the local hardware and accept new orders (PA_{final}) after registering at the directory services (from AMS, MTS, and DF). Since the customers' orders for products need to be decomposed into multiple different manufacturing tasks, to which the facility agents can respond, various CPPS aligned with the proposed architecture were implemented for this purpose (see Fig. 11).

A key benefit of the agent-based CPPS approach in the context of I4.0 is the linkage of heterogeneous controls. RIAN enables suitable controls to cooperate with adequate operating systems of various robot vendors (e.g., Raspberry Pi with Raspbian-Linux, Reiss robotics (now KUKA), robot controller with VX Works, FANUC robot controller with FANUC OS).

Both the implementation on suitable controls as well as on external computers is possible by using manufacturer provided interfaces. These interfaces enable data exchange between agents (CAs) and controls on the field level (RAs). Thus, the cost of changes in the software on the proprietary controls is minimized.

The configuration, changes, and adaptation of the control software (reconfigurability and reusability) are manageable by calling functions according to each manufacturer specification (CPPS) and adapting parameters or variables at runtime (Req1.4). An agent (AMS) retrieves status information from

all controllers containing the state of the plant and the processing progress. Based on this information, it decides the strategy of a production unit (Req1.5). Besides the agent (PAs) extensive knowledge about the process data, there exists an encapsulation to the overall network information (AMS). Over LAN, Wi-Fi, or mobile data connections, the current production time and the price of the service are provided for all participating agents (RAs, PAs, CAs, and the AMS), as shown in Fig. 10. An Internet server is required for linking various transport and production units via the Internet. The server (in addition to the infrastructural facilities) also creates agents instances (e.g., PAs) accessible on the cloud. Therefore, different hardware platforms, e.g., PC and PLC, are connected via the Internet. The open protocol of the MAS platform (based on TCP/IP) enables connections to other implementations (e.g., based on C++).

In Fig. 12, the agent-based CPPS architecture is shown considering the Functional and Communication Layers (regarding RAMI4.0) of RIAN. The proposed architecture was used to implement a distributed production environment on Automatica fair [69] as a collaboration of multiple companies (e.g., Martin Engineering, Schunk, Beckhoff, and others) in different exhibition halls (A4, A5, and B5). By using the proposed MAS approach, companies were able to realize the communication, design, and application with a specific implementation of different hardware and software platforms. Each hall represents an administration shell of an I4.0 component (CPPS A-E). In the industrial context, each hall could be represented by different worldwide plants, which collaborate in a unique production process (see Fig. 12).

The industrial partner of RIAN confirmed that they were amazed by the effectiveness and ease of the collaboration [69]. They implemented all necessary functionality needed to manage the production steps at the different facilities in the exhibition halls. The RIAN implementation required less than 3 months with less of four to six developers per academic and industrial partner.

7 Conclusion

Design patterns can help the MAS developers to set up their architectures with prepared solutions also for manufacturing control. They could design their own MAS in accordance with accepted MAS patterns in industry to ease the application of CPPS. Classification criteria also could aid in the initial information organization of design pattern, since there are many different approaches for MAS and automation domain.

Thanks to the preliminary analysis [18], and based on works by Lüder et al. [16] and Leitao et al. [8], this paper's authors have developed 13 classification criteria for MAS patterns. More than 20 MAS patterns were classified with the derived criteria for MAS revealing different terminologies, as well as new criteria to classify sub-agents. A CPPS architecture for manufacturing control for I4.0 components—regarding the RAMI 4.0—based on four sub-agents, was identified from the analysis of design patterns. They are as following, Resource Agent (RA), Process Agent (PA), Agent Management System (AMS), and the Communication Agent (CA). According to the proposed design pattern, these sub-agents should be considered mandatory for the agent-based CPPS architecture, since each of them fulfills fundamental functionalities. Regarding functional requirements identified, these are grouped into a Resource Access (RA*), Knowledge Base (KB), Coordination Process (CP), and Communication Interface (CI). All sub-agents often use the Knowledge Base in order to infer formal methods for its implementation. The Resource Access is a very necessary functional requirement to acquire and process data from physical resources with hard real-time capabilities; as well, the RA typically covers the lowest 0–2 automation levels. The Coordination Process contains procedures and sub-agents' delimitations for managing MAS in a higher hierarchy. This functionality is usually included in AMS on L1-L2 and PA on L2-L3 automation levels. The CI enables open communication between automation levels with multiple data formats, supporting the AMS's and CA's (L1-L3) tasks. CI's functionalities are frequently represented in all automation levels.

The proposed pattern of the four sub-agents can deliver relevant MAS features for developers in order to support new sub-models' designs with similar solutions. In addition, the pattern provides a proper information in order to reduce time to compare similar researches. This pattern provides MAS architecture that can help to cope with production complexity and adaptively as required by CPPS.

This document addresses the industrial sectors in multiple production systems domain: discrete manufacturing, continuous process, hybrid production. In addition, it takes into consideration the specificities of different MAS application (e.g., material flow systems, real-time capabilities, agent communications, and smart grids) and serves the needs of the RAMI 4.0 involving several partners and normativity.

To identify more patterns and to allow easier identification of such a pattern, in the future, the other 16 patterns from the FA 5.15 need to be analyzed with their authors support. Especially, non-functional requirements will be part of the further work of this manuscript's authors, since these requirements remaining can map quality attributes to the identified and newly MAS patterns.

Acknowledgments The authors especially thank the members of the technical committee 5.15 "Agent systems" of the Society Measurement and Automatic Control (GMA) within the Society of German Engineers (VDI) and German Electrical Engineers (VDE) for their close assistance. Also, regarding the specific feedbacks, this manuscript's authors would like to highlight the contribution of the colleagues: Aicher T., Badr I.,

Brehm R., Bruce-Boye C., Fischer J., Hoffmann M., Hofmann P., Pech S., Rauscher H., Redder M., Rehberger S., Schütz D., Theiss S., and Ulewicz S. Finally, Luis Alberto Cruz Salazar thanks the Colombian government grant of the department of science COLCIENCIAS –under grant "Convocatoria 756 Doctorados en el exterior"– and the Antonio Nariño University under grant "Programa de Formación de Alto Nivel - PFAN".

References

1. DIN (2016) Reference Architecture Model Industrie 4.0 (RAMI4.0). https://www.din.de/en/wdc-beuth:din21:250940128
2. Cheng Y, Zhang Y, Ji P et al (2018) Cyber-physical integration for moving digital factories forward towards smart manufacturing: a survey. Int J Adv Manuf Technol:1–13
3. Lu Y, Morris K, Frechette S (2016) Current standards landscape for smart manufacturing systems. Natl Inst Stand Technol NISTIR 8107:39
4. Cruz SLA Vogel-Heuser B (2017) Comparison of agent oriented software methodologies to apply in cyber physical production systems. In: 15th international conference on industrial informatics, INDIN. IEEE. Emden, Germany, pp 65–71. https://doi.org/10.1109/INDIN.2017.8104748
5. Ribeiro L (2017) Cyber-physical production systems' design challenges. In: IEEE 26th International symposium on industrial electronics, ISIE, pp 1189–1194
6. Xu X (2017) Machine Tool 4.0 for the new era of manufacturing. Int J Adv Manuf Technol 92:1893–1900
7. Ulewicz S, Schütz D, Vogel-Heuser B (2013) Flexible real time communication between distributed automation software agents. In: 22nd international conference on production research, ICPR 22, pp 1–7
8. Leitão P, Karnouskos S (2015) Industrial agents: emerging applications of software agents in industry, 1st edn. Elsevier, Amsterdam
9. Leitão P, Karnouskos S, Ribeiro L et al (2016) Smart agents in industrial cyber physical systems. Proc IEEE 104:1086–1101
10. Juziuk J, Weyns D, Holvoet T (2014) Design patterns for multi-agent systems: a systematic literature review. In: Agent-oriented software engineering: reflections on architectures, methodologies, languages, and frameworks, pp 79–99
11. Lüder A, Peschke J, Sanz R (2010) Design patterns for distributed control applications. In: Kühnle H (ed) Distributed manufacturing: paradigm, concepts, solutions and examples. Springer London, London, pp 155–175
12. Ribeiro L, Hochwallner M (2018) On the design complexity of cyber-physical production systems. Complexity 2018:1–13. https://doi.org/10.1155/8503
13. Cruz SLA, Rojas AOA (2014) The future of industrial automation and IEC 614993 standard. III international congress of engineering mechatronics and automation. CIIMA.:1–5. https://doi.org/10.1109/CIIMA.2014.6983434
14. Dai W, Vyatkin V (2013) A component-based design pattern for improving reusability of automation programs. In: IECON proceedings (industrial electronics conference). pp 4328–4333
15. Fuchs J, Feldmann S, Legat C, Vogel-Heuser B (2014) Identification of design patterns for IEC 61131-3 in machine and plant manufacturing. In: IFAC-PapersOnLine. pp 6092–6097
16. Lüder A, Calá A, Zawisza J, Rosendahl R (2017) Design pattern for agent based production system control—a survey. In: 13th IEEE conference on automation science and engineering, CASE. pp 717–722
17. Eckert K, Fay A, Hadlich T, et al (2012) Design patterns for distributed automation systems with consideration of non-functional requirements. In: IEEE International conference on emerging technologies and factory automation, ETFA. pp 1–9
18. Vogel-Heuser B, Ryashentseva D, Cruz S. LA, et al (2018) Agentenmuster für flexible und rekonfigurierbare Industrie 4.0/CPS-Automatisierungs-bzw. Energiesysteme (agent pattern for flexible and reconfigurable industry 4.0/CPS automation or energy systems). In: VDI-Kongress automation. VDI Verlag GmbH, Düsseldorf, pp 1119–1130
19. Rehberger S, Spreiter L, Vogel-Heuser B (2017) An agent-based approach for dependable planning of production sequences in automated production systems. At-Automatisierungstechnik 65:766–778
20. Farid AM, Ribeiro L (2015) An axiomatic design of a multiagent reconfigurable mechatronic system architecture. IEEE Trans Ind Informatics 11:1142–1155. https://doi.org/10.1109/TII.2015.2470528
21. Haoues M, Sellami A, Ben-Abdallah H, Cheikhi L (2017) A guideline for software architecture selection based on ISO 25010 quality related characteristics. Int J Syst Assur Eng Manag 8:886–909
22. Trentesaux D (2009) Distributed control of production systems. Eng Appl Artif Intell 22:971–978. https://doi.org/10.1016/j.engappai.2009.05.001
23. Kosanke K, Vernadat F, Zelm M (2015) Means to enable enterprise interoperation: CIMOSA object capability profiles and CIMOSA collaboration view. Annu Rev Control 39:94–101. https://doi.org/10.1016/j.arcontrol.2015.03.009
24. Morgan J, O'Donnell GE (2015) The cyber physical implementation of cloud manufactuirng monitoring systems. In: Procedia CIRP, vol 33, pp 29–34
25. Leitão P, Restivo F (2006) ADACOR: a holonic architecture for agile and adaptive manufacturing control. Comput Ind 57:121–130
26. Holvoet T, Weyns D, Valckenaers P (2009) Patterns of delegate MAS. In: SASO 2009—3rd IEEE international conference on self-adaptive and self-organizing systems
27. Wannagat A (2010) Development and evaluation of agent-based automation systems in order to increase the flexibility and reliability of manufacturing plants. PhD thesis, Faculty of Mechanical Engineering, Technical University of Munich
28. Schütz D, Schraufstetter M, Folmer J, et al (2011) Highly reconfigurable production systems controlled by real-time agents. In: IEEE international conference on emerging technologies and factory automation, ETFA. pp 1–8
29. Legat C, Lamparter S, Vogel-Heuser B (2013) Knowledge-based technologies for future factory engineering and control. In: Studies in computational intelligence. pp 355–374
30. Andrén F, Stifter M, Strasser T (2013) Towards a semantic driven framework for smart grid applications: model-driven development using CIM, IEC 61850 and IEC 61499. Informatik-Spektrum 36:58–68
31. Cruz SLA, Mayer F, Schütz D, Vogel-Heuser B (2018) Platform independent multi-agent system for robust networks of production systems. IFAC-PapersOnLine 51:1261–1268. https://doi.org/10.1016/j.ifacol.2018.08.359
32. Fischer J, Marcos M, Vogel-Heuser B (2018) Model-based development of a multi-agent system for controlling material flow systems. Autom 66:438–448
33. Karnouskos S, De Holanda TN (2009) Simulation of a smart grid city with software agents. In: UKSim 3rd European modelling symposium on computer modelling and simulation, EMS. pp 424–429

34. Lüder A, Schleipen M, Schmidt N, et al (2018) One step towards an industry 4.0 component. In: 13th IEEE conference on automation science and engineering, CASE. pp 1268–1273
35. Nieße A (2015) Verteilte kontinuierliche Einsatzplanung in Dynamischen Virtuellen Kraftwerken (distributed continuous resource planning in dynamic virtual power plants). PhD thesis, Faculty II—Computer Science. Economics and Law, Carl von Ossietzky University of Oldenburg, Oldenburg
36. Brussel H Van, Wyns J, Valckenaers P, et al (1998) Reference architecture for holonic manufacturing systems: (PROSA). Comput Ind 37:255–274
37. Regulin D, Schütz D, Aicher T, Vogel-Heuser B (2016) Model based design of knowledge bases in multi agent systems for enabling automatic reconfiguration capabilities of material flow modules. In: 12th IEEE conference on automation science and engineering, CASE. pp 133–140
38. Ryashentseva D (2016) Agents and SCT based self* control architecture for production systems. PhD thesis, Faculty of Mechanical Engineering, Otto-von-Guericke University Magdeburg
39. Theiss S, Kabitzsch K (2017) A Java software agent framework for hard real-time manufacturing control. - Autom
40. Vogel-Heuser B, Diedrich C, Pantförder D, Göhner P (2014) Coupling heterogeneous production systems by a multi-agent based cyber-physical production system. In: 12th IEEE international conference on industrial informatics, INDIN. pp 713–719
41. Hoffmann M (2017) Adaptive and scalable information modeling to enable autonomous decision making for real-time interoperable factories. PhD thesis, Faculty of Mechanical Engineering, RWTH Aachen
42. Pech S, Göhner P (2010) Multi-agent information retrieval in heterogeneous industrial automation environments. In: Cao L, Bazzan ALC, Gorodetsky V et al (eds) Lecture notes in computer science. Springer Berlin Heidelberg, Berlin, pp 27–39
43. Shehory O, Sturm A (2014) Agent-oriented software engineering: reflections on architectures, methodologies, languages, and frameworks, 1st edn. Springer-Verlag Berlin Heidelberg, Berlin Heidelberg
44. Cruz SLA (2018) Automatización Industrial Inteligente: Una estructura de control desde el paradigma holónico de manufactura (intelligent industrial automation: a control structure since the holonic manufacturing paradigm). Editorial Académica Española, Beau Bassin, Mauritius
45. Indriago C, Cardin O, Rakoto N, Castagna P, Chacòn E (2016) H2CM: a holonic architecture for flexible hybrid control systems. Comput Ind 77:15–28
46. Nieße A, Tröschel M, Sonnenschein M (2014) Designing dependable and sustainable smart grids—how to apply algorithm engineering to distributed control in power systems. Environ Model Softw 56:37–51. https://doi.org/10.1016/j.envsoft.2013.12.003
47. Platform Industrie 4.0 (I4.0) (2018) The structure of the administration shell: trilateral perspective from France, Italy and Germany. 64
48. Folmer J, Schütz D, Schraufstetter M, Vogel-Heuser B (2012) Konzept zur erhöhung der flexibilität von produktionsanlagen durch einsatz von rekonfigurierbaren anlagenkomponenten und echtzeitfähigen softwareagenten (concept for increasing the flexibility of production plants by using reconfigurable plant components). In: Informatik aktuell
49. Priego R, Iriondo N, Gangoiti U, Marcos M (2017) Agent-based middleware architecture for reconfigurable manufacturing systems. Int J Adv Manuf Technol 92:1579–1590. https://doi.org/10.1007/s00170-017-0154-z
50. Hanisch HM, Lobov A, Lastra Martinez JL et al (2006) Formal validation of intelligent-automated production systems: towards industrial applications. Int J Manuf Technol Manag 8:75
51. Rauscher M (2015) Agent based consistency check of heterogeneous models in industrial automation. PhD thesis, Faculty 5. Computer Science, Electrical Engineering and Information Technology, University of Stuttgart, Stuttgart
52. Hofmann P (2017) A fuzzy belief-desire-intention model for agent-based image analysis. In: Ramakrishnan S (ed) Modern fuzzy control systems and its applications. IntechOpen, Rijeka
53. Legat C, Vogel-Heuser B (2014) A multi-agent architecture for compensating unforeseen failures on field control level. In: Studies in computational intelligence. pp 195–208
54. Lee J, Bagheri B, Kao HA (2015) A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manuf Lett
55. Monostori L (2014) Cyber-physical production systems: roots, expectations and R&D challenges. Procedia CIRP 17:9–13
56. Komma VR, Jain PK, Mehta NK (2011) An approach for agent modeling in manufacturing on JADE^{TM} reactive architecture. Int J Adv Manuf Technol 52:1079–1090. https://doi.org/10.1007/s00170-010-2784-2
57. Vogel-Heuser B, Kegel G, Bender K, Wucherer K (2009) Global information architecture for industrial automation. Atp 1:108–115
58. Badr I (2011) Agent-based dynamic scheduling for flexible manufacturing systems. PhD thesis, Faculty 5. Computer Science, Electrical Engineering and Information Technology, University of Stuttgart, Stuttgart
59. Brehm R, Redder M, Flaegel G, Menz J, Bruce-Boye C et al (2019) A framework for a dynamic inter-connection of collaborating agents with multi-layered application abstraction based on a software-bus system. In: Czarnowski I., Howlett R., Jain L., Vlacic L. (eds) Intelligent Decision Technologies 2018. KES-IDT 2018 2018. Smart Innovation, Systems and Technologies, vol 97. Springer, Cham
60. Platform Industrie 4.0 (I4.0) (2017) Relationships between I4.0 components—composite components and smart production
61. Yao X, Lin Y (2016) Emerging manufacturing paradigm shifts for the incoming industrial revolution. Int J Adv Manuf Technol. 85:1665–1676. https://doi.org/10.1007/s00170-015-8076-0
62. Barenji RV, Barenji AV, Hashemipour M (2014) A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. Int J Adv Manuf Technol 71:1773–1791
63. Vargas C, Langfinger M, Vogel-Heuser B (2017) A tiered security analysis of industrial control system devices. In: 15th international conference on industrial informatics, INDIN. pp 399–404
64. Frayret JM et al (2004) Coordination and control in distributed and agent-based manufacturing systems. Prod Plan Control 15:42–54. https://doi.org/10.1080/09537280410001658344
65. Leitão P (2009) Agent-based distributed manufacturing control: a state-of-the-art survey. Eng Appl Artif Intell 22:979–991. https://doi.org/10.1016/j.engappai.2008.09.005
66. Garcia Valls M, Lopez IR, Villar LF (2013) ILAND: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. IEEE Trans Ind Informatics. 9:228–236. https://doi.org/10.1109/TII.2012.219866
67. Jimenez JF, Bekrar A, Zambrano-Rey G, Trentesaux D, Leitão P (2017) Pollux: a dynamic hybrid control architecture for flexible job shop systems. Int J Prod Res. 55:4229–4247. https://doi.org/10.1080/00207543.2016.1218087
68. Váncza J, Monostori L (2017) Cyber-physical manufacturing in the light of professor Kanji Ueda's Legacy. In: Procedia CIRP
69. Vogel-Heuser B, Bauernhansl T, ten HM (2017) Handbuch Industrie 4.0 Bd.2 (manual of industry 4.0 Vol.2), 2nd edn. Springer Berlin Heidelberg, Berlin, Heidelberg

Publication IV (MARIANNE)

at – Automatisierungstechnik. 70/6 (2022), pp. 580-598.

https://doi.org/10.1515/auto-2022-0008

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-

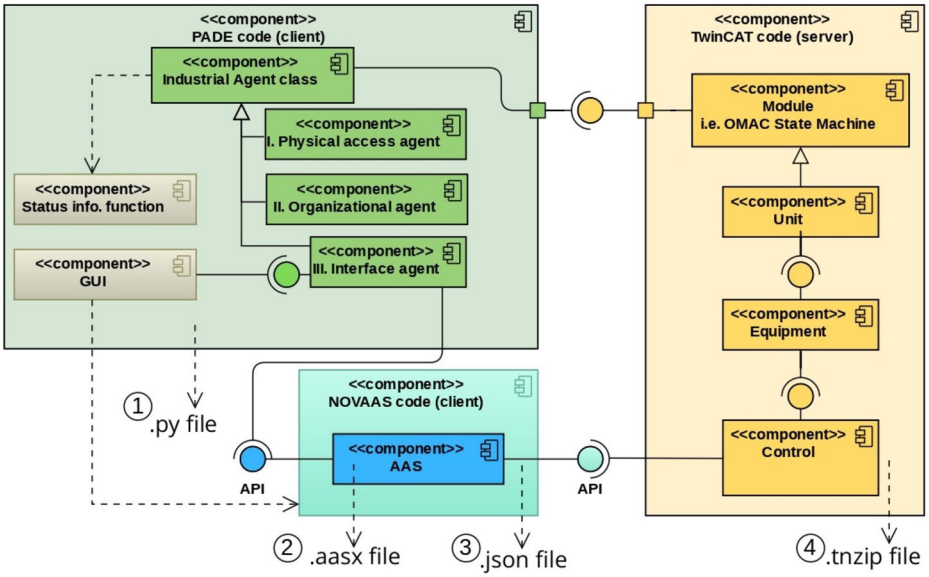

Figure 1: MARIANNE UML Component diagram. Codes for 1) PADE; 2) AASX Package Explorer; 3) Node-RED; and 4) TwinCAT.

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

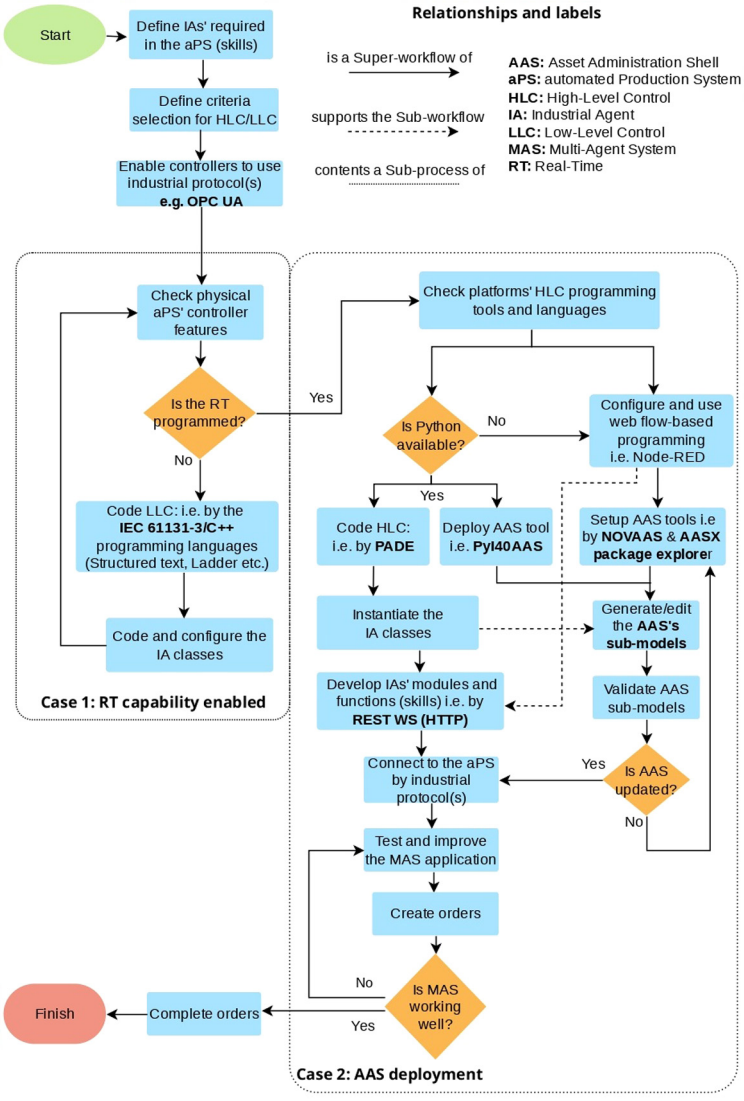

Figure 2: MARIANNE's guideline implementation flowchart and its relationships.

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

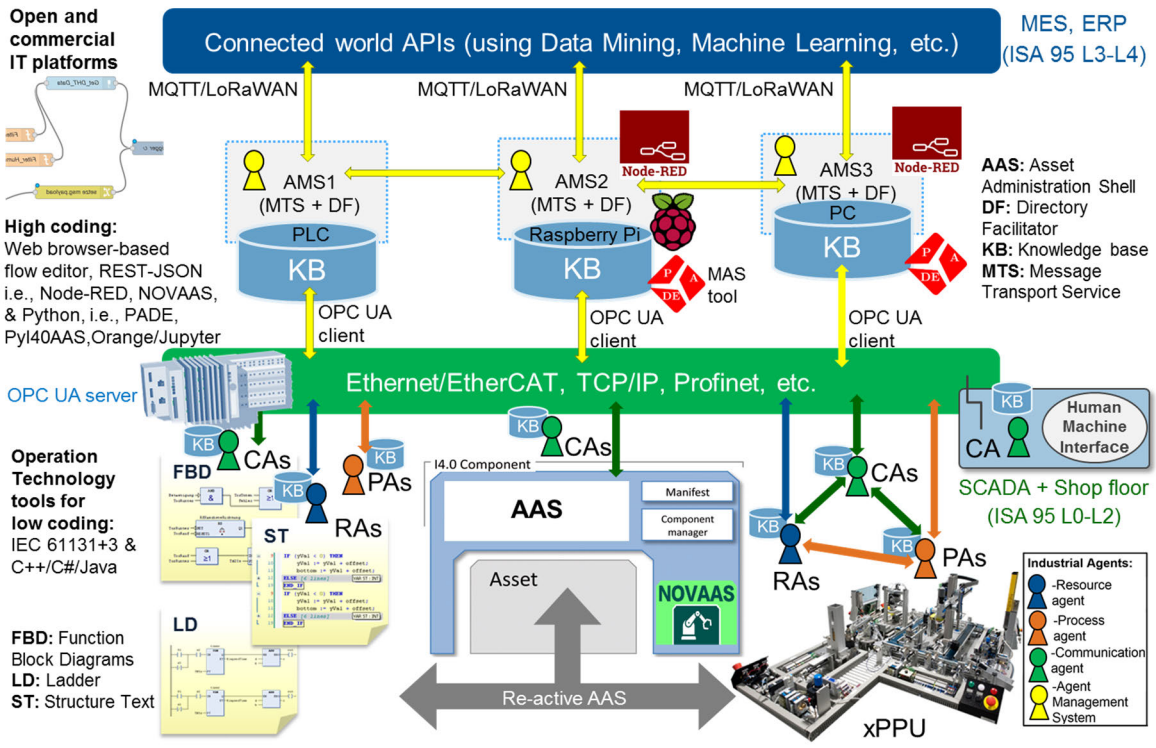

Figure 3: General landscape of the I4.0 scenario proposed with their I4.0 components and their IT/OT technologies.

[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

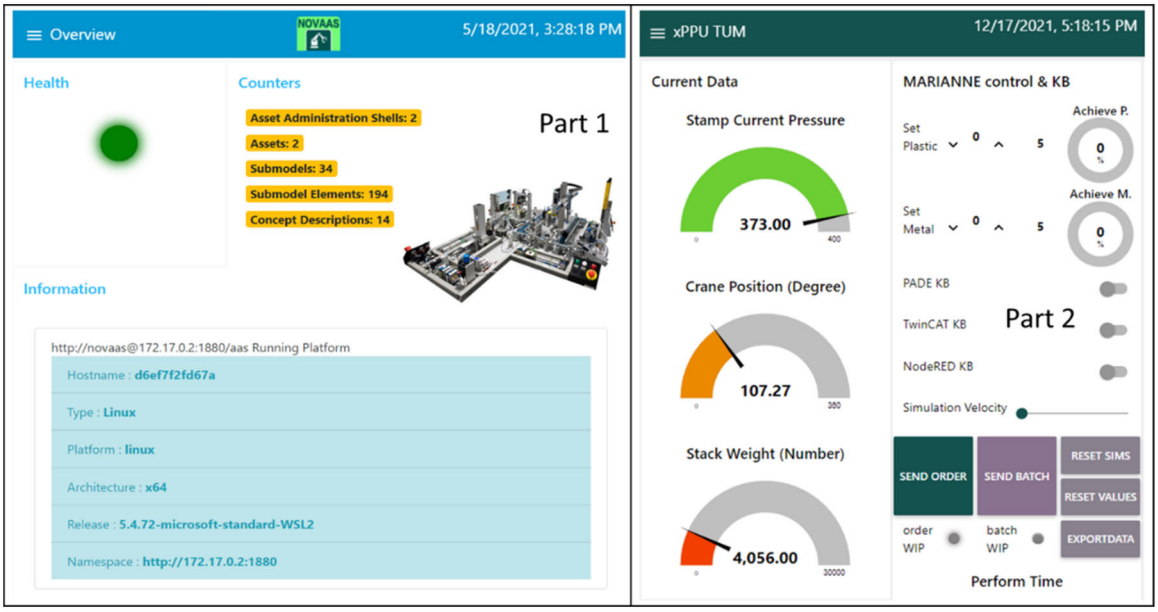

Figure 4: Dashboard and interfaces (GUI) of the implementation: 1) xPPU's NOVAAS dashboard, 2) CPPS' HMI in Node-RED.

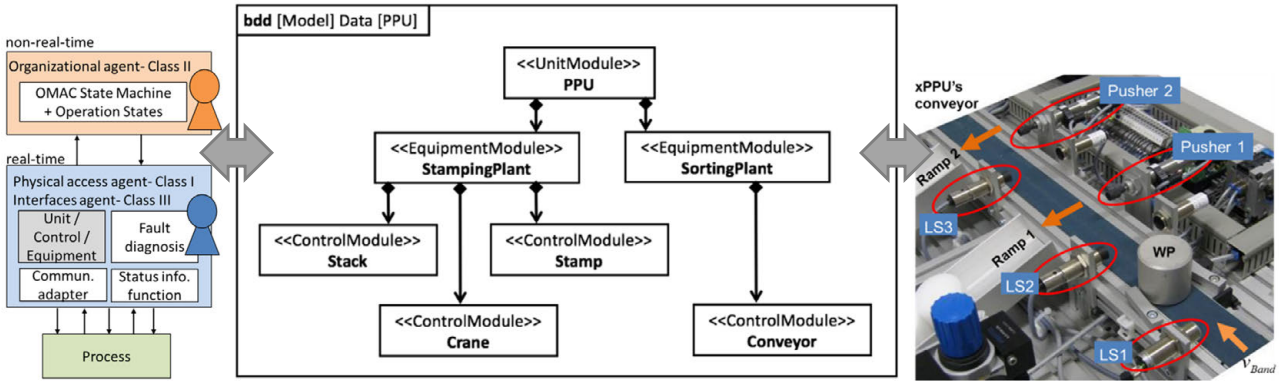

Figure 5: IAs and the SysML block definition diagram of separating the xPPU's HLC/LLC into modules. Adapted from [1].

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.

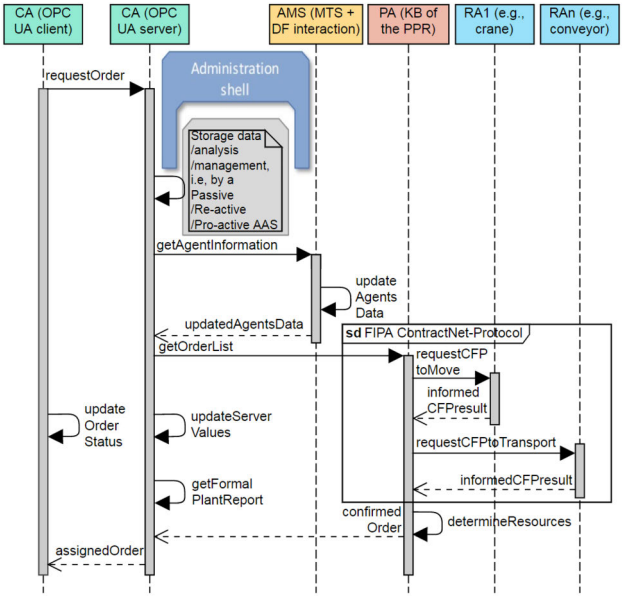

Figure 6: Sequence diagram to detail the IA patterns interactions in the CPPS network. CFP means "Call For Proposal" and refers to FIPA (see *FIPA Iterated Contract Net Interaction Protocol Specification* [10]).

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].

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 In-

dustrial 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

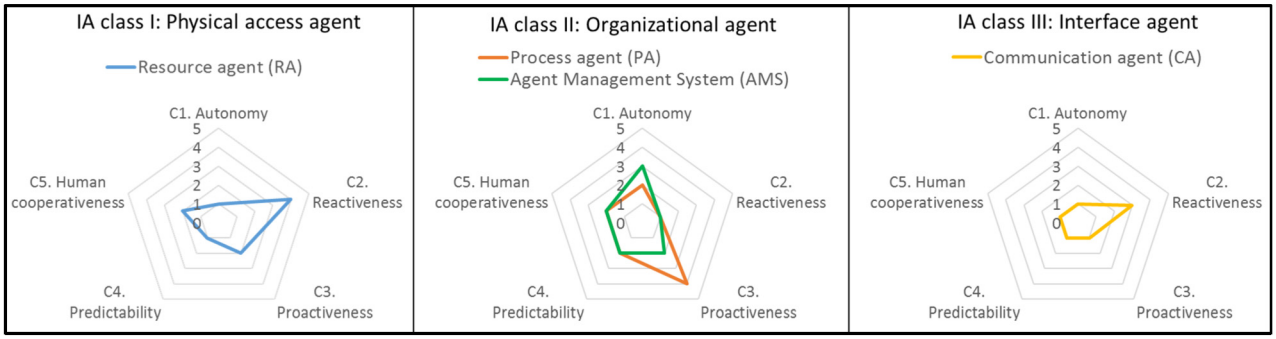

Figure 7: Industrial agents applied in MARIANNE architecture and their level of Industrial AI characteristics (see Table 5).

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.

Publication V (Agent4.0)

IEEE 20th International Conference on Industrial Informatics "INDIN" (2022), pp. 27 32.
https://doi.org/10.1109/INDIN51773.2022.9976159

Industrial Artificial Intelligence: A Predictive Agent Concept for Industry 4.0

Luis Alberto Cruz Salazar[1] (Student member, IEEE), Birgit Vogel-Heuser[1,2] (Senior Member, IEEE)

[1]*Institute of Automation and Information Systems, Department of Mechanical Engineering, TUM School of Engineering and Design, Technical University of Munich (TUM)*

[2]*Core Member of MDSI and Member of MIRMI*

(luis.cruz@tum.de; vogel-heuser@tum.de)

***Abstract*— "Artificial Intelligence in Industry 4.0", a technical report published by the working groups "Technological and Application Scenarios" and "Artificial Intelligence" (AI) of the Industry 4.0 (I4.0) platform, presents an innovative Industrial AI concept. Above all, it concludes that I4.0 experts and scientists must become accustomed to the behavior of autonomous AI-controlled systems, collaborate with them and comply with learnability requirements (predictability). Industrial AI instantly raises a set of concerns about existing norms and new standardizations. These frequently provide guidelines and, in some cases, offer procedures and implementations using design patterns. One way to produce AI in I4.0 systems is through Industrial Agents (IAs) due to their natural autonomy and additional intelligent characteristics, e.g., reactiveness, proactiveness, and human cooperativeness. Multi-Agent Systems (MASs) are particularly well suited for representing distributable AI that can develop I4.0 components being applied to various I4.0 scenarios. Considering the properties of IAs and the corresponding standards, an MAS architecture is used to understand the aspects of the flexible, intelligent, and automated Cyber-Physical Production System (CPPS). This article proposes a predictive IA for I4.0 (Agent4.0) to an agent-based CPPS architecture, leveraging IA design patterns and logical structure for implementing MAS. As a result, relevant standardized IA design patterns for I4.0 show how MAS can be created with the help of the Industrial AI requirements and Agent4.0 skills (functions) identified.**

***Keywords*— Agent4.0, Artificial Intelligence, Cyber-Physical Production Systems, Industry 4.0, Industrial Agents**

I. INTRODUCTION

Several industrial partners, academics, and researchers speak about distributed Artificial Intelligence (AI) and its potential benefits through Industrial Agents (IAs) in various domains, e.g., Manufacturing, Logistics, Smart Grids [1], [2]. Nevertheless, what exactly is an agent for Industry 4.0 (I4.0), and what are its AI characteristics?

From its Multi-Agent System (MAS) concept roots, an agent is an entity that "just acts" since the word "agent" is derived from the Latin verb "agere", which means "to do" [3]. Meanwhile, recently in the I4.0 context, according to German agents FA 5.15 VDI/VDE experts' standardization, an IA encapsulates hardware or software to reach objectives through its autonomous behavior by interacting with its environment and with other IAs [4], [5]. At the same time, TC-IA, by the IEEE P2660.1 working group, normalized the IA as an intelligent, agile, and robust software that describes and manages the functionalities and capabilities of industrial units [6]. Moreover, MAS experts define AI as supplemental technical systems with the capacity to process tasks independently and efficiently [7].

Nonetheless, those definitions are limited; they do not answer how the MASs acquire their intelligence and apply it to I4.0. Then, there are multiple and generally accepted definitions of both terms (agents and AI), which are ambiguous and far from identical within their communities.

AI specialists usually use agents to describe technologies that complete multiple tasks and can be trained with external data (from sensors or databases). The acquisition of data and *Machine Learning* methods like Artificial Neural Networks, Fuzzy Logic, and Linear Regression can support decisions derived from the information already known to the intelligent system. Depending on the lower complexity of Industrial AI characteristics such as autonomy and reactiveness, IAs algorithms can execute the actions that are considered "the best" for smart systems. However, most sophisticated learning algorithms must enable intelligent systems to learn from online/offline operations, e.g., predictability definition [3], [8]. Then, the trained models, data, and knowledge should be extended and made reusable [7]. In order to train IA's *Knowledge Base* models, highly complex and comprehensive data is required, resulting in a predictive IA [3].

This paper proposes a generic predictive IA concept for I4.0 (*Agent4.0*) that applies a supervised learning method to increase the predictability of automated production systems. The Agent4.0 can be implemented on a high variety of platforms through an MAS architecture to fulfill industrial use cases and increase interoperability. MAS can be applied to achieve a network of agent-based Cyber-Physical Production Systems (CPPSs) for I4.0 use cases. CPPSs are industrial *Cyber-Physical Systems* that are usually defined as the integration of virtual (software) with physical (hardware) processes [1]. The MAS architecture is derived from the already existing IA standards of VDI/VDE and TC-IA experts.

In contrast to other platforms such as the often-used Java Agent Development Environment "JADE" [9], [10], the proposed platform is the Python Agent Development "PADE" framework [11]. PADE is a good alternative for IA's development and execution platforms, allowing MAS to be written in a modern programming language by Python with an object-oriented paradigm, simple learning, and resources for distributed system development.

This work recognizes five Industrial AI characteristics for realizing a CPPS (sec. II) and their related works (sec. III). The study suggests an MAS including a logical architecture and software tools (sec. IV). This MAS already partially meets the AI requirements, and the others are assessed using a predictive

IA concept (sec. V). The last section (sec. VI) includes a summary and an outlook.

II. Industrial AI Characteristics and IA Patterns

This section summarizes five Industrial AI characteristics (C1-C5) and exemplifies the main IA patterns extended from [12]. After that, sec. III analyses their related work and gap.

(C1) Autonomy: An IA can independently master uncertain conditions in a delimited and automated manner, achieving its objectives systematically without external or human intervention. VDI/VDE experts define the Resource Agent (RA) that is often located in the lowest MAS's hierarchy without full autonomy due to their submissive heterarchy in the low-level control [4], [5].

(C2) Reactiveness: An IA can respond to a request to process its environment information from low- to high-level control (observation and communication responsiveness in real-time). MAS experts define the Communication Agent (CA) that may convert proprietary interfaces into multiple protocols [4], [5], e.g., a communication interface by TCP/IP (often reactive with the real-time capability).

(C3) Proactiveness: An IA can take the initiative for deciding and processing information whilst pursuing a goal (reasoning for deliberative tasks). Kovalenko et al. define the Product Agent that can apply a systematic, model-based optimization method during the decision-making process to achieve a common goal [9]. In some MAS compared in [4], [5], the Product Agent is often replaced by the Process Agent (PA).

(C4) Predictability: An IA can predict the subsequent outcomes of actions given the actions in the previous tasks and the self-learning (from past information). The Learning agent concept has a distinctive "learning element". It is often non-real-time [3], which is in charge of improving the agent's performance based on feedback and determining how the element should be updated to perform CPPS better in the future; see CPPS predictability definition in [8].

(C5) Human cooperativeness: An IA can apply the concept of *human-in-the-loop*. Karnouskos et al. introduce that IAs may be used to accomplish I4.0's idea of human-in-the-loop, in which shop-floor operators interact with their environment and CPPS with the help of IAs [1].

A traditional IA typology refers to response time and main behavior (or feature) by three types [12]: the Reactive IA (reacts to perception), the Proactive IA (performs deliberative actions), and the Predictive IA (anticipates by learning tasks). Here, four IA classes are categorized regarding their main capabilities, and Industrial AI characteristics are determined: Class I; Physical access agent (abstracting and connecting heterogeneous production equipment with the MAS), Class II; Organizational agent (managing and organizing the operation of the MAS), Class III; Interface agent (handling the interface and functionality of low- and high-level control) and Class IV; Human agent (acting humans as agents).

Figure 1 focuses on these IA classes (cp. center), integrating the AI typology (response time and main behavior, cp. top) and the level of Industrial AI characteristics (cp. bottom).

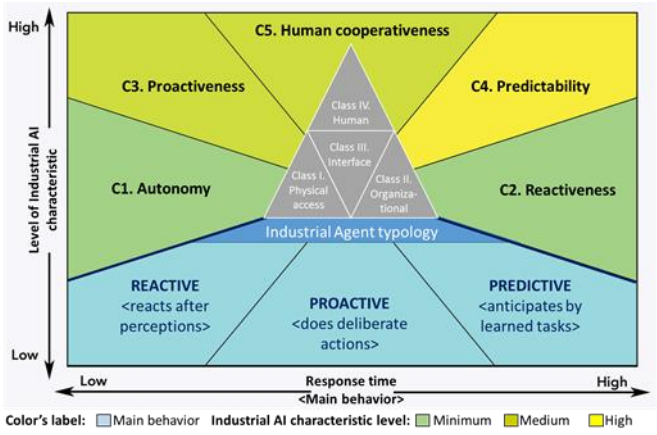

Fig. 1. Industrial agents' classes: typology and levels of Industrial AI characteristics C1-C5 (adapted from [12]). Inspired by the mindset state in terms of challenge/skills levels from Csikszentmihalyi's flow model.

III. Related Works

IAs are usually considered a type of distributed AI, but these are considered to be autonomous and reactive software (cp. C1-C2), as introduced in [6]. In contrast, Russell and Norvig introduce the *Rational Agent* concept as part of their AI categories (cp. C3), meaning it acts in order to achieve the best result or the best-expected result in the case of ambiguity [3]. They developed a taxonomy for the following AI system' categorization: i) *thinking humanly*, e.g., artificial neural networks and other cognitive methods (cp. C4); ii) *acting humanly* (cp. C5), e.g., humanoid robots with natural language processing; iii) *thinking rationally*, e.g., expert systems or rules of inference and optimization (cp. C4); and iv) *acting rationally*, e.g., intelligent software agents that are expected to achieve goals (cp. C1-C3).

While those definitions are helpful, they are far too general. Analyzing the roots of the IAs concept and AI facilitates contrasting both meanings with recent interpretations. Most recently, the authors Kaplan and Haenlein redefined AI as the ability of a system to understand external input accurately, learn from it, and apply what it has learned to fulfill specified goals and tasks through flexible adaptation (cp. C1-C4)[13]. Meanwhile, Lee et al. distinguish Industrial AI as a systematic discipline that integrates technical elements such as analytics, meaning not only the algorithm but also the algorithm's implementation modeling by humans (cp. C5) in certain settings and for specific goals [14]. These authors demonstrate in which ways Industrial AI enables the system to be self-aware, self-adaptive, and self-configuring (cp. C1-C4), easing the adoption of *Digital Twins.*

In order to address IA definitions, the root taxonomies of the word "agent" are analyzed. Unland defines an IA as a software entity that autonomously represents and manages industrial units' functionalities and capabilities (cp. C1) [6]. The same author defines a deliberative and reactive agent as the extreme

points within the range for the smartness of IAs (cp. C2-C3). In both cases, it is possible to replace or combine one deliberative agent with many reactive agents without losing quality [15].

Gangoiti et al. [10] apply those types of IAs by JADE, aiming at adding reactivity and flexibility to CPPS following with the digital twin concept (cp. C2), i.e., by the *Asset Administration Shell* (AAS). Another example is IntraMAS [16], an intralogistics domain model that appears in different forms according to the overall abstraction and proactive control software (cp. C3). Those IAs developers focus on the domain of CPPS to solve I4.0 issues such as autonomy, reactiveness, human collaboration etc. (cp. C1-5).

Most of the preliminary IA design patterns identified are standardized [4], [5], which are only reactive and proactive IAs types (C2-C3), with poor information on predictive IAs (C4). For instance, MAS experts mention this characteristic, but the analyzed IAs patterns do not demonstrate it [4]. The deliberative agents improve from their own experience or offline analyzed actions, for example, the *Learning agent* concept [3] (cp. C4). This distinction increases since probability is not predictability. A classical demonstration explains how the probability that a tossed coin will land up heads is 50%, but in no way can this accurately predict the next flip [17]. In this case, humans can train the agent to predict (cp. C5), on average, how many flips out of 200 will be heads, but it would not be able to predict the next flip. For this reason, a human applying typical machine learning methods are able to get predictive systems (cp. C4).

Most AI researchers generally mean an "agent" to be a computer system that, in addition to having the properties listed above, is developed using theories that are more typically encountered in humans (cp. C5); however, the IAs capabilities are still missing, getting unclear clear IAs types differentiation, i.e., level of IA's autonomy (cp. C1) [7]. Because of these definitions, it is viable that in a control system, a human agent and traditional agent behaviors are defined by a goal-orientated approach (cp. C5). An IA behavior represents a specific combination of tasks, but these tasks are not unique. IAs, therefore, are distinct entities with the ability to take a goal-oriented approach with real-time capabilities (cp. C2) [4], [6].

IAs frequently complete goals with autonomy, similar to humans, but (as far as recent experts demonstrate), they cannot reach the full AI autonomy level yet (cp. C1)[7]. Leitão et al. add modularity, flexibility, robustness, reconfigurability, and responsiveness to the IAs characteristics (cp. C1-C4), which are not entirely part of human behavior (cp. C5) [2]. For instance, Minsky et al. assume the human and computer can be "inextricably intertwined," while the *Human-computer interaction* field commonly refers to them as separate or individual entities [18]. However, humans (operators, developers) can be considered an external part of the MAS regarding the present human agent class.

Thus, the related approaches gap shows that MAS should consider more predictive agents (cp. C4) and human agents (cp. C5). In this sense, IAs can avoid the *AI effect*, which means that technology once thought to be intelligent will become outdated as machines become increasingly capable. As a result, the next sections present agent-based CPPS derived from IA patterns.

IV. IA Patterns and the Predictive IA Concept

This section presents an agent-based CPPS concept focusing on IA patterns and the fulfillment of the Industrial AI characteristics introduced. The *MAS architecture describes* the relations between its IAs and visualizes its design. The *MAS software* subsection describes tools used in order to enable a wide variety of platforms, ensuring CPPS's platform independence and interoperability.

A. MAS architecture derived from IA patterns

Derived from the Foundation for Physical Agents "FIPA" [19], the VDI /VDE 2653-4 [4], and the IEEE P2660.1 [6] norms, the MAS architecture shown in Fig. 2 was developed. Compared to the traditional MAS, the most crucial distinction in this paper's approach is the "learning element" embedded in Agent4.0 in collaboration with the *Administration shell*.

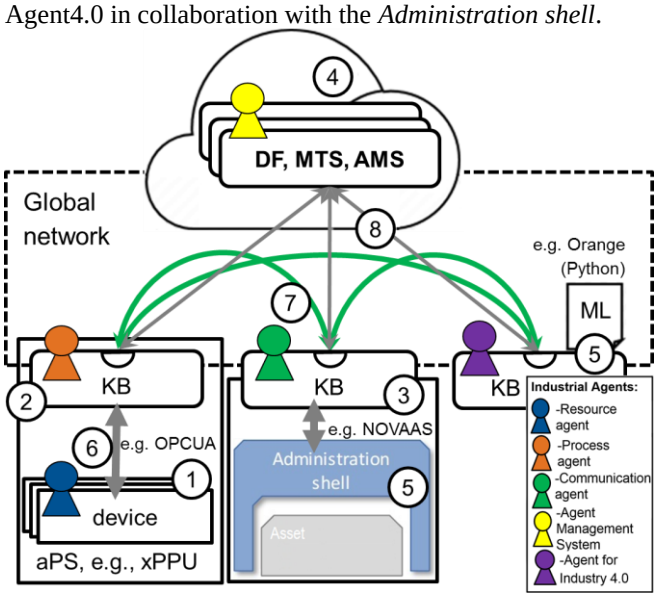

Fig. 2. Simple Logical Architecture of the MAS (extended from [20]). aPS, automated Production Systems; AMS, Agent Management System; DT, Directory Facilitator; KB, Knowledge Base; ML, Machine Learning; MTS, Message Transport System.

Protocols and messages, specified in sec. V (cp. Table II), enable IAs' communication and collaboration. An IA of the CPPS can have one of two aspects to autonomously support a variety of different use cases (cp. C1): each IA represents either a physical system (Fig. 2; 1) or an organizational entity (Fig. 2; 2, 4, 5) For instance, an AMS provides IAs diagnosis services or introduces production requests into the system by managing the PAs. RA usually represents a single device or a group of devices (Fig. 2; 1). RAs and PAs (Fig. 2; 2) can be found in a CPPS network in various numbers, modularizing software or separating hardware, enabling real-time reactiveness (cp. C2), and non-real-time proactiveness (cp. C3), respectively.

An AMS manages the IAs of the entire MAS and works with the *Message Transport System* and the *Directory Facilitator* for

discovery purposes [19]. All IAs register themselves with these organizational entities from FIPA (Fig 2; 8). The directory facilitator is a "yellow pages" service that supplies IA's skills and states [5], [11], [19]. Even if major parts of the FIPA standard are discontinued (deprecated or obsolete), AMS together with message transport system and the directory facilitator are still specifications considered stable and formally published [19].

The AMS supports bidirectional IP-address-to-agent-name mapping. This allows direct communication between agents (Fig. 2; 7), e.g., asking for an IA's IP address (identified by its name) and then using that data to establish a direct connection that is not dependent on other entities. By not defining specific interface communication into CAs (Fig. 2; 3), application, level, platform reactiveness (cp. C2) as well as AAS integration (Fig. 2; 5) are further enhanced. OPC UA (Fig. 1; 6) and other adequate industrial communication protocols are examples of possible implementations. To ensure high availability and predictability from single IAs (cp. C5), Agent4.0 monitors all RAs for availability on a normal behavior (cp. R4). The IAs directories are in the cloud (Fig. 1; 9) over numerous nodes, like internet name services. The logical structure here serves as a reusable specification for applying MAS architecture to extended CPPS domains, as given in [12]. The software used for the agent-based CPPS presented in the next section was selected as a result of this, considering the industrial AI needs (sec. II).

B. MAS software applied

The following technologies were chosen for the AAS deployment based on the previous prerequisites: Node-RED as integration middleware, Apache Kafka for the event/message-based communication pattern, REST services for the Request-Response communication IA patterns, and AutomationML as the supporting format for NOVAAS [21]. The CPPS control programs usually are performed at the low-level control of a development system's hierarchy. Because of the widespread use of PLCs today, it seems reasonable to assume that they are being used in the I4.0 era, as mentioned in [10].

I4.0 components are often founded on current PLC programming technology that relies on typical IEC 61131-3 but also distributed IEC 61499 standard. In addition, an AAS is configured using OPC UA, one of the most recommended AAS technologies [22] for AAS metamodel application. The AAS has been generated by transforming each AAS I4.0 demonstrator into nodes (*.js), as proposed by the authors in [21]. The OPC UA Server retains the description in OPC UA of the AAS, representing the MAS control and the plant's models based on NOVAAS. Following the modeling, the PLCopen XML file is created and imported into TwinCAT3; a PLC *integrated development environment*. The IEC 61131-3 program is then designed, variables are connected to unique in/external modules, and the application is ready to run on the xPPU's PLCs (i.e., real-time capable). The PLCs communicate with the plant by EtherCAT and publish an OPC UA server to contact the middleware, as introduced in [5]. In this scenario, the middleware, which includes an OPC UA client and a web server, is written in Java and runs on a Raspberry Pi. HTTP requests from remote users may be activated until the middleware is up and running. A web application through NOVAAS provides an overview of the AAS representation into the OPC UA client and gives the relevant description with integrated header and body sub-models [21]. Selected open sources and their graphical user interface recommendations for digital twin development in this architecture (based on AAS) are summarized in Table I.

TABLE I. OPEN SOURCE AAS DEVELOPMENT APPLICATIONS

Name	**Main feature**	**Open access* project**
AASX Package Explorer	Use C#, easy GUI, and server with AAS examples are available	https://github.com/admin-shell-io/aasx-package-explorer
BaSyx	Use Eclipse, wide range of functions/GUI	https://projects.eclipse.org/projects/technology.basyx
NOVAAS	Use Node-RED, GUI	https://gitlab.com/novaas
PyI40AAS	Use Python and simple pip, no GUI available	https://git.rwth-aachen.de/acplt/pyi40aas

*A repository is available online to use it. GUI refers to graphical user interface.

V. USE CASE AND EVALUATION

This section discusses the proof of concept in detail (sec. V.A) over scenarios of two agent-based CPPS demonstrators with a common MAS architecture (cp. Fig. 2 adapting [5]). After that, the main results are evaluated in sec V.B.

First CPPS is *myYoghurt*[1] demonstrator and the scenario is a revised solution by the authors in [20]. Second CPPS is the *Extended Pick and Place Unit (xPPU*[2]*)* that comprises storage of workpieces space, fabricating, and logistics. The scenario depicts a single production line's availability during processing of various types of workpieces. Each order of the xPPU is assigned to a new PA that coordinates the xPPU RAs [9]. The xPPU contrasts in aspects like openness via different protocols and modularity by hybrid platforms compared to myYoghurt plant. Table II results from testing those CPPSs and the applicable criteria based on [6]. Regarding IAs platforms, the main advantage of a PADE over a JADE is that it can send serialized objects (FIPA-ACL messages) and has multi-platform interaction based on web technologies [11]. PADE, as well the IA patterns implemented in this section are not limited for manufacturing, but also extended for smart grids domain [4].

TABLE II. QUALITATIVE ASSESSMENT OF IAS INTERFACES OF THE CPPS DEMONSTRATORS. EVALUATION BASED ON [6]

Pattern criteria*	**myYoghurt** [20]	**xPPU demonstrator**
Location	On-device	Hybrid
Interaction mode	Loosely coupled	Tightly coupled
API client	C++/C#, Java (JADE)	REST/JSON, Python (PADE)
Channel	FIPA-ACL, OPC UA	HTTP, FIPA-ACL, OPC UA
*Score**	2.56	3.20

*Criteria recommendation come from [6]. The score value is according to our expertise, providing a qualitative assessment of the IEEE 2660.1 interface practice into the CPPSs.

IEEE 2660.1 interface practices categorize I4.0 plants for example, regarding location (low/high-level control host), interaction mode (low/high-level control interactions),

[1]myYoghurt's web: http://i40d.ais.mw.tum.de

[2]xPPU's web: https://www.mec.ed.tum.de/ais/forschung/demonstratoren/ppu

application programming interface client, channel, score [6], and capability of using hybrid industrial communication protocols (e.g., Profinet, EtherCAT, among others), cp. Table II. This feature enables mixing open communication approaches by running tasks of an OPC UA server. Thus, a generic OPC UA Client will access the OPC UA server navigating the two AASs (from each plant) to learn more about the IEC 61131-3 code and the relationship among variables, in/output devices, and IAs. Moreover, they control internal/external modules, i.e., fault diagnosis, control, etc. A web app has been created, which provides an OPC UA Client to communicate with the OPC UA Server, enabling the web application to provide a web user with a summary of the whole MAS comprised of the PLC and managed by the xPPU demonstrator, as given in [5].

A. Application Example of the Agent-Based CPPS

The application of an agent-based CPPS scenario is summarized in this subsection. In the MAS, PAs have self-initiated behavior that endeavors to define the required services and resources to produce four types of workpieces: aluminum metal (MetalA), bronze metal (MetalB), black plastic (PlasticB), and white plastic (PlasticW). PAs take control of an MAS situation, making early changes (cp. C2), requesting and assigning RAs available to transport workpieces. An agent-soft sensor application has been validated through the RAs of the xPPU. RAs can replace autonomously individual parameters (cp. C1) at runtime to substitute missing actual values through mathematical estimations. For instance, a light barrier sensor for workpiece location is found at the xPPU's conveyors. In case of failure, RAs react to get real-time data (cp. C2) with a calculated accuracy value (values are given in percentage "%") through the workpiece's velocity and travel time.

Agent4.0 is provided with a supervised machine learning algorithm in the primary state to support (but not limited to) long-term production tasks. Then, the workpiece features of time, speed, weight, type of material, and distance traveled are used to estimate a sensor accuracy value (actual value). Agent4.0 uses Linear Regression, due to its low mathematical complexity, to find out the best fitting line to represent the relationship between these workpiece variables. Figure 3 is a screenshot from the *Orange* tool for training the Agent4.0. In practice, Linear Regression is helpful since RAs have an extensive dataset, both in the number of rows and columns, that may lead into a memory overflow (Fig. 3; right).

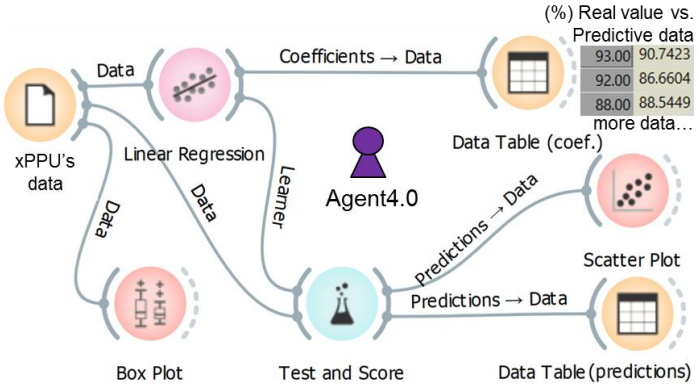

Fig. 3. Training Linear Regression model in Orange (Python-based platform) for Agent4.0 (left) to predict data for sensor values accuracy (right).

Given that the machine learning representation is a linear equation, solving it is as easy as solving an equation for a specific set of inputs to make predictions. Agent4.0 can process the data from RAs and obtain the prediction data accuracy (%) of each agent-based sensor. Not only workpieces positions but multiple methods are trained offline by human operators in Agent4.0 (cp. C5) to predict other target values, e.g., the type of material of the workpieces.

Fig. 4 plots the observations given through machine learning by different figures' colors; thus, a prediction is provided by the black regression line. Most of the observations are close to the regression line; hence we can say our Linear Regression is an adequate model to make accurate predictions (see Fig. 4; correlation coefficient "r" is 0.76). As a result, the initial sensor value is compared to the predicted value, providing an error margin. If the error value is reasonable (Fig. 4; range between blue lines), the estimated sensor value is considered accurate enough and accepted by the agent-based sensor.

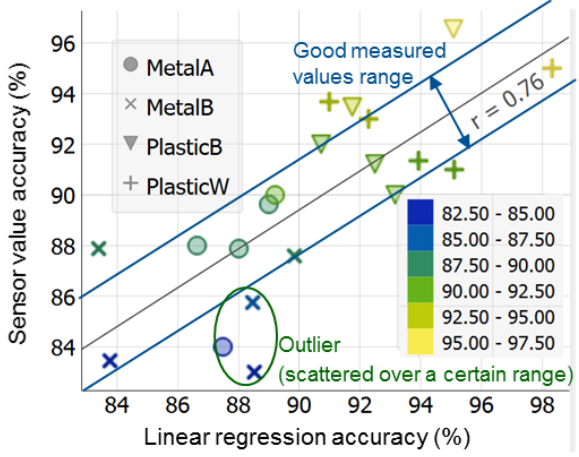

Fig. 4. Linear Regression plot of the Agent4.0. Predicted value and the correlated coefficient "r" (r>0.7 is considered a high correlation). The black line is the best fit of data; blue lines are the confidence band, out of this are outliers.

Through the machine learning model, the Agent4.0 is able to predict data such as the accuracy provided by the agent-based soft sensor from RAs (Fig. 3; right), increasing their availability. The proposed MAS architecture for CPPS was implemented in

a specific application scenario to examine the required application and its Industrial AI characteristics (cp. sec. II). Initially, an academic demonstration of mass-customized items was employed to test the agent-based CPPS, i.e., by the xPPU demonstrator.

B. Agent4.0's Qualitative Evaluation

According to the analyses from this study, current MAS implementation reach different Industrial AI levels (C1-C4) due to Agent4.0 being able to apply various functions (skills) with specific descriptions (cp. Table III and Fig. 5). The terms used in Table III are based on the IEEE P2660.1 guideline [6]; should" indicates it *is recommended to*, "may" indicates it *is permitted to*, and "can" indicates it *is able to*. Fig. 3 and Table III, resume each Agent4.0 skill descriptions that are drawn from the authors' evaluation, extending the preliminary work in [12].

TABLE III. AGENT4.0'S INDUSTRIAL AI CHARACTERISTICS EVALUATION

Industrial AI characteristics (C1-C5) evaluation / **Agent4.0 function (skill*) description**	***Autonomy***	***Reactiveness***	***Proactiveness***	***Predictability***	***Human cooperativeness***
Agent4.0 should increase its initial Knowledge Base competence because of the "learning element" (often non-real-time). Sec IV.A	●		●	●	●
Agent4.0 may operate in a time-predictable way, i.e., enabling short/medium/long-term production tasks. Sec. V.A	●	●		●	
Agent4.0 can predict data valuable to other IAs, by a central learning module. Sec. V.A	●				●
Agent4.0 can apply a supervised learning method, e.g., a Linear Regression algorithm, to achieve its goals. Sec. V.B				●	●
Agent4.0 usually does not fulfill hard/soft real-time requirements because predictability implies learning from the past and being located at the heterarchy top. Sec. II.C4				●	

●: needed. *See other IAs' skills in [12]

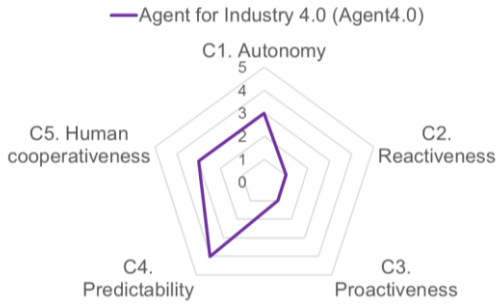

Fig. 5. Agent4.0 and its Industrial AI characteristics (cp. Table III).

VI. SUMMARY AND OUTLOOK

International MAS experts' widespread adoption of the IAs in the CPPS has contributed to the seemingly insurmountable I4.0 paradigm by IA patterns standardization. However, those patterns present a unique opportunity to measure Industrial AI characteristics in agent-based CPPS, which have so far plagued attempts to control how traditional automated production systems and equipment are made. The preliminary standardized IA patterns (RA, PA, CA, and AMS) demonstrate how to achieve autonomy, reactiveness, and proactiveness, providing Industrial AI in CPPS with platform independence. Meanwhile, the main contribution of this paper shows the Agent4.0 concept with an MAS architecture as the subsequent transformation of IAs for a CPPS evolution in order to enhance predictability, among other Industrial AI characteristics. Those industrial needs should be achieved and measured using concepts for predictive systems. The concepts in question enable MAS developers to predict CPPS behavior uncertainties and thereby decrease IAs' obsolescence, i.e., avoiding the AI effect (concept description at the end of sec. III). In this case study, the MAS application resides in a cloud environment-based Node-RED that is easily accessible with a network connection in order to reveal KPIs like OEE. The main benefit of applying IAs and reaching a high OEE is the ability to demonstrate CPPS competitiveness by highlighting weaknesses. For instance, RA increases the availability of the equipment by soft-sensors and PA delivers high-quality products to its customers. From these IA patterns, it is then possible to differentiate the Agent4.0 that increases predictability and human cooperativeness by offline training and predicting data trends within CPPS.

Regarding future work, using different machine learning models for Agent4.0 as a resource to identify critical CPPS situations in an unsupervised training environment could allow automatically performed cost-opportunity analyses to decide whether extra agent-based sensors should be included.

REFERENCES

[1] S. Karnouskos, P. Leitão, L. Ribeiro, and A. W. Colombo, "Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems: Multiagent Systems Entering Industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 14, no. 3, pp. 18–32, Sep. 2020.

[2] P. Leitão, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart Agents in Industrial Cyber–Physical Systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016, doi: 10.1109/JPROC.2016.2521931.

[3] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, 4th ed. Pearson, 2021.

[4] VDI/VDE, "2653 Sheet 4: Multi-agent systems in industrial automation - Selected patterns for field level control and energy systems," 2021. [Online]. Available: 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.

[5] L. A. Cruz S., D. Ryashentseva, A. Lüder, and B. Vogel-Heuser, "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.*, vol. 105, no. 9, pp. 4005–4034, Jul. 2019, doi: 10.1007/s00170-019-03800-4.

[6] IEEE, "IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions," *IEEE Std 2660.1-2020*, Jan. 2021.

[7] Plattform Industrie 4.0, "Technology Scenario, Artificial Intelligence in Industrie 4.0," Berlin, Germany, 2019.

[8] L. Ribeiro and M. Hochwallner, "On the Design Complexity of Cyberphysical Production Systems," *Complexity*, vol. 2018, pp. 1–13, Jun. 2018.

[9] I. Kovalenko, D. Ryashentseva, B. Vogel-Heuser, D. Tilbury, and K. Barton, "Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, Jul. 2019.

[10] U. Gangoiti, A. López, A. Armentia, E. Estévez, and M. Marcos, "Model-Driven Design and Development of Flexible Automated Production Control Configurations for Industry 4.0," *Appl. Sci.*, vol. 11, no. 5, p. 2319, Mar. 2021.

[11] L. S. Melo, R. F. Sampaio, R. P. S. Leão, G. C. Barroso, and J. R. Bezerra, "Python-based multi-agent platform for application on power grids," *Int. Trans. Electr. Energy Syst.*, vol. 29, no. 6, Jun. 2019, doi: 10.1002/2050-7038.12012.

[12] L. A. Cruz S. and B. Vogel-Heuser, "A CPPS-architecture and workflow for bringing agent-based technologies as a form of artificial intelligence into practice," - *Autom.*, vol. 70, no. 6, pp. 580–598, Jun. 2022, doi: 10.1515/auto-2022-0008.

[13] A. Kaplan and M. Haenlein, "Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence," *Business Horizons*. 2019, doi: 10.1016/j.bushor.2018.08.004.

[14] J. Lee, M. Azamfar, J. Singh, and S. Siahpour, "Integration of digital twin and deep learning in cyber-physical systems: towards smart manufacturing," *IET Collab. Intell. Manuf.*, vol. 2, no. 1, pp. 34–36, Mar. 2020, doi: 10.1049/iet-cim.2020.0009.

[15] P. Leitão and S. Karnouskos, *Industrial Agents: Emerging Applications of Software*

Agents in Industry, 1st Ed. Elsevier, 2015.

[16] J. Fischer, M. Marcos, and B. Vogel-Heuser, "Model-based development of a multi-agent system for controlling material flow systems," - *Autom.*, vol. 66, no. 5, 2018.

[17] B. Leybovich, "Probability is not Predictability," *Towards Data Science*, 2019. https://towardsdatascience.com/probability-and-predictability-b3d7ebb6952e (accessed Jan. 01, 2022).

[18] M. Minsky, R. Kurzweil, and S. Mann, "The society of intelligent veillance," in *2013 IEEE International Symposium on Technology and Society (ISTAS): Social Implications of Wearable Computing and Augmediated Reality in Everyday Life*, Jun. 2013, pp. 13–17, doi: 10.1109/ISTAS.2013.6613095.

[19] IEEE, "Foundation for Intelligent Physical Agents FIPA," 2005. http://www.fipa.org/repository/index.html (accessed Feb. 06, 2022).

[20] L. A. Cruz S., F. Mayer, D. Schütz, and B. Vogel-Heuser, "Platform Independent Multi-Agent System for Robust Networks of Production Systems," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1261–1268, 2018, doi: 10.1016/j.ifacol.2018.08.359.

[21] G. di Orio, P. Malo, and J. Barata, "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*, Oct. 2019, pp. 5505–5512, doi: 10.1109/IECON.2019.8927081.

[22] Platform Industrie 4.0, "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, 2020.