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).

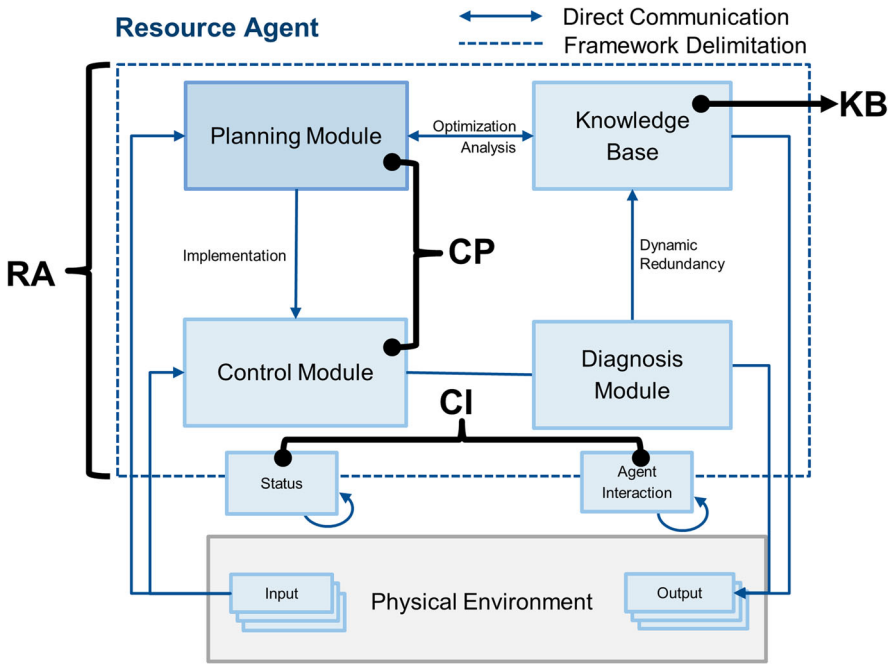

Fig. 1 Identifications of RA pattern in Wannagat's architecture [27]

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	

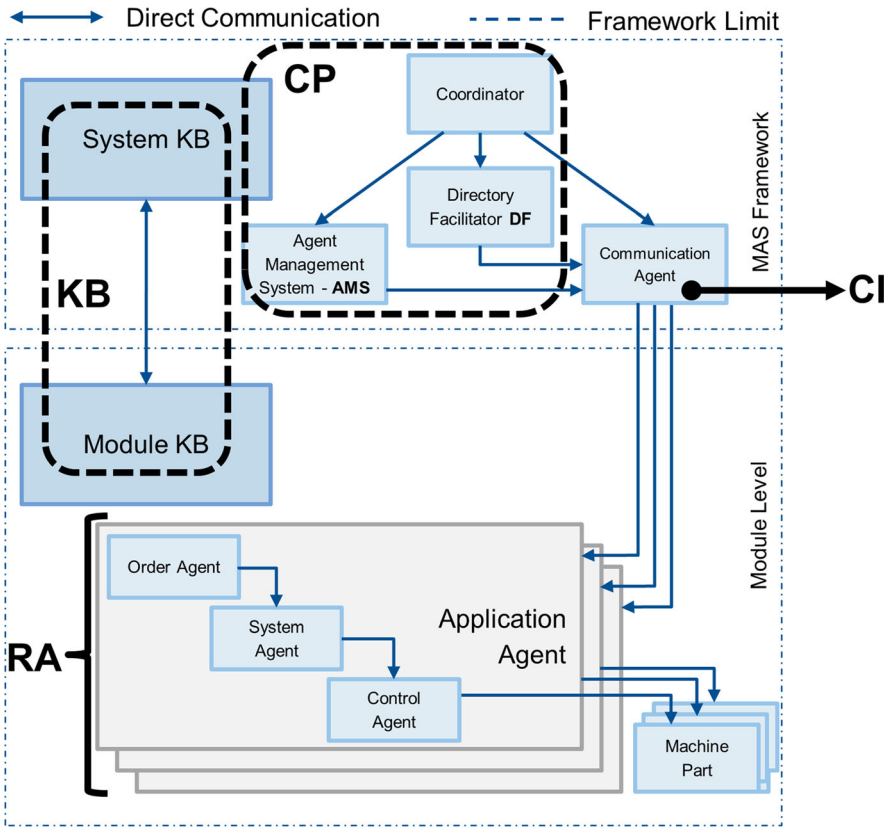

Fig. 2 Identification of MFS patterns in Fischer's architecture [32]

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

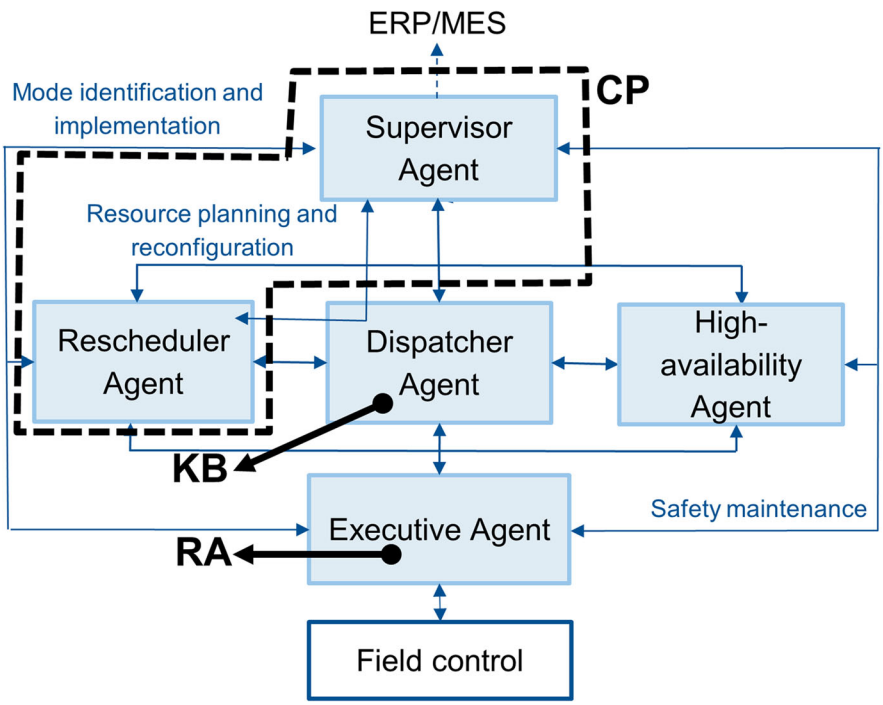

Fig. 3 Patterns for self*-control MAS in Ryashentseva's architecture [38]

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]

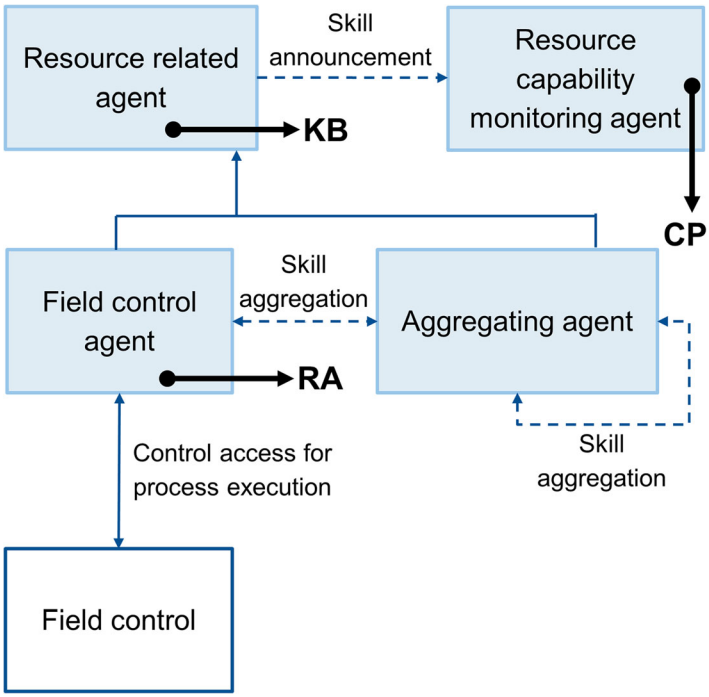

Fig. 4 Resource access design pattern from Lüder et al. [16]

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

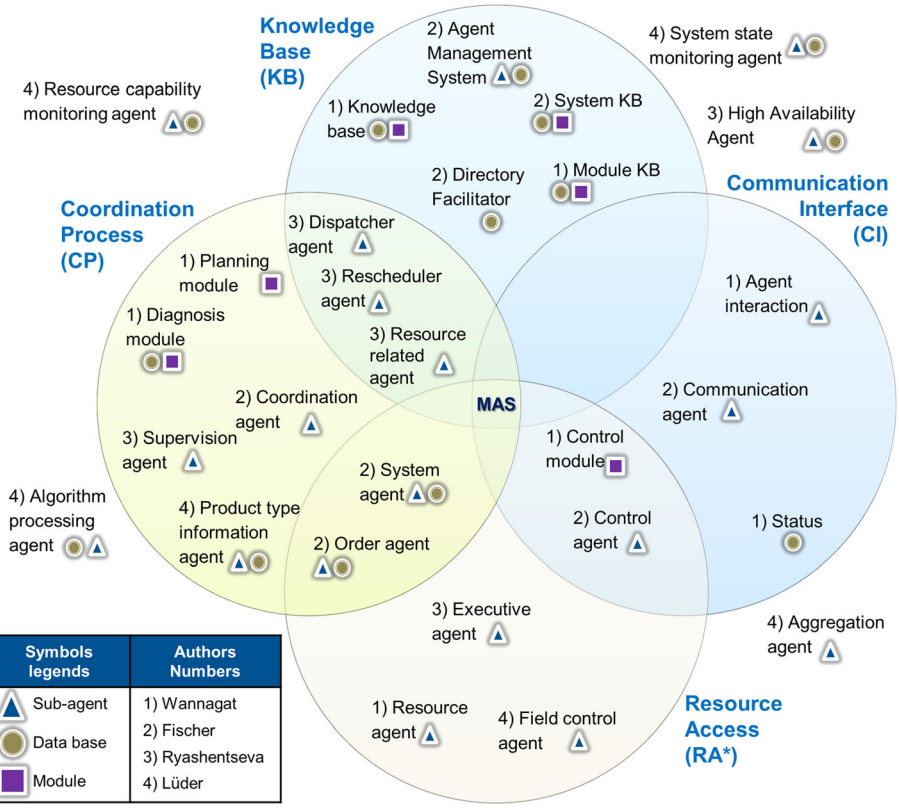

Fig. 5 Summary of the comparison of Wannagat, Fischer, Ryashentseva, and Lüder et al. MAS approaches to map common functional requirements

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

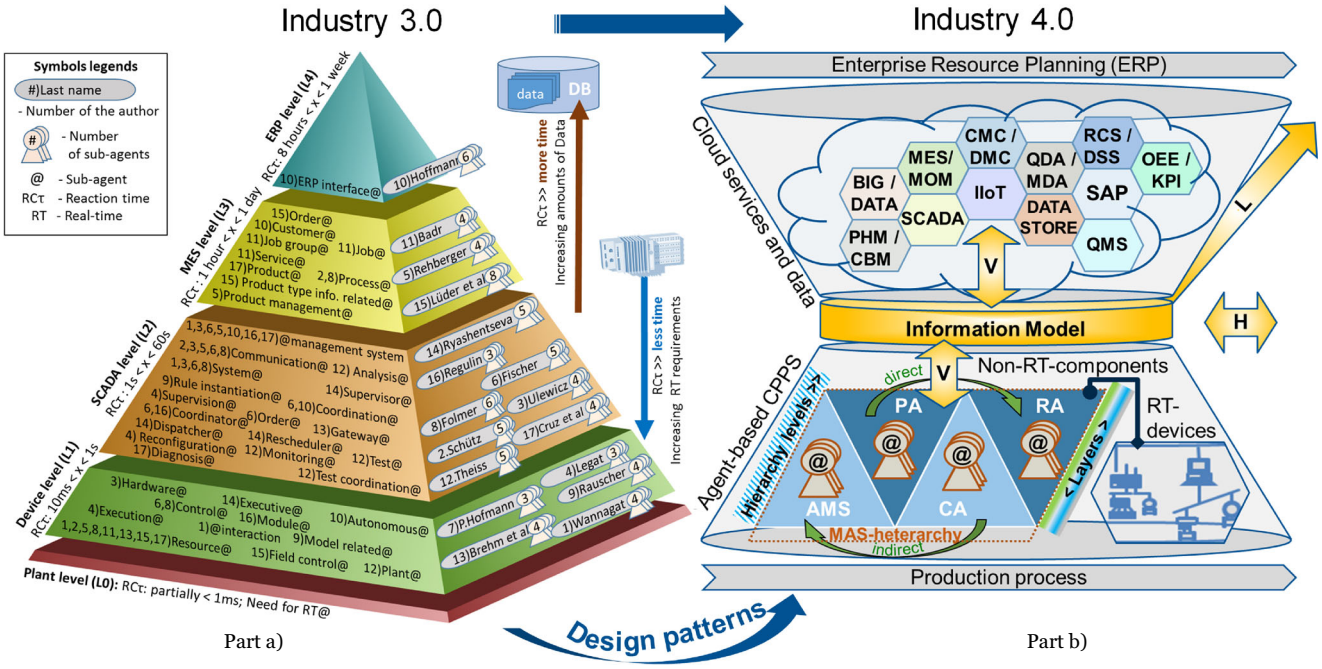

Fig. 6 Migration from the traditional levels of the automation pyramid (part a) to the "diabolo" [57] topology (part b). @: Sub-agent pattern; AMS, agent management system; CA, coordination agent; CBM, condition based monitoring; CMC, collaborative manufacturing community; CPPS, cyber physical production 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

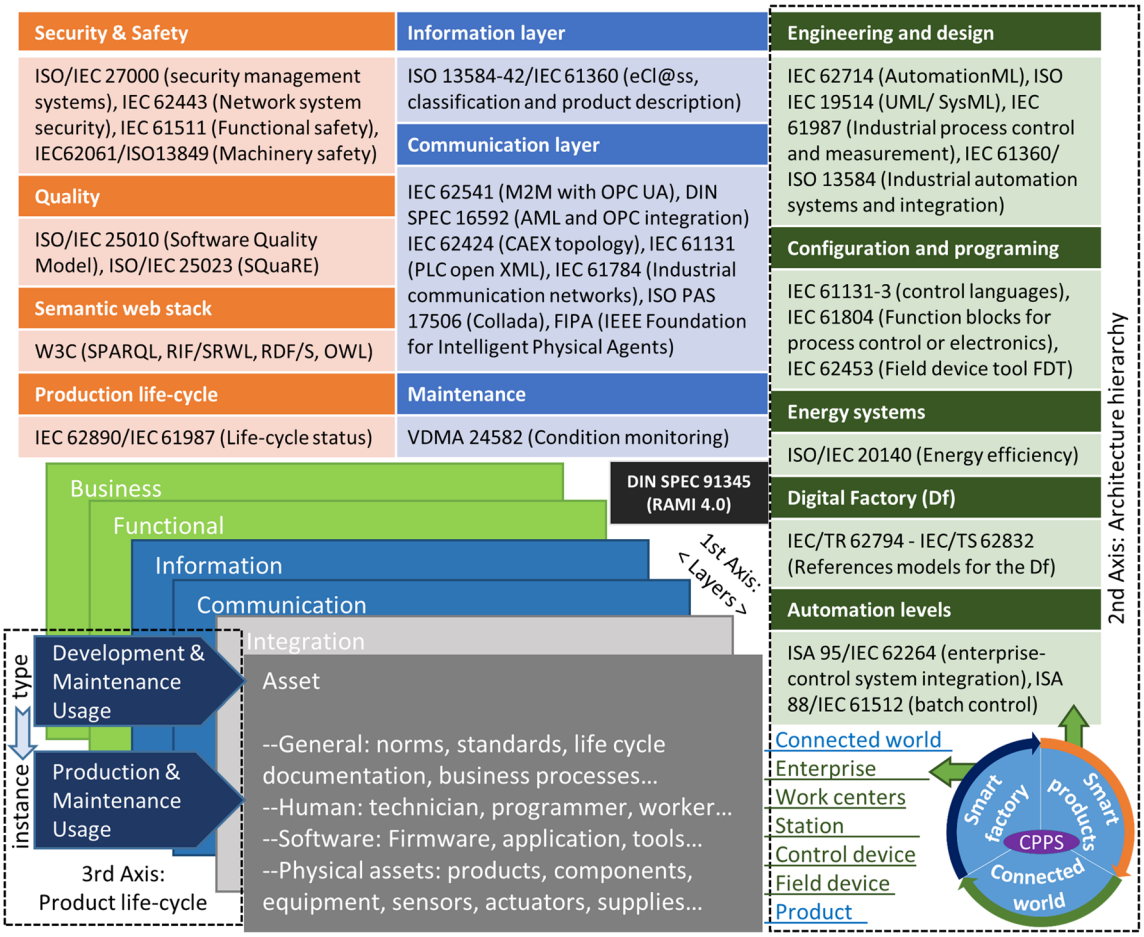

Fig. 7 The landscape of the RAM I4.0's axes and their optional norms

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

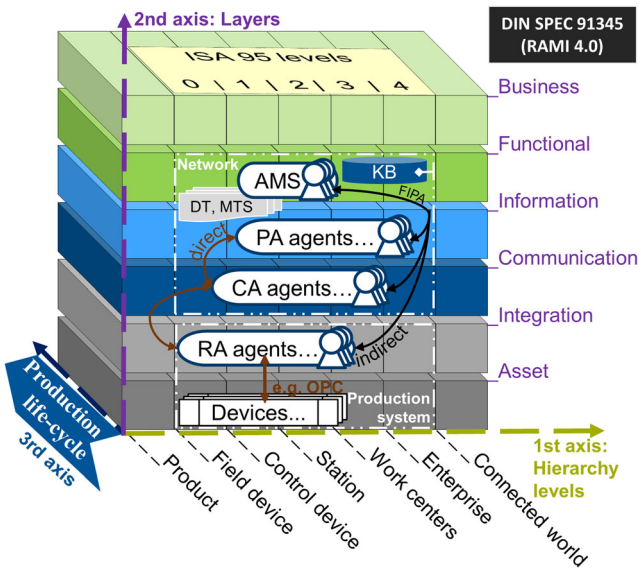

Fig. 8 Agent-based CPPS architecture aligned into two axes of RAMI 4.0

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:

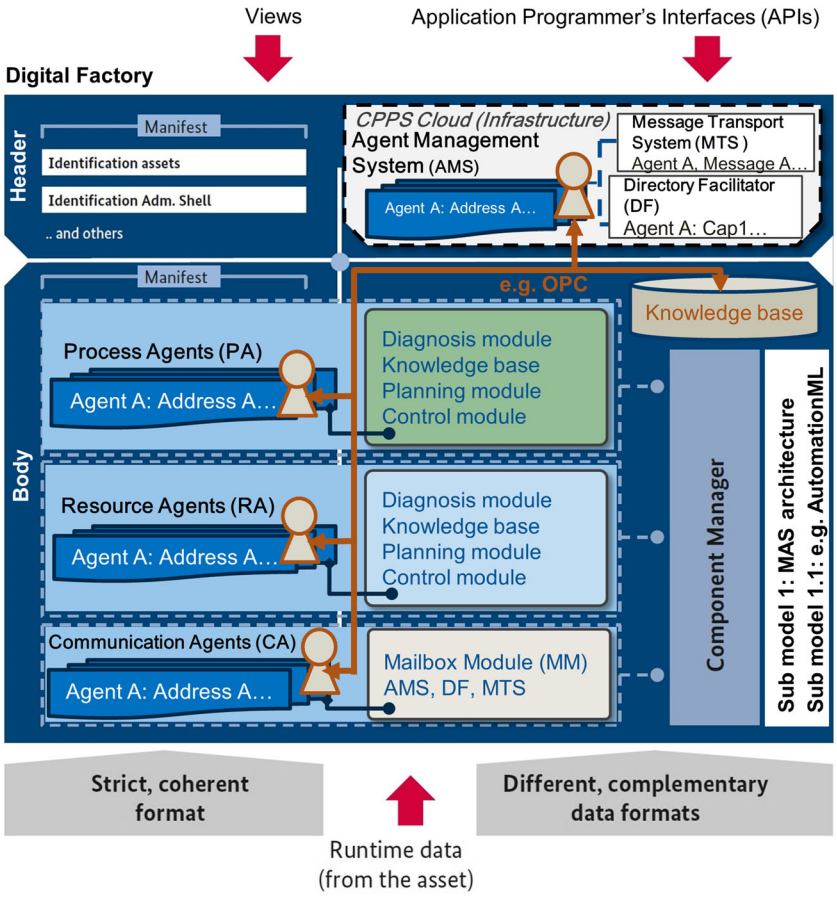

Fig. 9 CPPS's administration shell for I4.0 components (adapted from [60])

- 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

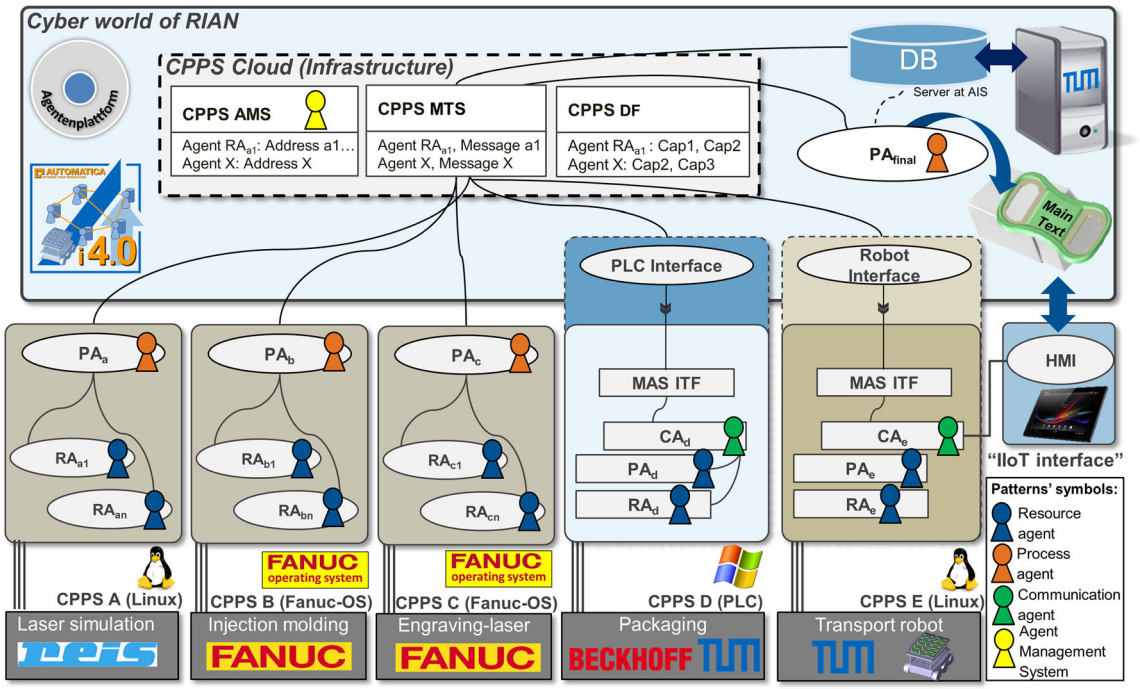

Fig. 10 The robot integrated agent network "RIAN" (adapted from [69])

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

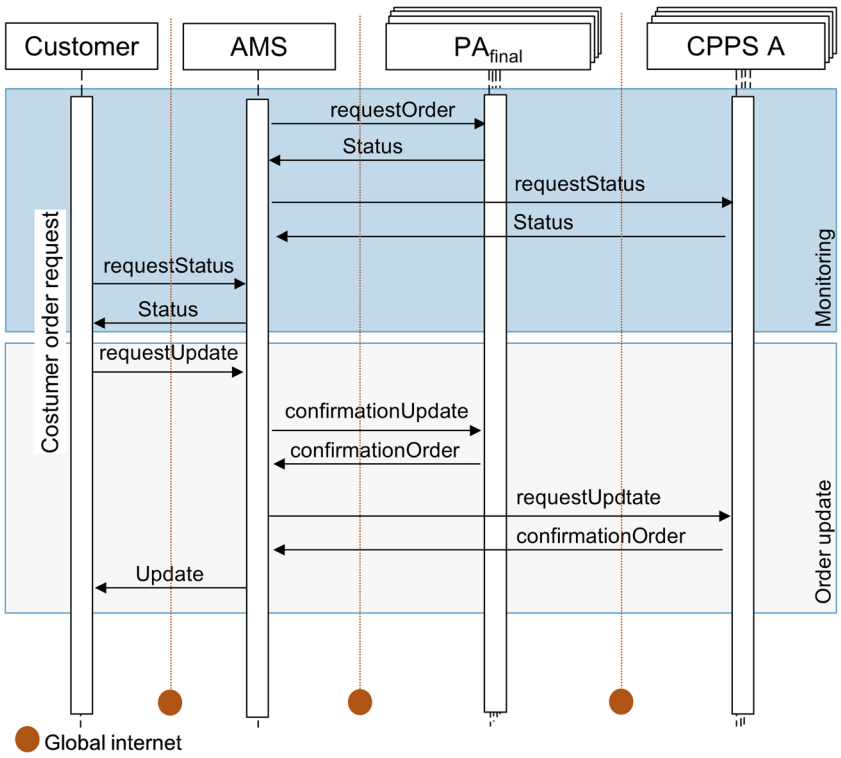

Fig. 11 Interaction in the agent-based CPPS network (adapted from [69])

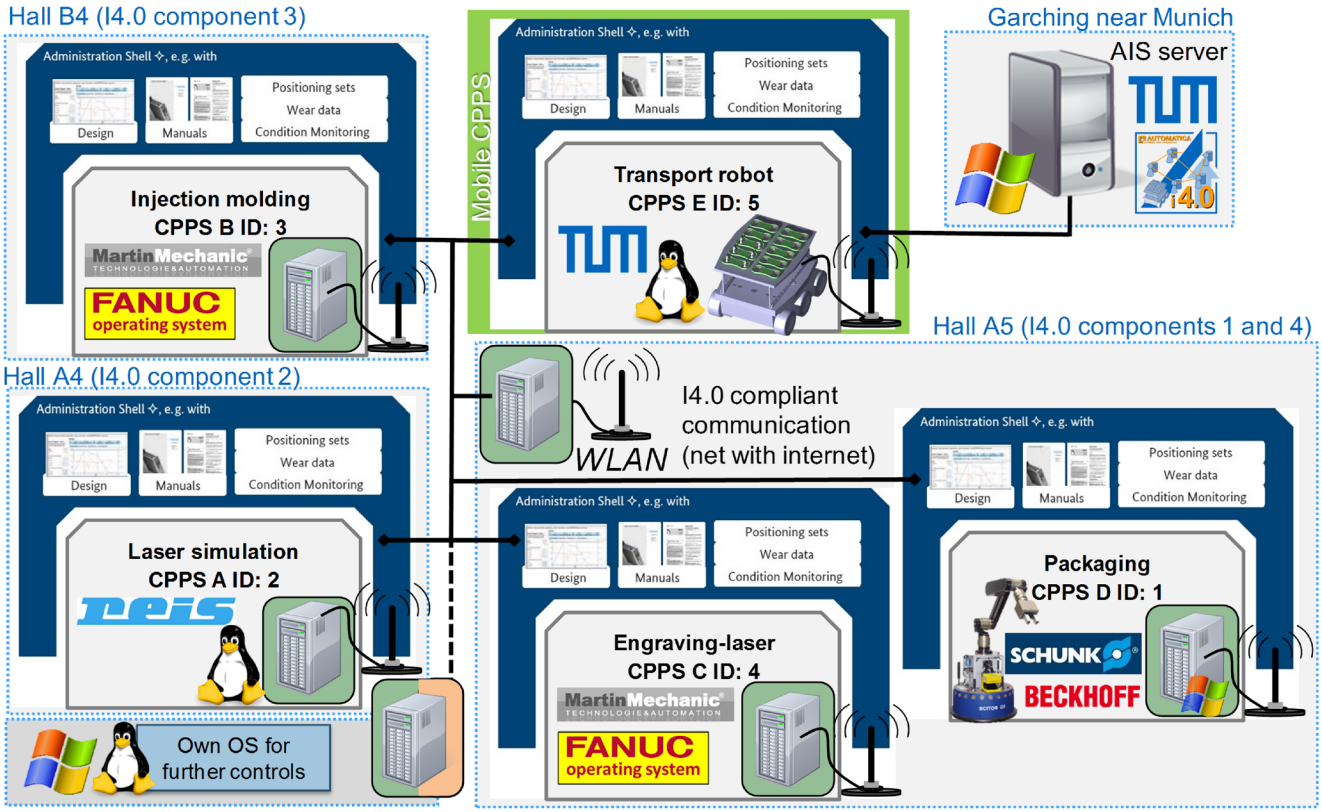

Fig. 12 The I4.0 components of the RIAN demonstrator

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