VNiVERSiDAD D SALAMANCA

DOCTORAL THESIS

An Industrial Agent-Based Approach to Designing Asset Administration Shells Type 3

Author:
Lucas Hiroaki Sakurada

Supervisors:
Fernando De la Prieta Pintado
Paulo Jorge Pinto Leitao

A thesis submitted in fulfillment of the requirements for the

Doctoral Degree in Computer Engineering

in the

Doctoral School "Studii Salamantini" of the University of Salamanca

2025

An Industrial Agent-Based Approach to Designing Asset Administration Shells Type 3

Ph.D. Thesis

University of Salamanca

Email: lsakurada@ipb.pt

ORCID: 0000-0003-0145-1834

Scopus Author ID: 57214751181

Google Scholar: mWBcaO8AAAAJ&hl=pt-PT&oi=ao

Ciência ID: 7913-F913-7D27

The doctoral thesis entitled "An Industrial Agent-Based Approach to Designing Asset Administration Shells Type 3", submitted by Mr. Lucas Hiroaki Sakurada in fulfillment of the requirements for the degree of Doctor in Computer Engineering at the University of Salamanca, has been conducted under the supervision of Dr. Fernando De la Prieta Pintado, Full Professor in the Department of Computer Science and Automation at the University of Salamanca, Spain, and Dr. Paulo Jorge Pinto Leitao, Full Professor in the Department of Electrical Engineering at the Polytechnic Institute of Bragança, Portugal.

Bragança, Portugal, July 2025

gov.br
Documento assinado digitalmente
LUCAS HIROAKI SAKURADA
Data: 23/07/2025 05:49:04-0300
Verifique em https://validar.iti.gov.br

Lucas Hiroaki Sakurada

Firmado por DE LA PRIETA PINTADO FERNANDO - ***1100** el día 23/07/2025 con un certificado emitido por AC FNMT Usuarios

Assinado por: **Paulo Jorge Pinto Leitão**
Num. de Identificação: 08446321
Data: 2025.07.24 11:13:25 +0100

Dr. Fernando De la Prieta Pintado
Full Professor
University of Salamanca

Dr. Paulo Jorge Pinto Leitao
Full Professor
Polytechnic Institute of Bragança

Acknowledgements

I would like to express my sincere gratitude to all the people who supported me throughout the development of this work, both professionally and personally. First and foremost, I am thankful to my parents and family for their support during this journey. Secondly, I am grateful to my supervisors, Prof. Paulo Leitao and Prof. Fernando De la Prieta, for the opportunity, encouragement, support, patience, and perseverance, as well as for sharing their knowledge and experience with me over the years.

I also thank all the friends and laboratory colleagues who have contributed directly or indirectly to the completion of this work.

I extend my thanks to FCT - Fundação para a Ciência e Tecnologia for providing financial support for my Ph.D. research, as well as to CeDRI - Research Centre in Digitalization and Intelligent Robotics, where I conducted my Ph.D. research.

This work was supported by FCT - Fundação para a Ciência e Tecnologia, I.P. by project reference 2020.09234.BD and DOI identifier 10.54499/2020.09234.BD.

Abstract

Over a decade ago, the term “Industrie 4.0” was first introduced at the Hannover Messe fair in Germany to describe a revolutionary paradigm focused on the digital transformation of industrial environments. This transformation is driven by the adoption of industrial Cyber-physical Systems (CPS), which enable unprecedented levels of flexibility, reconfigurability, and resilience in manufacturing systems. In these envisioned systems, every asset is embedded with intelligence and connectivity, enabling them to autonomously make decisions, collaborate, adapt to condition changes, and work together to optimize processes. As a result, Industry 4.0 (I4.0) creates a more agile, efficient, and responsive industrial ecosystem capable of anticipating and addressing market demands, driving continuous innovation, and maximizing productivity.

Despite significant efforts to leverage I4.0, its full realization remains a distant reality. Many companies continue to face challenges such as technological integration, the high complexity of designing such systems, substantial implementation costs, and the lack of standardized solutions that ensure seamless interoperability. To help address these challenges, the Reference Architecture Model Industrie 4.0 (RAMI4.0) has been introduced. RAMI4.0 is a three-dimensional model that formalizes all key aspects related to the digitization of industrial assets, aiming to provide guidelines and a shared understanding for participants in developing I4.0-compliant solutions based on industrial standards. At the core of RAMI4.0 lies the concept of the I4.0 component, a specific CPS category encompassing an asset and its digital counterpart, the Asset Administration Shell (AAS).

The AAS serves as a standardized digital representation of a physical or logical asset, encapsulating all relevant information throughout the asset’s lifecycle. By acting as the digital interface of an asset, the AAS ensures that all assets, regardless of their type or function, can be seamlessly integrated into the I4.0, facilitating interoperability and efficient data exchange across the entire production network. However, traditional AAS solutions, including AAS Types 1 and 2, cannot address all the requirements of modern industrial environments. To overcome these limitations, the AAS Type 3 extends beyond the conventional functionalities of traditional AAS by incorporating more sophisticated features, enabling I4.0 components to operate with greater autonomy, intelligence, and collaborative capabilities.

Despite its potential, AAS Type 3 remains an emerging and underexplored concept in the literature. While traditional AAS implementations (Types 1 and 2) have been widely studied and adopted, research on AAS Type 3 is still in its early stages, lacking comprehensive studies and standardized guidelines for its design and implementation. This gap highlights the need for further exploration to define its architecture, functionalities, and practical applications. By integrating autonomy, intelligence, and collaboration, AAS Type 3 represents a crucial step toward fully realizing the vision of I4.0.

In this context, the objective of this thesis is to address the lack of research on AAS Type 3 by specifying an approach for its realization using Multi-agent Systems (MAS). By leveraging the principles of MAS, this work aims to define an agent-based approach that enables AAS to operate with enhanced autonomy, intelligence, and collaboration. The proposed approach aims to provide a clear specification for designing and implementing AAS Type 3, allowing assets to interact dynamically, make decentralized decisions, and coordinate actions to achieve production goals efficiently. Moreover, as an innovative approach, this work contributes meaningfully to the state-of-the-art, serving as a valuable reference for researchers and practitioners interested in advancing the development and implementation of AAS Type 3 through an agent-based approach.

The proposed agent-based AAS approach was implemented and evaluated in two distinct case studies: a laboratory prototype and an industrial automotive production line. The experiments and assessments demonstrated that the proposed agent-based approach successfully realizes an AAS Type 3, enabling assets to transition from passive components to truly I4.0 components with enhanced autonomy, intelligence, and collaborative capabilities. Additionally, the proposed approach addresses essential requirements for I4.0, namely adaptability, pluggability, robustness, interoperability, and (re)usability.

Keywords: Asset Administration Shells, Cyber-physical Systems, Industry 4.0, Multi-agent Systems, Reference Architecture Model Industrie 4.0

Resumen

Hace más de una década, el término “Industrie 4.0”fue presentado por primera vez en la feria Hannover Messe en Alemania para describir un paradigma revolucionario centrado en la digitalización de los entornos industriales. Esta transformación está impulsada por la adopción de Sistemas Ciberfísicos (CPS), los cuales permiten niveles sin precedentes de flexibilidad, reconfigurabilidad y resiliencia en los sistemas de fabricación. En estos sistemas, cada activo está integrado con inteligencia y conectividad, lo que les permite tomar decisiones de manera autónoma, colaborar, adaptarse a cambios en las condiciones y trabajar conjuntamente para optimizar los procesos. Como resultado, la Industria 4.0 (I4.0) crea un ecosistema industrial más ágil, eficiente y receptivo, capaz de anticipar y satisfacer las demandas del mercado, impulsando la innovación continua y maximizando la productividad.

A pesar de los esfuerzos significativos por aprovechar la I4.0, su plena realización sigue siendo una realidad distante. Muchas empresas siguen enfrentándose a desafíos como la integración tecnológica, la alta complejidad en el diseño de tales sistemas, los altos costos de implementación y la falta de soluciones estandarizadas que aseguren interoperabilidad. Para ayudar a abordar estos desafíos, se ha introducido el Modelo de Arquitectura de Referencia para la Industria 4.0 (RAMI4.0). RAMI4.0 es un modelo tridimensional que formaliza todos los aspectos clave relacionados con la digitalización de los activos industriales, con el objetivo de proporcionar directrices y una comprensión compartida para los participantes en el desarrollo de soluciones compatibles con I4.0 basadas en estándares industriales. En el núcleo de RAMI4.0 se encuentra el concepto del componente I4.0, una categoría específica de CPS que engloba un activo y su contraparte digital, el *Asset Administration Shell* (AAS).

El AAS actúa como una representación digital estandarizada de un activo físico o lógico, encapsulando toda la información relevante a lo largo del ciclo de vida del activo. Al funcionar como la interfaz digital de un activo, el AAS asegura que todos los activos, independientemente de su tipo o función, puedan integrarse perfectamente en I4.0, facilitando la interoperabilidad y el intercambio eficiente de datos a lo largo de toda la red de producción. Sin embargo, las soluciones tradicionales de AAS, incluidos los tipos de AAS 1 y 2, no pueden abordar todos los requisitos de los entornos industriales modernos.

Para superar estas limitaciones, el AAS Tipo 3 va más allá de las funcionalidades convencionales de los AAS tradicionales al incorporar características más sofisticadas, lo que permite a los componentes I4.0 operar con mayor autonomía, inteligencia y capacidades colaborativas.

A pesar de su potencial, el AAS Tipo 3 sigue siendo un concepto emergente y poco explorado en la literatura. Mientras que las implementaciones tradicionales de AAS (Tipos 1 y 2) han sido ampliamente estudiadas y adoptadas, la investigación sobre el AAS Tipo 3 aún se encuentra en sus primeras etapas, careciendo de estudios integrales y directrices estandarizadas para su diseño e implementación. Esta brecha destaca la necesidad de una mayor exploración para definir su arquitectura, funcionalidades y aplicaciones prácticas. Al integrar la autonomía, la inteligencia y la colaboración, el AAS Tipo 3 representa un paso crucial hacia la realización plena de la visión de I4.0.

En este contexto, el objetivo de esta tesis es abordar la falta de investigación sobre el AAS Tipo 3 especificando un enfoque para su realización mediante el uso de Sistemas Multiagente (MAS). Aprovechando los principios de los MAS, este trabajo tiene como objetivo definir un enfoque basado en agentes que permita al AAS operar con mayor autonomía, inteligencia y colaboración. El enfoque propuesto busca proporcionar una especificación clara para el diseño e implementación del AAS Tipo 3, permitiendo que los activos interactúen de manera dinámica, tomen decisiones descentralizadas y coordinen acciones para lograr de manera eficiente los objetivos de producción. Además, como enfoque innovador, este trabajo contribuye de manera significativa al estado del arte, sirviendo como una referencia valiosa para investigadores y profesionales interesados en avanzar en el desarrollo e implementación del AAS Tipo 3 mediante un enfoque basado en agentes.

El enfoque propuesto basado en agentes para el AAS fue implementado y evaluado en dos casos de estudio distintos: un prototipo de laboratorio y una línea de producción en la industria automotriz. Los experimentos demostraron que el enfoque basado en agentes propuesto permiten poner en marcha con éxito un AAS Tipo 3, permitiendo que los activos pasen de ser componentes pasivos a componentes verdaderamente I4.0 con mayor autonomía, inteligencia y capacidades colaborativas. Además, el enfoque aborda los requisitos esenciales para I4.0, como la adaptabilidad, la conectividad, la robustez, la interoperabilidad y la (re)usabilidad.

Palabras clave: Asset Administration Shells, Sistemas Ciberfísicos, Industria 4.0, Sistemas Multiagente, Modelo de Arquitectura de Referencia para la Industria 4.0

Resumo

Há mais de uma década, o termo "Industrie 4.0" foi introduzido na feira Hannover Messe, na Alemanha, para descrever um novo paradigma focado na digitalização dos ambientes industriais. Essa transformação é impulsionada pela adoção de Sistemas Ciber-Físicos (CPS), que possibilitam níveis sem precedentes de flexibilidade, reconfigurabilidade e resiliência nos sistemas de manufatura. Nestes sistemas, cada ativo é equipado com inteligência e conectividade, permitindo que eles tomem decisões autônomas, colaborem entre si, se adaptem às mudanças de condições e trabalhem de forma coordenada para otimizar os processos. Como resultado, a Indústria 4.0 (I4.0) cria um ecossistema industrial mais ágil, eficiente e responsivo, capaz de antecipar e atender às demandas do mercado, promovendo inovação contínua e aumentando a produtividade.

Apesar dos esforços para adotar a I4.0, sua implementação completa ainda está distante. Muitas empresas enfrentam dificuldades, como a integração tecnológica, a alta complexidade de projetar tais sistemas, os elevados custos de implementação e a falta de soluções padronizadas que garantam interoperabilidade. Para ajudar a superar esses desafios, o Modelo de Arquitetura de Referência para Indústria 4.0 foi introduzido. O RAMI4.0 é um modelo tridimensional que formaliza os aspectos-chave da digitalização dos ativos industriais, oferecendo diretrizes e uma compreensão comum para os envolvidos no desenvolvimento de soluções compatíveis com a I4.0. No centro do RAMI4.0 está o conceito do componente I4.0, uma categoria específica de CPS que inclui um ativo e sua versão digital, o *Asset Administration Shell* (AAS).

O AAS funciona como uma representação digital padronizada de um ativo físico ou lógico, reunindo todas as informações relevantes durante o ciclo de vida do ativo. Atuando como a interface digital de um ativo, o AAS garante que qualquer tipo de ativo possa ser integrado de forma transparente à I4.0, facilitando a interoperabilidade e a troca eficiente de dados por toda a rede de produção. Contudo, as soluções tradicionais de AAS, como os Tipos 1 e 2, não conseguem atender a todos os requisitos dos ambientes industriais modernos. Para superar essas limitações, o AAS Tipo 3 vai além das funcionalidades convencionais dos AAS tradicionais, incorporando recursos mais avançados, permitindo que os componentes I4.0 operem com maior autonomia, inteligência e capacidades colaborativas.

Apesar de seu grande potencial, o AAS Tipo 3 é um conceito ainda emergente e pouco explorado na literatura. Enquanto as implementações tradicionais de AAS (Tipos 1 e 2) já foram amplamente estudadas e adotadas, a pesquisa sobre o AAS Tipo 3 ainda está em estágios iniciais, com poucos estudos e diretrizes padronizadas sobre seu design e implementação. Essa lacuna evidencia a necessidade de mais investigações para definir sua arquitetura, funcionalidades e aplicações práticas. Ao integrar autonomia, inteligência e colaboração, o AAS Tipo 3 representa um avanço significativo para a concretização plena da visão da I4.0.

Neste contexto, o objetivo desta tese é preencher a lacuna existente na pesquisa sobre o AAS Tipo 3, propondo uma abordagem para sua implementação utilizando Sistemas Multiagentes (MAS). A partir dos princípios dos MAS, este trabalho visa definir uma abordagem baseada em agentes que permita ao AAS operar com maior autonomia, inteligência e colaboração. A abordagem proposta busca fornecer uma especificação clara para o design e implementação do AAS Tipo 3, permitindo que os ativos interajam de forma dinâmica, tomem decisões de maneira descentralizada e coordenem ações para atingir os objetivos de produção de forma eficiente. Além disso, como uma abordagem inovadora, este trabalho contribui de maneira significativa para o estado da arte, servindo como referência para pesquisadores e profissionais interessados em desenvolver e implementar o AAS Tipo 3 por meio de uma abordagem baseada em agentes.

A abordagem proposta para o AAS baseada em agentes foi implementada e avaliada em dois estudos de caso distintos: um protótipo de laboratório e uma linha de produção automotiva industrial. Os experimentos realizados demonstraram que a abordagem baseada em agentes é eficaz na implementação de um AAS Tipo 3, permitindo que os ativos deixem de ser componentes passivos e se tornem partes ativas e autônomas no sistema I4.0, com capacidades aprimoradas de colaboração e tomada de decisões. Além disso, a abordagem atende aos requisitos essenciais da I4.0, como adaptabilidade, pluggabilidade, robustez, interoperabilidade e (re)usabilidade.

Palavras-chave: Asset Administration Shells, Sistemas Ciber-Físicos, Indústria 4.0, Sistemas Multiagentes, Modelo de Arquitetura de Referência para Indústria 4.0

Contents

List of Figures

List of Tables

List of Acronyms

AAS Asset Administration Shell

AASX Package file format for the Asset Administration Shell

ACC Agent Communication Channel

ACL Agent Communication Language

AF Adaptable Factory

AI Artificial Intelligence

AML AutomationML

AMR Autonomous and Mobile Robot

AMS Agent Management System

AOSE Agent-Oriented Software Engineering

API Application Programming Interface

BPME Business Process Model Engine

BPMN Business Process Model Notation

CPS Cyber-physical Systems

CSS Capability-Skill-Service

DF Directory Facilitator

DL Deep Learning

DT Digital Twin

ERP Enterprise Resource Planning

FIPA Foundation for Intelligent Physical Agents

HLC high-level control

HTTP Hypertext Transfer Protocol

I4.0 Industry 4.0

ICT Information and Communication Technology

IDTA Industrial Digital Twin Association

IIRA Industrial Internet Reference Architecture

IoT Internet of Things

JADE Java Agent DEvelopment framework

JSON JavaScript Object Notation

LLC low-level control

LLM Large Language Model

MAS Multi-agent Systems

MES Manufacturing Execution System

ML Machine Learning

MQTT Message Queuing Telemetry Transport

ONNX Open Neural Network Exchange

OPC UA Open Platform Communications Unified Architecture

OSI Open System Interconnection

PFA Portable Format for Analytics

PLC Programmable Logic Controller

PMML Predictive Model Markup Language

PPR Product-Process-Resource

RAMI4.0 Reference Architecture Model Industrie 4.0

REST Representational State Transfer

RFID Radio-Frequency Identification

SCADA Supervisory Control and Data Acquisition

SMEs Small and Medium-sized Enterprises

TD Thing Description

UML Unified Modeling Language

XML eXtensible Markup Language

ZDM Zero Defect Manufacturing

CHAPTER 1

Introduction

INDUSTRY 4.0 (I4.0) is driving the digitization of traditional manufacturing systems towards intelligent and adaptable factories, where industrial assets are interconnected and capable of collaborating both within and across companies to meet the growing demand for mass customization and high-quality products in a fast-evolving market. Despite many efforts to leverage I4.0, many companies worldwide have not yet fully embraced this digital transformation era. A key challenge in this digitization process is ensuring interoperability, i.e., the ability of heterogeneous systems, machines, and devices to communicate and collaborate seamlessly across industrial environments despite their wide variations in functional and communication capabilities.

Motivated by the importance of enabling seamless integration and communication among industrial assets, this thesis focuses on addressing the challenges regarding the design and development of a digitization approach that complies with I4.0 standards, reference architectures, and emerging digitization trends. In particular, it focuses on the Asset Administration Shell (AAS), a key aspect of the Reference Architecture Model Industrie 4.0 (RAMI4.0), which plays a crucial role in offering a standardized digital representation of an asset. Based on that, this thesis proposes an approach that not only facilitates interoperability but also enhances traditional AAS solutions by incorporating more advanced features, allowing I4.0 components to operate with increased autonomy, intelligence, and collaboration in industrial scenarios.

The remainder of this chapter is structured as follows:

- Section 1.1 provides the context and motivation for developing an I4.0-compliant digitization approach based on the AAS.

- Section 1.2 outlines the hypothesis and research questions that guide the investigation to meet the objectives of this work.
- Section 1.3 details the research methodology employed to direct the study and ensure rigorous and systematic analysis.
- Section 1.4 presents the organization of the chapters of this document that describe the research, design, development and experiments that were carried out in this work.

1.1 Context and Motivation

Traditional control approaches are based on centralized and hierarchical structures, typically following the well-known ISA-95 automation pyramid [1]. Despite these approaches being effective for optimizing production processes, they often lack the agility and adaptability to respond quickly to the ever-changing market, characterized by high-customized and high-quality products. However, with the advent of I4.0, there is a paradigm shift towards decentralized control structures to address these limitations through the implementation of Cyber-physical Systems (CPS) [2]. CPS play an important role in this vision, facilitating the design of large-scale systems endowed with intelligent, highly automated, and adaptable functions based on a set of distributed and autonomous entities, which combine cyber and physical counterparts, to interact and collaborate to reach the system's goals.

A key challenge in implementing CPS is ensuring seamless integration and communication between these cyber-physical components, which often leads to the development of ad-hoc solutions. Although these solutions may partially address the integration issues, they prevent establishing standardized, scalable, and interoperable systems, leading to compatibility issues, increased complexity, and higher maintenance costs, ultimately preventing the wide adoption and long-term sustainability of CPS-based systems. To address this challenge, DIN SPEC 91345 RAMI4.0 [3] is a three-dimensional model that formalizes all essential aspects related to the digitization of industrial assets, aiming to provide guidelines and a common understanding to the participants in developing I4.0-compliant solutions based on industrial standards. Central to RAMI4.0 is the concept of the I4.0 component, a specific CPS category encompassing an asset and its digital counterpart, the AAS [4].

The AAS, introduced in 2015, serves as a standardized digital representation of a physical or logical asset, encapsulating all relevant information throughout the asset's lifecycle into structured submodels [4]. About a decade later, AAS has become a key enabler of interoperability within I4.0 environments, where seamless integration and communication between various systems, devices, and platforms are essential. Today, the Industrial

Digital Twin Association (IDTA) oversees AAS, providing essential specifications and standardized submodel templates for digitizing industrial environments. According to recent IDTA publications and research works in the literature [5], the focus remains on consolidating and refining AAS traditional solutions, i.e., AAS Type 1 and 2 solutions, and providing standardized submodel templates. Despite these solutions provide standardized information about the asset and I4.0-compliant interfaces to ensure seamless interoperability and data exchange across the industrial environment, these solutions need to be enhanced to handle dynamic and complex industrial scenarios that demand a higher level of autonomy and intelligence from I4.0 components for decision-making and collaboration.

Looking ahead, the future vision for AAS involves the development and standardization of AAS Type 3. In summary, AAS Type 3, in comparison to other types, not only serves as a crucial facilitator of interoperability in I4.0 environments, enabling seamless integration and communication among different systems, devices, and platforms, but also offers a higher degree of autonomy, intelligence, and collaborative capabilities. These advanced features are indispensable for I4.0 components and play a critical role in realizing the complete vision of I4.0 of fully connected, intelligent, resilient, and adaptive industrial ecosystems. However, the progression toward AAS Type 3 is currently hindered by a lack of detailed specifications and official documentation. This gap presents significant challenges for developers and engineers in designing such components.

In this context, technologies like Multi-agent Systems (MAS) [6] are a suitable approach for realizing AAS Type 3, providing the required autonomy, intelligence, and collaborative capabilities, which are attributes inherent to the software agents [6]. MAS can be defined as a society of intelligent, autonomous, and cooperative agents that represent the physical or logical objects of a system. In such systems, the global behavior emerges from the interaction between individual agents through the implementation of interaction strategies, e.g., collaboration and negotiation. Several MAS-based solutions [7] have demonstrated the benefits of their usage in industrial settings, offering an alternative way to design complex and large-scale systems based on the decentralization of control. This overcomes the typical problems presented by the traditional centralized control approaches, which are not able to provide the flexibility, robustness, and reconfigurability required by the current industrial systems [8].

Having this in mind, this thesis is based on two key aspects that aim for a common objective: **(1)** addressing the lack of official documentation and limited research on AAS Type 3 design, and **(2)** proposing an agent-based AAS approach to support the design and development of an AAS Type 3 solution.

1.2 Hypothesis and Research Questions

Motivated by the aspects discussed in the previous section, this thesis addresses the challenge of designing and developing AAS Type 3, which extends beyond the conventional functionalities of traditional AAS by incorporating higher levels of autonomy, intelligence, and collaborative capabilities for I4.0 components. Based on that, the following hypothesis is addressed in this thesis:

> *An agent-based approach can enable the design and development of an Asset Administration Shell Type 3 solution. This allows the transition of assets from passive components to truly Industry 4.0 components (dynamic and active participants in the Industry 4.0 ecosystem), equipped with enhanced management, interoperability, autonomy, intelligence, and collaborative capabilities.*

In order to verify the hypothesis and achieve the objectives of this work, this thesis defines two research questions that encompass both the theoretical design (RQ1) and the practical application (RQ2). The dual focus on the system design and operational functionality ensures that the thesis will contribute both to academic theory and industrial practice.

RQ1: What aspects an agent-based approach must consider to design and develop an Asset Administration Shell Type 3 solution?

An agent-based approach for designing and developing an AAS Type 3 solution must consider several key aspects to ensure, beyond the common AAS functionalities, that it embodies autonomy, intelligence, and collaboration. Firstly, the approach must maintain compatibility with the official AAS metamodel, allowing for the addition of novel classes or modules without altering the core structure. This ensures adherence to AAS specifications and preserves interoperability with existing AAS Type 1 and 2 solutions, with these additions functioning as a higher-layer extension. Secondly, this higher-layer extension, represented by the agents, should not only handle intelligent and communication tasks but also interface directly with the asset, e.g., to enable rapid response for continuous monitoring and control, interpret data contained within the AAS, and use this information to drive actions and optimizations. Additionally, the adherence to standards is crucial to ensure seamless communication and integration across diverse industrial environments.

RQ2: How can the agent-based AAS approach enhance the digitization process in industrial environments?

While the proposed approach aims to guide the design and development of an AAS Type 3 solution, incorporating higher levels of autonomy, intelligence, and collaborative capabilities for I4.0 components, it should first be understood as a holistic I4.0 solution. This means it must deliver value in foundational areas of I4.0, such as interoperability,

adaptability, pluggability, and robustness. In contrast to RQ1, which focuses exclusively on the design of AAS Type 3, RQ2 delves deeper into the operational functionality, effectiveness, and potential benefits across various scenarios. By investigating these aspects, RQ2 seeks to provide deeper insights into how this approach can enhance the digitization process in industrial environments, particularly within the context of I4.0.

1.3 Research Methodology

The research conducted in this thesis follows a classical design science research methodology proposed by Peffers et al. [9], as shown in Figure 1.1. This methodology is based on a sequence of six activities for creating and evaluating artifacts to solve observed problems. An artifact in this context refers to any designed object with an embedded solution to a research problem. For instance, as described in Section 1.1, the objective of this thesis is to develop an agent-based approach that supports the design and development of an AAS Type 3 solution, enabling the implementation of truly Industry 4.0 components with enhanced management, interoperability, autonomy, intelligence, and collaborative capabilities.

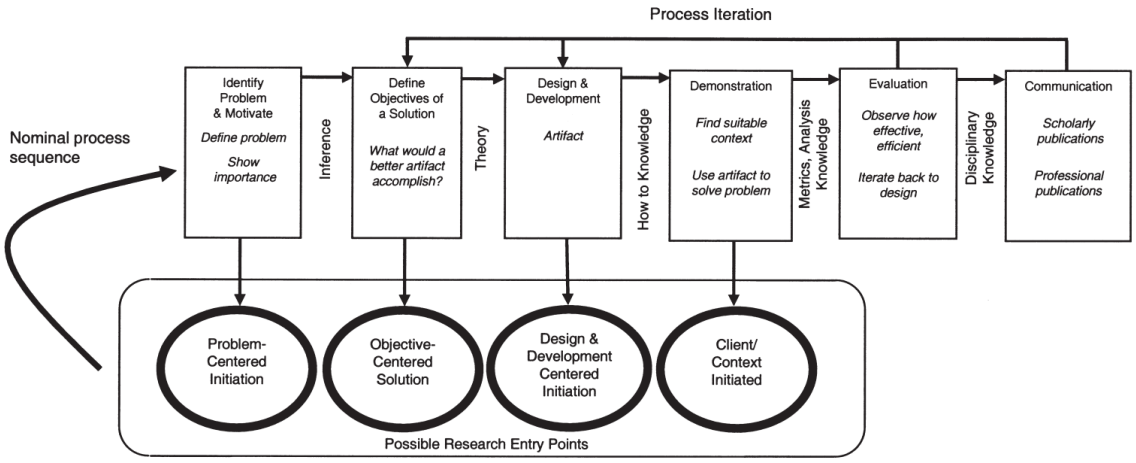

Figure 1.1: Design science research methodology [9].

As illustrated in Figure 1.1, this methodology involves an iterative process, i.e., at any step, the researcher can assess the outputs to decide if it is necessary to go back to previous steps to refine or enhance the solution artifacts. Therefore, the final solution can be achieved incrementally. In the following, the steps outlined in this methodology are briefly described, indicating how they are applied in this research.

1. Identify Problem & Motivate: This step comprises a literature review that contextualizes the domain and defines a specific research problem. It also provides the motivations for solving the problem by highlighting its importance and value. In this

work, these aspects are discussed in Chapter 1 and are supported by the literature survey presented and discussed in Chapter 2.

2. Define Objectives of a Solution: This step focuses on determining the objectives and requirements based on the research problem, considering the feasibility and practicality to address the stated problem. In this work, these aspects are discussed in Chapter 1, particularly Section 1.2, where two research questions are defined to guide the design and development of this artifact.

3. Design & Development: This step involves defining the structural and functional aspects of the artifact and developing the solution, whether it is a physical, digital, or conceptual entity. In this work, this is performed in Chapter 3 with the design of the proposed agent-based AAS approach and in Chapter 4 with its implementation.

4. Demonstration: This step involves the application of the artifact in one or more scenarios or case studies to show how it can address the research problem. In this study, the artifact includes a set of elements that are implemented in a laboratory prototype and an industrial case study, as discussed in Chapter 4.

5. Evaluation: This step measures the results of the artifact's application and determines how it fulfilled the expected objectives based on qualitative indicators. This work performs the evaluation in conjunction with the demonstration discussed in Chapter 4.

6. Communication: This step involves the dissemination of the problem, artifact and the results achieved, aiming the transfer of knowledge to the scientific community. In this work, this is performed by the preparation and presentation of various scientific publications in conferences and journals, discussing different aspects of the work reported in this document.

1.4 Doctoral Thesis Organization

The document is divided into five chapters, starting with the present chapter that discusses the context, motivation and objectives of this work, including the thesis hypothesis and related research questions. The remainder of this document is organized into the following chapters.

Chapter 2 provides a contextualization and literature review of the concepts, approaches, and technologies related to this thesis, specifically the CPS, RAMI4.0, AAS, and MAS research topics. This chapter aims to offer the necessary theoretical background for a better understanding and to highlight the current limitations, potential gaps, and open issues that require further investigation through the analysis of related works.

Chapter 3 introduces the proposed agent-based AAS approach. The discussion begins by introducing the proposed approach. The design of these components is then explained through the use of an information metamodel, which presents the classes for modeling agent-based AAS metamodel instances. Finally, each class is explained individually, accompanied by some illustrative design and development examples.

Chapter 4 presents the validation and assessment methodology and the experimental implementation of the proposed agent-based AAS approach. The chapter begins by describing the testing scenarios, assessment criteria, and case studies. Next, the agent-based AAS approach is designed and implemented based on the specifications of the scenarios and case studies. Finally, the assessment is conducted and discussed based on the experimental implementation and tests.

Chapter 5 discusses the conclusions reached during the development and assessment process, pointing out the contributions that are linked to the research questions and hypothesis of this work. It also points out the future work, discussing some aspects that could contribute to evolve the present work.

Finally, Annex A contains the Spanish summary of this thesis.

CHAPTER 2

Theoretical Background & Literature Review

OVER a decade ago, the term "Industrie 4.0" was first coined at the Hannover Messe fair in Germany to describe the revolutionary paradigm focused on the digitization of industrial environments. I4.0, also referred to as the Fourth Industrial Revolution, is characterized by the digitization of industrial sectors to achieve higher levels of flexibility, reconfigurability, and robustness in manufacturing systems that must cope with a demanding and ever-changing market driven by high-customized and high-quality products.

Since then, various digitization approaches have emerged to support this digital transformation, with the AAS being one of the most notable and recent developments. The AAS serves as a standardized digital representation of an asset, encapsulating all relevant information throughout its lifecycle. This facilitates interoperability and enables efficient data exchange across the entire production network. However, despite significant advancements in AAS research, challenges persist, particularly in the design and implementation of more advanced AAS solutions, such as AAS Type 3. Unlike traditional AAS, AAS Type 3 extends beyond conventional functionalities by incorporating more sophisticated features, enabling I4.0 components to operate with greater autonomy, intelligence, and collaborative capabilities. In this regard, technologies like MAS offer a promising solution for realizing AAS Type 3, facilitating autonomous and distributed decision-making in dynamic, complex industrial scenarios.

Based on that, this chapter analyzes existing literature and introduces relevant concepts to provide readers with the necessary theoretical background. The review serves as the foundation for addressing the main objective of this work: developing an agent-based approach for realizing AAS Type 3 using MAS. By examining current advancements and

identifying gaps in the literature, this chapter prepares readers to understand the context, significance, and potential impact of integrating MAS to enhance AAS capabilities.

The remainder of this chapter is structured as follows:

- Section 2.1 presents an overview of the main aspects of the I4.0 and CPS.
- Section 2.2 provides an overview of the RAMI4.0 model, describing its three dimensions, namely Lifecycle & Value Stream, Hierarchy Levels, and Layers.
- Section 2.3 presents relevant aspects of the I4.0 component and AAS. Additionally, it characterizes the current research related to AAS through a literature review, particularly a content analysis of AAS Type 3-related solutions to identify trends, challenges, and opportunities in the field.
- Section 2.4 introduces key concepts related to MAS and discusses several related works that propose MAS-based approaches in the context of I4.0, highlighting their applications and benefits in enhancing autonomy, intelligence, and collaboration within manufacturing environments.
- Section 2.5 discusses the parallels between MAS, AAS Type 3, and RAMI4.0, aiming to explore and discuss how these concepts are interrelated in order to later develop a specification for designing and implementing AAS Type 3 using MAS.
- Section 2.6 summarizes the key aspects discussed in this chapter.

2.1 Industry 4.0 and Cyber-Physical Systems

The term "Industrial Revolution" was initially used to describe a turning point in human history, marked by the transition from an agrarian and handicraft economy to one dominated by factory mechanization during the 18th century, leading to rapid industrialization of economies and significant societal changes [10, 11]. This era laid the foundation for subsequent industrial revolutions, from Industry 1.0 to Industry X.0. For simplicity, the term IX.0 will be used to reference these phases.

I1.0 occurred towards the end of the 18th century, characterized by early automation with the introduction of steam power and mechanized production, while I2.0 began around the turn of the 20th century, characterized by mass production, electrification and new modes of transportation. I3.0 emerged during the 1970s, driven by the rise of electronics, telecommunications, and computing, which enabled full automation and robotics in manufacturing processes. I4.0, which began in the early 21st century, represents the next phase of industrial transformation through the use of CPS [10, 11]. The latest and most debated phase, I5.0, complements and extends the core features of I4.0, incorporating a new focus on human-centricity, sustainability, and resilience in industrial processes [12,

13]. Unlike previous industrial revolutions, which followed a chronological progression, I5.0 is not a direct successor to I4.0 but rather a complementary perspective that integrates emerging societal needs into industrial development [14]. However, some organizations, such as the Research Council Industrie 4.0 and Plattform Industrie 4.0, argue that I5.0 lacks truly novel content, introducing unnecessary ambiguity. They arguee that I4.0 already encompasses key aspects of human-centricity and social impact, addressing many of the objectives attributed to I5.0. Furthermore, these entities warn that the term "Industry 5.0" may falsely suggest that I4.0 has already been fully realized [15].

Based on this perspective, this work does not engage in the debate regarding the validity of the term "Industry 5.0". However, based on recent research, it assumes the following: **(1)** the term "Industry 5.0" is gaining widespread adoption and will likely continue to be used in discussions related to human-centricity, sustainability, and resilience; **(2)** similar to previous industrial revolutions, I4.0 still has a considerable distance to cover before reaching full maturity. This ongoing evolution underscores the complexity and gradual nature of this transition, indicating that I4.0 will continue to evolve over time. For instance, research and industry reports indicate that the original vision of intelligent, fully flexible, and self-organizing factories remains far from being fully realized [16], with estimates suggesting that the complete transition to I4.0 could take around 20 years, potentially reaching maturity by 2035 [17]. Additionally, an evaluation conducted by the German National Academy of Science and Engineering in 2020 assessed seventy companies using the I4.0 Maturity Index, revealing that only 4% had made substantial progress towards I4.0 implementation [18].

Given these considerations, it is reasonable to assume that both I4.0 and I5.0 will coexist in the coming years, highlighting the importance of clearly defining the positioning of research works within this evolving landscape. In this context, this thesis firmly aligns with the scope of I4.0 (i.e., adhering to the initial vision outlined at its inception), offering relevant contributions to the ongoing implementation and evolution of I4.0. Specifically, this work proposes a digitization approach that enables assets to transition from passive components to dynamic and active participants in I4.0, equipped with enhanced management, interoperability, autonomy, intelligence, and collaborative capabilities.

The term "Industry 4.0" was first introduced in 2011 at the Hannover Messe fair in Germany as part of a strategic initiative by the government called "Industrie 4.0" to advance the German manufacturing sector through digital transformation [10]. Since then, many other countries have launched similar strategic initiatives tailored to their unique economic and industrial strengths, namely "Industrial Internet Consortium" in USA, "Industria 4.0" in Italy, "Produktion 2030" in Sweden, and "Made in China 2025" in China, to name a few [14, 19]. While these initiatives may vary in certain aspects, they share a common vision: transforming traditional factories into intelligent, flexible, adaptive, and reliable production environments, ultimately enhancing global competitiveness.

According to [10], I4.0 envisions a transformation characterized by enhanced collaboration among manufacturing resources (e.g., machines, robots, conveyors, warehousing systems, and production facilities) that are intelligent, autonomous, and knowledge-driven. These resources are envisioned to possess self-* properties and communication capabilities, enabling them to operate as connected, responsive elements within a larger, coordinated network. Smart products are another central component of the I4.0 vision. These products would be uniquely identifiable and locatable throughout the production process. Embedded with information about their own production requirements, including processing parameters and delivery instructions, smart products would be able to influence or even control specific stages of their production in a semi-autonomous manner. Moreover, a critical aspect of I4.0 is the ability to incorporate individual customer- and product- specific features into products at each stage of the lifecycle. This flexibility enables rapid adaptation to individual customer demands, making it economically viable to produce unique items and small batches, thereby meeting the growing demand for product customization.

To realize the vision of I4.0, three key types of integration must be achieved, namely horizontal integration, vertical integration, and end-to-end engineering integration [10]:

- Vertical integration: refers to the connection and synchronization of processes from the shop floor to the management and strategic decision levels. In a vertically integrated environment, data can move effortlessly from the shop floor to the top floor and back again. This bidirectional flow facilitates real-time decision-making, enhances operational efficiency, and ensures that production activities are tightly aligned with business objectives and strategies.

- Horizontal integration: focuses on connecting partners across the supply chain, including suppliers, manufacturers, distributors, and customers. This type of integration seeks to synchronize and coordinate operations across different companies to create a unified, cohesive, and efficient supply chain, allowing companies to respond more effectively to market demands, optimize resource utilization, and improve overall agility and performance.

- End-to-end integration: integrates the product information across its entire lifecycle, from concept and design to production, delivery, and recycling. For instance, manufacturers can use real-time data from smart products in the field to refine their designs or enhance production processes, while retailers and customers can access detailed insights into product performance and lifecycle status.

CPS play a central role in realizing the vision of I4.0 [2], as well as enabling these types of integration, serving as the backbone infrastructure for implementing intelligent large-scale systems exhibiting self-* properties, e.g., healing, adaptation, and organization. Complemented by emerging ICT technologies, e.g., the Internet of Things (IoT), Artificial Intelligence (AI), and Edge/Cloud Computing, CPS offers an alternative way to

design control systems based on a set of networked, distributed, and autonomous entities, combining cyber and physical counterparts. Unlike traditional embedded systems or traditional IoT architectures that merely connect physical and software elements for basic interaction and data exchange, CPS emphasize a high level of integration, coordination, cooperation, and autonomy among their components [2]. Such features allow CPS to not only respond to dynamic changes in their environment but also anticipate, adapt, and optimize their operations proactively. The realization of CPS involves several key features [20]:

- Autonomy: CPS components possess the ability to operate independently, controlling their own structural and behavioral properties without relying on centralized systems and without human intervention. For instance, in manufacturing, CPS components can self-organize their tasks, adjust workflows dynamically, and continue operating seamlessly even if other components experience downtime.
- Cooperation strategies: CPS components are designed to collaborate effectively with other components in the system. They interact and exchange information to achieve the system goals, employing strategies such as negotiation. For example, CPS components can negotiate with one another to distribute workloads efficiently, adapt production schedules in real-time, and ensure optimal use of resources.
- Context-awareness: enables CPS components to recognize and interpret their operating environment, including the status of surrounding systems and their internal state. This feature is essential for adaptive and situational decision-making. For instance, CPS components can adjust delivery routes based on traffic conditions, ensuring timely and cost-effective deliveries.
- Cognitive computation: empower CPS to process data, reason about their current state, and anticipate future needs or challenges. For instance, through predictive analytics, CPS can forecast system failures, optimize energy consumption, or identify inefficiencies. This capability is critical in applications such as predictive maintenance, where CPS components can identify potential issues before they lead to downtime, minimizing operational disruptions.
- Learning and adaptation: CPS components are equipped with learning algorithms that enable them to learn from past experiences and improve their performance over time. This adaptability is crucial in dynamic environments where conditions can change frequently. For example, CPS can learn from production data to identify patterns, optimize processes, and reduce waste, ultimately leading to continuous improvement in productivity and quality.

Despite the advancements in CPS solutions, realizing I4.0 requires a standardized and interoperable approach to ensure seamless integration and communication between these cyber-physical components. Often, ad-hoc solutions are developed to address specific

integration challenges. However, while these solutions may partially address the integration issues, they prevent establishing standardized, scalable, and interoperable systems, leading to compatibility issues, increased complexity, and higher maintenance costs, ultimately preventing the wide adoption and long-term sustainability of CPS-based systems. In this context, this thesis addresses the challenge of achieving seamless integration and communication between cyber-physical components. Specifically, it focuses on implementing AAS Type 3, a key element for enhancing I4.0 components (i.e., a CPS category aligned with the RAMI4.0 model) that facilitates the development of standardized and interoperable systems.

2.2 Reference Architecture Model Industrie 4.0

Several reference architectures[1] have been defined to support the implementation of I4.0. These reference architectures follow a high level of abstraction and a common, standard vocabulary to ensure that all participants involved in the design and development of I4.0-compliant solutions share a common perspective and understand each other. Although multiple architectures can be derived from a reference architecture, they all share a common foundation. Consequently, these architectures are easily understandable by I4.0 participants and can be seamlessly integrated, promoting interoperability and collaboration across diverse systems. Currently, RAMI4.0 [3] and Industrial Internet Reference Architecture (IIRA) [21] comprise the two major reference architectures that define a conceptual framework for designing and developing I4.0-compliant systems and components. RAMI4.0 is a German initiative that is more established in Europe, while IIRA is a USA initiative. Despite their differences in features, models, and coverage of different aspects, this work aligns with the RAMI4.0 model, particularly focusing on the concept of AAS that reflects the upper five layers of RAMI4.0.

RAMI4.0 is a three-dimensional model that formalizes all essential aspects related to the digitization of industrial assets, aiming to provide a common understanding to the participants in developing I4.0-compliant solutions. As illustrated in Figure 2.1, RAMI4.0 depicts a basic reference architecture for I4.0 comprised of three dimensions, namely Lifecycle & Value Stream, Hierarchy Levels, and Layers.

[1]The terms “architecture” and “reference architecture” are related but have distinct meanings, particularly in the context of system design and development. An architecture refers to the fundamental structure of a system or solution, including its components and relationships. It is tailored to the requirements and constraints of a particular system and directly guides the implementation of a system. On the other hand, reference architecture is a generalized template that provides a high-level framework for designing systems within a specific domain. It offers standardized guidelines, patterns, and best practices that can be reused across multiple projects. In simpler terms, a reference architecture provides the foundation for creating specific architectures or solutions.

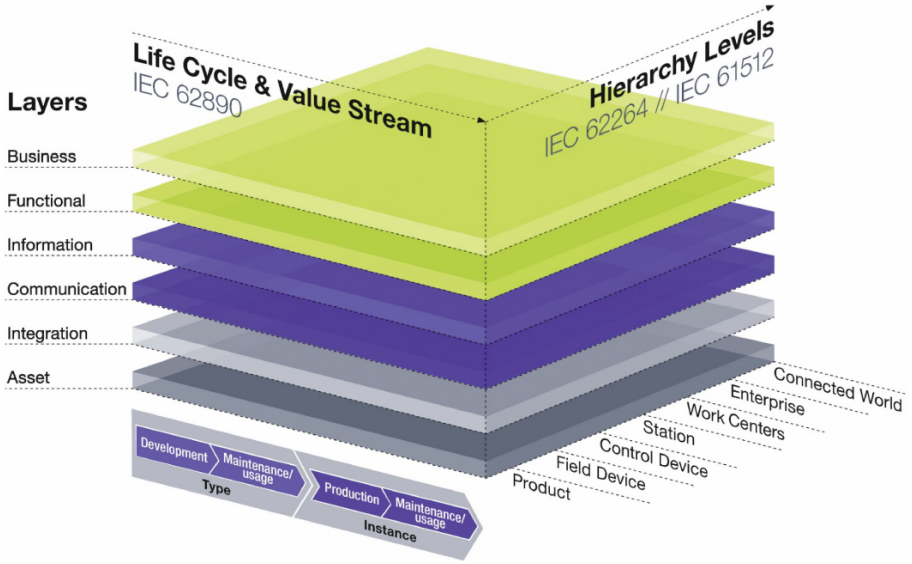

Figure 2.1: Reference architecture model Industrie 4.0 [3].

The "Lifecycle & Value Stream" dimension, based on the IEC 62890 [22] standard, covers aspects related to the lifecycle of an asset and the value-added process, i.e., it describes an asset at a particular point in time during its lifetime, e.g., from planning and development, rapid prototyping, construction to its usage and disposal. Every asset undergoes different phases but can be categorized into "type" and "instance". The "type" phase represents the asset's development from concept to prototype, while the "instance" phase begins with manufacturing and extends through real-world usage. Importantly, data generated during an asset's usage feeds back into the "type" phase, enabling continuous improvements. For instance, operational insights gained from an asset's use can inform refinements in its design, enhancing future performance [3, 23, 24].

The "Hierarchy Levels" dimension, based on the IEC 62264 [25] and IEC 61512 [26] standards, is related to the position that an asset occupies in the hierarchy within the organization where it performs its functions. RAMI4.0 proposes to partially follow the hierarchy proposed in these standards, reflecting the classic automation pyramid, namely "Control Device", "Station", "Work Centers", and "Enterprise". Additionally, new levels have been added to reflect the needs of I4.0, namely "Product", "Field Device", and "Connected World". The "Field Device" level involves intelligent devices bridging the physical and information worlds. The "Product" level has been incorporated because products will eventually have the capability to influence their own manufacturing process, becoming part of the I4.0 solution spectrum. Lastly, the "Connected world" level has been included to support I4.0's emphasis on interconnecting factories to enable collaboration beyond internal boundaries. Unlike the rigid communication hierarchy of the automation pyramid, in which communication occurs from one level to the next, the new hierarchy model allows direct communication between all participants connected in the I4.0 network [3, 23, 24].

The "Layers" dimension presents the different aspects relevant to the digitization process of assets [3, 23, 24]. RAMI4.0 defines six layers:

- Asset layer: represents the asset that exists in the real world, e.g., physical or logical entities with value for a company.
- Integration layer: represents the transition from the physical world to the information world. Each significant event in the real world corresponds to an event in the virtual world, i.e., if there are changes in reality, the event is reported to the Integration Layer through suitable mechanisms. This integration can be achieved, e.g., by using RFID readers, QR codes, sensors, actuators, etc.
- Communication layer: describes how information is exchanged between assets, regardless of their respective levels in the hierarchy. In this context, communication refers to data transmission between two or more participants. The Communication layer is arranged similarly to the well-known Open System Interconnection (OSI) model. The contents of the transmission, such as data, are positioned and described in the Information layer.
- Information layer: bundles the relevant information of an asset. This layer is responsible for ensuring data integrity, integration of different data, and persistence of the data. It is also responsible for acquiring, processing, and adapting this data and providing it in a structured manner through I4.0-compliant interfaces. For instance, data from sensors in a factory can be processed and structured into standardized information models.
- Functional layer: provides formal descriptions of functions offered by an asset. Rules and decision-making logic are included in this layer.
- Business layer: describes the commercial view (e.g., general legal and regulatory conditions, business models , monetary aspects, etc.). It also orchestrates the services provided by the Functional layer.

2.3 I4.0 Component and Asset Administration Shell

Along with the RAMI4.0 model, which provides a common understanding to the participants in developing I4.0-compliant solutions, the so-called I4.0 component also plays a crucial role in the digitization process of industrial assets. This component can be viewed as a specific CPS category aligned with RAMI4.0. As seen in Figure 2.2 (bottom), its primary purpose is to encompass each asset relevant to I4.0 in an AAS, which reflects the asset in the digital world in the upper five layers of RAMI4.0, namely integration, communication, information, functional, and business.

2.3.1 Definition, Structure and Types

In the context of I4.0, assets (i.e., every physical or logical object with value for the industry) are sourced from different vendors and often exhibit significant variations in their functional and communication capabilities. This diversity poses challenges when designing solutions that aim to enable the seamless integration and collaboration of such heterogeneous assets, which initially remain "unknown" within the system. In this regard, AAS plays a key role in addressing these challenges. Once an asset is equipped with its own AAS, it transforms into an I4.0 component, enabling it to be recognized, accessed, and managed in the information world through I4.0-conform interfaces. This transformation facilitates interoperability and new collaboration forms that were previously difficult or impossible to achieve. With AAS, collaboration extends beyond company boundaries, paving the way for a truly connected world.

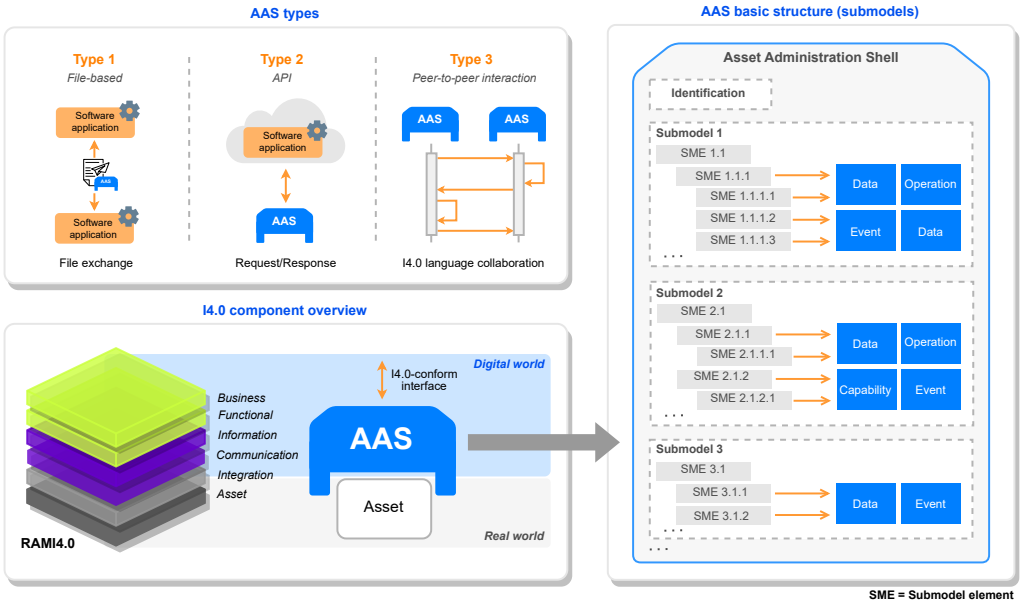

Figure 2.2: I4.0 component overview (bottom), AAS types (top), and AAS structure (right).

Formally, the AAS is defined as a *"standardized digital representation of an asset"* [27], capturing all relevant information (ranging from design, operational, and business-related aspects to end-of-life) about the asset throughout its lifecycle and organizing it in a standardized structure. This standardization helps address the silo problem, where asset-related information is fragmented across systems and departments, hindering integration and reuse. By offering a common data model, the AAS enables consistent communication, seamless data exchange, and informed decision-making across the entire value chain, benefiting various stakeholders (e.g., manufacturers, operators, service providers, and regulatory bodies) with accessible and valuable information.

As illustrated in Figure 2.2 (right), the AAS structure comprises an identification (globally unique identifier) and a collection of submodels following the architectural principle

"separation of concerns" [27]. In this context, each submodel can be viewed as a modular building block that encapsulates specific information about the asset, such as identification, capabilities, technical data, and operational data. Moreover, each submodel contains a set of submodel elements (include data properties, operations, events, and other elements), which can be organized hierarchically to represent information across varying levels of granularity in a clear and structured manner. For a more technical perspective of the AAS, submodel, and submodel elements structure, see [27].

Today, the IDTA is responsible for standardizing the AAS, providing a set of specifications and submodel templates to support AAS implementations. For instance, *Part 1: Metamodel* specification [27] describes a technology-neutral AAS information metamodel, presenting the main classes and their relationships to develop AASs. On the other hand, *Part 2: Application Programming Interfaces* specification [28] defines APIs for enabling the access to the information provided by an AAS. Additionally, submodel templates guide the creation of submodels based on predefined structures, ensuring standardization, interoperability, reusability, and faster development.

- Standardization: Submodel templates follow a standardized structure defined by organizations like IDTA.
- Interoperability: By using predefined submodel templates, different systems and components can understand and exchange information seamlessly.
- Reusability: Templates can be reused across multiple projects and assets. This saves time and effort in defining submodels from scratch and helps maintain consistency.
- Faster development: Templates provide a ready-made structure for common asset features (e.g., digital nameplate, technical data, capabilities), speeding up the design and deployment of AAS-based solutions.

For instance, "Asset Interfaces Description" [29] submodel template specifies an information model and a common representation for describing the interfaces of an asset service or asset-related service. Further examples of submodel templates can be found in [30].

Finally, as shown in Figure 2.2 (top), the AAS can be classified according to their interaction pattern to exchange information and degree of autonomy to make decisions [31]. These types are defined as follows:

- Type 1: represents the simplest type of AAS (file-based AAS) that stores the asset information throughout its lifecycle and can be exchanged as a whole in the form of a machine- and human-readable file, e.g., XML, JSON, and AASX.
- Type 2: acts as an Application Programming Interface (API) that reacts to external requests but lacks the capability to take initiatives and decisions. Such API enables the online access to asset information and can be specified in a technology-neutral way.

- Type 3: represents a more advanced and extended form of AAS, acting as a decision-making entity that autonomously interacts with other AASs to exchange information and collaborate (e.g., through negotiation).

2.3.2 Asset Administration Shell Type 3

As previously discussed in Section 1, AAS Type 3 remains an emerging and underexplored topic in the literature. In this context, this section aims to examine and clarify the concept of AAS Type 3 by analyzing how it is defined in official documents, e.g., those provided by the IDTA and Plattform Industrie 4.0, along with some definitions from selected academic research. The objective is to offer a clearer understanding of its key characteristics and intended functionalities. Based on this analysis, a general definition and the characteristics/requirements for AAS Type 3 are established.

Table 2.1: Summary of AAS Type 3 definitions.

Ref.	Definitions
[31]*	*"(...) can participate in protocol-based interactions, as defined, for example, in the VDI/VDE 2193 standard for the bidding process (...) The goal is to design decentralized processes that rely on a certain degree of autonomy or decision-making capability of the AASs"*
[32]*	*"Interactions which need peer-to-peer interactions between AAS of I4.0 components. The exchange of messages may be structured according to a grammar of an Industrie 4.0 language or may be implemented by using standardized APIs. Type 3 AAS interactions enable I4.0 components to become pro-active components which can support system integration and system interaction in a peer-to-peer environment between I4.0 components, e.g., to enable AAS assisted plug & play scenarios between I4.0 components"*
[33]*	*"An AAS Server Application with a corresponding interface for the I4.0 language is at the same time a client and a server and is a so called proactive AAS"*
[34]	*"AASs are able to interact autonomously with each other via a standardized interface with a common syntax and semantics base, thus realizing peer-to-peer AAS interaction. (...) AASs can understand and cooperate with each other. (...) An AAS with such autonomy is called a proactive one"*
[35]	*"(...) proactive AAS can implement intelligence. The proactive AAS, that is not fully specified yet, can realize semi- or full-autonomous interactions which enables order driven production"*
[36]	*"AASs with the ability to interact as equals among themselves, offering and requesting services when necessary to meet their goals"*
[37]	*"(...) can take decisions and participate in protocol-based interactions"*
[38]	*"(...) proactive AAS also provides CRUD (create, read, update, delete) capabilities but can additionally communicate and bid with other AAS using the I4.0 language"*

*Plattform Industrie 4.0 official document.

As summarized in Table 2.1, the definitions of AAS Type 3 presented in official documents, namely [31–33], as well as in selected research works, are brief, generic and remain limited in scope, evidencing the early stage of conceptual development. However, they consistently emphasize several common and core aspects: the ability to interact and collaborate through the I4.0 language (VDI/VDE 2193) [39, 40] and the integration of autonomy and

intelligence to support decision-making processes. Additional requirements, as well as concrete guidelines for the design and implementation of such components, are still in the early stages of development, particularly in official documents, which only highlight the importance of adopting the I4.0 language for communication.

Based on these definitions, this work proposes the following definition for AAS Type 3: *AAS Type 3 represents the next evolutionary step of the AAS concept, extending AAS Type 1 and Type 2 solutions. AAS Type 3 not only serves as a key enabler of interoperability in I4.0 environments but also introduces a higher degree of autonomy, intelligence, and collaborative capabilities. It acts as a decision-making entity capable of autonomously interacting with other AAS Type 3 instances to exchange information and collaborate, e.g., through negotiation. Furthermore, it possesses the intelligence to self-manage its associated asset, leveraging AI techniques to support the decision-making process and adaptive behavior throughout the asset's lifecycle.*

From the available definitions, it is possible to identify key characteristics for AAS Type 3, namely autonomy, intelligence, and collaboration. Although this set of characteristics is still limited, these elements provide a useful basis for recognizing and classifying an AAS solution as Type 3, serving as a first step toward the development of more comprehensive design guidelines and implementation strategies. In this context, and to provide a clearer understanding of these characteristics, this work defines them as follows:

- Autonomy: *"the ability to make your own decisions without being controlled by anyone else"* (Cambridge Dictionary). In this work, autonomy does not mean absolute freedom to decide arbitrarily but rather the capability to make decisions based on predefined criteria.

- Intelligence: *"the ability to learn and understand things quickly and easily"* (Cambridge Dictionary). In this work, intelligence can have different levels, e.g., from basic rule-based mechanisms to advanced AI techniques, as well as collective intelligence.

- Collaboration: *"involving two or more people or organizations working together for a particular purpose"* (Cambridge Dictionary). In this work, collaboration refers to AASs working together for a particular purpose. Unlike simple communication, collaboration goes beyond the exchange of information and comprises a deeper, interdependent process where all participants play crucial roles in achieving a common goal.

It is important to note that these are the expected minimum characteristics to be obtained by implementing an AAS Type 3 solution. However, this does not exclude the fact that the implemented solution can present other characteristics, e.g., adaptability, robustness, and pluggability, which are related to and/or resulted from these minimum characteristics.

2.3.3 Positioning the Role of Asset Administration Shell Type 3 in Industry 4.0

Despite the advancements brought by I4.0, including initiatives like RAMI4.0 and the concepts of CPS and I4.0 components, many companies still rely on systems based on the ISA-95 standard. ISA-95 provides a hierarchical approach for designing manufacturing systems across different levels, ensuring structured operations and integration between enterprise and production systems. However, its rigid structure limits flexibility, adaptability, and real-time responsiveness, making it difficult for companies to adjust quickly to evolving market demands.

I4.0 aims to enhance rather than replace the ISA-95-based systems, introducing more flexibility, adaptability, robustness, and interoperability into manufacturing systems. While AAS Type 1 and AAS Type 2 solutions improve interoperability and asset management by providing digital representations of assets, enabling systems at different levels of ISA-95 to share information in a structured manner, they do not fully implement the vision of I4.0. These types are limited in their capabilities when it comes to achieving the full potential of I4.0, which demands more dynamic, autonomous, intelligent, and collaborative systems. In this context, the AAS Type 3 plays a pivotal role in allowing companies to integrate and gradually transition to more flexible, adaptive, and interoperable systems aligned with I4.0 and RAMI4.0 without completely abandoning their existing infrastructure.

The adoption of AAS Type 3 should not be seen as a replacement for existing systems. Instead, AAS Type 3 can seamlessly integrate with established systems, e.g., Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES). It can work alongside MES to optimize production processes and provide feedback from the factory floor while complementing the broader business planning and resource management capabilities of the ERP. Rather than replacing these systems, AAS Type 3 adds autonomy, intelligence, and collaboration at the asset level, ensuring both operational and business decisions are better aligned and more efficient. By integrating AAS Type 3, companies can unlock the full benefits of I4.0, improving operational efficiency and decision-making while preserving their current systems' value and stability.

In summary, AAS Type 3 is a key enabler for transitioning traditional systems toward I4.0-compliant systems by facilitating the implementation of truly I4.0 components, a specific category of CPS aligned with RAMI4.0. Table 2.2 presents some examples of its applicability, demonstrating how AAS Type 3 extends the benefits of AAS Types 1 and 2 while surpassing traditional approaches.

2.3.4 Characterization of Asset Administration Shell Current Research

As previously discussed, the AAS is a standardized digital representation of an asset that encapsulates all relevant asset information throughout its lifecycle and enables stan-

Table 2.2: Comparison between traditional and AAS Type 3 approaches.

Traditional approach	AAS Type 3 approach*
Follows a strict hierarchy where ERP → MES → SCADA → PLCs → field assets control the flow of information. If an asset in the factory floor detects an issue, it must send a report to the upper levels (e.g., MES or ERP) for decision-making. This process introduces latency, and by the time a command is received, the issue may have already escalated	Decision-making happens locally at the asset level. Each asset has its own intelligence and can autonomously make decisions and adjust parameters. **Example:** When a machine detects excessive vibration, indicating tool wear or imbalance, the AAS enables local decision-making without involving MES or ERP systems. It can autonomously adjust parameters like spindle speed to reduce vibration or switch to a backup tool, ensuring smooth operations and preventing further wear
If critical central systems fail, production can be significantly impacted since assets rely on top-down commands. In the event of a system failure or communication breakdown, assets on the factory floor may not receive necessary instructions or updates, leading to production delays or inefficiencies. In some cases, operations may even come to a halt due to the lack of proper guidance	Assets can operate independently even when central systems are unavailable. **Example:** Each asset possesses its own knowledge base and decision-making capabilities. If a network failure occurs and disconnects the factory from the central system, assets continue to operate autonomously, utilizing local knowledge of their functions, parameters, and operating rules
Different assets often use distinct communication protocols, requiring custom middleware for interoperability. Asset-to-asset communication on the factory floor is limited, with data flowing primarily in a vertical hierarchy through central systems	Uses standardized interfaces, enabling decentralized asset-to-asset communication. **Example:** Assets can autonomously collaborate or negotiate with each other for load balancing, exchange operational data in real-time, and make decentralized decisions without relying on central systems
Maintenance is often time-based (scheduled servicing) or reactive (fixing after failure). Unexpected failures lead to downtime, costly repairs, and production losses	Collects real-time data and applies AI techniques to detect early warning signs of failure. **Example:** A factory using AAS Type 3 detects vibration anomalies in a motor, signaling potential failure. Instead of waiting for it to break, the system triggers predictive maintenance, avoiding costly downtime
If a factory needs to produce a new product, engineers must manually reprogram the system. This process is time-consuming and prone to errors	Assets can automatically reconfigure themselves based on production requirements. **Example:** A factory switches from product A to product B. Instead of engineers manually updating every machine, the AAS autonomously reconfigures assets based on preloaded product parameters

*In the column "AAS Type 3 approach", when referring to an asset's ability to perform a task, this capability is solely enabled by AAS Type 3. Without AAS Type 3, assets remain passive entities, relying on external systems or manual intervention for instructions.

dardized data exchange between I4.0 participants. This capability makes AAS a crucial facilitator of interoperability within I4.0 environments, where seamless integration and communication between various systems, devices, and platforms are essential. In this context, a bibliographic search was conducted to explore the AAS topic, particularly the AAS Type 3 design and implementation practices. This analysis not only provides a general overview of AAS solutions found in the literature but also focuses on identifying key enabling technologies, challenges, and research opportunities. These findings will provide a solid foundation and guidance for developing this thesis.

Overview

The bibliographic search was conducted in the Scopus scientific database on the topic "Asset Administration Shell", occurring in the title, abstract, or keywords. The search covered the period from 2015 to 2024, as the AAS concept was introduced only in 2015. The search was limited to papers written in English and documents of type Conference paper, Book chapter, and Article, as well as in the subject area of engineering and computer science. As a result, the query returned a total of 438 documents.

Despite being a relatively new topic with less than a decade of development, Figure 2.3 illustrates a rise in AAS-related publications, especially from 2018, when the initial AAS specifications were released. It is important to note that this survey primarily focuses on the AAS from an academic perspective. However, many companies within the industry are already integrating AAS into their solutions. For instance, at Hannover Messe fair 2023, Siemens, Bausch+Ströbel, Bosch Rexroth, CADENAS, Festo, HARTING, SICK, Phoenix Contact, and WAGO jointly demonstrate how the AAS can be implemented in practice [41].

Currently, the IDTA plays a crucial role in providing the necessary specifications for AAS and developing standardized submodel templates. The IDTA has around 100 members, including medium-sized companies and well-known large enterprises that are global technology leaders. This indicates that interest in AAS is growing in both the scientific community and the industry. Therefore, the relatively small number of scientific papers does not fully reflect the success and widespread adoption of AAS in practice, but they do offer valuable insights and foster new research and innovation in AAS solutions.

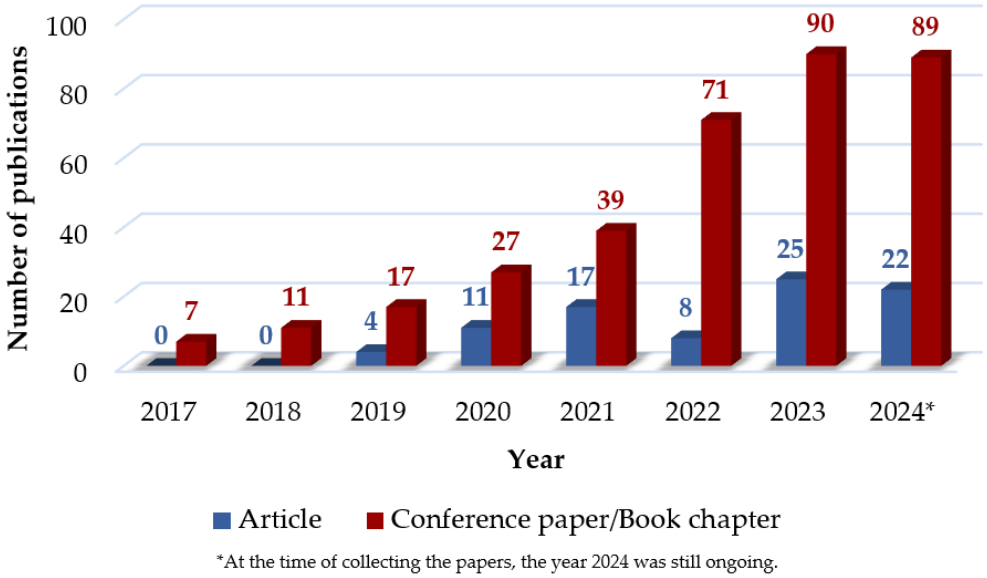

Figure 2.3: Number of scientific publications in recent years on the AAS topic.

Recurring to the VOSviewer tool[2], a software designed for creating and visualizing bibliometric networks, the publications were analyzed in more detail to identify the general research topics in this field based on the author's keywords. This tool utilizes the CSV file exported from Scopus with the query results, enables the analysis of co-occurrences, and generates a network map of author keywords. Figure 2.4 depicts the most common terms obtained from this analysis, representing the concepts, technologies, approaches, and application domains researched in the context of AAS.

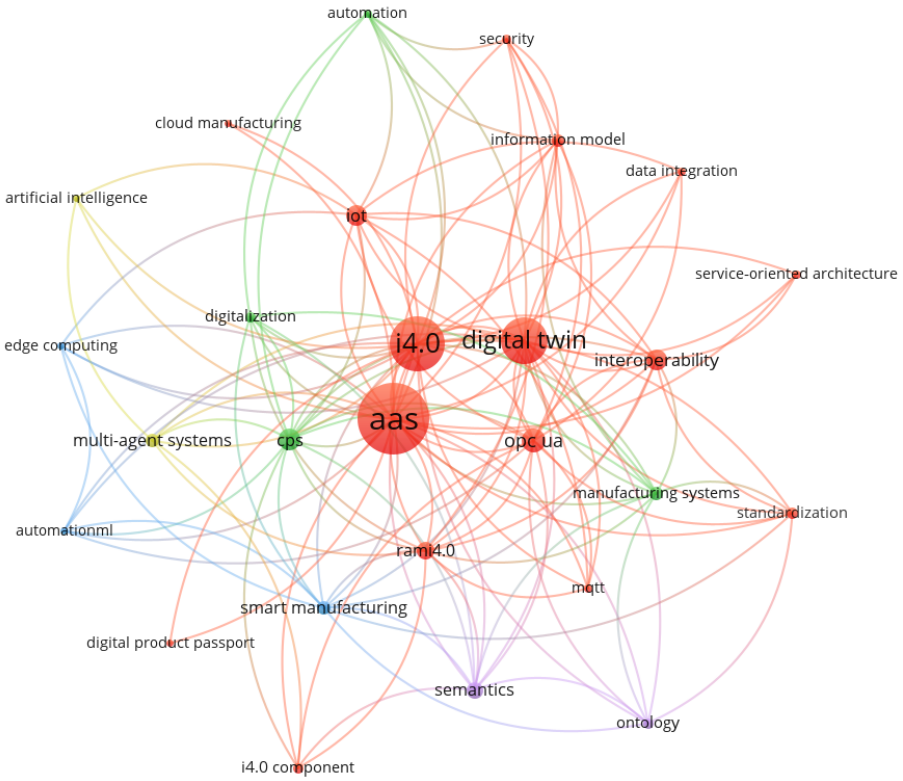

Figure 2.4: Co-occurrence network with the most common terms found in scientific publications in the AAS topic.

In this type of network, the size of the nodes indicates the frequency of the keywords, while lines between nodes represent links between the keywords. It is also important to consider the proximity of the keywords, which represents the set of keywords that are frequently used together.

As shown in Figure 2.4, the term AAS ("aas") represents the searched term, which explains its size, given that it appears as a keyword in most documents. This term is connected to related terms such as "rami4.0", "i4.0", "i4.0 component", and "cps". This connection is expected as the AAS reflects the asset in the digital world in the upper five layers of RAMI4.0, a reference architecture model for developing I4.0-compliant solutions. Furthermore, the AAS and its asset form an I4.0 component, a specific CPS category.

[2] https://www.vosviewer.com/

Along the "aas" term, it is also possible to see the main applications domains ("manufacturing systems" and "automation") and some terms related to its roles/functions, namely "digitalization", "standardization", "data integration", "information model", and "interoperability". In addition, supporting technologies including "opc ua", "artificial intelligence", "multi-agent systems", "iot", "automation ml", "mqtt", "edge computing", and "cloud manufacturing" are related, along with essential concepts like "semantics", "ontology", "security", and "service-oriented architecture" relevant to the implementation of AAS solutions.

Finally, the term "aas" is also associated with "digital twin" and "digital product passport", which are currently popular concepts related to digitization but with different purposes. In this sense, it is important to clarify the purpose of these concepts to avoid any misunderstandings, mainly between AAS and Digital Twin (DT). In the industrial and research community, it is often mentioned that an AAS implements a DT for I4.0, and to some extent, this can be true based on how the AAS is implemented and its intended functions. Despite both are digital representations of assets, DT focuses on replicating the dynamic behavior and state of an asset in a virtual environment. It enables real-time monitoring, simulation, and analysis of the asset, facilitating, e.g., predictive maintenance, optimization, and decision-making processes. The core idea behind a DT is to provide a near-real-time, data-driven virtual counterpart of the asset, which can interact with the physical world and offer insights based on its operational data. On the other hand, AAS focuses on providing a structured, standardized information model for an asset. It serves as a digital interface for the asset, enabling interoperability and seamless data exchange between different systems and stakeholders in an industrial context.

While AAS acts as a digital representation of the asset, it does not inherently include the dynamic behavior replication that characterizes DTs. However, it can be extended to include such functionalities if integrated with real-time data collection and control mechanisms and AI-based services/functions. This integration, however, is not a requirement for AAS but rather a possible enhancement that depends on the application's specific needs. When such an extension occurs, the AAS might adopt some DT functionalities, but it remains fundamentally different in its core purpose.

In this work that focuses on implementing an enhanced version of a traditional AAS, referred to as agent-based AAS (AAS Type 3), the AAS is enhanced to include some aspects that might overlap with those of a DT, as well as other functionalities not typically associated with DTs, such as autonomy and collaboration. However, it is crucial to clarify that the proposed solution is still classified as an AAS solution even with these enhancements. This distinction is essential for correctly positioning the proposed solution within the I4.0 and for understanding the scope and limitations of what it aims to achieve.

Content Analysis

From the 438 documents obtained in the bibliographic search, a methodology was defined to select the relevant documents for more in-depth analysis, aligning with the objectives of this thesis, which is to propose an approach to design and implement an AAS Type 3 solution. This methodology involves reading the abstract, title, and keywords of each document to identify which papers are related to the design and/or implementation of an AAS Type 3 or partial approaches/solutions that could potentially contribute to its design and implementation. The criteria for identifying if a paper relates to an AAS Type 3 solution is whether the proposed approach demonstrates some aspect related to intelligence, autonomy, or collaborative capabilities as described before. After this initial screening process, the next step involves thoroughly reading the identified documents (67) from the previous phase to verify that they are indeed related to AAS Type 3.

It is important to note that the described methodology was adopted since the search on the topic "Proactive Asset Administration Shell" or "Asset Administration Shell Type 3" or a similar search string on the Scopus database does not provide any significant result. However, this does not necessarily mean that there are no documents related to this topic, but rather that they do not contain the specific search terms in the title, abstract, or keywords, or may be named differently. For this reason, it was necessary to identify papers that discussed AAS Type 3 solutions by using the described methodology.

Based on this screening and review process, it was identified that only 32 documents are associated with AAS Type 3. Figure 2.5 shows the distribution of AAS research works categorized as "AAS Type 3 solutions" (includes solutions focusing on the design and/or implementation of an AAS Type 3 and partial approaches/solutions that could potentially contribute to its design and implementation) and "Other AAS solutions" (which includes AAS Type 1 and 2 solutions and specific solutions supporting the AAS, such as semantics, security, middlewares, development platforms, etc.). This work does not delve further into the reasons that may explain the limited number of documents associated with AAS Type 3. However, it raises some possible reasons: **(1)** AAS is a recent topic under development for less than a decade. In recent years, the focus has been on AAS Type 1 and 2 solutions, which is understandable as they are considered less complex to design and implement than AAS Type 3; **(2)** the lack of specifications and official documents to guide the development of the AAS Type 3 presents a challenge in designing such components.

From the selected documents (32) categorized as "AAS Type 3 solutions", the main research works are summarized in Table 2.3. The table includes information about the solution description, the supporting technologies, and whether the proposed solution possesses attributes such as autonomy, intelligence, and collaboration. Once more, it is important to clarify that an AAS Type 3 solution in this review process may not be a complete AAS Type 3 solution. It can also be a partial solution that addresses certain aspects related to its

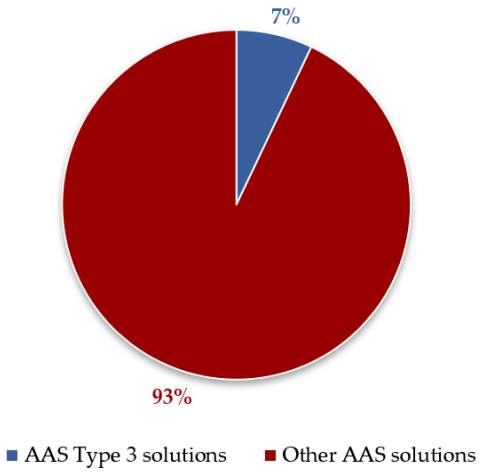

Figure 2.5: Distribution of AAS research works.

implementation, such as autonomy, intelligence, and collaboration. Partial solutions were also considered, as they provide relevant insights that could help in the implementation of an AAS Type 3. The criteria for determining whether a solution is Type 3 complete or partial is based on its attributes (as indicated in Table 2.3). If the solution possesses all of them, it can be considered complete. Otherwise, it is considered partial. Furthermore, varying levels can still be observed within these criteria. For instance, one solution may exhibit a higher degree of collaboration than another. However, this work focuses solely on identifying and discussing these criteria, as most existing research follows a binary approach (i.e., either it works or it does not), which can lead to subjective and inconsistent comparisons.

Table 2.3: Summary of the main research works related to the AAS Type 3.

Research work	Solution description	Supporting technologies	Attributes		
			Autonomy	Intelligence	Collaboration
Ye and Hong [42]; *Ye et al.* [43]; *Ye et al.* [44]	- Presents an implementation example of deploying AASs, particularly demonstrating the exchange of information between AASs using OPC UA [42]; Similar solution but focused on enabling the Plug-and-Produce strategy [43]; Proposes a method for implementing AASs based on a three-layer architecture comprised by field assets, edge AAS deployment, and cloud AAS management [44] - **Asset:** laboratory prototype (robots, conveyors, sensors and PLCs) [42]; industrial robots and turntable [43, 44]	OPC UA and AML [42, 43]; OPC UA, Edge and Cloud [44]	–	✓	✓

Continued on next page

Research work	Solution description	Supporting technologies	Attributes		
			Autonomy	Intelligence	Collaboration
Cavalieri and Salafia [45]	- Proposes an AAS-based model for predictive maintenance using logical blocks to abstract all required functionalities for the process. Basic blocks are implemented within AAS submodels at the Edge level, while advanced blocks based on ML are implemented at the Cloud level - **Asset:** milling machines	Edge, Cloud and ML	–	✓	–
Sidorenko et al. [46]	- Proposes an OPC UA model for I4.0 language messages necessary for a skill execution interaction protocol that can be utilized for collaboration between AASs - **Asset:** product and transport	OPC UA	–	–	✓
Ocker et al. [47]; *Vogel-Heuser, Ocker and Scheuer* [38]	- Discusses that the asset information described in the AAS may be used as a standardized agent's knowledge representation, i.e., the agents get all the relevant information to make decisions from the AAS submodels [38, 47] - **Asset:** demonstrator (a stack, a crane, a stamp, and a conveyor with ramps and pushers) [38, 47]	MAS [38, 47]	✓	✓	✓
Arm et al. [48]	- Presents some collaborative aspects of AAS Type 3, namely the negotiation process between AASs based on production priorities and requirements - **Asset:** virtual assembly line (machines, transport robots and storage racks)	OPC UA and MQTT	–	–	✓
Zhou et al. [49]	- Proposes an AAS-based production management platform that covers the demand and production sides. This platform enables vertical communication between AASs using OPC UA and presents modules to decouple complex production functionalities - **Asset:** robotic arm, laser maker, camera, PLC, screw machine and servo motor	OPC UA and ML	–	✓	✓
Jungbluth et al. [50]	- Concentrates on modeling AAS submodels for resource and product agents to use in dynamic replanning applications - **Asset:** demonstrator (a module for transportation, assembly, flashing of information on the product and two quality control modules)	MAS	✓	✓	✓
Simon et al. [51]	- Presents an approach for implementing a shared production scenario using the AAS and the I4.0 language, which involves a negotiation to find suitable manufacturers for customized products - **Asset:** product and factories	–	–	–	✓

Continued on next page

Research work	Solution description	Supporting technologies	Attributes		
			Autonomy	Intelligence	Collaboration
López, Casquero and Marcos [52]	- Proposes a design pattern for implementing AASs based on industrial agents - **Asset:** product and industrial robot	MAS	✓	✓	✓
Grunau et al. [37]	- Introduces possibilities for transitioning from AAS Type 2 to Type 3, along with modules/applications designed to support this transition, particularly to enable message exchange between AASs using MQTT, while ensuring alignment with the I4.0 language - **Asset:** storage and product	MQTT	–	–	✓
Sakurada, Leitão and De la Prieta [53]	- Presents an agent-based approach to enhance the digitization process of assets, particularly by enabling the development of intelligent and collaborative AASs - **Asset:** assets simulated in a virtual environment	MAS	✓	✓	✓
Kannan et al. [54]	- Describes an I4.0-compliant digitization of a laser cutter module using AAS and the adoption of I4.0 language to facilitate the interaction with this module - **Asset:** laser cutter module	MQTT	–	–	✓
Sidorenko et al. [55]	- Proposes a framework that combines MAS with AAS to enable the human-centered production paradigm. The focus is on the active involvement of operators in the industrial collaboration process with other assets - **Asset:** human, robot and camera	MAS, REST and OPC UA	✓	✓	✓
Xia, Jazdi and Weyrich [56]	- Proposes an approach that integrates LLM with AASs to plan and control production processes - **Asset:** transport robots and painting stations	LLM	–	✓	–
Stolze et al. [35]	- Proposes an approach for implementing the bidding protocol of the I4.0 language for AAS Type 3 communication using BPMN workflows - **Asset:** products and resources	–	–	–	✓
Shin and Um [57]	- Presents a method for generating and deploying data analytics models in AASs for an energy-efficient manufacturing scenario - **Asset:** production machines	AI	–	✓	–

Continued on next page

Research work	Solution description	Supporting technologies	Attributes		
			Autonomy	Intelligence	Collaboration
Bernhard et al. [58]	- Introduces a MAS architecture designed to enable shared production. The proposed architecture is grounded in holonic principles, and the AASs serve as the self-description mechanism for these holons, providing the necessary information for their operation, interaction, and integration - **Asset:** robot module, assembly module, 3D-printer module, connector module, conveyor module and quality module	MAS	✓	✓	✓

Notation: – Not Addressed; ✓The solution as a whole has this attribute.

In summary, the research works presented in Table 2.3 highlights Open Platform Communications Unified Architecture (OPC UA) and MAS as key technologies for AAS Type 3. OPC UA is widely used for message transmission between AASs, while MAS enhances autonomy, intelligence, and collaboration in AAS solutions. This is evident in the "attributes" column of Table 2.3, where research incorporating MAS encompasses all listed attributes. Additionally, other technologies such as Edge and Cloud are involved in hosting AASs and distributing intelligence, and Machine Learning (ML) and Large Language Model (LLM) are used to enhance the intelligence of the AAS solutions.

The authors in [42–44, 49] propose solutions using OPC UA to enable communication between AASs. However, their approaches separate the intelligence and decision-making process from the AAS. For instance, in [49], a higher management layer focuses on service-oriented functions such as monitoring, scheduling, control, and anomaly detection. This upper layer retrieves asset information using OPC UA to monitor and control the process and make decisions. While the overall system can be considered intelligent, the AAS itself is not an autonomous and intelligent entity.

Aligned with the VDI/VDE 2193 standard (I4.0 language), [46] suggests an OPC UA model for I4.0 language messages necessary for a skill execution interaction protocol. Also, [54] illustrates an I4.0-compliant digitalization of a laser cutter module using AAS, where communication with the AAS is facilitated through the use of I4.0 language via Message Queuing Telemetry Transport (MQTT). This implementation involves the exchange of I4.0 messages in JSON format between the service provider and requester, following the bidding protocol specified in VDI/VDE 2193-2. Although these research works do not specifically address the design and implementation of an AAS Type 3, the proposed model and interaction protocols can be utilized for collaboration between AASs that support OPC UA and MQTT for transmitting the I4.0 language messages.

A predictive maintenance model based on AAS is proposed in [45], which mainly involves a data flow between AASs at the Edge and advanced logical blocks based on ML in the Cloud. The asset data is analyzed in the Cloud, where decisions are made and transmitted to the AASs. Despite the solution introducing some level of intelligence to the system, each AAS lacks the autonomy to make decisions on its own, as decisions are exclusively made in the Cloud. Additionally, there is no collaboration between the AASs, as only pre-designed data flows for data collection and aggregation are present.

A systematic method for generating and deploying data analytics models in AASs is presented in [57]. This approach focuses on generating energy prediction models using AI techniques and converting these models into standardized formats. After conversion, these models are integrated into AAS submodels, which external clients can request and utilize for energy prediction. While this method significantly enhances interoperability in manufacturing intelligence, it does not address the autonomy and collaborative aspects.

In [48], the main focus is introducing a "ConfigWizard" software designed to facilitate the semi-automated generation of AASs. However, the paper also presents some aspects of AAS Type 3 during experimental tests, particularly focusing on the negotiation process between AASs based on production priorities and requirements. The paper also considers scenarios involving failure and repair times.

The authors in [51] expand the factory boundaries by proposing an AAS-based solution to enable shared production scenarios. In this setup, trusted partners can dynamically share manufacturing capabilities and services over digital interfaces throughout the product lifecycle. The process involves a negotiation to find suitable manufacturers for customized products. This negotiation is facilitated by a platform, so the interaction is not directly between AAS entities. The paper focuses more on modeling AAS submodels and presenting examples of interaction patterns using the I4.0 language rather than providing details about practical implementation and adopted technologies.

An innovative approach that integrates LLMs with AASs is proposed in [56]. This work uses LLM to interpret complex information, generate insights, and support decision-making processes in industrial automation systems. The information is primarily sourced from AASs, which provide descriptive details about a specific asset. Although LLM can enhance the decision-making process, the authors highlight several practical challenges that should be considered before adopting this type of solution. Furthermore, aspects related to autonomy and collaboration are not addressed in detail in this work.

Among the research works more aligned with the concept of AAS Type 3, [35, 37] explicitly propose solutions to support the implementation of AAS Type 3. [37] introduces possibilities for transitioning from AAS Type 2 to AAS Type 3, along with modules/applications designed to support this transition, particularly to enable message exchange between AASs using MQTT, while ensuring alignment with the I4.0 language. On the

other hand, [35] presents an architecture for implementing AAS Type 3. The solution employs a Business Process Model Engine (BPME) capable of interpreting and executing behavior descriptions modeled as Business Process Model Notation (BPMN) workflows. This study primarily focuses on the communication/collaboration aspect, presenting how to translate the bidding protocol of the I4.0 language into the BPMN model and proposing a submodel to support this process. Although both research works propose solutions that can exhibit intelligence, autonomy, and collaboration, they focus mainly on the communication and collaborative aspects of AAS Type 3. To provide a clearer and more systematic implementation process for AAS Type 3, an extended specification of these works are required.

In contrast, other research efforts focus on utilizing MAS from diverse perspectives. For example, [52] proposes a pattern for implementing AASs based on industrial agents, and [53] presents an agent-based approach to enhance the digitization process of industrial assets, particularly by enabling the development of intelligent and collaborative AASs. Other research works mainly focus on developing solutions that use the AAS to leverage MAS solutions. For instance, [38, 47] discuss that the asset information described in the AAS may be used as a standardized agent's knowledge representation, i.e., the agents get all the relevant information to make decisions from the AAS submodels. Following a similar strategy, [50] concentrates on modeling AAS submodels for resource and product agents to use in dynamic replanning applications, and [58] introduces a MAS architecture designed to enable shared production grounded in holonic principles. In this approach, AASs serve as the self-description mechanism for the holons, providing the necessary information for their operation, interaction, and integration. Moreover, [55] proposes an approach that combines MAS with AAS to enable the human-centered production paradigm. Similar to the traditional collaboration between AAS products and AAS resources, this approach also aims to include AAS representing operators in this process.

The research works mentioned above clearly show the potential and feasibility of solutions aligned with the concept of AAS Type 3, especially using MAS. While the research works [59, 60] primarily present conceptual ideas that explore the use of MAS for implementing AASs, other studies [52, 53, 58] remain at a preliminary stage or require further extensions to support broader industrial applications. In contrast, more comprehensive approaches are proposed in [38, 50], which could potentially be adapted for use beyond their original, specific cases. However, even these approaches would require modifications or enhancements to become more widely applicable. Specifically, they would need to be abstracted to a higher level in order to generalize beyond their initial target applications. This may involve the development of a detailed specification to guide the design of AAS Type 3, ensuring that these systems are sufficiently generic, independent of specific use cases, and capable of supporting various collaboration strategies.

2.4 Software Agents, Industrial Agents and Multi-agent Systems

As discussed in the previous section, MAS represent a key enabling technology for implementing AAS Type 3. In this context, this section introduces important concepts related to software agents, industrial agents, and MAS, as well as discusses some of their applications in the context of I4.0 and CPS found in existing literature and R&D projects.

2.4.1 Overview

MAS is a paradigm derived from distributed AI that has been intensively studied since the 1980s. The development of agent-based applications has gained significant attention due to the distributed and autonomous nature of these software entities. Agents can operate independently, adapt to their environments, and coordinate with one another, making MAS particularly suitable for addressing complex, real-world problems [6]. In this context, the following section begins by introducing the concept of an individual agent to provide a foundational understanding of the building blocks of MAS. Additionally, particular attention is given to the so-called industrial agent, a specialized type of agent designed to meet the specific demands of dynamic industrial environments.

Although no universally accepted definition exists, and the term remains a subject of ongoing debate and controversy [6], in this work an agent will be defined as being:

> *"An autonomous component, that represents physical or logical objects in the system, capable to act in order to achieve its goals, and being able to interact with other agents, when it does not possess knowledge and skills to reach alone its objectives"* [61].

Just as the definition of an agent lacks universal consensus, various classifications exist for categorizing agents. This work does not aim to discuss and compare these different typologies, e.g., reactive, deliberative, and hybrid (see [61–63]). However, it is generally accepted that an agent possesses several essential characteristics at its core [6, 62, 63]:

- Encapsulation: Agents represent physical or logical objects within an environment.
- Autonomy: Agents operate without direct human or external intervention and have control over their actions and internal state.
- Learning ability: Agents can learn from their experiences.
- Reactivity: Agents perceive their environment and respond in a timely manner to changes in order to fulfill their design objectives.
- Proactiveness: Agents exhibit goal-directed behavior by taking initiative to achieve their design objectives.

- Social ability: Agents interact with other agents (and possibly humans) to achieve their design objectives.

This work focuses on agents specifically designed to meet the I4.0 requirements, known as industrial agents (from now on when simply referring to the term agent, it means industrial agent), a specialization of traditional software agents, which inherit their characteristics and are enhanced with additional capabilities to comply with several industrial requirements, namely hardware integration, reliability, fault-tolerance, scalability, industrial standard compliance, quality assurance, resilience, manageability and maintainability [8].

> *"An industrial agent is an agile and robust software entity that intelligently represents and manages the functionalities and capabilities of an industrial unit. While it reveals the common features of an advanced agent, it also has some specifics. It understands and efficiently handles the interface and functionality of (low-level) industrial devices. Usually it belongs to an agent-based industrial application system within which it acts and communicates in an efficient, intelligent, collaborative, and goal-oriented way. In principle, it is an autonomous and self-sustained unit. Nevertheless, it accepts and follows company guidelines, codes of conduct, general laws, and relevant directives from higher levels. Moreover, especially in emergency and real-time scenarios, its autonomy may be compromised in order to permit fast and efficient reactions"* [64].

Generally, industrial agents follow a two-layered approach (see Figure 2.6) comprising a high-level control (HLC) and a low-level control (LLC) [8, 65]. The HLC is responsible for providing intelligence and adaptation features to support decision-making processes, e.g., through monitoring, diagnosis, prediction, and optimization functions. However, it is incapable of delivering real-time performance. On the other hand, LLC is responsible for executing low-level control tasks that guarantee real-time constraints, often through PLCs or industrial PCs running control programs compliant with IEC 61131, IEC 61499, or similar standards. This separation allows the HLC to handle complex, high-level decision-making, while the LLC maintains stable, real-time operations.

This structure aligns closely with the CPS, where the HLC serves as the "cyber" part, providing intelligence and adaptability, and the LLC functions as the "physical" counterpart, executing real-time control tasks [65]. This alignment makes the industrial agent approach suitable for implementing cyber-physical components, as well as I4.0 components, as long as adaptations are made to meet the specific requirements of the AAS, ensuring compliance with AAS specifications, enabling interoperability, seamless integration, and effective communication within I4.0 environments.

Addressing complex real-world problems often requires the combined and cooperative efforts of multiple agents, as a single agent has limited capabilities. As illustrated in Figure 2.6, a MAS can be described as a society of intelligent, autonomous, and cooperative agents that may represent physical or logical objects of a system. In contrast to tradi-

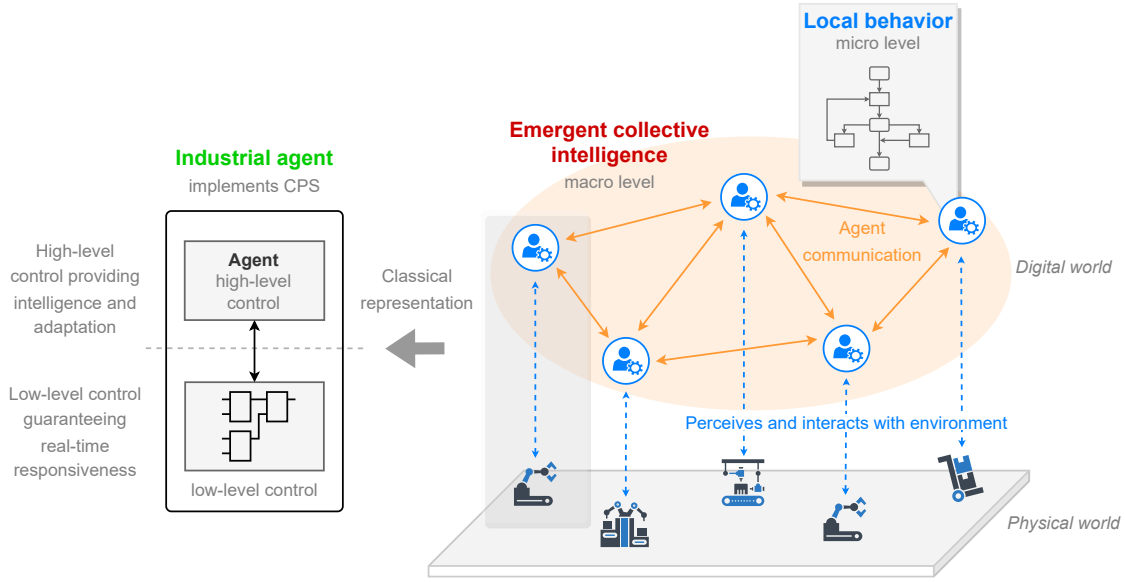

Figure 2.6: Multi-agent systems overview.

tional centralized control strategies, MAS offers a decentralized approach to designing large-scale, complex systems by decentralizing the system's control and distributing it to autonomous and cooperative agents rather than being concentrated in a single authority. The overall functionality of the system emerges through interactions among these individual agents, e.g., through collaboration, negotiation, and delegation of responsibilities, where each agent contributes with unique knowledge and skills toward achieving both individual and shared objectives. Furthermore, MAS enables the decomposition of complex tasks into manageable subtasks, each assigned to a capable agent, ensuring that even intricate challenges can be effectively addressed in a decentralized manner [6, 20].

Traditional MAS elements, such as ontologies (e.g., [61, 66]), Agent-Oriented Software Engineering (AOSE) methodologies (e.g., [67, 68]), and design patterns (e.g., [69, 70]), are crucial for the design and functioning of MAS. However, this work does not focus deeply on these elements, as they do not directly align with the specific objectives of the proposed approach. Instead, this work proposes an agent-based approach specifically tailored to realize an AAS Type 3. Unlike a traditional MAS, the goal here is not to create a purely MAS-centric system but to integrate agents to enhance the autonomy, intelligence, and collaborative capabilities of traditional AASs. As a result, the proposed design process for developing an AAS Type 3 does not strictly adhere to the methodologies and design patterns typically associated with MAS, as these may not be directly applicable. Future research could explore how MAS elements could be adapted to complement or further enhance the approach introduced in this work.

2.4.2 Multi-agent System Approaches in Industry 4.0

MAS can be applied to solve a wide range of problems across several industrial applications in a CPS context, namely smart production, smart electric grids, smart logistics, and smart healthcare [7]. Several research works and European R&D projects have demonstrated the benefits of introducing MAS in industrial environments, particularly offering alternatives to overcome the typical problems presented by the traditional centralized control approaches, which are not able to address the flexibility, robustness, and reconfigurability imposed on the current industrial systems.

While some agent-based AAS research works have already been discussed in Section 2.3.4 ("Content Analysis"), this section focuses on presenting additional applications and advantages of MAS in industrial scenarios based on existing literature and R&D projects, reinforcing that MAS is a key technology for realizing CPS, particularly providing advanced features for AAS. Given the availability of several surveys and discussion papers on MAS (e.g., [7, 8, 59, 60, 71–73], to name a few), which extensively cover their applications, as well as past, present, and future trends, the goal here is not to provide an exhaustive review. Instead, this section aims to offer an overview for readers, illustrating how MAS have been applied in industrial environments.

The methodology employed involves a targeted review of highly mature research works (including only journal articles and Ph.D. theses) and R&D initiatives that offer a solid theoretical and practical foundation for MAS approaches in I4.0, particularly focusing on agent-centric solutions, i.e., solutions that address the roles and interactions of agents within MAS and how their collective behavior enhances overall system performance. The solutions discussed are designed to be as generic as possible, prioritizing the architecture and principles of agent behavior over the specifics of algorithms or embedded techniques. Below is a summary of the most relevant research works reviewed.

Starting with classic holonic control architectures, which laid the groundwork for many of the features now associated with I4.0 and CPS, and to modern agent-based systems, these architectures emphasize decentralized, flexible, agile, robust, and adaptive control by structuring systems into autonomous and cooperative units called holons[3]. Each holon combines data processing and physical functions, mirroring the integration of cyber and physical elements seen in CPS and industrial agents. In this context, some relevant holonic architectures deserve mention due to their contributions and inspiration to the development of subsequent MAS-based approaches.

[3]Despite the differences between agents and holons, the holon capabilities described here are analogous to those of software and industrial agents. Thus, the holonic architectures presented are interpreted from a MAS perspective.

PROSA (Product-Resource-Order-Staff Architecture) [74] is one of the earliest and most influential holonic architectures designed for manufacturing systems to enhance adaptation and flexibility. It introduces holons to represent key elements of the production process: products, resources, and orders, which interact autonomously in a decentralized manner. In addition, staff holons can be incorporated to provide assistance and advice to the basic holons. While the basic holons remain responsible for decision-making, staff holons act as external experts offering guidance, especially in cases where problems are too complex to resolve through decentralized decision-making alone.

ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) [61], inspired by PROSA, addresses the agile reaction to emergence and change, increasing the robustness and flexibility of manufacturing systems. The architecture is designed to move between a centralized approach for global optimization and a more heterarchical approach to handle unexpected events and system modifications. ADACOR defines four types of holons: product, task, and operational holons, which represent manufacturing components and operate autonomously, coordinating their tasks in a decentralized manner. Additionally, a supervisor holon is introduced to provide coordination and global optimization within the decentralized control. Building on the original architecture, ADACOR2 [75] extends these principles by incorporating self-organization capabilities to achieve evolvable and reconfigurable manufacturing systems.

These holonic architectures have made significant contributions, inspiring future MAS-based approaches. Since their inception at the close of the 20^{th} century and the beginning of the 21^{st}, MAS-based approaches have continued to evolve, particularly with the advent of I4.0 and CPS, which have further refined and expanded their applications. In this sense, the following summarizes some research works from the last decade, providing an overview of recent advances in this field.

In [76], the author presents a MAS control architecture designed to enhance system flexibility by distributing control decisions across five types of agents. The High-availability agent is responsible for dealing with external disturbances, the Rescheduler agent is responsible for data processing within the process control, the Executive agent is applied to interact with legacy control systems, the Supervisor agent is applied to perform supervisor tasks, and the Dispatcher agent deals with the knowledge base to ensure sustainable control. This architecture was designed for three distinct application scenarios, demonstrating its reusability and adaptability for various industrial contexts.

The authors in [77] model shop floor assets as intelligent agents (namely machining, conveying, product, and supplementary agents) endowed with autonomous decision-making and cooperative capabilities. This approach enables the production of multiple products while addressing challenges like deadlocks that disrupt workflows and reduce efficiency. By employing negotiation mechanisms, these agents collaborate to optimize operations and prevent conflicts. The authors also identify the specific conditions that lead

to deadlocks and propose four strategies to mitigate these risks through agent interactions, supported by big data based feedback and coordination.

An agent-based cloud-assisted self-organized architecture, called CASOA, is presented in [78]. CASOA comprises four types of agents: suggestion agents, product agents, machining agents, and conveying agents, each focusing on specific functions within the manufacturing system. These agents operate in a decentralized manner, exchanging reasoning information via their knowledge bases. In addition, a cloud-assist mechanism represented by the suggestion agent infers suggestion plans for optimizing lower agents' scheduling rather than directly allocating the tasks. Experimental results validated the architecture's efficiency and reliability, demonstrating significant advantages of the dynamic scheduling method over traditional static scheduling, as well as high robustness and adaptability to frequent product changes and disturbances.

In [79], a MAS architecture is proposed to enable dynamic, online, and decentralized service reconfiguration. The architecture primarily involves resource and product agents that interact based on service-oriented principles. Resource agents act as service providers, encapsulating machine operations such as drilling or welding as services, which are published in a repository and accessible through discovery mechanisms. Product agents, acting as service consumers, utilize these services to complete tasks and meet production demands. These agents employ various strategies to proactively identify reconfiguration opportunities, such as facing service failures, service performance degradation, or production changeover. The experimental results demonstrated the feasibility and benefits of the agent-based solution, exhibiting adaptability and responsiveness in the face of changing conditions.

A self-organized cyber-physical conveyor system using MAS is presented in [80]. In this system, each conveyor is managed by an agent embedded with self-organization techniques that can autonomously interact with each other and adapt to condition changes. For instance, when a conveyor agent is initiated and connected to the transfer system, it starts without knowledge of its position in the sequence. Through interaction with other agents, the conveyors collaboratively determine their sequence positions, ensuring seamless transfer operations. This functionality becomes crucial when system changes occur, such as adding, removing, or swapping modules, enabling a modular and reconfigurable conveyor transfer system.

The author in [81] proposes an agent-based CPS architecture that distributes data analysis across Edge and Cloud layers. Edge agents focus on real-time, simple data processing to ensure fast responses, while Cloud agents handle advanced, computationally demanding tasks for decision-making and system optimization. This architecture enables vertical collaboration by allowing Cloud agents to supervise and guide Edge agents, while horizontal collaboration plays a critical role in further decentralization by giving agents a high degree of independence and autonomy from other computational layers. Successfully

implemented in an automotive manufacturing plant, the system enabled early detection of defects through the decentralized and collaborative data analysis performed by the agents.

Furthermore, it is important to highlight some works that, beyond focusing on the functionality of the MAS, strive to align with the RAMI4.0. For instance, [82] presents a MAS architecture aligned with both RAMI4.0 and the main standards concerning industrial agents (e.g., [65, 83]). On the other hand, [36] proposes an I4.0 platform aligned with RAMI4.0, which offers an infrastructure for managing and supporting I4.0 components implemented as industrial agents. This alignment ensures that the developed MAS solutions are not only operationally robust but also fully compatible with the principles and guidelines of I4.0, enabling seamless integration into industrial environments.

In addition to the research found in the literature, several European R&D initiatives have demonstrated MAS applications in real-world industrial environments. GRACE [84] project adopts a MAS approach that focuses on integrating the process and quality control in a washing machine production line, enabling the real-time monitoring, feedback control loops and online self-adaptation for adjustment of process/product variables. ARUM [85] project combines MAS with service-based architectures to improve decision support in planning and control, aiming to respond faster to unexpected events in a production of highly customized products, such as aircraft and ships. IDEAS [86] and PRIME [87] projects focus on using MAS to enable the plug-and-produce configuration of industrial production systems, allowing rapid reconfiguration and deployment, and evolutionary system adaptation. Finally, GO0D MAN [88] project uses a MAS-based zero defect manufacturing solution to develop an adaptive system for real-time monitoring and early detection of defects along the different processes in an automotive assembly line.

While research in this area continues to grow, standardization and research initiatives led by working groups such as the IEEE-IES Industrial Agents Technical Committee[4] and the VDI/VDE-GMA Technical Committee 3.35 "Agent Systems"[5] are playing a key role in advancing the application of industrial agents in real-world scenarios. These efforts aim to bridge the gap between theoretical developments and practical implementations, promoting the widespread adoption of MAS-based solutions in industrial environments. It is also important to mention the past efforts of the Foundation for Intelligent Physical Agents (FIPA)[6], which, despite being an older initiative, introduced a series of specifications that facilitate the development of agent-based solutions and remain relevant today.

Furthermore, with the advent of AAS and DT, alongside the recent popularity of AI, particularly through LLM techniques, new perspectives are emerging for future advances

[4] https://tcia.ieee-ies.org/
[5] https://www.mec.ed.tum.de/en/ais/committee-works-and-memberships/vdi-vde-gma-fa-335/
[6] http://www.fipa.org/

in MAS solutions. Specifically, the proposed agent-based AAS approach could play a significant role in this trend, leveraging agent capabilities to enhance the AAS.

2.5 Exploring Synergies Between MAS, AAS and RAMI4.0

The discussions in the previous sections have addressed AAS Type 3 and MAS in isolation, although some research on agent-based AAS related to AAS Type 3 has been discussed. From Section 2.3, the conclusions can be summarized as follows: AAS Type 3 remains an underexplored topic, but emerging research links it to MAS, which bring autonomy, intelligence, and collaboration to AAS solutions. While the benefits of existing agent-based AAS solutions are evident, they must be abstracted to a higher level to generalize beyond their initial applications. This may involve the development of a detailed specification to guide the design of AAS Type 3, ensuring that these systems are sufficiently generic, independent of specific use cases, and capable of supporting various collaboration strategies.

On the other hand, Section 2.4 presented that research on MAS have significantly advanced in recent years, particularly with the emergence of I4.0 and CPS. MAS solutions reviewed in the literature and through R&D initiatives exhibit several common characteristics, including adaptability, robustness, scalability, reconfigurability, reusability, pluggability, responsiveness, and the ability to design complex, large-scale CPS-based systems. Additionally, the development of MAS is supported by international technical committees that aim to advance the application of industrial agents in real-world scenarios. These committees also promote standardization initiatives, promoting the widespread adoption of MAS-based solutions in industrial environments.

Based on these conclusions, it is reasonable to consider MAS as a viable and effective approach for implementing AAS Type 3. In this context, this section seeks to establish parallels between MAS, AAS Type 3, and RAMI4.0, aiming to explore and discuss how these concepts are interrelated to develop a specification for designing and implementing AAS Type 3 using MAS. Additionally, the goal is to provide a brief, conceptual, and non-technical overview to support the reader's understanding before moving on to more technical specifications in the subsequent chapter.

This section adopts two complementary viewpoints to elucidate the relationships between MAS, AAS Type 3, and RAMI4.0. The first viewpoint compares agents and AAS Type 3, highlighting the synergy between the two concepts. The second viewpoint aligns more closely with RAMI4.0, emphasizing how agents can enhance the digitization process across the layers defined by RAMI4.0, which are directly related to AAS. Both viewpoints are valid and complementary, aiming to demonstrate how agents can support the implementation of AAS Type 3.

2.5.1 Viewpoint 1: Leveraging MAS to Enhance AAS

Agents and AAS, particularly AAS Type 3, share similarities despite being distinct concepts developed for different purposes. The primary objective of the AAS is to facilitate interoperability, enabling seamless integration and communication across diverse systems, devices, and platforms. AAS Type 3, however, represents an enhanced version that extends beyond interoperability by incorporating autonomy, intelligence, and collaborative capabilities. As previously described in Section 2.3, AAS Type 3 must present an active behavior capable of autonomously initiating communication, negotiation, and decision-making, which is achieved through autonomy, intelligence, and collaborative capabilities. Interestingly, this definition aligns closely with the definition and characteristics of agents, as outlined in Section 2.4. To illustrate this concept in an accessible manner, a system composed of AASs Type 3 can be conceptualized as a MAS, where each AAS is an autonomous and intelligent entity representing a specific asset. These AASs not only manage the lifecycle information of their respective assets but also engage in interaction strategies, e.g., collaboration and negotiation, in order to achieve the system goals.

It is important to highlight that this analogy does not imply that an agent is equivalent to an AAS or vice versa, nor that one can fully implement the other. Instead, agent should be understood as a complementary technology that enhance the functionality of an AAS rather than fully replicating it. In this context, when this work refers to implementing AAS Type 3 using MAS/agent technology, it specifically means leveraging MAS/agent capabilities to provide additional or complementary functionalities that extend traditional AAS solutions (i.e., Types 1 and 2) to achieve the advanced characteristics of AAS Type 3.

The structure of an AAS, as defined by the official metamodel in [27], is based on a set of submodels that describes the asset information throughout its lifecycle, and this structure can be extended to function like an API (i.e., AAS Type 2) [28]. Currently, there is no official specification for the design of AAS Type 3, which would require additional structural components/modules to enable autonomous decision-making and collaborative interactions with other AASs. Agents can provide this additional structure, particularly by providing the necessary intelligence, autonomy, and communication capabilities. Furthermore, in the case of industrial agents, they offer the ability to interface with assets in compliance with industrial requirements. A significant advantage of this approach is its modularity, as it avoids altering the original AAS structure, requiring only the addition of supplementary components/modules.

This first viewpoint highlights that MAS technology can provide the autonomy, intelligence, and collaborative capabilities required for AAS Type 3, establishing a conceptual basis and starting point for the specifications presented in the Chapter 3.

2.5.2 Viewpoint 2: Alignment with RAMI4.0

As aforementioned, the AAS reflects the asset in the digital world and is conceptually aligned with the upper five layers of RAMI4.0, namely integration, communication, information, functional, and business (see Section 2.2). However, the main AAS specifications (Part 1 [27] and Part 2 [28]) primarily address the communication and information layers. While the existing specifications provide a solid foundation for organizing and exchanging asset information, the integration, functional, and business layers are not explicitly defined. These layers require complementary technologies and approaches, particularly to achieve the intelligence, autonomy, and collaboration capabilities envisioned for an AAS Type 3. In this context, this section explores and reflects how agents can implement or enhance each of these layers. To illustrate this, three examples are presented: the first discusses key digitization requirements based on the RAMI4.0 layers; the second demonstrates how the AAS contributes to this digitization; and the third shows how agents can further enhance this process.

First, the digitization process of a drilling machine is analyzed as an example, focusing solely on the RAMI4.0 layers without integrating AAS or agent concepts. The objective is to first define the key requirements for digitization, independent of specific technologies or technical aspects. This approach enables a clear understanding of the expected outcomes of a RAMI4.0-compliant digitization process before assessing the role of AAS and the potential enhancements brought by agents toward implementing AAS Type 3. This example highlights fundamental aspects to consider when digitizing a drilling machine, particularly for monitoring and improving failure management from a high-level perspective.

Example 1

- Asset: The drilling machine is equipped with vibration and temperature sensors.
- Integration: Connects the physical drilling machine to the digital world, e.g., integrating data from sensors.
- Communication: Enables seamless data exchange between the drilling machine and other systems.
- Information: Represents drilling machine data (e.g., vibration and temperature) in a standardized format for further analysis in the upper layers, e.g., to predict failures.
- Functional: Represents the functions that the drilling machine can perform using data from the information layer to detect and manage failures.
- Business: Utilizes drilling machine data for decision-making related to preventive maintenance scheduling and resource allocation. This data is derived either directly from the information layer or through functions at the functional layer.

When applying AAS to this example and considering the main AAS specifications (Part 1 and Part 2) to date, the primary focus is on the information and communication layers. Part 1 defines a structured framework for representing asset information in standardized submodels, while Part 2 defines an API that allows seamless exchange of asset information. However, despite covering these layers, AAS does not fully address other critical aspects of digitization. The current specifications present certain gaps, particularly in the integration, functional, and business layers. The integration layer, for instance, can be addressed indirectly through communication protocols like OPC UA, MQTT, or HTTP, but these are not directly defined within the AAS. The functional layer is another area where AAS can describe asset functions through submodels but lacks execution capabilities. Similarly, while the business layer can be covered by storing relevant business-related data in submodels, AAS does not explicitly define how business processes should be handled or automated. Given these limitations, the current specifications align more closely with an AAS Type 2 solution rather than a fully AAS Type 3 implementation. The example below describes this process.

Example 2

- Asset: The drilling machine is equipped with vibration and temperature sensors.
- Integration: For instance, an adapter may be employed to gather data from the drilling machine, preprocess it, and store it in an AAS submodel.
- Communication: Communication is based on client-server schema (supported by AAS Part 2).
- Information: Data from the drilling machine, such as vibration and temperature, as well as other information, is stored in a standardized and structured manner using AAS submodels (supported by AAS Part 1).
- Functional: An external system, application, or service retrieves information from the submodels to detect and manage failures.
- Business: An external system, application, or service accesses information from the submodels and/or the output from the functional layer to perform decision-making in line with company business strategies.

The integration, functional, and business layers can theoretically be implemented using various approaches. However, it is crucial to select an approach that also offers the intelligence, autonomy, and collaboration capabilities necessary for AAS Type 3. In this context, agents are particularly well-suited to address these requirements due to their unique characteristics. Agents bring the essential capabilities of autonomy, decision-making, and collaboration, which are critical for enabling AAS to act as a self-managing, intelligent entity. The integration layer can be enhanced through industrial agents, which are typically designed to interface with assets to collect data or to adapt control. The functional

layer can be improved by incorporating agents that utilize AI techniques for tasks such as monitoring, diagnosis, optimization, and predictive analysis. Additionally, the business layer can benefit from agents' ability to make autonomous decisions, optimizing processes like resource allocation, predictive maintenance scheduling, and fault management in alignment with business strategies. Additionally, agents can enhance the communication layer by facilitating decentralized communication through strategies like collaboration and negotiation.

Example 3

- Asset: The drilling machine is equipped with vibration and temperature sensors.
- Integration: Industrial agents interface with assets (e.g., drilling machine) to collect data or to adapt control (supported by the IEEE 2660.1 standard).
- Communication: Data exchange follows a client-server schema (supported by AAS Part 2). Additionally, agents enable decentralized communication through interaction strategies like collaboration and negotiation (supported by FIPA specifications and VDI/VDE 2193 standards).
- Information: Data from the drilling machine, such as vibration and temperature, as well as other information, is stored in a standardized and structured manner using AAS submodels (supported by AAS Part 1).
- Functional: Agents can autonomously execute adaptive functions (e.g., based on AI techniques), dynamically detecting and managing failures.
- Business: Agents facilitate self-optimizing, decentralized decision-making to enhance operational efficiency in line with company business strategies.

Comparing examples 2 and 3, integrating MAS with AAS transforms digitization from a basic data management process to a decentralized, intelligent, and self-managing system capable of autonomous decision-making and enhanced business processes. This evolution opens new avenues for creating more resilient, adaptable, and efficient industrial systems. Figure 2.7 illustrates the difference between both approaches, highlighting example 3 as the goal to be achieved in this thesis by proposing a specification of an agent-based AAS approach for guiding the design and development of an AAS Type 3 solution.

The discussions and reflections presented in this section, along with the conclusions from the literature review on AAS and MAS, strongly support the initial hypothesis that an agent-based approach can effectively enable the design and development of an AAS Type 3 solution. After the review and reflection on AAS, MAS, and RAMI4.0, the key conclusions from this chapter can be highlighted as follows:

- AAS Type 3 as an emerging topic: AAS Type 3 remains an underexplored topic, with its full potential yet to be realized. One of the key challenges is the lack of an

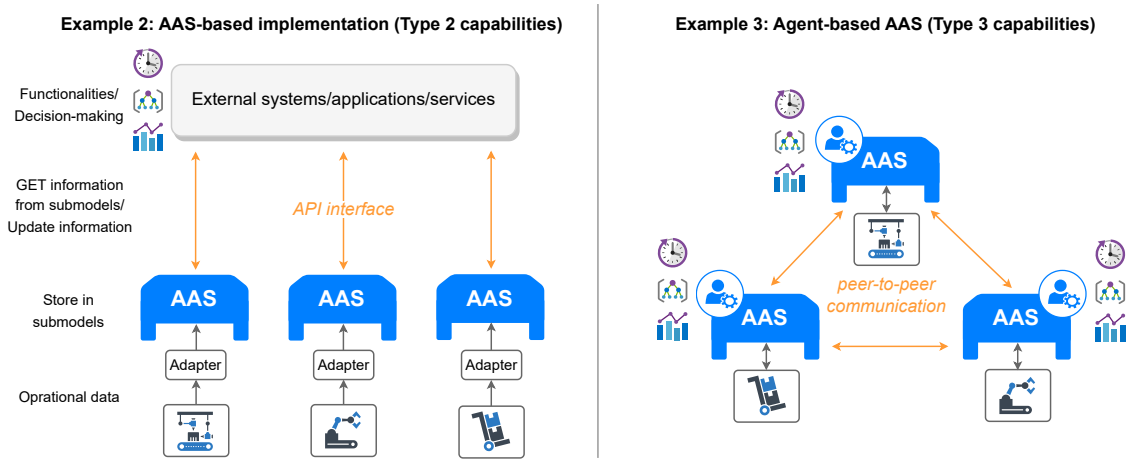

Figure 2.7: Comparison between AAS solutions.

official specification, which creates uncertainty in its design and implementation. However, this also presents an opportunity for innovative approaches to shape its development, with MAS emerging as a strong candidate for enabling its realization.

- Advancing industrial systems with MAS: Several research works and R&D initiatives have demonstrated the benefits of applying MAS in industrial environments, namely adaptability, robustness, scalability, reconfigurability, reusability, pluggability, responsiveness, and the ability to design complex, large-scale CPS-based systems. Additionally, the development of MAS is supported by international technical committees that aim to advance the application of industrial agents in real-world scenarios. These committees also promote standardization initiatives, promoting the widespread adoption of MAS-based solutions in industrial environments.

- Extending the AAS structure: MAS should complement and extend the official AAS structure by providing the necessary intelligence, autonomy, and communication capabilities. In the case of industrial agents, they also enable seamless interfacing with physical assets while complying with industrial requirements. A significant advantage of this approach is its modularity, as it avoids modifying the core AAS structure. Instead, it enhances AAS functionality by introducing supplementary components/modules that enable autonomous and intelligent behavior.

- Enhancing RAMI4.0 layers: The benefits of MAS are reflected across multiple layers of RAMI4.0, particularly the integration, communication, functional, and business layers. MAS facilitates the seamless integration of assets in I4.0, supports decentralized communication through strategies such as collaboration and negotiation, empowers assets to perform autonomous functions based on AI techniques, and improves decision-making processes.

These findings confirm that MAS is a viable and effective approach to addressing the challenges of AAS Type 3. The following chapters will focus on specifying and implementing the proposed approach, considering the conclusions and reflections made in this chapter.

2.6 Summary

This chapter reviewed existing literature and introduced key concepts to provide readers with the necessary theoretical background. The review formed the basis for achieving the main objective of this work: developing an agent-based approach for realizing AAS Type 3 using MAS. By exploring current advancements and highlighting gaps in the literature, this chapter established the context, underscored the significance, and revealed the potential impact of integrating MAS to enhance AAS capabilities.

Section 2.1 contextualized and positioned the basis of the work by exploring the concept of I4.0 and CPS. It aimed to establish a clear understanding of the vision and objectives of I4.0, focusing on how it drives the digital transformation of manufacturing and how CPS can contribute to realizing I4.0's vision. Section 2.2 outlined the role of RAMI 4.0 in providing a common understanding to the participants in developing I4.0-compliant solutions. Key points raised in these sections include: **(1)** despite the emergence of I5.0, this work remains positioned within the I4.0 context, as it is still an ongoing revolution and far from being fully realized; **(2)** CPS is a critical enabler for realizing the goals and vision of I4.0; **(3)** CPS requires a standardized and interoperable approach to ensure seamless integration and communication between cyber-physical components, e.g., through the I4.0 component, a CPS category aligned with the RAMI4.0 model.

Section 2.3 conducted a characterization and content analysis through a bibliographic search to examine the AAS, particularly concerning the design and implementation practices of AAS Type 3. This analysis offered a comprehensive overview of AAS solutions reported in the literature while focusing on identifying key enabling technologies, challenges, and research opportunities. From this review process, several insights were obtained: **(1)** although AAS is a relatively recent topic, it is rapidly gaining traction in both the scientific community and industry as a key enabler of interoperability in I4.0; **(2)** there is a strong connection between AAS and DT, which often leads to debates about whether AAS constitutes a DT or vice versa. From the author's perspective, such debates should be avoided, emphasizing instead the importance of clearly defining the roles and functionalities of proposed AAS or DT solutions to prevent misunderstandings; **(3)** AAS Type 3 remains an underexplored topic, but emerging research links it to MAS, which bring autonomy, intelligence, and collaboration to AAS solutions; **(4)** while the benefits of existing agent-based AAS solutions are evident, they need to be abstracted to a higher level to generalize beyond their initial applications. This may involve the development of a detailed specification to guide the design of AAS Type 3, ensuring that these systems are

sufficiently generic, independent of specific use cases, and capable of supporting various collaboration strategies.

Section 2.4 built upon the findings of the AAS bibliographic search, which identified MAS as a key technology for realizing AAS Type 3. MAS emerges as a critical enabler for enhancing the autonomy, intelligence, and collaborative capabilities of AAS solutions. Their ability to support decentralized decision-making and dynamic interactions aligns closely with the vision and requirements of AAS Type 3, making MAS a foundational approach for developing these components. From early holonic architectures to modern agent-based systems, MAS have evolved significantly, particularly with the advent of I4.0 and CPS. The MAS solutions reviewed in the literature and R&D initiatives share common features, namely adaptability, robustness, scalability, reconfigurability, reusability, pluggability, responsiveness, and the capacity to design complex, large-scale CPS-based systems. Additionally, MAS have been supported by international technical committees that aim to advance the application of industrial agents in real-world scenarios, e.g., through standardization initiatives, thereby promoting the widespread adoption of MAS-based solutions in industrial environments.

Section 2.5 highlighted the main conclusions obtained from the chapter, serving as a starting point for specifying the proposed approach. The goal was to provide a brief, conceptual, and non-technical overview to support the reader's understanding before moving on to more technical specifications in the subsequent chapter. In this context, this section adopted two complementary viewpoints to elucidate the relationships between MAS, AAS Type 3, and RAMI4.0. The first viewpoint compared agents and AAS Type 3, highlighting the synergy between the two concepts. The second viewpoint aligned more closely with RAMI4.0, emphasizing how agents could enhance the digitization process across the layers defined by RAMI4.0, which are directly related to AAS.

In summary, this chapter, through a literature review and the definition of key concepts, presented arguments highlighting AAS Type 3 as a significant research opportunity, recognizing it as a concept still in its early stages of maturity and lacking a detailed specification to guide its design. Furthermore, it identified MAS as a suitable and promising technology for realizing AAS Type 3, given their ability to enhance autonomy, intelligence, and collaboration in industrial environments. The following chapter will focus on specifying an agent-based AAS approach for guiding the design and development of an AAS Type 3 solution using MAS.

CHAPTER 3

An Agent-Based Approach to Designing Asset Administration Shells Type 3

LOOKING ahead, the future vision for AAS involves the development and standardization of AAS Type 3, which extends beyond the conventional functionalities of traditional AAS by incorporating more sophisticated features, enabling I4.0 components to operate with greater autonomy, intelligence, and collaborative capabilities. However, the progression towards AAS Type 3 is currently hindered by a lack of official documentation and limited research work. This gap presents significant challenges for developers and engineers in designing components that can deliver the advanced features expected of AAS Type 3, as the existing AAS Type 1 and 2 solutions do not address these features. In this context, as discussed in the previous chapters, and based on the assumption that MAS are the key technology for providing the required autonomy, intelligence, and collaborative capabilities to enhance the AAS, this chapter proposes an agent-based AAS approach for guiding the design and development of an AAS Type 3 solution using MAS technology, addressing both the lack of official documentation and the limited research focused on AAS Type 3 design.

The remainder of this chapter is structured as follows:

- Section 3.1 provides an overview of the proposed agent-based AAS approach and provides a technical perspective on the metamodel, which details the elements and their relationship for modeling agent-based AAS instances.
- Section 3.2 describes fundamental submodels that enable agent-based AAS to effectively manage and interact with its corresponding asset in the industrial environment, as well as to facilitate interaction and collaboration among agent-based AASs.

- Section 3.3 discusses the standardized interface mechanisms to enable the agent-based AAS to access and manage the information contained in the submodels.
- Section 3.4 presents the integration patterns for interfacing agent-based AASs with physical assets.
- Section 3.5 details the vocabulary and structure of the messages, along with examples of interaction patterns between agent-based AASs aligned with the I4.0 language and FIPA standards.
- Section 3.6 discusses some approaches to implementing intelligence in agent-based AAS, ranging from simple rule-based mechanisms to advanced AI techniques.
- Section 3.7 summarizes the key aspects discussed in this chapter.

3.1 Proposed Agent-Based AAS Approach

This section overviews the agent-based AAS approach and provides a technical perspective on the metamodel, which details the available elements for modeling agent-based AAS instances. The approach is designed to enhance the capabilities of AAS by integrating software agents that enable higher levels of autonomy, intelligence, and collaboration. Moreover, the metamodel serves as a blueprint, defining the elements and their relationships, offering flexibility for implementation while maintaining compatibility with the existing AAS specifications and industrial standards.

3.1.1 Overview

As shown in Figure 3.1, the agent-based AAS approach seamlessly integrates MAS into the existing AAS structure with the objective of implementing the AAS Type 3, enabling I4.0 components to operate with greater autonomy, intelligence, and collaboration. In this approach, MAS facilitates the creation of decentralized and highly adaptable autonomous systems through the interactions among multiple autonomous, intelligent, and cooperative agents, capable of making decisions, negotiating, sharing knowledge, and executing tasks with more flexibility and efficiency compared to traditional centralized systems. Each agent is designed with specific roles and responsibilities, and their capabilities can be further enhanced with data analysis techniques and AI algorithms, introducing new functionalities to I4.0 components, e.g., to support monitoring, diagnosis, prediction, and optimization tasks. At the same time, the AAS provides a standardized asset information model that serves as a knowledge representation or information source for the agents. With this standardized model, agents can access, exchange, and utilize asset-specific information in a structured and consistent manner, which is critical for ensuring that agents

have comprehensive knowledge about the assets they represent, supporting the decision-making process and collaborative tasks. Moreover, the modularity of the proposed approach allows reusing the implemented AAS Type 1 and 2 solutions, extending them to AAS Type 3 solutions through the integration with agents.

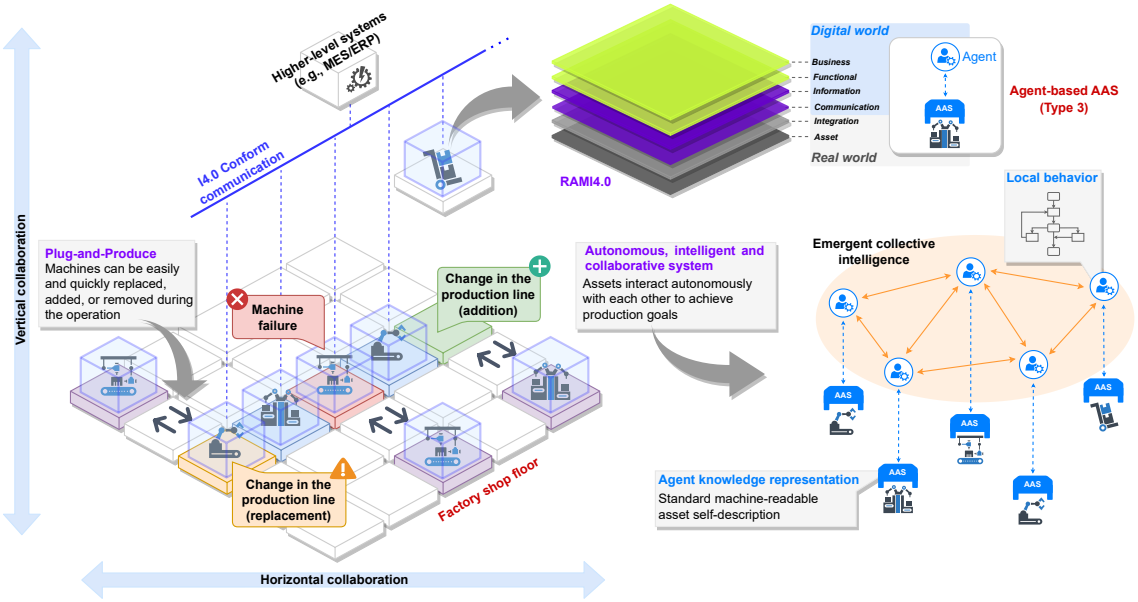

Figure 3.1: Overview of the agent-based AAS approach.

Additionally, by leveraging the communication capabilities of the agents, the agent-based AAS approach should enable both vertical and horizontal collaboration. Vertical collaboration is performed by components of different factory levels, e.g., shop floor and business levels. In this context, higher-level systems, e.g., MES and ERP, are able to exchange information with shop floor devices, such as products and resources (e.g., machines and production units), to plan and optimize production. On the other hand, horizontal collaboration is performed by components at the same factory level, typically between products and resources, which are able to exchange production-related information and autonomously adapt to dynamic production requirements. The versatility of horizontal and vertical collaboration allows the system to move between hierarchical and heterarchical configurations depending on the requirements and conditions of the system. For example, higher-level systems can optimize production orders and allocate tasks to shop floor devices, while shop floor devices can have the intelligence and autonomy to collaborate with each other and make decisions to respond quickly to unexpected events, e.g., downtimes due to machine failure.

3.1.2 Information Metamodel

By integrating software agents to develop an approach that realizes AAS Type 3, some key aspects should be considered. Firstly, the approach must maintain compatibility

with the official AAS metamodel, allowing novel classes or modules to be added without altering the core structure. This ensures adherence to AAS specifications and preserves interoperability with existing AAS Type 1 and 2 solutions, with these additions functioning as a higher-layer extension. Secondly, this higher-layer extension, represented by the agents, should not only handle intelligent and communication tasks but also interface directly with the asset, e.g., to enable the rapid response for continuous monitoring and control, interpret data contained within the AAS, and use this information to drive actions and optimizations. With these considerations in mind, Figure 3.2 illustrates the required elements and their relationships for modeling agent-based AAS metamodel instances.

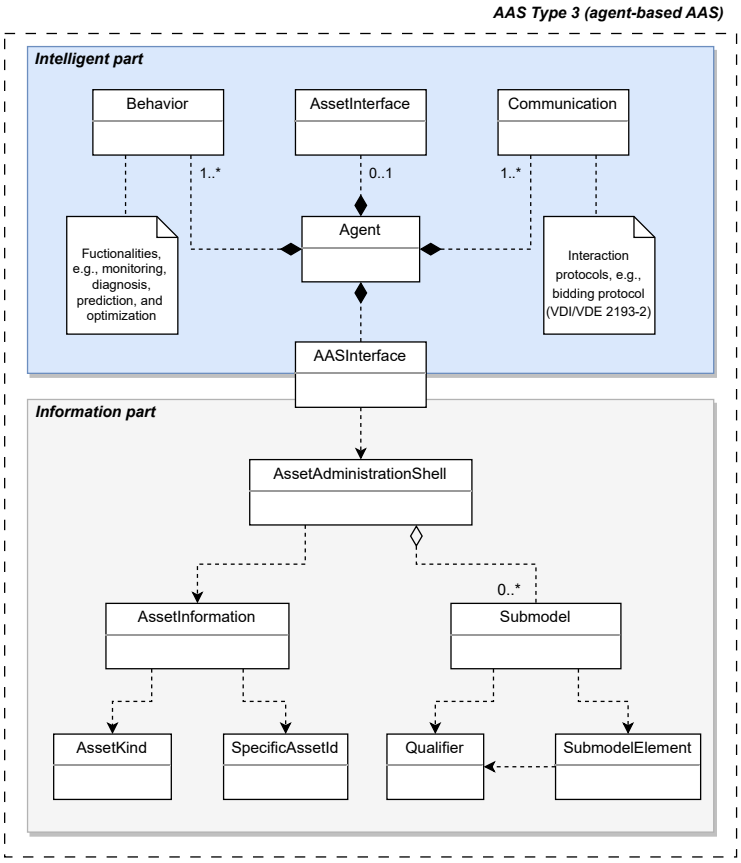

Figure 3.2: Agent-based AAS metamodel.

The metamodel consists of two major parts, namely information and intelligent parts. The information part presents the main classes to describe the asset information in the AAS submodels. The intelligent part presents the main classes to enhance the AAS with autonomy, intelligence, and collaborative capabilities provided by the agent. Regarding the information part, it will not be covered in this document since it follows the specifications defined in [27]. However, in summary, the *AssetAdministrationShell* is mainly composed of *Submodels* that describe the content or functional aspects of an asset in a standard manner. On the other hand, the intelligent part describes four classes that constitute an *Agent*, namely *Behavior*, *Communication*, *AASInterface*, and *AssetInterface*.

- The *Behavior* defines how the agent acts and makes decisions in various situations to achieve specific goals. Moreover, the behavior of an agent is fundamental as it encapsulates the intelligence and essential functionalities of the agent, such as monitoring, diagnosis, prediction, and optimization.
- The *Communication* encompasses the rules and interaction protocols that agents adhere to when interacting and collaborating with other agents, components, or systems, defining how agents exchange information, coordinate their actions, and participate in cooperative tasks.
- The *AASInterface* represents the agent's ability to access and retrieve information from the submodels, enabling the agent to build or improve its knowledge representation based on the information provided by these submodels. In this sense, the agent can better understand the system it operates and make more informed decisions.
- The *AssetInterface* allows the agent to interact with an asset, defining how the agent can request functions, access data, or set parameters on this asset. However, this interface is optional and only needed if the agent represents an asset that performs real-world functions. For example, if the agent represents a product that does not perform skills like drilling or assembling, the agent does not need to interface with the asset, e.g., to request those skills.

Based on this metamodel, the structure of an agent-based AAS can be set up in different ways. For example, the intelligent and information parts of an agent-based AAS can be decoupled and deployed along the Edge-Cloud layers, or if the asset has computational capabilities, the intelligent and information parts can be embedded directly within the asset. Independent of where the different parts are deployed, it is fundamental to define the proper interfaces that enable the interaction between the parts and with the asset.

3.2 Modelling the Information Part of an Agent-Based AAS

As previously discussed, the information part of the agent-based AAS holds all information about the asset organized in a structured manner using submodels. Nevertheless, not every submodel has value or relevant information for the decision-making process, being necessary to define the proper submodels to help in the dynamic behavior of the agent-based AAS, particularly to support collaborative tasks. For this purpose, this section aims to describe fundamental submodels that enable agent-based AAS to effectively manage and interact with its corresponding asset in the industrial environment, as well as to enable collaboration with each other.

These submodels are mainly based on the concepts of capabilities and skills from the capability- and skill-based engineering domain and are formally specified in the Capability-

Skill-Service (CSS) reference model [89]. As illustrated in Figure 3.3, the CSS model builds upon and extends the widely adopted Product-Process-Resource (PPR) model [89, 90]. It offers a structured framework to define and categorize the concepts of capability, skill, and service, facilitating a clear distinction between them and clarifying their interrelationships.

- Capability: An abstract description of a function or operation that an asset can perform, which results in tangible real-world effects, such as drilling. This description includes specific properties and constraints that define the scope and limitations of the operation. For example, a drilling capability might be specified as "drilling with a depth range of 20 cm and a diameter of 15 mm into certain types of metals".
- Skill: The concrete implementation of a function provided by an asset, as defined by a capability. A skill enables the practical execution of a capability and typically involves parameters, including inputs and outputs, that allow for the control and monitoring of the skill's execution. Each skill must be accessible through at least one interface, which provides interaction with the skill's state machine, as well as its parameters.
- Service: Specifies how a service provider can deliver one or more capabilities to a service requester. A service outlines the mechanisms for requesting and receiving the execution of specific capabilities, enabling assets to offer their capabilities to other components within the system.

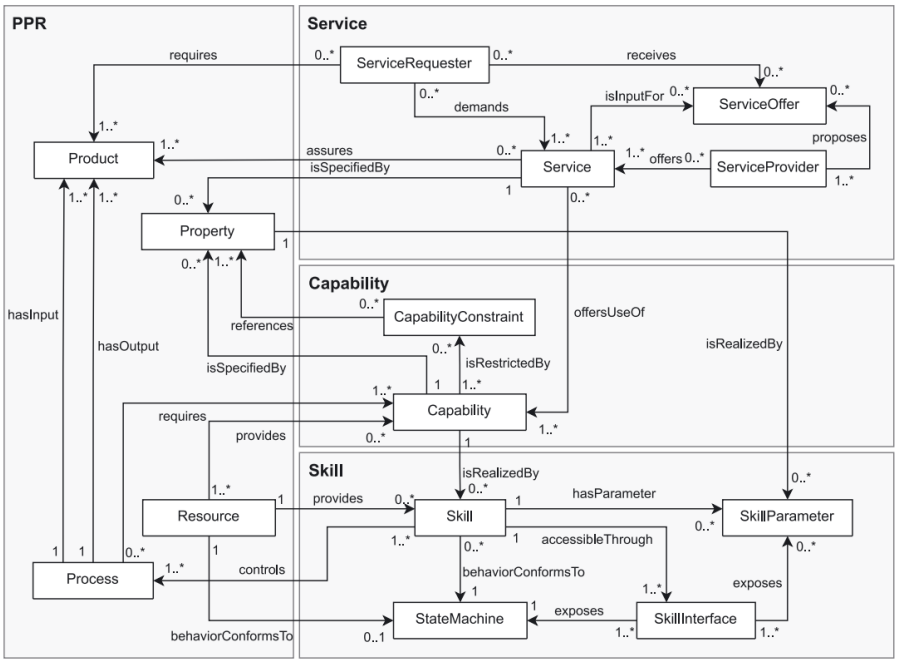

Figure 3.3: Capability-Skill-Service reference model [89].

Figure 3.4 provides an illustrative example of how the concepts of capability, skill, and service from the CSS model can be applied to the agent-based AAS approach, enabling collaboration in industrial scenarios. In this context, information about capabilities and

skills is organized into submodels, which the agent-based AASs utilize to manage their respective assets. As a result, agent-based AASs representing resources possess knowledge of the capabilities their assets can provide and the skills to execute these capabilities. Meanwhile, agent-based AASs representing products may have a work plan structured in a submodel that outlines the steps and requirements for production based on the necessary capabilities. This knowledge allows products and resources to collaborate seamlessly to complete the production process. In such scenarios, for example, resources may act as service providers by offering their capabilities as services, while products act as service requesters, seeking services aligned with their production requirements. The capabilities of resources are tested against product requirements to ensure that the functions offered by the resources match the capabilities needed to fulfill the product's production plan.

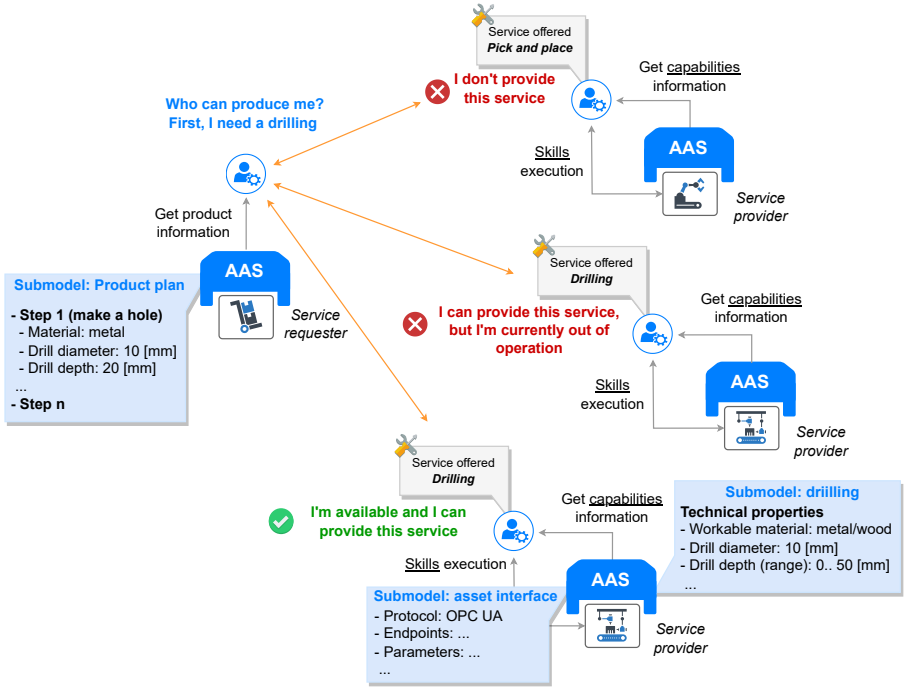

Figure 3.4: Mapping of capabilities, skills, and services to an agent-based AAS.

Based on that, the following subsections describe some submodels based on the concepts of capabilities and skills. This work employs official submodel templates and specifications provided by the IDTA to ensure greater interoperability. Nevertheless, there are several submodels templates that are currently under development or analysis for publication (see the benefits of using submodel templates in Section 2.3.1). Therefore, in specific cases, this work develops custom submodels that adhere to the AAS specification to fulfill the necessary requirements.

3.2.1 Capabilities: Description of Automation Functions

The CSS model defines capability as an abstract description of a function/operation performed by an asset with effects in the real world, e.g., drilling and laser cutting. Designing manufacturing systems based on the required capability for each step of the production process offers more agility and adaptability to the system. For example, this approach enables the easier and quicker adaptation in scenarios like production replanning due to machine breakdowns, where having detailed capability descriptions can simplify the task of finding alternative machines with similar capabilities. On the other hand, in lot-size-one scenarios, the capability description allows for the identification of whether it is possible to manufacture a new product by evaluating which assets are capable of offering the required capabilities. In this context, it is fundamental to describe capabilities in a standard manner.

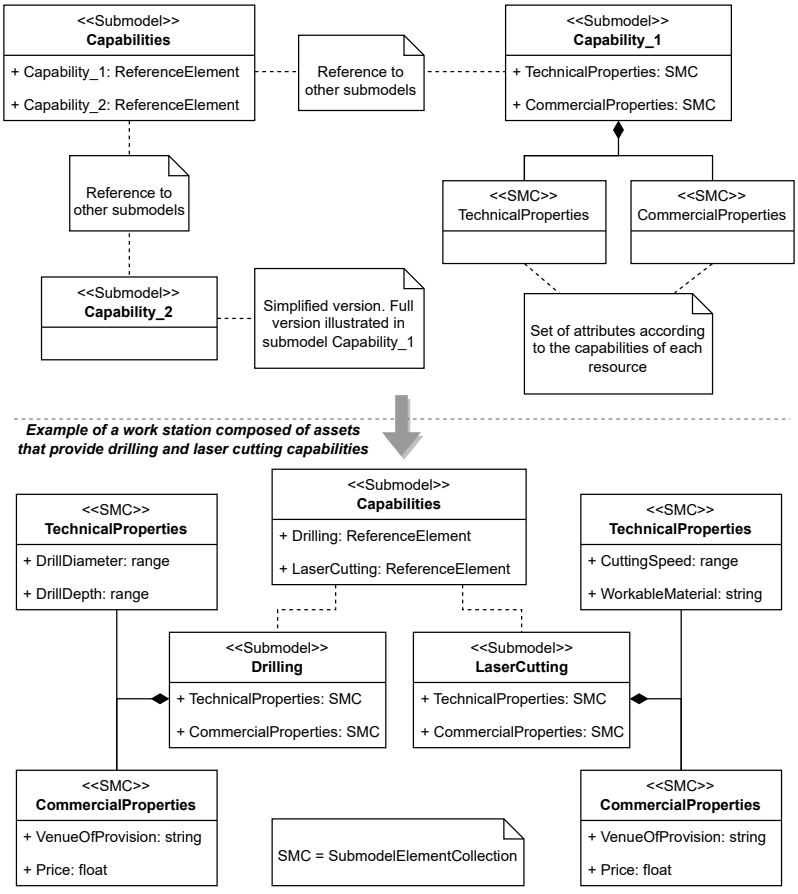

Figure 3.5: Capabilities submodel template.

Currently, there is a submodel under development to express the capability of a production resource [91]. Since this submodel has not yet been officially released, this work considers modeling asset capabilities based on the submodel developed in [54], illustrated in Figure 3.5, which can be composed of a set of capabilities. For each capability, a specific submodel is instantiated and referenced, which consists of two *SubmodelElementCollection*

classes, namely *TechnicalProperties* and *CommercialProperties*. These classes are customized with different properties according to the capabilities of each asset. *TechnicalProperties* refer to the description of the offered capability, for instance, a drilling machine can present the following properties: drilling diameter and depth. *CommercialProperties* refer to cost-related parameters, such as venue of provision and price. In this context, ECLASS [92] and IEC CDD [93], two dictionaries of standardized semantics based on the IEC 61360 [94] standard, are recommended to describe these properties in an unambiguous, machine-readable, and industry-independent manner.

3.2.2 Skills: Interaction with Automation Functions

In manufacturing systems, physical assets are often connected to low-level controllers, such as programmable logic controllers, which follow the IEC 61131 [95] and IEC 61499 [96] standards to ensure real-time responsiveness and effective execution of the skills of their connected assets. According to the CSS model, a skill is the implementation of an encapsulated function/operation provided by a specific asset, i.e., a skill enables the realization of a capability. In this context, to facilitate the interaction with these assets, their skills should be exposed through standardized interfaces. In this sense, the description of these interfaces should adhere to a standard format. This standardization is fundamental for ensuring seamless communication and integration across different components or systems in the manufacturing environment.

Based on that, IDTA provides standardized submodels that describe interaction patterns to facilitate the utilization of skills through standardized interfaces, namely *ControlComponentType* [97], *ControlComponentInstance* [98], and *AssetInterfacesDescription* [29]. The first two are specific to the type of asset, whether it is a "type" or "instance". Since this work considers assets as instances, the *ControlComponentInstance* submodel, complemented with the *AssetInterfacesDescription* submodel, is considered to model the asset skills.

The *ControlComponentInstance* submodel (see Figure 3.6) comprises two main *SubmodelElementCollection* classes, namely *Skills* and *Endpoints*. *Skills* refer to the collection of skills offered by the component, including information such as the parameters used to configure the skill, a reference to the error codes associated with the component that may be triggered by that skill, the designation of the operation in which the skill can be executed (e.g., manual, semi-automatic, and automatic), and references to other skills that are used by that particular skill. On the other hand, *Endpoints* serve as references to other submodels that describe how to interface with an asset based on communication protocols. In this sense, the *AssetInterfacesDescription* submodel can be used for this purpose since it specifies an information model for describing the interfaces of an asset service or asset-related services.

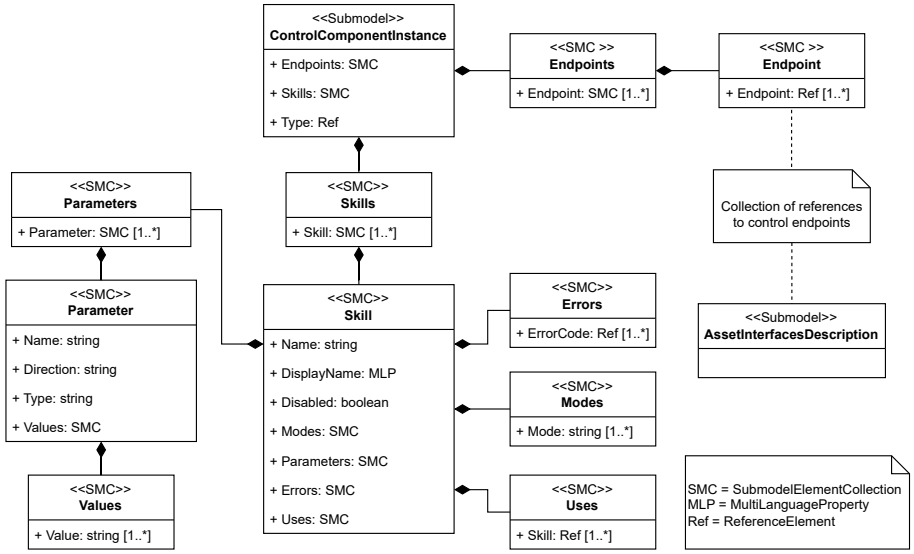

Figure 3.6: Control component instance submodel template (adapted from [98]).

The *AssetInterfacesDescription* submodel is based on the Thing Description (TD) [99] standard to define the content and structure of the submodel, describing in a standardized manner how to interface with assets that exhibit different protocols and payload formats, particularly using Modbus, HTTP, and MQTT. Since it is an extensive submodel with different schemas to support specific protocols, this work will not go into detail about all aspects of this submodel, describing only two fundamental *SubmodelElements*, namely *base* and *href* properties. The *base* property is the entry point for the asset connection, which is specified according to the protocol, e.g., the IP address and port of the asset. The *href* provides the description of asset endpoints. For instance, if the asset provides some service based on the HTTP protocol, the combination of *base* and *href* properties, e.g., `http://{address}:{port}/{endpoint}`, describes how to execute some service of the asset. For a complete view of the submodel, refer to Figure 2 on page 12 of the reference [29].

3.2.3 Product Requirements and Manufacturing Plan

Based on the described submodels, a product should have a submodel to describe the required capabilities involved in its manufacturing process. In this context, it can be evaluated which assets are able to offer the required capabilities and skills, resulting in the product manufacturing plan. The product manufacturing plan can be described in a submodel structure, including the manufacturing steps, schedule, state, and the assigned resources to provide the capability in each step.

Since there are no official submodels related to product manufacturing, two submodels are designed to deal with these aspects. Figure 3.7 (left) illustrates the designed *ProductRequirements* submodel, which contains the required capabilities to manufacture a product. On the other hand, Figure 3.7 (right) shows the designed *ProductManufacturingPlan* submodel,

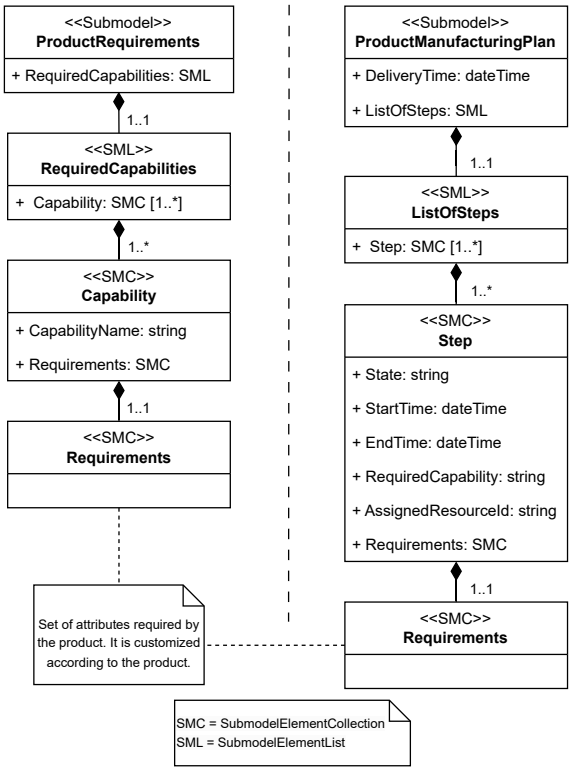

Figure 3.7: Submodel templates related to product manufacturing.

which contains a *ListOfSteps* that describes one or more steps for producing a product. Each step results in a sub-product that becomes the final product when its manufacturing plan is complete. In this context, each *Step* describes the information regarding the state, the scheduled start and end time, the required operation, the assigned resource to perform the operation, and a collection of requirements to perform the operation. The *Requirements* from both submodels depend on the type of product, so it is not possible to define a generic model since a product may require a drilling capability with the appropriate drilling requirements, such as diameter and depth, or a laser cutting capability with requirements such as cutting speed.

3.2.4 Complementary Submodels

It is important to highlight that the proposed approach does not limit itself only to the previously described submodels. However, they provide the basic information to enable the collaborative tasks of agent-based AASs. Additional submodels can be introduced to describe specific functionalities performed by the agent-based AAS, such as monitoring, optimization, and diagnosis, which may rely on AI techniques. To support these functionalities, the adoption of AI-specific submodels, namely *AI Dataset* [100], *AI Model-Nameplate* [101], and *AI Deployment* [102] could be adopted.

The main idea of the referred AI-related submodels is to cover the AI lifecycle in a standardized manner, ensuring that every stage, from dataset preparation to model deployment, is systematically documented. Such standardization not only enhances the accessibility, reproducibility, and reusability of AI models but also provides greater transparency for stakeholders regarding the implemented functionalities and the decision-making criteria of agent-based AASs, building trust in AI-based systems by providing clear insights into how models are trained, deployed, and used.

Figure 3.8 illustrates a simplified version of these submodels. The aim of the *AI Dataset* submodel is to identify and explain the dataset that is used to train and thus instantiate an AI model. On the other hand, the *AI ModelNameplate* submodel contains information about the model training process and the trained model itself. Finally, the *AI Deployment* submodel describes all relevant information for deploying trained AI models. For a detailed information about these submodels, see [100–102].

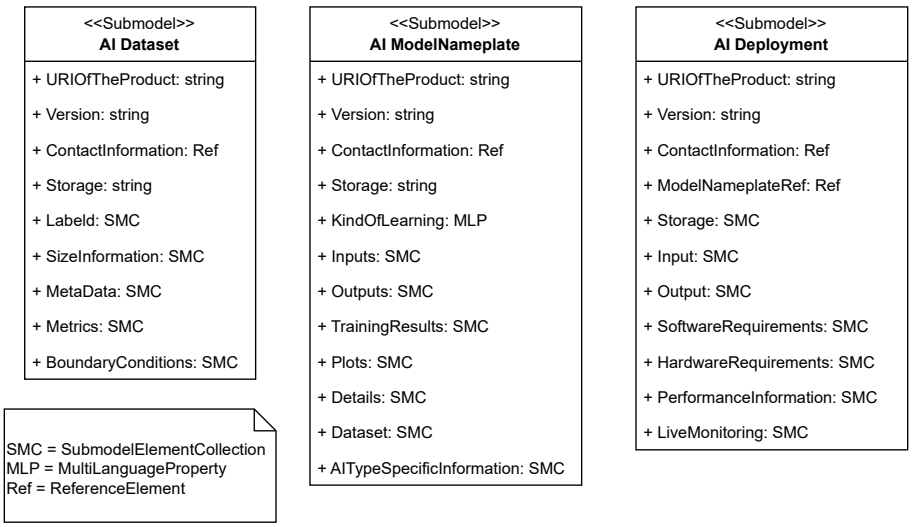

Figure 3.8: AI-specific submodel templates (adapted from [100–102]).

Despite the benefits of accessibility, reproducibility, and reusability of AI models that these submodels can offer, the main focus during the operational phase of the agent-based AAS is on the *AI Deployment* submodel. Besides the other features, this submodel is particularly relevant since it stores the AI model directly or provides a file path for quick access. This AI model should be described using standardized, machine-readable interchange format, such as Predictive Model Markup Language (PMML) [103], Portable Format for Analytics (PFA) [104], and Open Neural Network Exchange (ONNX) [105], which enables the description of AI models in a structured and interoperable manner. In this way, the agent-based AAS only needs to consume the AI model using a suitable scoring engine. A scoring engine is a specialized software component or algorithm that takes input data and processes it through the AI model to produce, e.g., predictions or inferences. The scoring engine is able to interpret the specific structure of an AI model (which can be described using PMML, PFA, or ONNX) and applies the model to input data in real time.

Moreover, the scoring engine does not require the agent to be aware of the details of how the AI model is implemented. It only needs to pass the input data and receive the output, making the process transparent to the agent. A more detailed discussion of this process will be made later since the focus here is only on the description of submodels.

In this context, the referred generic AI-specific submodels designed to cover the AI lifecycle, from model training to deployment, are highly flexible and can be readily adapted to store not only PMML, PFA, or ONNX but also other or possible future standardized, machine-readable interchange formats for AI models. This approach would provide a standardized solution for managing diverse AI models in manufacturing systems, particularly enhancing the agent-based AAS approach to accommodate a wide range of AI-driven functionalities.

Notably, using PMML, PFA, ONNX, or other standardized formats is highly recommended but not mandatory. These formats may bring several benefits, such as representing AI models in a standardized manner, enhancing interoperability, facilitating seamless integration across diverse systems, and reducing the time required to make a model production-ready. However, other approaches for representing and exchanging AI models can also be adopted depending on specific requirements or constraints, such as unique application needs and legacy system compatibility.

This section, "Complementary Submodels", provided an overview of additional submodels that can be considered when designing an agent-based AAS. While these submodels are not essential for supporting the collaborative tasks of an agent-based AAS, they can enhance the system's functionality. For example, another submodel that can be considered is the *Time Series Data* submodel [106], which allows for creating an interoperable description of time series data in industrial automation across the entire asset lifecycle. In this context, this submodel can define endpoints or queries that enable the agent-based AAS to access time series data of an asset, as well as specify how to store new data, e.g., in a database.

3.3 Interface for Access to Asset Information

Modeling the information part of an agent-based AAS is an important requirement to enable effective management and interaction with its corresponding asset. However, a crucial aspect is how the agent, the intelligent component of an agent-based AAS, can access this information. In this context, the AAS specification part 2 (version 3.0) [28] defines APIs for enabling access to the asset information that is contained in submodels, which can be specified in different technologies, e.g., HTTP/REST, MQTT, and OPC UA.

At this moment, the AAS specification part 2 provides only guidelines for a HTTP/REST API, particularly describing the operations to enable access to the submodel information.

From these operations, different HTTP/REST paths result according to the deployment of the AAS (information part of the agent-based AAS). For instance, an AAS can be deployed on a device, or several AASs can be deployed in an AAS repository that allows for the management of these AASs. Table 3.1 presents some examples of the operations that can be used in an AAS repository to manage several AASs following the AAS specification part 2. In this sense, it is possible to see that the information from the submodels can be easily obtained through the GET method, as well as the possibility of editing or updating this information through the POST, PUT, and DELETE methods. Other REST paths for the AAS repository service specification and the use case of a standalone AAS can be found in [107, 108], respectively.

Table 3.1: AAS repository API.

REST path	Description
GET /shells/{aasIdentifier}/submodels/{submodelIdentifier}	Returns the submodel
GET /shells/{aasIdentifier}/submodels/{submodelIdentifier}/ submodel-elements/{idShortPath}	Returns a specific submodel element from the Submodel at a specified path
PUT /shells/{aasIdentifier}/submodels/{submodelIdentifier}	Updates the submodel
PUT /shells/{aasIdentifier}/submodels/{submodelIdentifier}/ submodel-elements/{idShortPath}	Updates an existing submodel element at a specified path within submodel elements hierarchy
POST /shells/{aasIdentifier}/submodels/{submodelIdentifier}/ submodel-elements	Creates a new submodel element
POST /shells/{aasIdentifier}/submodels/{submodelIdentifier}/ submodel-elements/{idShortPath}	Creates a new submodel element at a specified path within submodel elements hierarchy
DELETE /shells/{aasIdentifier}/submodels/{submodelIdentifier}	Deletes the submodel from the AAS
DELETE /shells/{aasIdentifier}/submodels/{submodelIdentifier}/ submodel-elements/{idShortPath}	Deletes a submodel element at a specified path within the submodel elements hierarchy

Path parameters
- {aasIdentifier}: AAS unique id (UTF8-BASE64-URL-encoded);
- {submodelIdentifier}: The Submodel's unique id (UTF8-BASE64-URL-encoded);
- {idShortPath}: IdShort path to the submodel element.

This specification makes it possible to integrate the intelligent part (agent) and the information part (submodels) using an API-enabled approach, particularly using HTTP/REST, as shown in Figure 3.9 (top). Other approaches can also be considered, e.g., "OPC 30270: Industry 4.0 Asset Administration Shell" [109] provides guidelines for mapping the AAS metamodel to the OPC UA information model in order to represent AAS and their submodels in the address space of an OPC UA server, which enables OPC UA clients to request the submodels information from the server. In this context, the information part and the intelligent part of the agent-based AAS approach can also be integrated using OPC UA. On the one hand, an OPC UA server can host the information part of the agent-based AAS approach. On the other hand, the intelligent part represented by the agent can be an OPC UA client that requests information from the server, as shown in Figure 3.9 (bottom). It is important to highlight that the OPC 30270 specification is based on the old

version of the AAS information metamodel. Therefore, some adaptation may be required, considering the current version of the AAS metamodel (version 3.0).

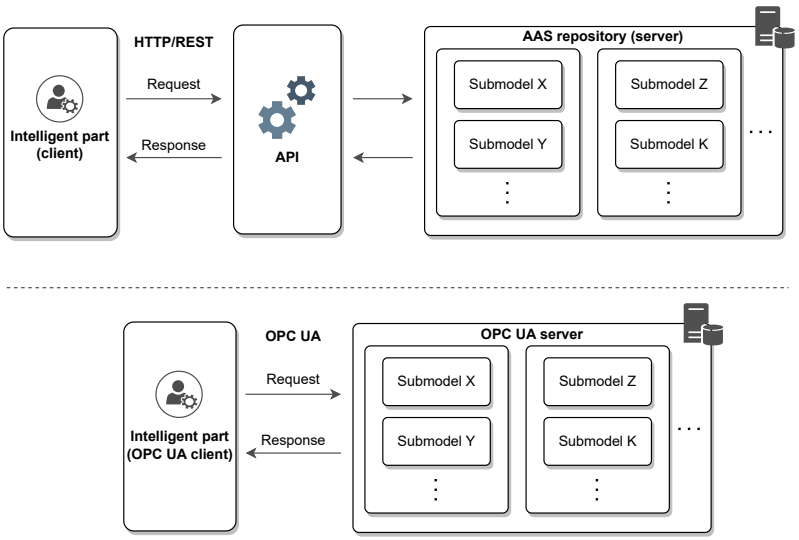

Figure 3.9: Integration between the intelligent and information part.

To illustrate the exchange of information using both approaches, consider the classic example of the AAS servo DC motor, illustrated in Figure 3.10, which contains four submodels, namely *Identification*, *TechnicalData*, *OperationalData*, and *Documentation*.

Figure 3.10: Example of the AAS servo DC motor.

Based on the scheme shown in Figure 3.9 (top) and the paths described in Table 3.1, the intelligent part can request the information of the *OperationalData* submodel, e.g., using the GET method {`serverAddress`}`/shells/`{`aasIdentifier`}`/submodels/`{`submodelIdentifier`}, where {`aasIdentifier`} and {`submodelIdentifier`} are specific information of the unique id of the AAS and the unique id of the *OperationalData* submodel, respectively. An example of the requested submodel is provided in the JSON structure, as shown in the following:

```
{
  "idShort": "OperationalData",
  "id": "http://i40.customer.com/instance/1/1/AC69B1CB44F07935",
  "kind": "Instance",
  "semanticId": {
    "type": "ExternalReference",
    "keys": [
      {
        "type": "GlobalReference",
        "value": "0173-1#01-AFZ615#016"
      }
    ]
  },
  "submodelElements": [
    {
      "category": "VARIABLE",
      "idShort": "RotationSpeed",
      "semanticId": {
        "type": "ExternalReference",
        "keys": [
          {
            "type": "ConceptDescription",
            "value": "http://customer.com/cd//1/1/18EBD56F6B43D895"
          }
        ]
      },
      "valueType": "xs:integer",
      "value": "4370",
      "modelType": "Property"
    },
    {
      "category": "VARIABLE",
      "idShort": "Torque",
      "semanticId": {
        "type": "ExternalReference",
        "keys": [
          {
            "type": "ConceptDescription",
            "value": "http://customer.com/cd//1/1/18EBD56F6B43D896"
          }
        ]
      },
      "valueType": "xs:float",
      "value": "117.4",
      "modelType": "Property"
    }
  ],
  "modelType": "Submodel"
}
```

On the other hand, based on the scheme shown in Figure 3.9 (bottom), the intelligent part can request the information of the *OperationalData* submodel by using OPC UA. Figure 3.11 illustrates an example of the AAS servo DC motor mapped in the OPC UA information model and hosted in an OPC UA server, and an OPC UA client that can request information contained in node addresses, e.g., the node "ns=3;i=416", which is

the *OperationalData* submodel represented in the OPC UA information model. Unlike the previous approach, where the response follows a JSON structure, in this case, the format of the response will depend on the OPC UA library adopted.

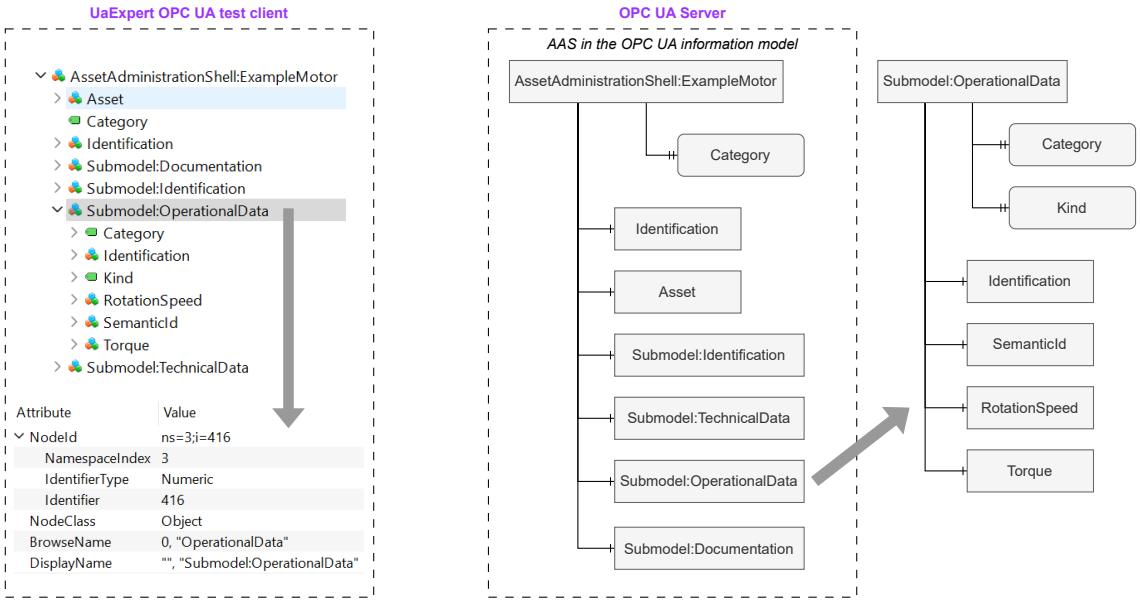

Figure 3.11: Example of AAS mapped in the OPC UA information model.

3.4 Integration Patterns for Interfacing Agent-Based AASs with Physical Assets

Integration with the asset is another important aspect of the agent-based AAS approach. This is made possible by the intelligent part since the information part is made up of submodels that follow the AAS information model but are not meant to do this kind of integration. Instead, they only describe how to do it, for example through the *AssetInterfacesDescription* submodel (see Section 3.2.2).

The intelligent part represented by the agent is suitable for realizing I4.0 components by interfacing with automation and control devices. As an example, consider an agent associated with a drilling machine that requires interaction with a PLC controlling the automation system to write or read variables. In this context, this approach goes beyond common I4.0 component solutions, where usually the AAS and asset do not present this level of integration, i.e., the AAS only stores the asset information along its lifecycle, and it does not perform any interaction with the asset, e.g., to adapt its control.

In order to achieve interoperability and facilitate this integration, it is fundamental to have an easy, transparent, and reusable way of interconnecting agents with different assets. In this context, the IEEE 2660.1 standard [65] presents generic interface practices based on interaction mode and location levels. These interface practices can be used in the context of

this work, particularly to show the different possibilities and guide the selection of the best interface practice to integrate agent-based AASs with assets, depending on the use case.

Figure 3.12 illustrates the integration patterns outlined in the IEEE 2660.1 standard. All these patterns are applicable to the proposed approach and should be considered. The standard defines and compares four integration patterns based on two key factors: the interaction mode between the agent and the asset and the agent's deployment location.

The interaction mode refers to the way the agent interacts with the asset:

- Tightly coupled: This interaction mode is characterized by a direct, permanent, and non-mediated connection between the agent and asset. The participants are connected via direct network communication or shared memory and follow a traditional request-response schema. This schema typically involves the agent sending a request to the asset, which processes the request and sends back a response.
- Loosely coupled: This interaction mode is characterized by an indirect connection between the agent and asset, usually mediated by one or more message brokers, following a publish-subscribe schema. Messages are sent asynchronously, meaning that the agent does not have to wait for an immediate response from the asset.

Regarding the deployment location, two possibilities are identified:

- Hybrid: The agent operates on a separate computational platform from the asset controller.
- On-device: The agent operates on the same computational platform as the asset controller.

In industrial settings, any proposed solution has to be integrated with the already existing and deployed systems. This scenario implies the preservation of the physical infrastructure, while software changes are permitted. These physical constraints, in a certain way, may determine the integration pattern to be adopted. For instance, if the existing device cannot accommodate an agent, the hybrid form is the only solution. Additionally, the communication protocol that the asset supports most of the time determines the interaction mode. For example, an asset where the interface is based solely on the Modbus protocol, the client-server scheme (tightly coupled) is the only way to establish this interaction. In this sense, it is important to notice that the selection of one of the four integration patterns does not depend only on the application domain but also on the existing system constraints. However, some systems may present fewer constraints and more flexibility, allowing the selection of more than one integration pattern for the same use case. In such circumstances, it is important to take into account the advantages and disadvantages of each integration pattern, and the recommendation method [65, 111] offered by the IEEE 2660.1 standard can help with this decision.

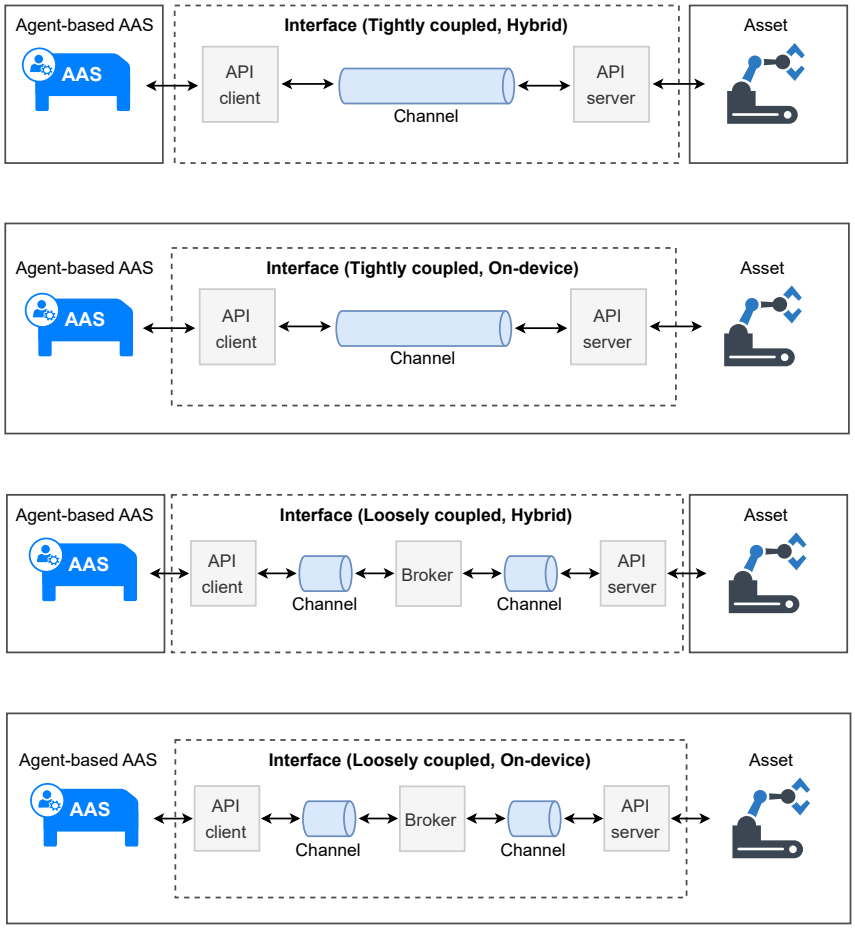

Figure 3.12: IEEE 2660.1-2020 generic integration patterns (adapted from [110]).

The IEEE 2660.1 recommendation method recommends the best interface practices to interconnect agents and physical assets, particularly low-level automation devices, based on user-selected criteria. These criteria include functions (e.g., control, monitoring, or simulation), application domains (e.g., factory automation, building automation, or energy and power systems), technological constraints (e.g., the ability to host agents in the asset controller), as well as the relevant characteristics for the application scenario (e.g., response time, scalability, and reusability). Based on these criteria defined by the user, the method uses a recommendation engine that evaluates existing interface practices stored in a repository, which is continuously updated with interface practices collected from several experts who have implemented and used different interface practices. Subsequently, the recommendation engine generates an output of recommended interface practices.

To illustrate the application of the IEEE 2660.1 recommendation method, consider the digitization of an inspection gauge placed in a production line performing inspection tasks to detect deviations in product production. In this context, the application domain is "factory automation", the asset cannot host agents locally, and the functionality provided by the agent is the "control" of the asset, e.g., to adjust its measurement parameters that need to be changed according to the specifications of the product. In such configuration,

"response time" and "reusability" are the most important characteristics to be considered (weighted as response time 80% and reusability 20%). The results provided by the IEEE 2660.1 recommendation method are illustrated in Figure 3.13.

Results

Recomended interface practice

Id practice	Location	Interaction mode	API client	Channel	Broker	Score
HT-7	Hybrid	Tightly coupled	Java	OPC UA		3.20

Details

Id practice	Location	Interaction mode	API client	Channel	Broker	Score
HT-7	Hybrid	Tightly coupled	Java	OPC UA		3.20
HT-5	Hybrid	Tightly coupled	Apache Milo	OPC-UA		3.20
HT-6	Hybrid	Tightly coupled	Java	Ethernet/IP		2.56
HT-3	Hybrid	Tightly coupled	Java	HTTP		2.56
HT-2	Hybrid	Tightly coupled	Java	Sockets		1.82
HT-4	Hybrid	Tightly coupled	C/C++/SQL	Sockets		1.63
HL-1	Hybrid	Loosely coupled	Apache Paho	MQTT	Eclipse Mosquitto	1.63
HT-1	Hybrid	Tightly coupled	Java	Modbus		1.56
HL-2	Hybrid	Loosely coupled	Java	OPC-UA		1.45
HL-3	Hybrid	Loosely coupled	C/C++/SQL	Sockets	DBMS	1.15
HL-4	Hybrid	Loosely coupled	Apache Paho	MQTT		0.41

Figure 3.13: Results of the recommended interface practices for the inspection gauge.

The recommended practice ("HT-7") involves implementing a tightly coupled interface utilizing the OPC UA protocol. Due to hardware constraints preventing the hosting of agents on the asset, the suggested strategy relies on a hybrid location setup. This implies that the client operates remotely, potentially leading to response time being influenced by the communication infrastructure. If the asset is not capable of supporting the OPC UA protocol, other alternative interface practices can be considered, e.g.,"HT-6" and "HT-3", which considers Ethernet/IP and HTTP protocols, respectively.

3.5 Communication Between Agent-Based AASs

Section 3.2 focused on describing some fundamental submodels for the agent-based AAS, particularly providing the required information about the asset related to capabilities and skills to support the management and interaction with the corresponding asset and the realization of collaborative tasks. Based on that, this section aims to present the vocabulary and structure of the messages and some examples of interaction patterns between agent-based AASs.

This work does not impose restrictions on communication protocols, such as HTTP, OPC UA, or MQTT, needed to exchange information between agent-based AASs. These protocols should be defined on a case-by-case basis according to the system's requirements. However, it is important to note that even if the AASs follow the same vocabulary and structure of the messages, seamless communication is not guaranteed if they do not support the same communication protocol. This work does not go into detail to address this

challenge but a middleware for the application layer interoperability can be a suitable solution, as proposed in [112]. For the sake of simplicity, agent-based AASs are implemented to support the same communication protocols in the experimental part of this work, which are message-based interactions provided by the framework used to develop the solution.

Despite the fact that the proposed approach is agent-based, it must be abstracted and seen as an AAS Type 3 solution. For this purpose, the agent-based AAS should be able to interact with other AAS-based solutions by exchanging messages using a common language and protocol. In this context, the I4.0 language (VDI/VDE 2193 standards) is recommended to support the interactions between AAS Type 3 solutions and serve as the foundation for I4.0-compliant interactions. As illustrated in Figure 3.14, these standards include the vocabulary of the language and the structure of the messages (VDI/VDE 2193-1) [39], as well as interaction protocols (VDI/VDE 2193-2) [40].

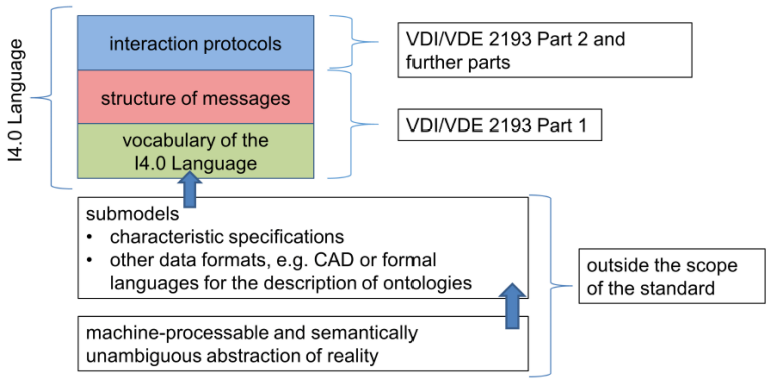

Figure 3.14: Structure of the VDI/VDE 2193 standards [39].

Although the I4.0 language is suitable for providing I4.0-compliant interactions, at this moment, the VDI/VDE 2193-2 standard specifies only the Bidding protocol, which restricts solutions that require a greater variety of interaction patterns. In this sense, besides the I4.0 language standards, the proposed agent-based AAS can also reuse the interaction protocols specified in the FIPA Agent Communication standards and adapt them when necessary to be I4.0-compliant. The focus is to provide an adaptable and standardized solution that ensures versatility for different collaborative scenarios.

3.5.1 Vocabulary and Structure of the Messages

The vocabulary and structure of the messages are based on the VDI/VDE 2193-1 standard, which is adapted from FIPA standards, particularly the FIPA Agent Communication Language (ACL) Message Structure Specification [113] and FIPA Communicative Act Library Specification [114].

The vocabulary of the message is related to the semantic aspects of the information exchange. In the I4.0 context, this information can be contained in individual AAS submodels and should be exchanged inside but also beyond company boundaries. Despite the fact that these submodels provide a standardized structure that follows the AAS information metamodel, their content can present incoherence about their correct meaning. For this purpose, in this information exchange process, all the communication partners need to share a unique vocabulary. This vocabulary should be based on property-based machine-readable descriptions, i.e., the attributes of an asset in the physical world that denote its characteristics. In this context, as mentioned before, ECLASS and IEC CDD are standardized dictionaries that enable the description of properties in an unambiguous, machine-readable, and industry-independent manner. Although the use of such dictionaries is fundamental in industrial applications, this topic is out of the scope of this work, which will not concern the semantic aspects but only the structure of the messages and interaction patterns of the agent-based AAS approach, assuming that the content of the messages always expresses their correct meaning.

Regarding the structure of a message, it consists of two parts, namely *interactionElements* and *frame*. The *interactionElements* includes the content of a message, e.g., the submodels or other data elements. While the *frame* identifies the message type, the interaction protocol used, the sender and receiver of a message, and other attributes related to conversation identification. Table 3.2 (part indicated by "VDI/VDE 2193-1") summarizes this structure.

Table 3.2: Message structure (adapted from [39]).

VDI/VDE 2193-1				**FIPA ACL**
Message area	Message element	Description	Use	Equivalent msg. parameter
`interaction-Elements`	`-`	Content of a message (submodels or another data elements)	O	`content`
	`type`	Purpose, i.e., intention of a message	M	`performative`
	`sender`	Sender of a message (AAS ID)	M	`sender`
	`receiver`	Receiver of a message (AAS ID)	O	`receiver`
	`conversationId`	ID of a conversation	O	`conversation-id`
`frame`	`messageId`	ID of a message	M	`-`
	`replyTo`	Expression referencing the original message	O	`reply-with`
	`replyBy`	Time by which the response must be received	O	`reply-by`
	`semanticProtocol`	Identification of the used semantic protocol	M	`protocol`
	`role`	Role of the sender of the message	O	`-`

M - mandatory; O - optional.

Table 3.2 also relates the parameters of a message specified in the FIPA ACL Message Structure with the message elements defined in VDI/VDE 2193-1, showing the equivalence between both. This similarity allows the I4.0 language message structure to be used in conjunction with the interaction protocols specified in FIPA, not requiring major changes

in the specified protocols and offering a larger catalog of interaction protocols for AAS Type 3 solutions.

As exemplified in the VDI/VDE 2193-1 standard, the messages between AASs Type 3 can be exchanged in JSON format. For instance, below is a sample message for communication between AASs of Type 3 in JSON format:

```
{
  "frame": {
    "semanticProtocol": {
      "keys": [
        {
          "type": "string",
          "idType": "string",
          "value": "string",
          "local": "boolean"
        }
      ]
    },
    "type": "string",
    "messageId": "string",
    "sender": {
      "identification": {"id": "string","idType": "string"},
      "role": {"name": "string"}
    },
    "receiver": {
      "identification": {"id": "string","idType": "string"},
      "role": {"name": "string"}
    },
    "replyBy": "number",
    "inReplyTo": "string",
    "conversationId": "string"
  },
  "interactionElements": []
}
```

3.5.2 Examples of Interaction Patterns

This section presents some examples of interaction patterns defined in VDI/VDE 2193-2 and FIPA standards that the agent-based AAS approach can use for communication. In these interaction patterns, the exchange of messages should follow the message structure described in the previous section and summarized in Table 3.2. Furthermore, these interactions should be complemented by the communicative acts listed in Table 3.3, which provides an overview of the message element "type" (see Table 3.2).

Bidding Protocol

As an example, Figure 3.15 shows the VDI/VDE 2193-2 Bidding protocol [40], which can support negotiation processes between agent-based AASs in manufacturing environments,

Table 3.3: List of types [39].

Type	Action of the sender/receiver
General status	
Consent (Agree)	I will execute the function you requested
Refusal (Refuse)	I refuse to perform the activity you requested
Error (Failure)	I was not able to execute your request
Not understood (Not understood)	I did not understand what you want me to do
Contract initiation	
Call for proposals (Call for Proposal)	Please send me offers for the following call for proposals
Offer (Propose)	I offer to execute this offer
Offer acceptance (Accept Proposal)	I accept your offer
Offer rejection (Reject Proposal)	I reject your proposal
Exchange of knowledge	
Informing (Inform)	Believe me when I tell you that the following is true or untrue in my model of the world
Confirming (Confirm)	The statement received is consistent with my model of the world
Contradicting (Disconfirm)	The statement received is not consistent with my model of the world
Spreading (Propagate)	Believe me when I tell you the following and pass it on
Subscribing (Subscribe)	Send me constant updates
Queries	
Query as to whether (Query If)	Notify me as to whether the following statement is true in this model of the world
Query ref (Query Ref)	Notify me of the values for the following query
Requirement (Request)	Perform the following activity
Requirement as to when (Request When)	Execute the following activity if the condition occurs
Requirement as to how often (Request Whenever)	Like requirement as to when, but repeatedly when the condition occurs
Cancel	
Cancelling (Cancel)	Command reversal, cancellation of dialogues in general
Mediating	
Proxy (Proxy)	Forward the message included here to the I4.0 components specified

where they can act as service requesters and/or service providers. This protocol is particularly useful for scenarios, e.g., adaptable factory and order-controlled production [115], where assets need to efficiently negotiate based on predefined rules and criteria to achieve the manufacturing goals. In this context, the service requester can represent a product that needs a certain service (e.g., the capabilities provided by a resource) to meet its requirements, and the service provider can represent a resource that offers its capabilities (e.g., drilling and milling) as a service at a certain cost. In this interaction pattern, the service requester initiates the interaction by performing a "call for proposals" containing a description of the required service and waits for responses from service providers. Service providers can then refuse or propose to provide the service by checking their feasibility. In case of acceptance, they send proposals with a price and other conditions for the execution of the service. Finally, the service requester evaluates the received proposals and selects the best one based on the desired criteria.

This protocol presents versatility to meet different scenarios, namely resource allocation based on vertical collaboration and system reconfiguration due to condition changes based on horizontal collaboration. In the first scenario, a higher-level system can negotiate with several shop floor resources based on the capabilities they provide to determine which resources will be responsible for performing operations on the products to be manufactured. On the other hand, in the second scenario, if a machine designated to perform an operation breaks down, the product may be able to autonomously find and negotiate with other resources that are capable of performing the required capability.

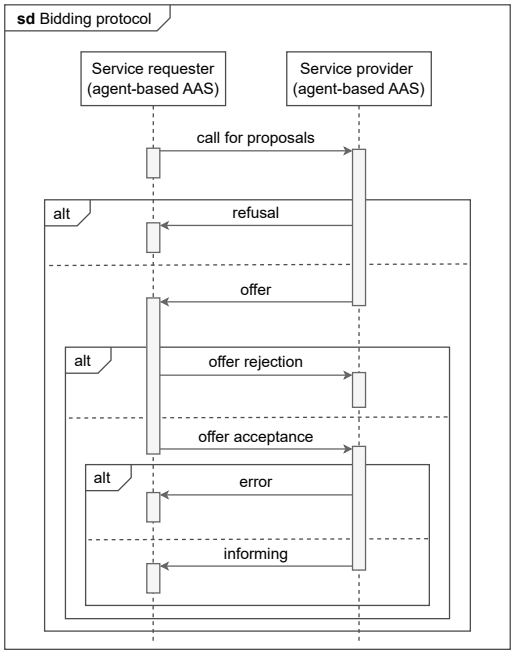

Figure 3.15: Bidding protocol (adapted from [40]).

Request Interaction Protocol

Figure 3.16 illustrates the interaction pattern for the operation execution process based on the FIPA Request Interaction protocol [116], where agent-based AASs can assume the role of initiators and/or participants. This interaction allows for a flexible and adaptive manufacturing process through the dynamic adjustment of process parameters based on real-time conditions and requirements. In such scenario, machines need to constantly adjust their operational parameters (skill parameters) to meet the production requirements. For instance, a resource (initiator) can request the process parameters for a product being produced (participant). The product sends the parameters, and then the resource can decide whether to accept or refuse to perform the operation by checking the feasibility of the requested parameters. If accepted, the resource adjusts its operational parameters and performs the operation, informing the results at the end.

Publish-Subscribe Protocol

This interaction pattern is based on the FIPA Subscribe Interaction protocol [117] to continuously receive information of interest or event notifications. For instance, if a machine breaks down, its corresponding agent-based AAS can publish a message notifying that the machine cannot perform the scheduled operations. In such scenario, all products designated for that machine are informed about the unexpected event and can attempt to redefine a new manufacturing plan based on the negotiations defined by the Bidding protocol (see Figure 3.15). In this context, agent-based AASs can assume the roles of

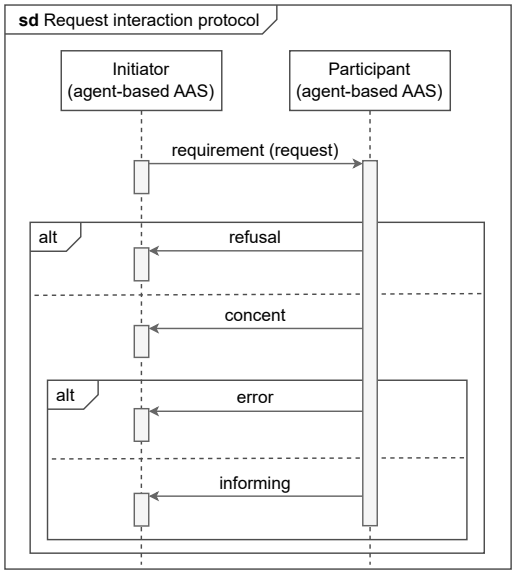

Figure 3.16: Request interaction protocol (adapted from [116]).

subscribers and/or publishers, as shown in Figure 3.17. The subscriber needs to subscribe to a specific topic to receive the desired information, which can be periodic data updates or a warning when an event is triggered. On the other hand, the publisher will publish information on that topic when these circumstances are satisfied. Typically, this publish-subscribe model involves a broker, an intermediary entity that manages the exchange of messages between subscribers and publishers.

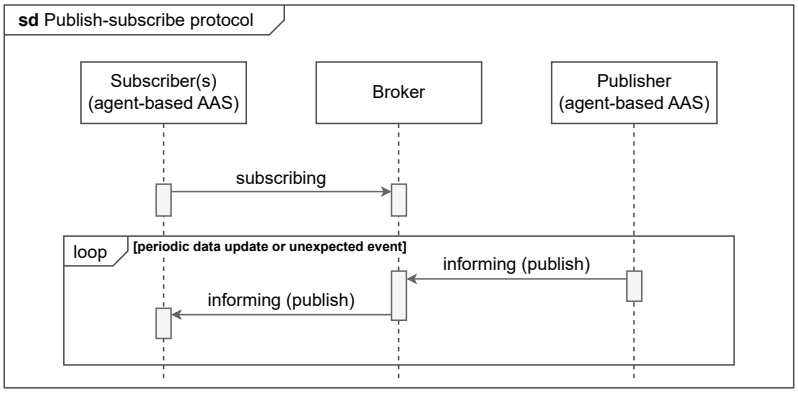

Figure 3.17: Publish-subscribe protocol.

For this interaction pattern, a submodel containing a list of topics can be modeled, and the agent-based AAS can use this submodel to know which topics it needs to subscribe to in order to receive information. Based on the information received, the agent-based AAS can be designed to take the proper actions according to the specific use case. This protocol is particularly useful for broadcast messages, where a component needs to communicate with several other components. For instance, in a fault-tolerance scenario, an Autonomous and Mobile Robot (AMR) that fails can publish a message to advise other AMRs not to use a congested route.

Other Supporting Protocols

The following interaction pattern examples are based on the FIPA Propose Interaction Protocol [118], the FIPA Query Interaction Protocol [119], and the FIPA Request When Interaction Protocol [120]. These interaction patterns do not focus on attending to a specific scenario but can be useful in several scenarios, particularly enabling the request and exchange of information, which is fundamental in decentralized systems, where the entities have only a partial knowledge and view of the system and may need more information for the decision-making process.

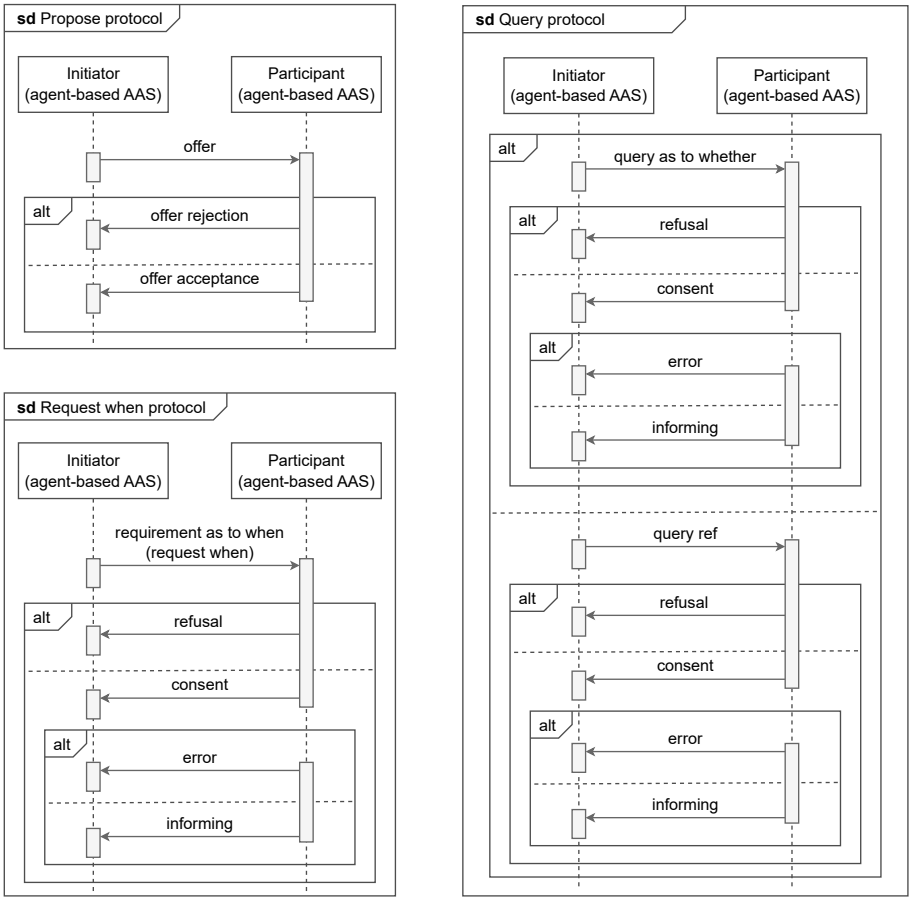

Figure 3.18: Supporting interaction protocols (adapted from [118–120]).

In the propose interaction protocol, the initiator initiates by sending an offer message to the participant, indicating its intent to perform a specific action. After receiving the offer message, the participant then responds with either acceptance or rejection. Contrarily, in the request when interaction protocol, the initiator sends a message to the participant, instructing them to carry out a particular action as soon as a certain precondition becomes true. Lastly, in the query protocol, the initiator seeks information from the participant through the communicative methods: "query as to whether" or "query ref". The "query as to whether" communication is utilized when the initiator wants to determine the truth of

a given proposition, while the "query ref" is employed when the initiator desires specific information from the participant.

Exceptions

For all interaction patterns, the receiver, at any moment of the communication flow, can notify the sender if it does not comprehend the communication by sending a message indicating "not understood". Moreover, at any moment in the communication flow, one of the entities (sender or receiver) may decide to terminate the interaction by sending a message indicating "cancelling". In such cases, the interaction is terminated, rendering any commitments made during the communication null and void. For instance, if a negotiation interaction is canceled for some reason, any agreement resulting from this negotiation must be disregarded, i.e., the system should be returned to the state before the interaction. This process may involve several actions that should be considered in the implementation, but are not addressed in the experimental part of this work. The following are some examples of these actions:

- Rollback of changes: Any change made during the interaction process should be reversed, e.g., any updates to databases, changes in statuses, or alterations in records.
- Error handling: The system should be designed to handle cancellations, ensuring that no unintended side effects occur as a result of the cancellation process.
- Notification: Relevant participants should be notified about the cancellation. This could involve notifying participants involved in the interaction, as well as any other stakeholders who might be affected.
- Logging: The cancellation event should be logged for future reference. This log should include details such as the reason for cancellation, the participants involved, and any actions taken during the interaction.

3.6 Strategies for Developing AI-Based Behaviors

According to the metamodel described in Section 3.1.2, the *Behavior* class is responsible not only for implementing how the agent behaves and acts in the system but also for implementing the intelligent aspects of the agent-based AAS. From an individual perspective, each agent-based AAS should have an intelligent behavior ranging from simple rule-based mechanisms to advanced AI techniques. On the other hand, from a collective perspective, the global intelligent behavior of the system can emerge from the interactions between the agent-based AASs, even when they exhibit simple behaviors and limited intelligence. In this context, this section presents some examples of AI techniques that can be considered

and an approach for implementing intelligence in an individual agent-based AAS, which can ultimately contribute to the system's collective intelligence.

3.6.1 Examples of AI Techniques

Integrating AI into agent-based AASs significantly enhances their intelligence, enabling them to effectively handle complex and collaborative tasks. These techniques can range from simple rule-based mechanisms, which can be simple and nearly immediate, to more sophisticated ML and Deep Learning (DL) approaches that can handle more complex tasks with greater adaptability.

Rule-Based Mechanisms

A rule-based mechanism is a basic type of AI model that uses a set of predefined set of rules to make decisions and solve problems. These rules are typically created by developers leveraging human expert knowledge in the relevant domain and are usually structured as "if-then" statements that dictate the system's behavior in different situations. A classic example is the Nelson Rules [121], which are a process control method used to determine whether any measured variable is out of control. Figure 3.19 presents an example of an agent-based AAS endowed with a rule-based mechanism. This mechanism consists of acquiring information by sensing the environment using sensors or through the arrival of messages from other agent-based AASs, making a decision based on this information and the predefined set of rules, and taking the proper actions, such as triggering an actuator or sending a message to other agent-based AAS.

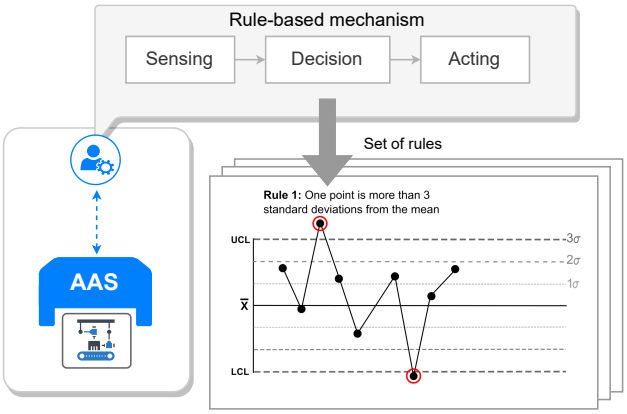

Figure 3.19: Example of rule-based mechanisms in the agent-based AAS approach.

Although the rule-based mechanism results in the lowest level of intelligence, it can provide several benefits. Rule-based mechanisms are easy to design and implement. The logic is often easy to understand, modify, and debug, which can reduce development time. Rule-based mechanisms can provide immediate responses to specific conditions without

the need for complex computations or learning phases. The decisions are typically more transparent, explainable, and predictable than those made by more complex AI techniques. This can be important in domains where understanding the decision-making process is necessary for validation, compliance, or user trust. Furthermore, rule-based mechanisms usually require less computational power compared to advanced AI techniques. This makes them suitable for environments with limited processing capabilities or for real-time applications where quick decision-making is essential.

Machine Learning and Deep Learning Approaches

Including advanced AI techniques into agent-based AASs, namely ML and DL, empowers them to move beyond predefined rules, significantly improving their decision-making capabilities to address complex and dynamic scenarios. These approaches leverage training datasets to develop AI models that can, e.g., learn patterns, predict outcomes, and adapt to evolving conditions. Some examples of these techniques include:

- Supervised Learning is a ML category that involves learning from labeled data. For example, an agent can be equipped with supervised learning techniques for predictive maintenance. The agent is trained on historical data from a machine, with labeled instances of machine failures and normal operations. By learning the patterns associated with machine breakdowns, the agent can analyze real-time data and predict when a failure is likely to occur.

- Unsupervised Learning is a ML category that involves learning from unlabeled data, i.e., the machine learns to find patterns and relationships in the data without any predefined labels. For example, an agent monitoring a machine's vibration could utilize unsupervised learning to identify anomalies. The agent is provided with raw, unlabeled vibration data over time. It learns the normal patterns of the machine's vibration during typical operations. If the machine starts vibrating at a higher frequency than usual, the agent can flag this as an anomaly, even though it was not explicitly trained on what constitutes a failure or normal operation.

- Reinforcement Learning is a ML category where an agent learns to make decisions by interacting with its environment to achieve a specific goal. The agent learns through trial and error, receiving feedback in the form of rewards or penalties for the actions it takes. For instance, an agent managing a conveyor system could use reinforcement learning to optimize operations. The agent can choose to increase, decrease, or maintain the conveyor speed based on real-time conditions. The agent receives a positive reward when it successfully increases throughput, such as when the number of items processed per minute rises. On the other hand, the agent receives a negative reward when it causes delays, or if speeding up the conveyor

belt leads to the accumulation of items due to slow processing at a workstation. This continuous feedback loop allows the agent to learn and refine its speed adjustments.

- Neural networks are the foundation of DL, where multiple layers of neurons enable learning of complex patterns from large datasets. For example, in manufacturing, an agent equipped with neural networks could analyze product images to automatically and accurately detect defects. This detection process improves production quality and reduces the need for manual inspection.

Moreover, recently, LLMs have revolutionized the landscape of AI by significantly enhancing the capabilities of intelligent systems. These models are trained on extensive datasets and demonstrate exceptional skills in comprehending and producing natural language. In the context of agent-based AASs, LLMs can enhance language processing abilities, enabling them to interpret detailed instructions, extract relevant information from vast amounts of text, and communicate effectively with humans and other agents. For example, figure 3.20 illustrates an interaction between two LLM agent-based AASs, which can communicate using natural language and adhere to the VDI/VDE 2193-1 message structure in a request interaction protocol. This is an illustrative example that can be further complemented to perform more complex tasks and decision making.

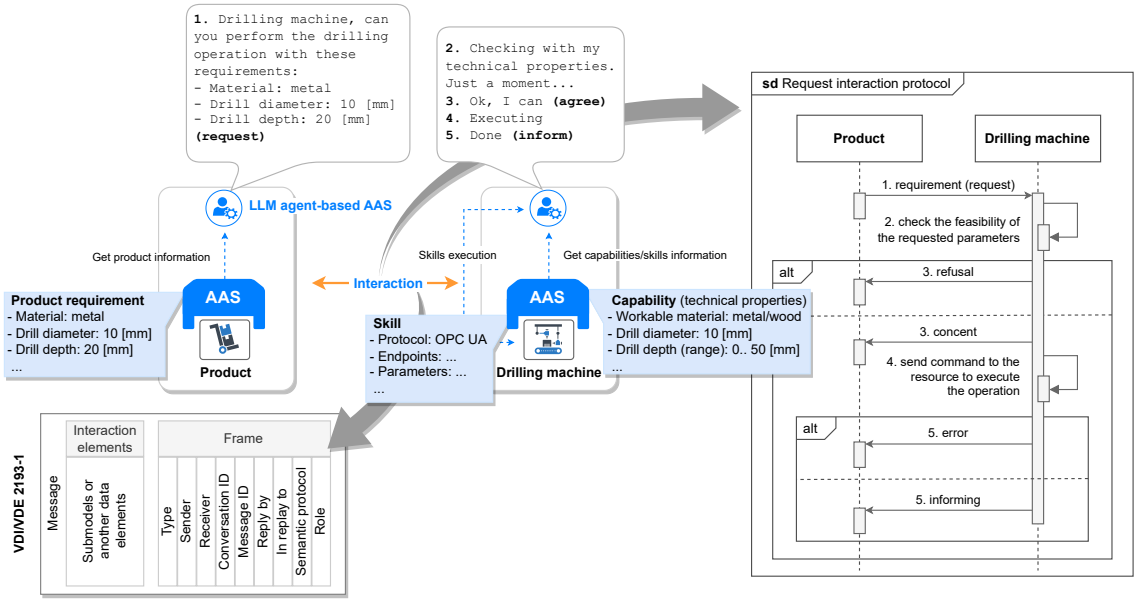

Figure 3.20: Example of LLMs in the agent-based AAS approach.

In general, both rule-based and more advanced AI techniques have their strengths and weaknesses, and they are often used together in many applications to complement each other. Table 3.4 summarizes the strengths and weaknesses of both approaches to provide guidance on selecting the appropriate strategy.

Table 3.4: Comparison between rule-based mechanisms and AI techniques.

Features	Rule-based mechanisms	ML/DL
Learning Capability	None (rules are static and do not evolve unless manually updated)	High (AI models can learn from new data, improving performance and adapting to new conditions over time)
Transparency	High (rule are explicitly defined and easily understood)	Low ("black box")
Data Requirement	Low (requires minimal data, typically real-time operational data)	High (requires large datasets for training)
Simplicity	High (easy to understand and implement)	Low (complex models)
Consistency	High (follow predefined rules that always produce the same output for a given input)	Variable (while AI can be highly accurate, slight variations in data or the learning process may lead to inconsistent behavior in some cases)
Adaptability	Low (rules are fixed and require manual updates when conditions change)	High (learns from new data and adapts without manual intervention)
Computational Requirements	Low (requires minimal resources due to simplicity)	High (demands significant computational power)
Reactivity	High (fast, deterministic decisions based on predefined rules)	Variable (simple models can be fast, but complex models may introduce delays)
Industry Applicability	High (well-suited due to explicit rules and predictability)	Low to Medium (difficult due to lack of transparency)
Maintenance	High (requires manual rule adjustments when conditions change)	Medium (models need retraining, but less frequent manual intervention is required)

3.6.2 Deployment of AI Models Into the Agent-Based AAS

The implementation of rule-based mechanisms can be simple and direct, typically using "if-then" statements that define clear decision-making logic. However, for more complex, dynamic environments, integrating advanced AI models based on ML/DL into an agent-based AAS significantly enhances its capabilities. In this sense, this section provides an approach to implementing and deploying such AI models into an agent-based AAS.

As illustrated in Figure 3.21, the agent-based AAS should act as a model consumer, where AI models developed in a training/modeling environment are deployed to the deployment/operational environment for execution. The (re)design phase encompasses two key steps: first, the *model producer* (e.g., experts, systems, or tools) is responsible for developing, training, and validating AI models; second, the *modeling submodel* handles the conversion and exportation of these models into standardized, machine-readable interchange formats, such as PMML, PFA or ONNX, for inclusion in appropriate submodels (e.g., the *AI Deployment* submodel as described in Section 3.2.4). On the other hand, the operational phase involves deploying the AI models. During this phase, the *model consumer*, represented by the agent-based AAS, retrieves the AI models described in PMML, PFA, ONNX, or another standardized format from the designated submodels and executes them seamlessly.

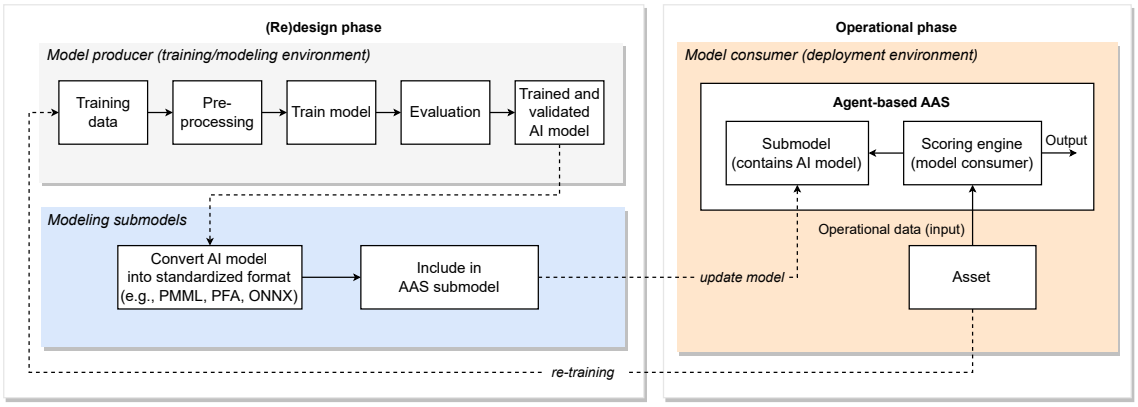

Figure 3.21: Deployment of AI models into the agent-based AAS.

This decoupling, combined with the use of standardized, machine-readable interchange formats for AI models, facilitates AI development while enabling interoperability and seamless sharing of models across various applications and systems. These standardized formats ensure that AI models, developed in specialized environments using appropriate programming languages and tools, can be deployed in environments that might be different than where the models were initially trained, facilitating their integration into diverse operational settings. Moreover, integrating these standardized AI models into AAS submodels offers an I4.0-compliant approach to describing and representing AI within the I4.0 context.

Based on that, once an AI model is included in a specific submodel, the agent-based AAS needs to retrieve this model from the proper submodel and consume it using an appropriate scoring engine approach. A scoring engine is a software component that processes input data through an AI model to produce, e.g., predictions or inferences. It is able to interpret the specific structure of an AI model and applies the model to input data in real time. Furthermore, as shown in Figure 3.21, as new data is generated, the AI model can be retrained and updated within the submodel, ensuring continuous improvement and adaptation to evolving operational conditions.

As an illustrative example, consider the classic Iris dataset (often referred to as the "hello world" of AI) translated into a manufacturing context, such as the monitoring of an asset. While labeled data is valuable yet often scarce in manufacturing, this proof-of-concept example assumes the availability of labeled data for asset parameters to demonstrate the approach. The emphasis here is not on the AI algorithm itself but on demonstrating how the agent-based AAS can support and consume different AI models tailored to specific applications.

As shown in Figure 3.22, the asset dataset consists of various parameters collected from the asset, such as temperature, pressure, vibration, and humidity, recorded over a period of time. These parameters are typically labeled to indicate whether the asset is operating

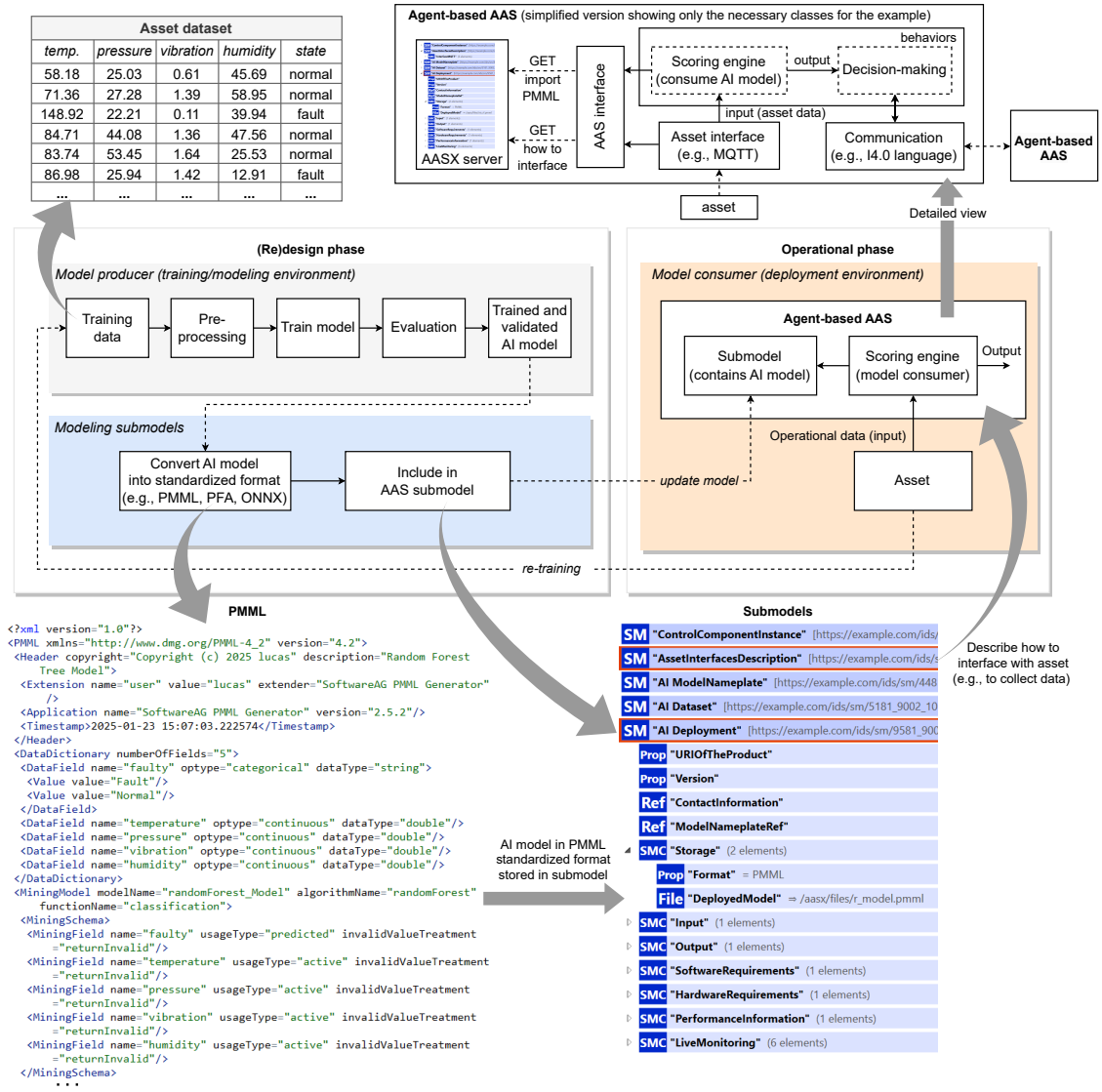

Figure 3.22: Example of AI model deployment for asset monitoring in agent-based AAS.

in a normal or fault state. In a training/modeling environment, the dataset is used to train an AI model based on the random forest algorithm (e.g., utilizing tools or programming languages such as Python, R, or others). Once the model is trained and validated, it is converted into PMML format and included in the *AI Deployment* submodel.

In the operational phase, the agent-based AAS, which may be developed in a Java environment, retrieves the model from the *AI Deployment* submodel and consumes it using a scoring engine (e.g., through Java PMML API). By using this model, the agent-based AAS can predict asset states based on input data and trigger alerts when potential faults are detected. Although this example specifies particular technologies for implementation, the proposed approach is technology-neutral and generic. This means that alternative AI models developed in any environment, using any programming language or tool, can also be utilized. Furthermore, as already discussed in Section 3.2.4, describing AI models in

standardized, machine-readable interchange formats can bring several benefits, such as enhanced interoperability and portability. However, some challenges, such as the limited availability of APIs and frameworks that support these standardized models, could impact their adoption. This does not invalidate the proposed solution, as alternative approaches, such as model serialization and de-serialization, can be adopted, taking into consideration portability issues.

3.7 Summary

This chapter introduced the specification of the agent-based AAS approach for designing AAS Type 3 solutions. This approach enables the transition from passive components to truly I4.0 components with enhanced management, interoperability, autonomy, intelligence, and collaborative capabilities. In summary, this chapter described the work developed to answer the first research question of this thesis: *What aspects an agent-based approach must consider to design and develop an Asset Administration Shell Type 3 solution?*

The lack of specifications and limited research regarding AAS Type 3 design presents a challenge in developing such components that extend beyond the conventional functionalities of AAS Type 1 and 2 solutions by incorporating more sophisticated features, enabling I4.0 components to operate with greater autonomy, intelligence, and collaborative capabilities. In this context, an information metamodel (Section 3.1) is proposed to address this issue by offering a specification for designing AAS Type 3 through an agent-based approach. The information metamodel consists of two parts: the information part and the intelligent part. The information part includes the main classes for describing asset information in the submodels, and the intelligent part includes the main classes for enhancing the AAS with autonomy, intelligence, and collaborative capabilities provided by the agent.

Section 3.2 presented the modeling of the information part of the agent-based AAS, describing some fundamental submodels that enable agent-based AAS to effectively manage and interact with its corresponding assets in the industrial environment, as well as to enable collaboration with each other. These submodels comply with the specifications provided by the IDTA to ensure greater interoperability and are mainly based on the concepts of capabilities and skills from the capability- and skill-based engineering domain, which are formally specified in the CSS reference model.

The remaining sections focused on the intelligent part of the agent-based AAS. Section 3.3 presented the integration between the intelligent and information part, highlighting two approaches. The first approach is based on AAS specification part 2 (version 3.0), which defines APIs for accessing asset information stored in submodels. The second approach is based on the "OPC 30270: Industry 4.0 Asset Administration Shell", providing guidelines

for mapping the AAS metamodel to the OPC UA information model. Section 3.4 described the integration patterns for interfacing agent-based AASs with physical assets based on the generic interface practices provided by the IEEE 2660.1 standard, as well as the applicability of the IEEE 2660.1 recommendation method in the proposed approach. Section 3.5 presented the vocabulary and structure of the messages and some examples of interaction patterns between agent-based AASs based on the VDI/VDE 2193 and FIPA standards. Finally, Section 3.6 covered some examples and an approach for implementing intelligence in individual agent-based AAS.

In summary, this chapter presented the conceptual aspects of the design of the agent-based AAS approach. This approach aims to be broadly applicable and use-case independent, supporting diverse collaboration strategies. The following chapter will focus on implementing this approach in practice and testing it in various scenarios and case studies with varying requirements and different types of assets.

CHAPTER 4

Experimental Implementation and Validation

THE proposed agent-based AAS approach was developed to guide the design of AAS Type 3 solutions using MAS technology, enabling I4.0 components to operate with more intelligence, autonomy, and collaboration. It is important to emphasize that the proposed approach does not aim to replace the already existing system architecture but rather to provide an I4.0-compliant, modular solution that can enhance the existing system architecture, which currently lacks these advanced features. Furthermore, the proposed approach offers the potential to extend AAS Type 1 and 2 solutions to Type 3, maintaining compatibility with the official AAS metamodel while integrating agents into the structure. In this context, this chapter intends to validate the proposed agent-based AAS approach in two different case studies, the first based on a laboratory prototype (small-scale production system) and the second on an industrial automotive production line. For both case studies, agent-based AASs were designed and implemented to enhance the existing system with autonomy, intelligence, and collaborative capabilities.

The remainder of this chapter is structured as follows:

- Section 4.1 outlines the assessment methodology for the agent-based AAS approach, detailing the assessment scenarios, evaluation criteria, and case studies.
- Section 4.3 explains the general process of implementing an individual agent-based AAS using proper development platforms.
- Section 4.4 presents the deployment of the agent-based AAS approach for the small-scale production system case study.
- Section 4.5 presents the deployment of the agent-based AAS approach for the automotive production line case study.

- Section 4.6 discusses how the implemented agent-based AAS approach meets the characteristics of an AAS Type 3 and established assessment criteria.
- Section 4.7 summarizes the key aspects discussed in this chapter.

4.1 Assessment Methodology

This work uses ISO/IEC 25040:2011[1] as a reference to develop the assessment methodology of the agent-based AAS approach. Although this standard is target for software products, it can be adapted for CPS-based solutions. As shown in Figure 4.1, the proposed methodology comprises four steps. The first step consists of defining the purpose of the assessment and identifying the assessment scenarios (Section 4.1.1). The second step aims to specify the assessment criteria, namely evaluation metrics, scores, and criteria for judging (Section 4.1.2). The third step focuses on testing the proposed approach in case studies considering the defined scenarios (Sections 4.2, 4.3, 4.4, 4.5). Finally, the fourth step runs the assessment by comparing and analyzing the results with the defined assessment criteria (Section 4.6).

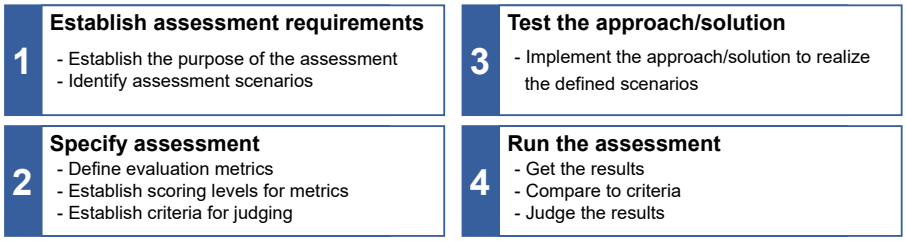

Figure 4.1: Assessment methodology of the agent-based AAS approach.

4.1.1 Scenarios

Agent-based AASs can be designed to realize different collaborative strategies depending on the system's needs. In this context, this work defines some scenarios to guide the assessment process of how the proposed approach can handle some production processes that require collaboration between assets. These scenarios describe how a manufacturing system should behave and function to achieve I4.0. In this work, these scenarios are derived from the Adaptable Factory (AF) scenario, which is one of the application scenarios [115] proposed by the Plattform Industrie 4.0 that describe a vision for the digital transformation of industrial systems towards the realization of I4.0. In this regard, the proposed approach is designed and implemented, aiming to realize such scenarios.

[1] ISO/IEC 25040:2011 contains requirements and recommendations for the evaluation of software product quality. This evaluation process can be used for different purposes and approaches.

The AF scenario envisions a manufacturing environment that is highly flexible and responsive to changing demands. A core concept of this scenario is plug-and-produce, which focuses on the idea that machines and equipment should be easily connected and interoperable, allowing them to be quickly integrated into the production process with minimal or no reprogramming/reconfiguration. In addition, machines and equipment should be designed to communicate and collaborate autonomously. The goal is to create a manufacturing environment that can quickly adapt to changing conditions, reducing reliance on static production lines and manual intervention.

Since the AF scenario is multifaceted, it can be broken down into many different sub-scenarios. These sub-scenarios contemplate different aspects involved in the AF scenario that require a certain level of collaboration between assets, namely resource allocation, operation execution, and system reconfiguration due to unexpected events. This separation into sub-scenarios allows for targeted evaluation of the proposed approach in critical interactions between assets rather than testing the entire process from start to finish, making it possible to verify the system's capacity to coordinate and adapt in specific situations.

It is crucial to understand these sub-scenarios as discrete conditions that may occur independently of each other. While they can occur sequentially or be interlinked based on production conditions, it is essential to recognize them as specific situations that can happen at certain moments within the manufacturing system that demand collaboration strategies for resolution. This perspective allows for a more problem-oriented evaluation of how the proposed approach can handle some collaborative production processes or some challenges that may arise within the manufacturing environment.

Operation Execution Scenario

This sub-scenario deals with the execution of manufacturing operations in an adaptable factory environment. It involves coordinating and executing tasks between various assets, namely products and resources, to fulfill production requirements. Products have knowledge about their own manufacturing process, enabling them to navigate autonomously through various stages of production. As it progresses through each station, the product communicates with the corresponding resources to adjust process parameters according to its unique specifications. Meanwhile, resources within the factory, such as machines and robots, operate in harmony with the products, responding to their needs. This interaction allows for dynamic adjustments based on the specific needs of each product, enhancing flexibility within the production process. In summary, the realization of this sub-scenario requires the following conditions:

- Products and resources must have communication capabilities to exchange information related to manufacturing operations, e.g., process parameters.

- Products must have knowledge about their own manufacturing process.
- Resources must have the ability to adjust their own skill parameters according to the product's needs.

Autonomous Negotiation Scenario

This sub-scenario focuses on the dynamic allocation of resources within the manufacturing environment. It involves determining which machines and equipment should perform specific functions based on negotiation strategies and asset capabilities. This approach allows for efficient utilization of machinery and equipment by matching their capabilities with the requirements of specific manufacturing tasks or products. The self-description of asset capabilities is crucial in this process, as it allows for the identification of which assets are capable of offering the required capabilities for manufacturing a new product or performing a specific task. Negotiation strategies for resource allocation may consider multiple criteria beyond just asset capabilities, such as cost, lead time, and quality requirements. In summary, the realization of this sub-scenario requires the following conditions:

- Products and resources must have the ability to negotiate with each other.
- Products must have knowledge about their own manufacturing requirements, i.e., the capabilities needed to be produced, e.g., a product may require drilling and laser cutting operations.
- Resources must provide their capabilities as services that other assets (e.g., products) can search for and request.

Unexpected Event Scenario

This sub-scenario focuses on the system's ability to adapt and reconfigure in response to unexpected events, e.g., a machine breakdown or malfunction. The system can initiate automated response protocols when it detects an unexpected event. For example, if a machine breakdown is imminent or has already occurred, the system can automatically reconfigure production schedules and resource allocations to minimize the impact on overall operations. In summary, the realization of this sub-scenario requires the following conditions:

- The system must be capable of automatic reconfiguration in response to unexpected events, meaning that assets must collaborate to resolve the issue.
- Resources must continuously monitor their health conditions and notify other assets in case of an unexpected event, such as a breakdown or malfunction.

- Products must have knowledge about their own manufacturing process and requirements.

4.1.2 Assessment Criteria

As previously mentioned, the proof of concept of the proposed approach consists of testing it in different manufacturing scenarios where collaboration between assets is required. In this context, it is fundamental to define the assessment criteria to evaluate the proposed approach in these scenarios. The objective is not to quantitatively evaluate the approach's performance, since the agent-based AAS acts as a higher layer that complements the already implemented existing system, enhancing it with dynamic and adaptive functions based on autonomous entity collaboration that can operate in soft real-time. In this regard, quantitative metrics related to time constraints are much more dependent on the low-level functions, for which the proposed approach is not responsible. Other metrics based on the time needed for agent-based AASs to negotiate or interact with each other in order to make decisions could also be analyzed, as these metrics could affect the system performance when multiple entities are exchanging information simultaneously. However, various factors, such as the development technology and the computational platforms used for implementation, can influence such evaluation metrics. In this case, a comparative analysis of different technologies and strategies to implement the proposed solution would be necessary, which is outside the scope of this work.

Based on that, this work discusses the implemented solution in terms of qualitative criteria, namely adaptability, pluggability, robustness, interoperability, and (re)usability, which are important requirements to realize I4.0-compliant solutions, particularly for the AF scenario and its derived sub-scenarios. Since these requirements cover a wide spectrum, their understanding and interpretation can be subjective and difficult to evaluate. In this sense, these requirements are broken down into sub-requirements, which helps in clarifying and evaluating specific aspects of each requirement (see the example for adaptability in Table 4.2). This method facilitates a more granular understanding and assessment, reducing ambiguity and subjectivity in interpretation. Moreover, by identifying the individual sub-requirements, it is possible to quantitatively measure the level of each requirement fulfillment using the Equation 4.1.

$$requirement = \frac{5}{n} \times \sum_{i=1}^{n} \beta_i \leq 5 \tag{4.1}$$

where n is the number of sub-requirements of a requirement and β_i is a score of fulfillment of a sub-requirement, ranging from not applicable ($\beta = 0$), low ($\beta = 0.25$), medium ($\beta = 0.5$), or high ($\beta = 1.0$). Each requirement has a maximum value of 5, since the objective is to express the overall score on a 0 to 5 scale.

For instance, Table 4.1 shows an example of a requirement with five sub-requirements ($n = 5$). In this sense, considering the score of fulfillment (β_i) for each sub-requirement shown in Table 4.1 and applying Equation 4.1, this illustrative requirement has a value of 3 out of 5. Note that the obtained result is indicative and may not accurately represent the actual performance of the proposed approach in meeting the requirement since the assessment of each individual sub-requirement is performed qualitatively. This assessment method should be seen as a tool to identify the strengths and areas for improvement in the proposed approach.

Table 4.1: Example to evaluate a requirement.

Sub-requirement	β			
	N/A (0)	**Low (0.25)**	**Medium (0.5)**	**High (1.0)**
Sub-requirement 1	✓			
Sub-requirement 2			✓	
Sub-requirement 3				✓
Sub-requirement 4				✓
Sub-requirement 5			✓	

The following sections further describe the proposed requirements and sub-requirements to be used in this work. The requirement (level 1) provides a general definition, and the sub-requirements (level 1.x) provide further details that a solution should consider, aiming to meet the requirement.

Adaptability

The pursuit of highly adaptable systems has been a longstanding goal, even before the emergence of the Fourth Industrial Revolution and the widespread use of CPS to achieve such a level of adaptability in industrial processes. In today's market, characterized by volatile consumer demands and the growing preference for high-quality and customized products, adaptability has become even more imperative for companies to remain competitive. The adaptability enables such companies to adapt and respond quickly to changing conditions. Table 4.2 describes the adaptability concept and the set of sub-requirements that a solution should consider to achieve adaptability in manufacturing systems.

Pluggability

Another important requirement for manufacturing systems is pluggability, or the most well-known term, "plug-and-produce". The idea behind this concept is to simplify the setup and reconfiguration of manufacturing systems, allowing for rapid deployment of new equipment, quick adaptation to changing production needs, and seamless integration of diverse technologies and subsystems. Table 4.3 describes the pluggability concept and

Table 4.2: Adaptability requirement.

Requirement	**Requirement description**
Adaptability 1	Adaptability refers to the ability of industrial systems to learn from and respond to changing conditions over time, improving their performance autonomously
Sub-requirements	**Sub-requirement description** (The proposed solution should...)
Adaptability 1.1	have the perception and action capabilities to effectively interact with and respond to different contexts or situations
Adaptability 1.2	enable seamless collaboration among the system's components, allowing them to share information, make coordinated decisions, and work together to solve complex tasks
Adaptability 1.3	be able to learn from new data and experiences to improve its performance and decision-making processes over time
Adaptability 1.4	be able to modify its behavior in response to new information or changing environments

the set of sub-requirements that a solution should consider to achieve pluggability in manufacturing systems.

Table 4.3: Pluggability requirement.

Requirement	**Requirement description**
Pluggability 1	Pluggability refers to the ability of machines, devices, and components to automatically recognize and self-configure to work together seamlessly without the need for manual intervention or extensive setup
Sub-requirements	**Sub-requirement description** (The proposed solution should...)
Pluggability 1.1	be capable of automatically detecting and configuring connected devices, minimizing the need for manual setup and intervention
Pluggability 1.2	be able to quickly reconfigure manufacturing systems to accommodate changes in product specifications or production requirements
Pluggability 1.3	enable components from different manufacturers to seamlessly communicate and collaborate to perform complex tasks without requiring extensive customization or integration effort
Pluggability 1.4	be modular and scalable to easily incorporate additional features or modules as needed
Pluggability 1.5	be able to perform the capability assessment. This involve interpreting the asset description and matching the production requirements to the asset capabilities

Robustness

Robustness is a fundamental requirement in I4.0, where industrial systems must operate reliably to ensure continuous production and avoid costly downtime. Robust systems have the ability to remain working correctly and relatively stable, even in presence of

disturbances. Table 4.4 describes the robustness concept and the set of sub-requirements that a solution should consider to achieve robustness in manufacturing systems.

Table 4.4: Robustness requirement.

Requirement	**Requirement description**
Robustness 1	Robustness refers to the system's ability to maintain its performance and stability in the face of potential external and internal disturbances
Sub-requirements	**Sub-requirement description** (The proposed solution should...)
Robustness 1.1	continue functioning, potentially at a reduced level, despite component failures
Robustness 1.2	continue functioning normally as new components are added to the system
Robustness 1.3	anticipate potential failures before they occur, allowing for timely maintenance actions that prevent downtime
Robustness 1.4	automatically identify issues and initiate corrective actions without human intervention
Robustness 1.5	ensure that data remains accurate, consistent, and unaltered during transmission and storage (e.g., using cryptographic techniques and secure protocols), as well as protect data from unauthorized access and attacks

Interoperability

Although the previous requirements are fundamental for realizing more intelligent and collaborative production systems, the "interconnectivity" of such systems is often limited, where assets from different vendors are not compatible. This limitation often leads to the development of ad-hoc solutions, where custom integration efforts are required to bridge the gap between disparate assets. These ad-hoc solutions may involve the development of custom interfaces, middleware, or translation layers to facilitate communication and data exchange between incompatible assets. To address these challenges and unlock the full potential of intelligent and collaborative production systems, efforts are required to promote greater interoperability within the industrial ecosystem. Table 4.5 describes the interoperability concept and the set of sub-requirements that a solution should consider to achieve interoperability in manufacturing systems.

(Re)usability

(Re)usability is a crucial aspect to consider when designing and implementing solutions in industrial environments. Unlike flexibility, pluggability, or robustness requirements, which focus on the behavior and operational aspects of manufacturing systems, (re)usability pertains more to the feasibility and practicality of the proposed solution within the context of industrial deployment. Table 4.6 describes the (re)usability concept and the set of

Table 4.5: Interoperability requirement.

Requirement	**Requirement description**
Interoperability 1	Interoperability refers to the ability of different systems, machines, and software applications to connect, communicate, and work together seamlessly, ensuring they can understand and exchange information effectively, regardless of their manufacturer or the technology they use
Sub-requirements	**Sub-requirement description** (The proposed solution should...)
Interoperability 1.1	be aligned with reference architectures and standards to support the design of such systems
Interoperability 1.2	adopt standardized information models to structure information
Interoperability 1.3	adopt standardized communication protocols, data formats, and structures to enable seamless exchange of information between systems
Interoperability 1.4	adopt standardized semantics to describe asset properties in an unambiguous, machine-readable, and industry-independent manner
Interoperability 1.5	enable the integration of data from different sources (vertically and horizontally), allowing for comprehensive analysis and decision-making

sub-requirements that a solution should consider to achieve (re)usability in manufacturing systems.

Table 4.6: (Re)usability requirement.

Requirement	**Requirement description**
(Re)usability 1	(Re)usability refers to the ability to efficiently (re)use resources, components, technologies, and processes across different applications, systems, and stages of production within the industrial environment
Sub-requirements	**Sub-requirement description** (The proposed solution should...)
(Re)usability 1.1	include generic models, templates, and guidelines to facilitate its design and implementation
(Re)usability 1.2	be compatible with existing hardware and software infrastructures, enabling easy deployment in different use-cases without extensive modifications
(Re)usability 1.3	allow for the efficient (re)use of modules, services, and classes in both small-scale and large-scale applications
(Re)usability 1.4	include mechanisms or a platform for managing the system, such as state and configuration, as well as interfaces that are accessible to users with diverse backgrounds, abilities, and preferences

4.2 Description of the Case Studies

This section describes the two case studies used to deploy, test, and validate the proposed approach, aiming to realize the defined scenarios presented in Section 4.1.1.

4.2.1 Small-Scale Production System

To test and validate the collaborative aspects of the proposed solution, the agent-based AAS approach was implemented and experimentally tested in a small-scale production system demonstrator based on a Fischertechnik infrastructure[2], as illustrated in Figure 4.2. This demonstrator serves as a versatile testing ground for I4.0 solutions, providing the ability to test different production process scenarios on a small-scale prototype, including the previous ones defined in Section 4.1.1.

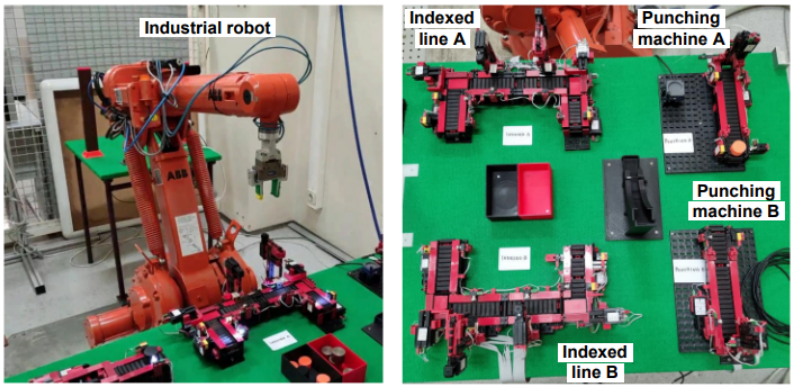

Figure 4.2: View of the small-scale production system demonstrator.

As shown in Figure 4.2, the demonstrator consists of several assets, namely two punching machines, two indexed lines, and a transport robot. These assets realize different functionalities that emulate real machines' operation on a small-scale model. For example, they can provide capabilities such as transporting, punching, and drilling workpieces. The low-level logic control of these machines is implemented as IEC 61131-3 programs running in a PLC. On the other hand, the transport robot executes its operations using programs developed in an ABB proprietary programming language that runs in the robot controller. Furthermore, the demonstrator has an API interface that allows for the easy integration with the assets, such as checking a machine state, starting or stopping an operation, and configuring some conditions, such as a machine breakdown, using MQTT or HTTP/REST protocols.

The demonstrator can be easily configured to perform some production process (e.g., initially processing a workpiece in the punching machine and subsequently in the indexed line) based on a traditional solution that is able to perform the specified tasks and produce the expected volumes. However, the challenge is that when the order level is driven by high product customization and the system needs to quickly adapt to changing conditions, this kind of solution becomes inefficient and costly to reconfigure. In this context, the agent-based AAS approach is able to decentralize intelligence and perform collaborative

[2] https://www.fischertechnik.de/en/products/industry-and-universities

tasks to address these challenges, as well as realize the previously described scenarios. The goal to be achieved by applying agent-based AASs to the demonstrator is to show the versatility of the proposed approach in performing collaborative tasks using AASs Type 3.

4.2.2 Automotive Production Line

In addition to testing and evaluating the proposed approach in a laboratory case study, another objective of this thesis was to validate certain aspects of the proposed approach in a real-world context. To achieve this, the agent-based AAS approach was implemented and experimentally tested, considering an industrial automotive case study. This case study, part of the European openZDM[3] project, focuses on the final assembly stage, where an inspection gauge device is used to automatically verify the gap and flush in the assembled parts of the car (e.g., frame and doors), replacing the current operation performed using a manual gauge.

This inspection device, called G3F, uses a laser technology to perform the gap and flush measurements and needs to know the color of the car in order to adjust the laser parameters for an accurate measurement. In such situation, two kinds of assets are considered: i) the G3F asset, and ii) the car as a product being produced. Since the G3F needs to know the color of the car and the limits for the gap and flush measurements, the AASs of these assets should be of Type 3, allowing for the establishment of a collaboration for the exchange of information. In this context, the goal to be achieved by applying agent-based AASs to the system is to provide more proactivity in the dynamic adaptation of process and inspection parameters, where G3F devices and products autonomously collaborate with each other to achieve the implementation of I4.0-compliant Zero Defect Manufacturing (ZDM) strategies.

4.3 Implementation of the Agent-Based AAS Approach

Since for both case studies the agent-based AAS approach was developed using the same development platforms, this section describes the implementation process of an individual agent-based AAS using these development platforms, particularly to implement the main classes described in the information metamodel presented in Section 3.1.2.

[3] https://www.openzdm.eu/

4.3.1 Information Part

The AAS submodels were developed using the Eclipse AASX Package Explorer[4] tool V3.0, a C#-based viewer and editor for the development of AASs according to the information metamodel of the AAS [27]. This tool enables the export of AASs in various file formats, including XML, JSON, and AASX[5]. This allows for cross-company data exchange through file exchange methods and also offers the option to host these files on a server that supports a standardized API interface for accessing the information within these file-based AASs. So far, it is one of the best applications for generating AAS with all its elements, since it is very easy for non-programmer users to structure all asset information into submodels through a user-friendly graphical interface. Figure 4.3 shows the graphical interface of the Eclipse AASX Package Explorer tool, presenting some of the submodels instantiated for one of the case studies.

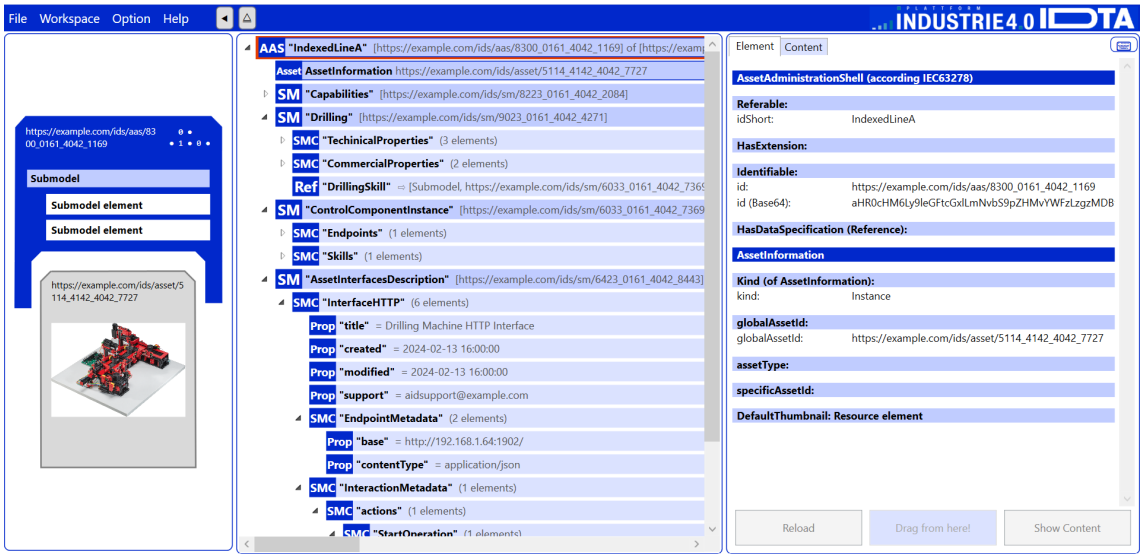

Figure 4.3: Application of the Eclipse AASX Package Explorer tool to develop the information part.

4.3.2 Intelligent Part

The agents (intelligent part) were implemented using the Java Agent DEvelopment framework (JADE)[6] version 4.5, which implements an efficient agent platform and supports the development of MAS that complies with the FIPA specifications. Regarding its features, JADE can be easily deployed on different computational platforms, providing advanced mechanisms to support the development of MAS for general applications, namely agent communication mechanisms, a library of FIPA interaction protocols ready to be used, a

[4] https://github.com/eclipse-aaspe/package-explorer
[5] .aasx file is a a common data format for AAS.
[6] https://jade.tilab.com/

graphical user interface to manage several agents, and a set of useful tools to debug the developed agents.

JADE uses a platform model defined by FIPA that provides the infrastructure in which agents can be deployed, allowing the existence, operation, and management of agents. This model includes the Agent Management System (AMS), the Directory Facilitator (DF), and the Agent Communication Channel (ACC). The AMS is responsible for supervising the platform and managing the lifecycle of agents. The DF provides the yellow pages service, where agents can publish the services they provide and find other agents providing the services they need. The ACC manages communication between the agents.

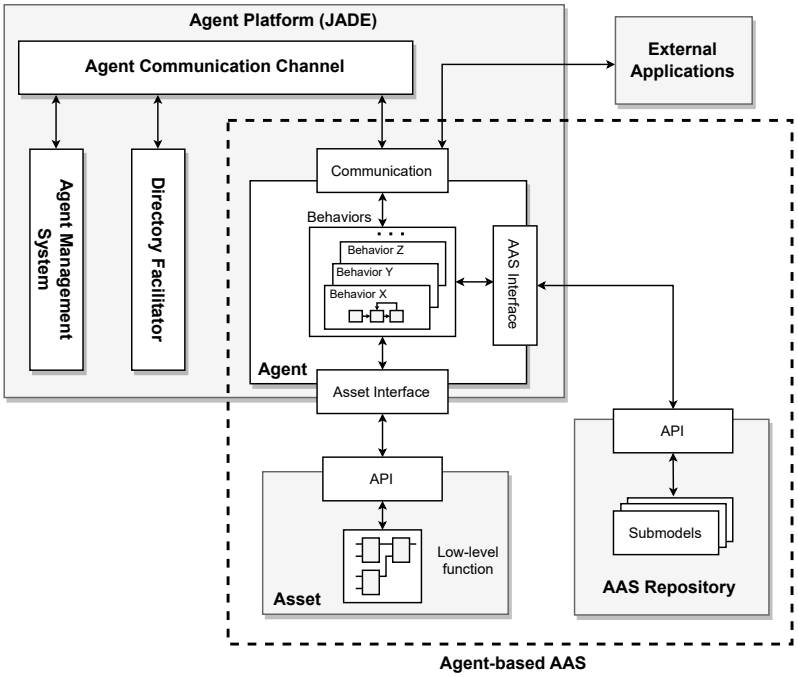

Figure 4.4: JADE agent platform to manage the intelligent part of an agent-based AAS.

Figure 4.4 illustrates the intelligent part (agent) of the agent-based AAS approach deployed in the JADE agent platform. In this context, JADE is responsible for providing all software infrastructure to manage the agents, while the information part (submodels) and the asset have their own management platforms.

In this work, the agents (intelligent part) are Java classes that extend the `Agent` class provided by the JADE, inheriting basic functionalities such as registration services, remote management, and sending and receiving ACL messages. As previously described in Section 3.1.2, an agent is comprised of four classes, namely *Behavior*, *Communication*, *AASInterface*, and *AssetInterface*. Each of these classes were implemented using the functionalities provided by JADE, namely the Behavior classes (e.g., `CyclicBehaviour`, `OneShotBehaviour`, `TickerBehaviour`, `WakerBehaviour`, and `FSMBehaviour`) and the library of ready-to-use FIPA interaction protocols (e.g., `AchieveREInitiator`, `AchieveREResponder`, `ContractNetInitiator`, and `ContractNetResponder`), as well as non-JADE

libraries to support the use of communication protocols, e.g., MQTT (Eclipse Paho) and HTTP (Unirest).

Even though it is important to use the VDI/VDE 2193 standard to create communication structures in compliance with I4.0, this work used the FIPA ACL for message exchange instead, because JADE provides specific classes for using FIPA message structures. This option does not invalidate the proof of concept, as it only reflects the structure and elements notation of the messages, not the collaboration and operational aspects of the proposed approach. Solutions requiring the VDI/VDE 2193-1 message structure can easily adapt this structure. For example, Table 3.2 relates the parameters of a message specified in the FIPA ACL Message Structure with the message elements defined in VDI/VDE 2193-1, showing the equivalence between both.

4.3.3 API-Enabled Approach

Eclipse AASX Server[7] V3.0 complements the use of Eclipse AASX Package Explorer by providing a local service to host and serve AASX packages, enabling access to the asset information through the HTTP/REST, OPC UA, and MQTT protocols. The Eclipse AASX Server adheres to the AAS specification part 2 (version 3.0), and therefore it can perform the operations described in [107]. This specification provides only guidelines for a HTTP/REST API, particularly describing the operations to enable access to the submodels information and their elements.

This API-enabled approach is crucial, since it provides standard interfaces for the agents to obtain information from their corresponding AASs using the HTTP/REST communication protocol. For example, an agent can invoke a function that requests specific asset information from the Eclipse AASX Server, as depicted in the following code snippet:

```
protected JSONObject serverRequest(String serverAdress, String endpoint) {
  try {
    HttpResponse<String> response;
    // GET request to obtain a JSON representing a particular submodel
    response = Unirest.get(serverAdress + endpoint).asString();
    jsonObject = new JSONObject(response.getBody());
  } catch (UnirestException e) {
    e.printStackTrace();
  }
  return jsonObject;
}
```

This function returns a `jsonObject` that needs to be interpreted via programming, i.e., by other specific functions that extract information from a JSON object representing a particular submodel. Although these functions require customization to align with each

[7] https://github.com/eclipse-aaspe/server?tab=readme-ov-file

submodel, the standardized submodel structure promotes code reusability, as a function developed to get information from a specific submodel can be reused by all agent-based AASs that have the same submodel.

Below is an example of a simplified function that handles information about the *ProductManufacturingPlan* submodel, described in Section 3.2.3, which is reused by the agent-based AASs implemented in this work. This code snippet aims to provide the agent-based AAS with knowledge about the steps of a product's production process. As a result, the agent-based AAS can effectively represent the product during the production process, managing, making decisions, and interacting with other agent-based AASs to complete the process.

```
private void submodelProductManufacturingPlan(jsonObject obj) {
  JsonNode rootNode;
  rootNode = parseJSON(obj);
  JsonNode submodelElementListOfSteps = rootNode.get("submodelElements").get(1).get("
      value");

  for (JsonNode element : submodelElementListOfSteps) {
    String assignedResourceId;
    String requirements;
    JsonNode valueArray = element.get("value");
    for (JsonNode value : valueArray) {
      if (value.get("idShort").asText().equals("AssignedResourceId")) {
        assignedResourceId = value.get("value").asText();
      }
      if (value.get("idShort").asText().equals("Requirements")) {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
          requirements =
          objectMapper.writeValueAsString(value.get("value"));
        } catch (JsonProcessingException e) {
          e.printStackTrace();
        }
      }
      // continue if condition
    }
    // simplified version - object Steps have more parameters
    listOfSteps.add(new Steps(assignedResourceId, requirements, ...));
  }
}
```

Another example is a simplified function that handles information about the *AssetInterfacesDescription* submodel, described in Section 3.2.2. The code snippet below aims to establish an interface with the asset, enabling data collection based on the submodel description, such as via the MQTT or HTTP protocol.

```
private void submodelAssetInterfacesDescription(jsonObject obj) {
  rootNode = parseJSON(obj);
  // MQTT interface
  if (rootNode.has("InterfaceMQTT")) {
    String broker;
    broker = rootNode.get("InterfaceMQTT")
                     .get("EndpointMetadata")
```

```
                    .get("base")
                    .asText();
    mqtt = new MQTT(2, broker, this.getLocalName());
    mqtt.mqttConnection();
    JsonNode allTopics = rootNode.get("InterfaceMQTT")
                                 .get("InteractionMetadata")
                                 .get("properties");
    // subscription to receive data from the asset
    for (JsonNode eachTopic : allTopics) {
      String topic = eachTopic.get("forms").get("href").asText();
      mqtt.subscribe(topic);
    }
    // ...
  }
  // HTTP interface
  if (rootNode.has("InterfaceHTTP")) {
    // procedures for HTTP interface
  }
  // other interfaces...
}
```

4.3.4 Asset Integration

This section describes the integration patterns between the agent-based AASs and the assets for the two case studies using the IEEE 2660.1-2020 standard (see Section 3.4).

Since the small-scale production system case study offers two interface options, HTTP and MQTT, for interacting with assets, the IEEE 2660.1 recommendation method[8] can be used to determine the best of the two interface options. A key requirement for using the recommendation method is to have available scoring agent practices in a repository. This study involved a repository containing 23 interface practices that experts assessed (repository data provided by [65]).

In this context, the application domain is "factory automation", the asset cannot host agents locally, and the functionality provided by the agent is the "control" of the asset, e.g., to start or stop an operation and adjust the production parameters that need to be changed according to the specifications of the product. The most important considerations in this configuration are "response time" and "scalability", each given a 50% weight. Figure 4.5 shows the results obtained from the IEEE 2660.1 recommendation method. It indicates that the best practices for the selected criteria are the "HT-7" using OPC UA and "HT-3" using HTTP (score 3.2 out of 5). Despite the OPC UA interface (HT-7) being

[8]The IEEE 2660.1 recommendation method recommends the best interface practices to interconnect agents and physical assets, particularly low-level automation devices, based on user-selected criteria. These criteria include functions (e.g., control, monitoring, or simulation), application domains (e.g., factory automation, building automation, or energy and power systems), technological constraints (e.g., the ability to host agents in the asset controller), as well as the relevant characteristics for the application scenario (e.g., response time, scalability, and reusability). For more details see Section 3.4.

highly recommended for industrial environments, it was not selected because it is not available in the demonstrator setup. Additionally, the available MQTT interface (HL-1) was excluded due to its poor performance based on the defined criteria (score 1.44 out of 5). Therefore, this work adopts the interface using HTTP (HT-3).

Results

Recomended interface practice

Id practice	Location	Interaction mode	API client	Channel	Broker	Score
HT-7	Hybrid	Tightly coupled	Java	OPC UA		3.20

Details

Id practice	Location	Interaction mode	API client	Channel	Broker	Score
HT-7	Hybrid	Tightly coupled	Java	OPC UA		3.20
HT-3	Hybrid	Tightly coupled	Java	HTTP		3.20
HT-5	Hybrid	Tightly coupled	Apache Milo	OPC-UA		2.80
HT-6	Hybrid	Tightly coupled	Java	Ethernet/IP		2.00
HT-4	Hybrid	Tightly coupled	C/C++/SQL	Sockets		1.92
HT-2	Hybrid	Tightly coupled	Java	Sockets		1.92
HT-1	Hybrid	Tightly coupled	Java	Modbus		1.62
HL-3	Hybrid	Loosely coupled	C/C++/SQL	Sockets	DBMS	1.44
HL-1	Hybrid	Loosely coupled	Apache Paho	MQTT	Eclipse Mosquitto	1.44
HL-2	Hybrid	Loosely coupled	Java	OPC-UA		1.36
HL-4	Hybrid	Loosely coupled	Apache Paho	MQTT		0.48

Figure 4.5: Recommended interface practices for the small-scale production system.

In the context of the automotive production line case study, the asset (G3F) can only be interfaced through the MQTT protocol. Consequently, this case study does not require the IEEE 2660.1 recommendation method. The integration pattern for this asset is hybrid and loosely coupled (see Figure 3.12), with the agent hosted separately from the asset and based on a publish-subscribe schema. Even though this integration pattern may not be the ideal choice, it is necessary because the asset only supports this technology.

For both case studies, the information on how the agent-based AAS should interface with the asset and its functionalities/services is described in submodels (e.g., *AssetInterfacesDescription* submodel). With this information, the agent-based AAS can interact automatically with the asset to manage its skills.

4.4 Deployment of the Proposed Approach in the Small-Scale Production System

This section presents the deployment of the agent-based AAS approach for the small-scale production system demonstrator described in Section 4.2.1. As shown in Figure 4.6, for each asset in the demonstrator, namely the industrial robot, indexed lines, punching machines, and products, an agent-based AAS was designed and implemented.

Although the small-scale production system demonstrator enables several experimental tests, it is essential to increase the number of assets in the demonstrator to enable the

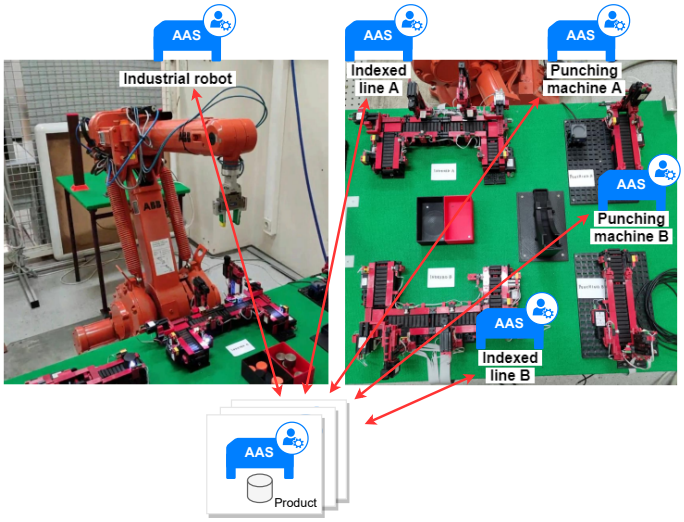

Figure 4.6: Conceptualization of the proposed approach applied in the demonstrator.

creation of a more dynamic and complex environment that requires more significant interactions between the assets. In this sense, the experimental plan for this work is similar to the one adopted in [61]. The aim is to use an extended version of the current demonstrator setup through an emulation platform that abstracts real physical assets. This emulation platform enables replicating each asset and all its functionalities and interfaces (see Figure 4.7). From the API's perspective, it is indistinguishable whether it is connected to the emulation platform or the real system. This approach allows for various experiments, such as trial and error tests, and scales the system regarding the quantity of assets in a flexible and safe virtual world. The extended version of the demonstrator includes three transport robots, three punching machines, three indexed lines, and n products.

The testing process for each scenario outlined in Section 4.1.1 involves launching several products in the system at specific intervals. The interaction between the products and resources follows the behavior models and interaction protocols defined for each scenario in the subsequent sections. Behavior models can be modelled using different formalisms, e.g., Unified Modeling Language (UML)[9] or Petri nets [122]. In this sense, this work adopts UML diagrams.

When dealing with decentralized systems with multiple entities, analyzing the success of the interactions of each entity is a complex task. In this work, the success of an experimental test considers the completion of all products' manufacturing processes within a reasonable time as an indicator of successful interactions. For example, if n products are launched into the system with an expected time t for completion, all n products should ideally be finished around t time. This metric is not intended for qualitative assessment but rather indicates that the test was conducted correctly.

[9] https://www.uml.org/)

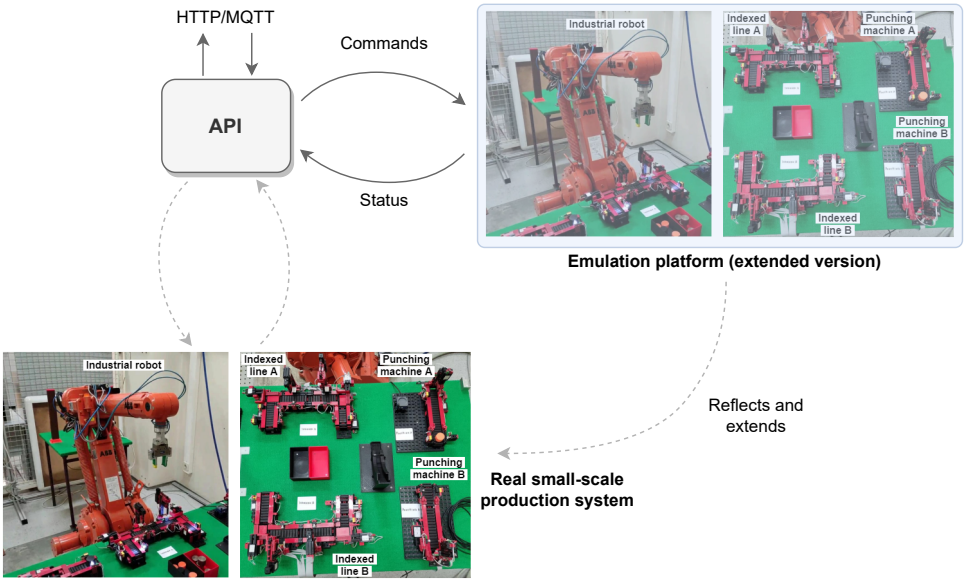

Figure 4.7: Real system and emulation platform for experimental tests (adapted from [61]).

4.4.1 Operation Execution Scenario

This scenario (see Section 4.1.1) focuses on the coordination between products and resources to meet the production requirements following a predefined production plan. In this regard, the products must have knowledge about their own manufacturing process to interact with several resources to complete its productions process. On the other hand, the resources must have the ability to manage its skills, e.g., start or stop an operation, or adapt some parameters according to the product requirements.

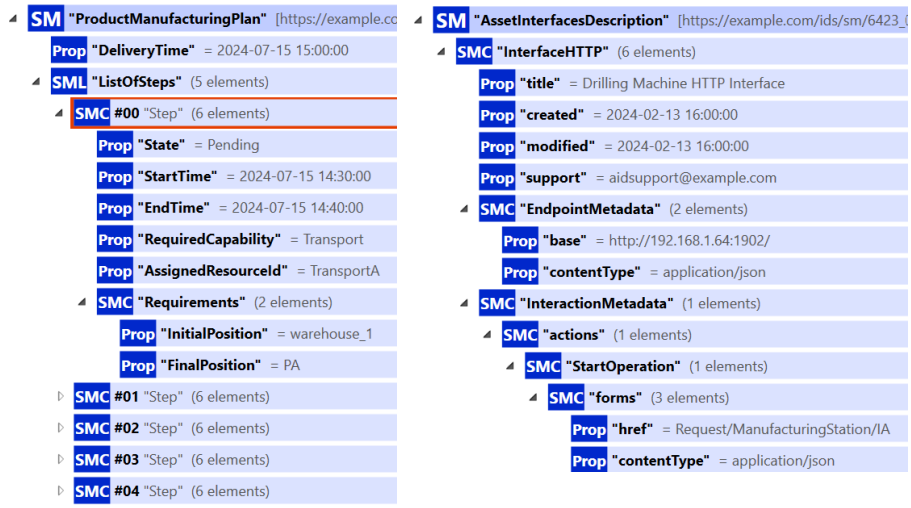

Figure 4.8: Short version of the developed *ProductManufacturingPlan* (left) and *AssetInterfacesDescription* (right) submodels.

In this scenario, the defined production plan of a product comprises the following steps:

$$transport\ (load) \rightarrow punching\ machine \rightarrow transport \rightarrow indexed\ line \rightarrow transport\ (unload)$$

These steps are described and structured in the *ProductManufacturingPlan* submodel, as illustrated in Figure 4.8 (left). Among the properties of each step in *ListOfSteps*, *Requirements* provides the parameters to inform the resource that will perform the required operation. For example, the target positions are informed for the transport robot, and for the punching machine and indexed line, parameters related to the punching and drilling functions are informed.

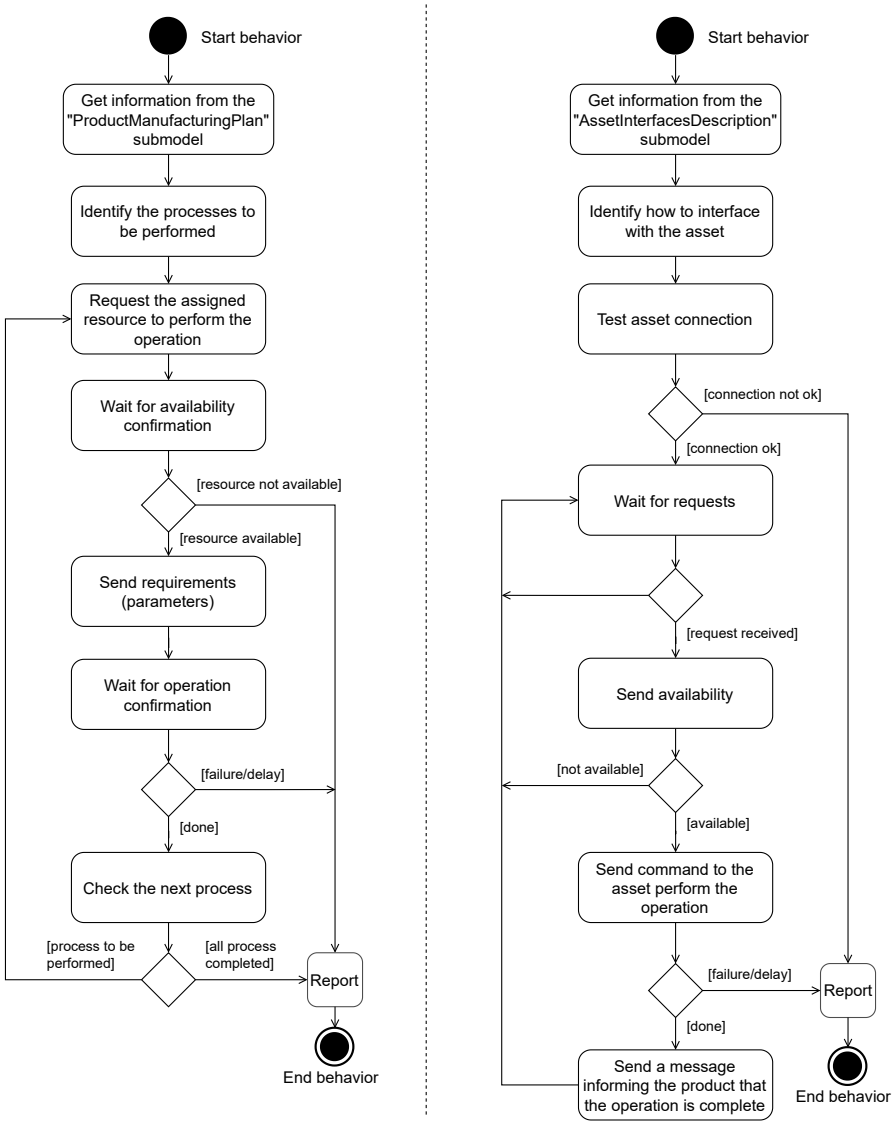

Figure 4.9: UML diagram of the operation execution behavior of an agent-based AAS representing a product (left) and a resource (right).

Regarding the interface with the asset to manage its skills, Figure 4.8 (right) illustrates the *AssetInterfacesDescription* submodel, where information about how to interface with the asset is provided. In this case, the interface with an indexed line is performed via HTTP protocol, and one of the available services that starts the machine operation can be invoked through the endpoint `Request/ManufacturingStation/IA`.

Figure 4.9 illustrates the UML activity diagram describing the behavior of the agent-based AASs for the operation execution scenario. Figure 4.9 (left) shows the behavior of an agent-based AAS representing a product, which focuses on getting the information contained in the *ProductManufacturingPlan* submodel to know all processes involved in the product production. As further presented in Section 3.2.3, this submodel describes crucial aspects involved in each production process, such as the planned start and end times, assigned resource, state of the process, and requirements for the process. Based on this information, the agent-based AAS can interact with the agent-based AASs that represent the resources, requesting operations with the required parameters during the appropriate time slot. This behavior is cyclic, which means that it ideally ends when all planned processes are completed.

On the other hand, Figure 4.9 (right) shows the behavior of an agent-based AAS representing a resource. This behavior, in the first moment, aims to interface with the asset based on the description provided in the *AssetInterfacesDescription* submodel (see Section 3.2.2). After that, the agent-based AAS waits for requests to execute the requested operation.

Some conditions, such as failure or lack of connection with the asset, can prevent the production process. In these instances, the agent-based AAS is designed to emit a warning message. However, the primary objective in this scenario is to make sure that the production process is successfully completed. As a result, no specific actions have been implemented to handle these conditions. A particular condition, such as failure, is addressed in the unexpected event scenario.

The exchange of information between the agent-based AASs followed the interaction pattern for the operation execution described in Figure 4.10. In this interaction protocol, the products act as the "Initiator", requesting operations from the resources, which act as the "Participant". The resources receive the request along with the process parameters and then decide whether to accept or refuse to perform the operation by checking the feasibility of the requested parameters. If accepted, the resource adjusts its operational parameters and carries out the operation, providing the results at the end.

4.4.2 Autonomous Negotiation Scenario

This scenario focuses on the negotiation between products and resources in order to meet production requirements. In this context, products must know their own manufacturing requirements, and resources must know the capabilities they provide. This negotiation is based on a service-oriented principle, where the resources offer their capabilities as services, and the products can search for and request these services.

In this scenario, products launched on the factory floor must autonomously negotiate with the available resources based on their required capabilities:

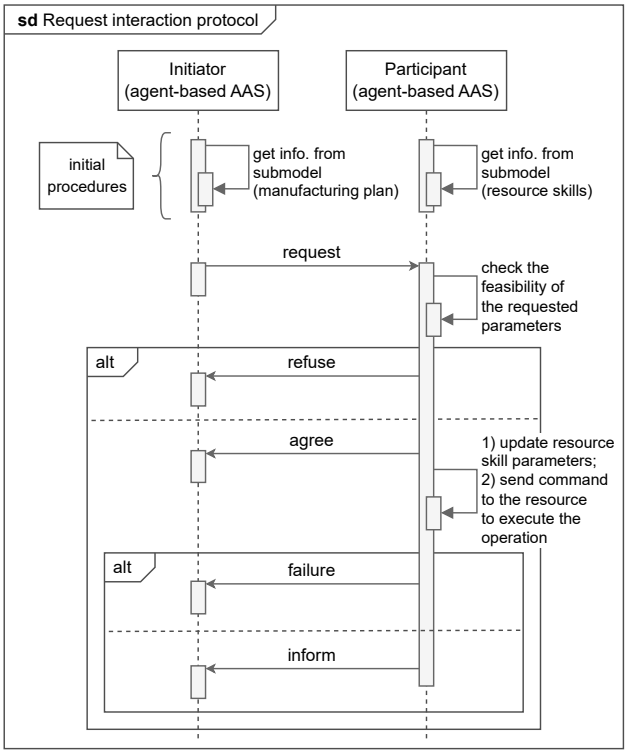

Figure 4.10: Interaction protocol for the operation execution scenario.

$$capability_1 \rightarrow capability_2 \rightarrow capability_3 \rightarrow ... \rightarrow capability_n$$

In this context, products possess knowledge about the required capabilities, described and structured in the *ProductRequirements* submodel, as shown in Figure 4.11 (left). For example, a product may require a drilling capability with specific requirements such as depth and diameter.

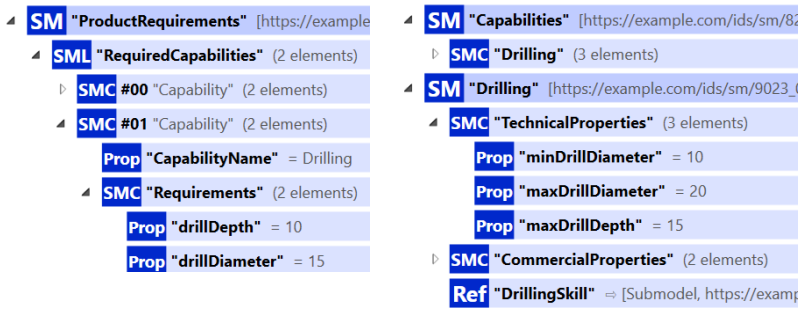

Figure 4.11: Short version of the developed *ProductRequirements* (left), *Capabilities* (right), and *Drilling* (right) submodels.

On the other hand, resources have a set of capabilities they can provide, described in the *Capabilities* submodel. Each element of this set refers to a specific submodel that provides a detailed description of the capability. As shown in Figure 4.11 (right), the *Capabilities* submodel presents only one capability (drilling) that refers to a specific *Drilling* submodel, which describes the technical and commercial properties of the drilling capability.

Figure 4.12 depicts the UML activity diagram that describes the behavior of the agent-based AASs in a negotiation scenario. Figure 4.12 (left) presents the behavior of the agent-based AAS representing a product, which focuses on acquiring information from the *ProductRequirements* submodel to understand the required capabilities for the product production process. Within this scenario, the product must engage in negotiations with the resources providing the required services, such as punching and drilling, as well as the transport between resources and warehouses (loading and unloading). The decision-making process for selecting the best proposal can be facilitated by decision-making algorithms, e.g., based on heuristic functions that consider the proposed price, the location of the resource, the proposed due date, and the confidence about the service provider (e.g., as proposed in [61]). In this work, the decision is based on choosing the cheapest price to facilitate the proof of concept.

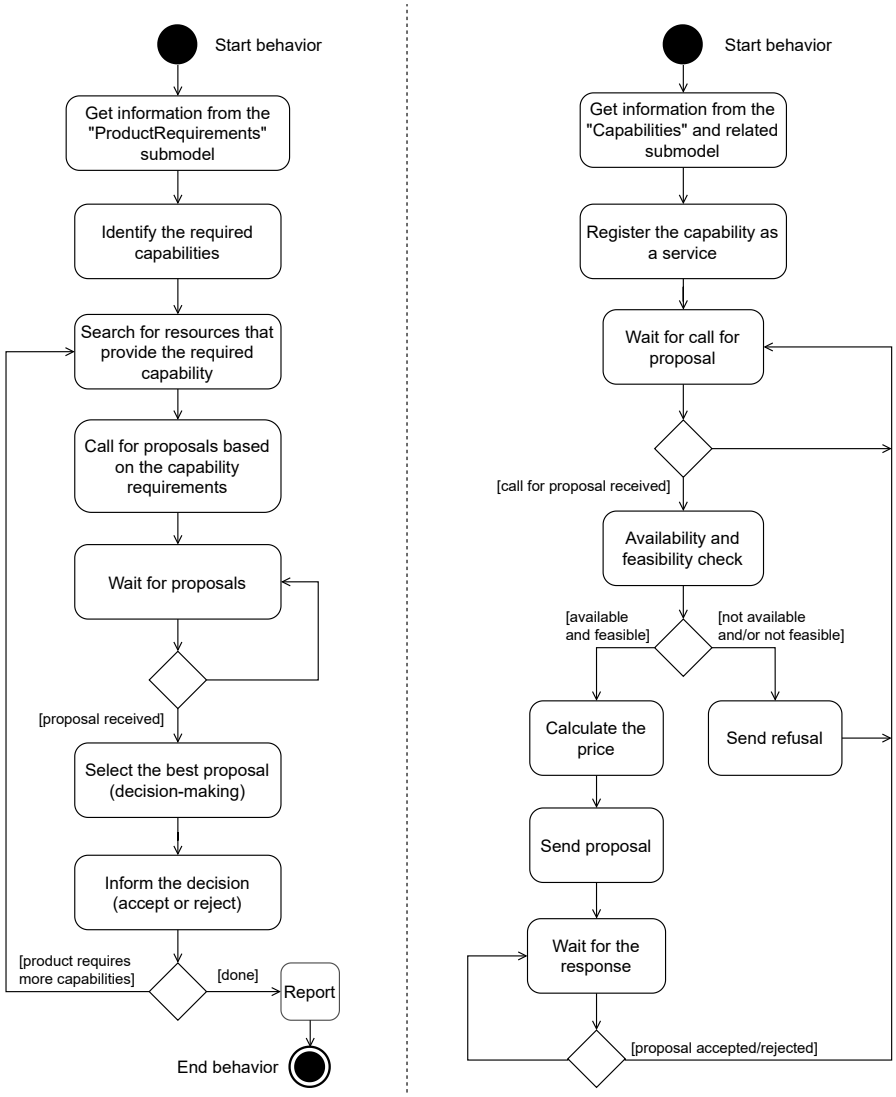

Figure 4.12: UML diagram of the negotiation behavior of an agent-based AAS representing a product (left) and a resource (right).

On the other hand, Figure 4.12 (right) shows the behavior of the agent-based AAS representing a resource. The behavior begins with the agent acquiring knowledge about the capability of its associated asset and registering this capability as a service using the Yellow Pages service implemented by JADE. The next step involves waiting for the call for proposals when products requiring the capabilities initiate negotiation by sending their production requirements. At this point, the resource checks the availability of the resource and uses an algorithm to evaluate if the requirements match the technical specifications of the resource. This algorithm examines each requirement to ensure the resource can meet it. Based on this approach, the resource may choose to reject or continue the negotiation. If the resource agrees, it calculates the price and sends an offer.

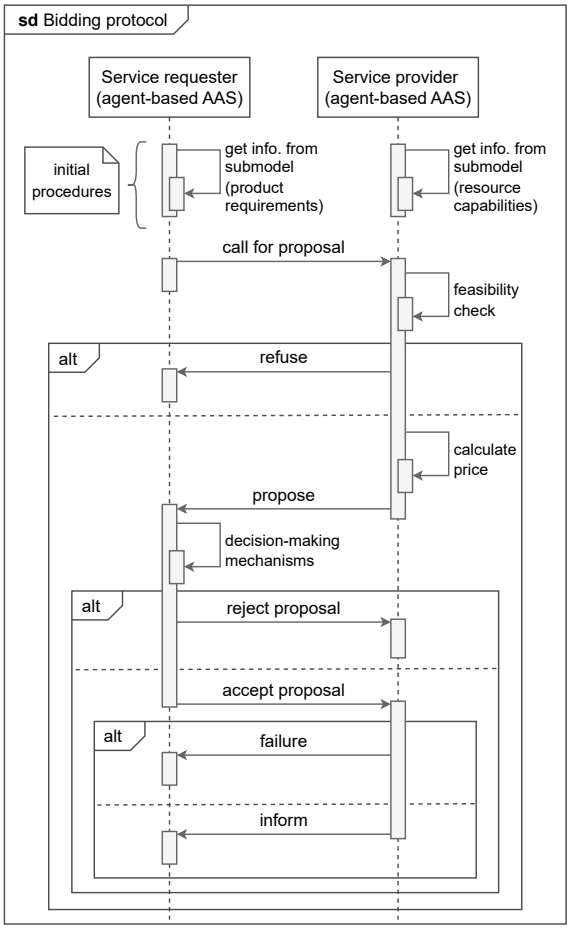

Figure 4.13: Interaction protocol for the autonomous negotiation scenario.

When calculating the price, various factors need to be considered, such as setup time and costs, expenses related to acquiring new tools, operational costs, resource investments, and the level of assigned work. This calculation process should be tailored to the specific interests of the company and is, therefore, customized for each case. Nevertheless, the dynamic bid price calculation presented in [61] is used as an example to illustrate this process in this work, which considers fixed costs and the profit margin adapted according to the market laws.

The exchange of information between the agent-based AASs followed the interaction pattern for the negotiation process described in Figure 4.13. In this interaction protocol, the products act as the "Service requester", initiating the negotiation process with the resources acting as the "Service provider". When the resources receive the call for proposal, they can analyze and decide to continue the negotiation by offering proposals for the products. In this case, the products analyze the received proposals and decide on the best one.

4.4.3 Unexpected Event Scenario

This scenario illustrates how a production system handles unexpected events, e.g., a machine malfunction that disrupts its normal operation. The focus is on the moment when the unexpected event occurs and how the agent-based AASs interact with one another to address this issue and ensure the completion of the production process.

Similar to the operation execution scenario, the system's initial state can be represented by the conditions outlined in the operation execution scenario. This includes the requirement for products to understand their manufacturing process in order to interact with various resources and for resources to have the capability to interface with the asset to manage their skills.

Just like in the operation execution scenario, the production plan for the products consists of the following steps:

$$transport\ (load) \rightarrow punching\ machine \rightarrow transport \rightarrow indexed\ line \rightarrow transport\ (unload).$$

These steps are defined and organized in the *ProductManufacturingPlan* submodel, as depicted in Figure 4.8 (left). However, at a specific time, one of the assigned resources for carrying out one of these operations experiences a failure, necessitating the products to reschedule their production plans in order to complete the production process.

Figure 4.14 depicts the UML activity diagram that describes the behavior of the agent-based AASs in an unexpected event scenario. Figure 4.14 (right) shows the monitoring behavior of an agent-based AAS representing a resource, which focuses on monitoring the health condition of its asset along its operation lifecycle. The behavior can utilizes various methods, from simple rule-based approaches to advanced AI algorithms, to monitor the asset. For instance, this work adopts an approach based on the Nelson rules [121] to determine whether any measured variable is out of control. By employing this approach, the agent-based AAS can identify and prevent problems and make decisions to stop its operations in order to avoid downtimes or potential issues that could compromise asset integrity. In such a situation, the agent-based AAS notifies all assets with planned operations about canceling those operations.

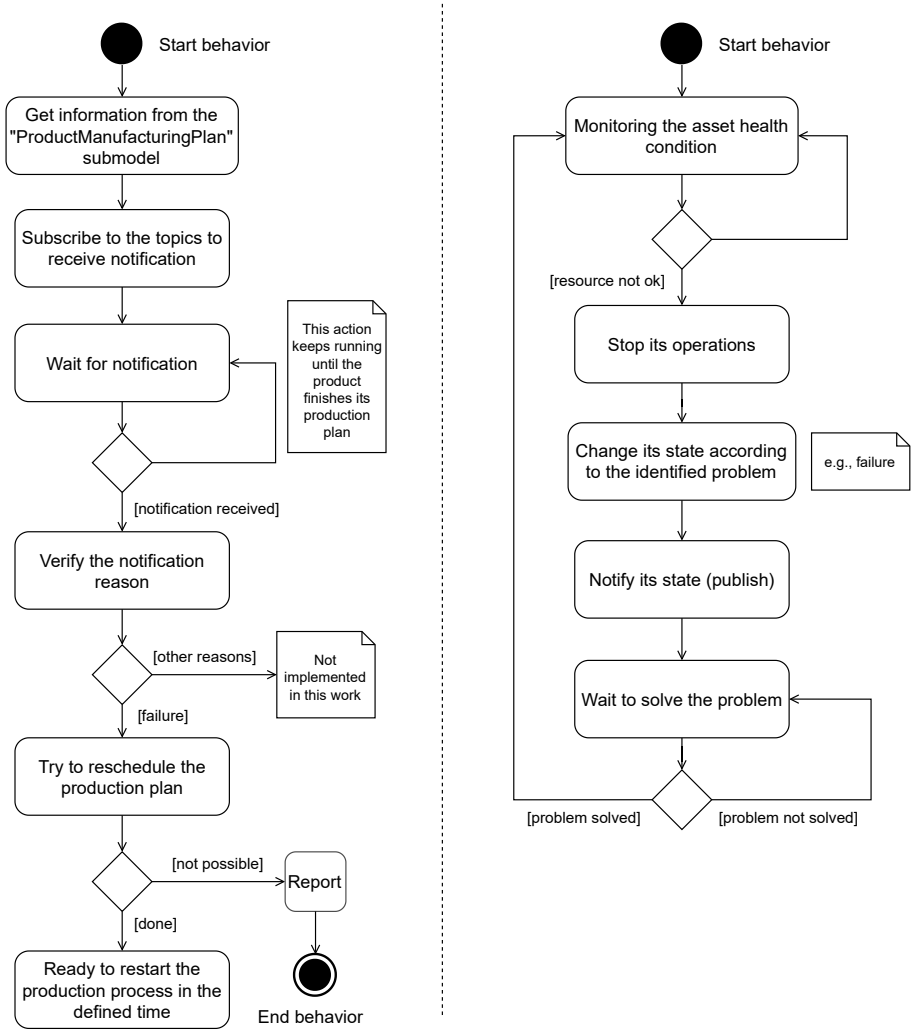

Figure 4.14: UML diagram of the unexpected event behavior of an agent-based AAS representing a product (left) and a resource (right).

Figure 4.14 (left) presents the behavior of the agent-based AAS representing a product. This behavior focuses on waiting for notifications to make decisions based on the received information. In this case, the agent-based AAS subscribes to specific topics where the information will be published. The topic is defined in the design phase and included in a submodel. As the notified information may cover a wide range of subjects, it is important to develop proper treatment and actions for each case. In this scenario, the subject involves problems related to failures. Therefore, specific actions are performed in those conditions. For instance, when a product is notified about a resource failure, the product needs to find another resource capable of performing the same operation. This process can be done by searching for services provided by the resources (similar to the negotiation scenario). If one or more resources are available to provide this service, the product can negotiate with the resources or request the operation from a specific one. After the product reschedules its production plan, the process can continue as if it were an operation execution scenario.

The exchange of information between the agent-based AASs followed the interaction pattern for the unexpected event described in Figure 4.15. This interaction is based on the publish-subscribe schema, where the exchange of messages is topic-based and managed by a broker, which facilitates the communication between an entity that needs to communicate the same information to various others. In this interaction protocol, the products assume the role of "Subscriber" and the resources assume the role of "Publisher". The "Subscriber" subscribes to specific topics to receive notification from the "Publisher(s)" when some unexpected event occurs.

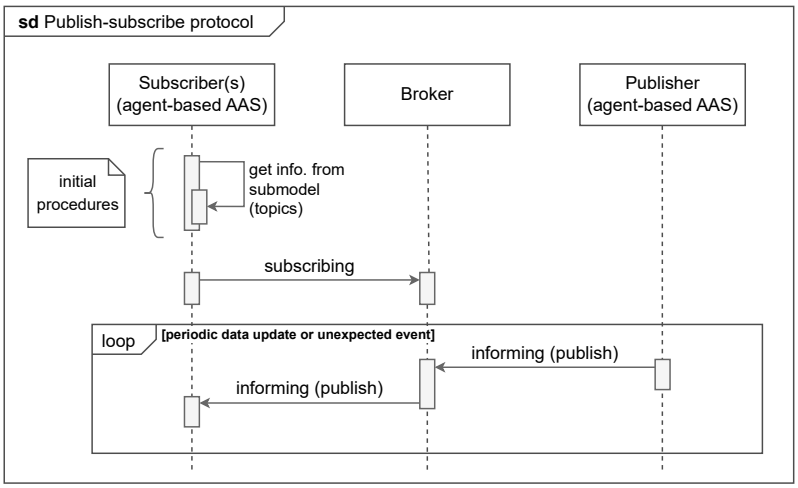

Figure 4.15: Interaction protocol for the unexpected event scenario.

Note that the protocol depicted in Figure 4.15 specifically outlines the interaction for event notification. Additionally, the protocols shown in Figures 4.10 and 4.13 can be used to facilitate the subsequent processes following the event notification. For example, these protocols support activities such as searching and negotiating with resources, as well as requesting the operation of a resource.

4.5 Deployment of the Proposed Approach in the Automotive Production Line

This section describes how the agent-based AAS approach was designed and implemented in the automotive production line as outlined in Section 4.2.2. Even though it is more straightforward and limited in test scenarios compared to the previous case study based on a laboratory prototype, this example allows for evaluating the practicality of the proposed approach in an industrial environment. In this case study, only the operation execution scenario was implemented, as it aligns with the actual production line operation and requirements. Implementing the other scenarios would require impractical modifications in the production line's current operation.

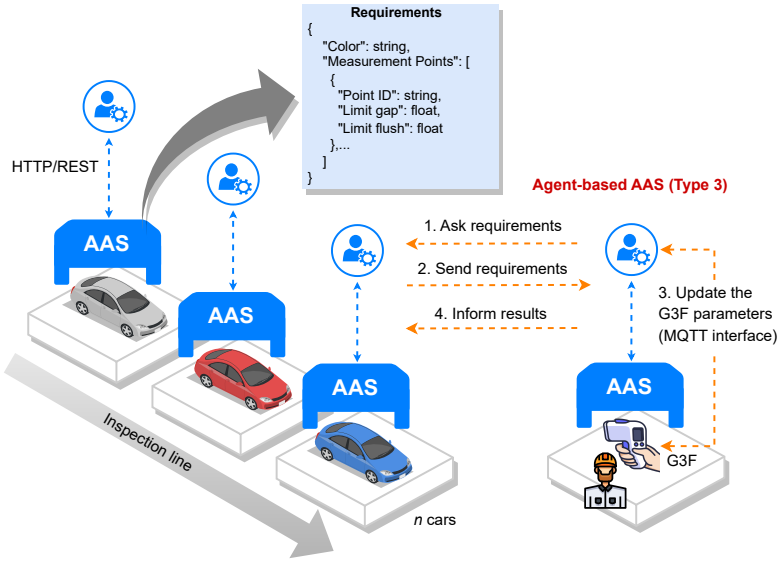

Figure 4.16: Implementation of the agent-based AAS approach in a production line.

As shown in Figure 4.16, an agent-based AAS was designed and implemented for each asset in the production line, namely the G3F and products (cars as products being produced). The implementation of the agent-based AAS was slightly similar to the operation execution scenario of the first case study. While there are minimal differences, these can be observed in the behavior, interaction protocol, and asset interface, which form the dynamic aspects of the agent-based AAS and are adjusted based on the system and asset needs. However, the submodel's structure remains unchanged as it adheres to a standardized structure.

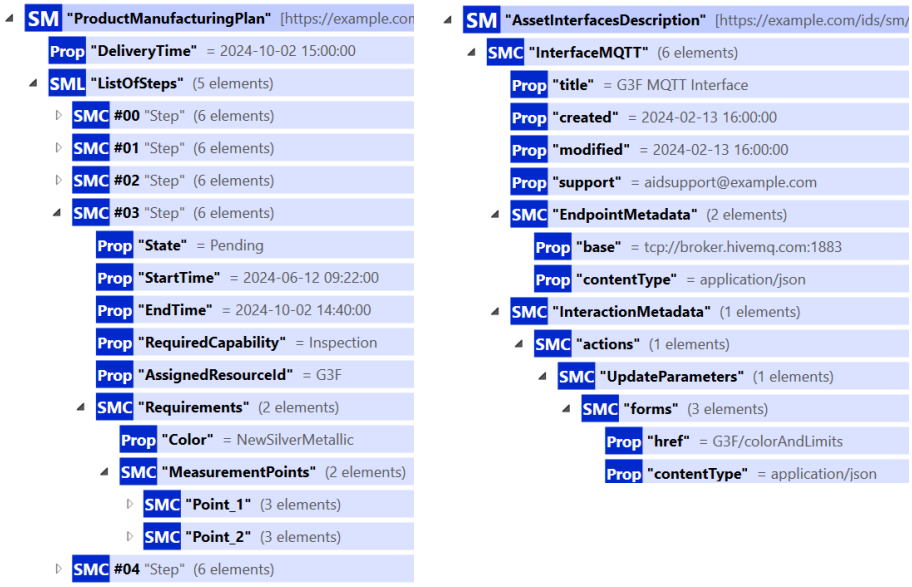

Figure 4.17: Short version of the developed *ProductManufacturingPlan* (left) and *AssetInterfacesDescription* (right) submodels for the automotive production line case study.

The testing process is identical to the one used in the previous case study. This means that products are being launched into the system at specific intervals. The product and resource interaction follows the behavior models and interaction protocols defined below.

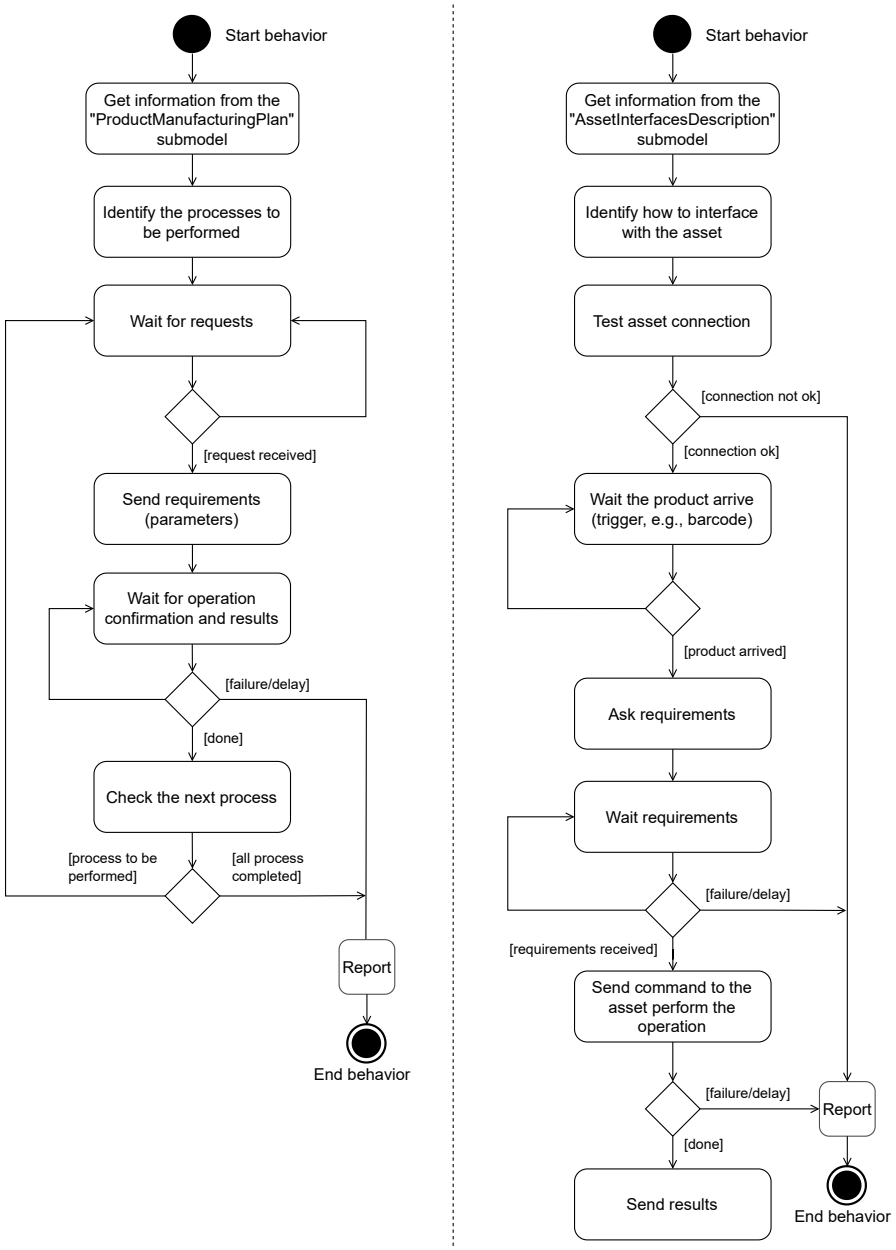

Figure 4.18: UML diagram of the operation execution behavior of an agent-based AAS representing a product (left) and a G3F (right).

The product has a *ProductManufacturingPlan* submodel (see Figure 4.17) that contains information about all the steps required for its manufacturing process, and in this case, there is a specific step for the inspection operation. This step contains information about the planned time, the step state, the assigned G3F to perform the inspection operation, and the requirements, i.e., the process parameters to be informed to the G3F, namely color and measurement points (gap and flush limits). On the other hand, the G3F has

specific submodels, namely the *AssetInterfacesDescription* (see Figure 4.17), which provide information about how to interface with the G3F and adjust its parameters. In this case, the interface is performed by using the MQTT protocol, where there is a defined publish topic to send the parameters to the G3F following a JSON content type.

The UML diagram in Figure 4.18 depicts the behavior of the agent-based AAS for the operation execution scenario of the automotive production line case study. A comparison with the operation execution scenario from the first case study (see Figure 4.9) reveals a difference in who initiates the interaction process. In the previous case study, the product initiated the interaction with the resources. However, in this case, the G3F must initiate the interaction by requesting the product requirements. This change is necessary due to the conditions imposed by the production line. The objective is to enhance the current process by autonomously and dynamically adjusting the process and inspection parameters, avoiding generating significant changes in the way the factory's current system works.

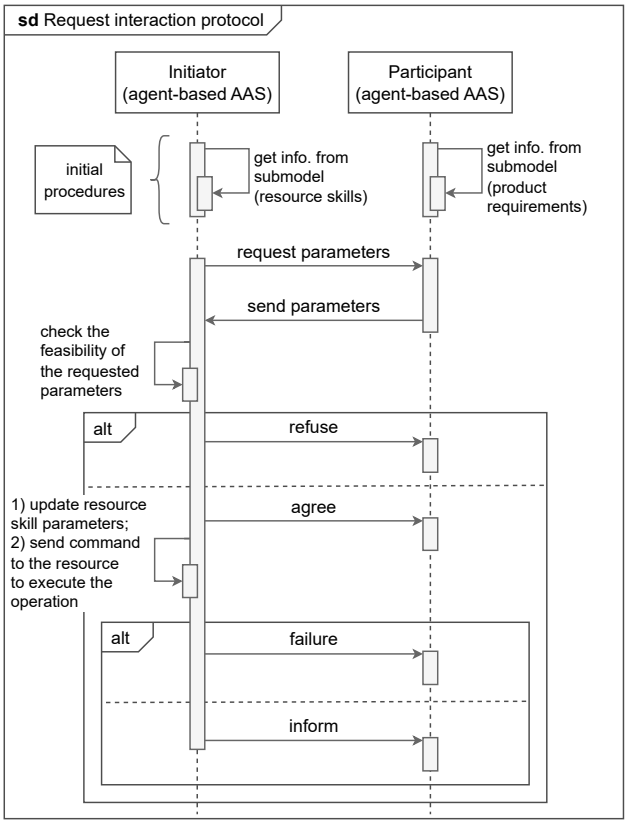

Figure 4.19: Interaction protocol for the automotive production line case study.

The exchange of information between the agent-based AASs followed the interaction pattern for the operation execution described in Figure 4.19. The G3F agent assumes the role of the "Initiator", asking each product agent (role of "Participant") for the process parameter information, i.e., color and measurement points. The product agent provides the requested information to the G3F agent, which interfaces with the G3F device to send

and receive data using MQTT, e.g., to inform the process parameters. After performing measurements, the G3F agent informs the product agent about the measurement results.

4.6 Analysis of Experimental Results and Discussions

As defined in Section 4.1.2, the assessment criteria for the agent-based AAS approach comprises a qualitative analysis involving five requirements: adaptability, pluggability, robustness, interoperability, and (re)usability. These are essential requirements to consider when it comes to developing I4.0-compliant solutions. However, it is important to emphasize that the proposed approach primarily focuses on designing and implementing an AAS Type 3. Therefore, the initial analysis should determine if the proposed approach fulfills the requirements that define an AAS as Type 3, which include autonomy, intelligence, and collaboration, as discussed in Section 2.3.1. Afterward, the remaining I4.0 requirements will be evaluated. This qualitative assessment aims to be impartial and grounded in factual observations gathered throughout the design, implementation, and test stages of the proposed approach in the two case studies.

4.6.1 AAS Type 3 Minimum Requirements

This section outlines the minimum requirements that define the implemented agent-based AAS approach as an AAS Type 3 solution. These requirements include autonomy, intelligence, and collaboration, as discussed in Section 2.3.1. The level of each requirement can vary depending on the application goals and is highly influenced by the adopted algorithm/technique. It is evident that incorporating advanced AI techniques, such as ML, DL, and other sophisticated algorithms, can yield a higher degree of autonomy, intelligence, and collaboration. However, this work does not focus on the algorithm itself. Instead, the emphasis is on designing components that meet these requirements rather than on the specifics of individual algorithms.

Given that the proposed approach is modular replacing algorithms is relatively straightforward. This allows different AI techniques to be implemented as needed to optimize autonomy in decision-making, improve intelligence in adapting to new contexts, and reinforce collaboration in dynamic environments.

In this section, the terms “agent-based AAS” and “asset” (which may include product and resource types) are sometimes used interchangeably. When referring to an asset performing a task, negotiating, collaborating, or making decisions, it is understood that these capabilities are only enabled by AAS Type 3. Within the scope of this work and its case studies, assets without AAS Type 3 remain passive entities, relying on external systems or manual intervention for instructions, and lack autonomy, intelligence, and

collaborative capabilities. A discussion about how the implemented agent-based AAS approach meets these minimum requirements is presented below.

Autonomy

Autonomy is defined as *"the ability to make your own decisions without being controlled by anyone else"* (Cambridge Dictionary). Autonomy does not mean absolute freedom to decide arbitrarily but rather the capability to make decisions based on predefined criteria. For instance, in the operation execution scenario, despite the agent-based AAS representing products following a predefined production plan, they independently make decisions and interact with resources to complete the plan without external intervention, ensuring that the system functions effectively while respecting predefined constraints. On the other hand, resources receive the product's requests and autonomously adjust their skill parameters to meet the product specifics.

Other examples include negotiation and unexpected event scenarios. In the negotiation scenario, assets in the system negotiate with each other based on predefined criteria to achieve both beneficial results. This negotiation process is autonomous since each asset makes decisions independently, following defined rules and protocols without external control.

In the unexpected event scenario, the system's autonomy is demonstrated by its ability to reschedule production plans in response to disturbances, such as equipment failures. In this sense, the assets can autonomously assess the situation and reallocate resources to mitigate the impact. This ability to respond to condition changes without human intervention demonstrates the system's capability to operate autonomously within the boundaries of its predefined criteria.

Intelligence

Intelligence is defined as *"the ability to learn and understand things quickly and easily"* (Cambridge Dictionary). In the proposed approach, intelligence can have different levels, e.g., ranging from basic rule-based mechanisms to advanced AI techniques. Basic rule-based mechanisms involve simple decision-making processes based on predefined rules and logic. On the other hand, advanced AI techniques include ML, DL, and other sophisticated algorithms that enable the system to learn from data, adapt to new situations, and improve over time. For instance, in the unexpected event scenario, the monitoring behavior of the agent-based AAS is based on a rule-based algorithm that analyses operational data from the asset to identify possible problems that can lead to downtimes or potential issues that could compromise the integrity of the asset.

Although this work does not address advanced AI techniques (e.g., ML algorithms and data analysis) to achieve intelligence, the intelligent behavior of the system emerges mainly from the interaction and collaboration among a group of distributed and autonomous entities (agent-based AASs) endowed with simple behaviors and/or limited intelligence. This collective intelligence provides an alternative approach to designing intelligent systems, where the traditional centralized pre-programmed control is replaced by a distributed functioning in which the interactions between individuals result in the emergence of intelligent global behavior.

Collaboration

Collaboration is defined as the capacity of *"involving two or more people or organizations working together for a particular purpose"* (Cambridge Dictionary). In this work, collaboration refers to AASs working together for a specific purpose. Unlike simple communication, collaboration goes beyond the exchange of information and comprises a deeper, interdependent process where all participants play crucial roles in achieving a common goal. In this regard, the implementation of the proposed approach in the different scenarios demonstrated different types of collaboration between the agent-based AASs.

In the operation execution scenario, assets collaborate by interacting in real-time to perform operations and adjust production parameters dynamically. This type of collaboration is critical for maintaining efficiency and responsiveness in dynamic production environments, which involves exchanging data on current production status and requirements and coordinating actions to ensure smooth operation without human intervention.

In the autonomous negotiation scenario, assets negotiate with each other to autonomously reach agreements or decisions. This form of collaboration is essential for optimizing resource utilization and involves proposing and evaluating different proposals and reaching a consensus or making decisions without direct human oversight.

In the unexpected event scenario, assets play a crucial role in responding to unforeseen circumstances or disruptions. This type of collaboration is essential for maintaining resilience and minimizing the impact of unexpected events on production, which involves detecting and recognizing unexpected events and coordinating responses and actions among multiple assets.

4.6.2 Industry 4.0 Requirements

Once the implemented solution is evaluated based on the minimum requirements that characterize an AAS Type 3 solution, it can be assessed from another perspective, focusing on its alignment with the fundamental requirements to realize I4.0. In this context, this section evaluates and discusses the implemented agent-based AAS approach based on

the previously defined I4.0 requirements: adaptability, pluggability, robustness, interoperability, and (re)usability. As described in Section 4.1.2, each requirement comprises several sub-requirements, which help clarify and evaluate specific aspects of each requirement. The requirement (level 1) provides a general definition, and the sub-requirements (level 1.x) provide further details that an approach/solution should consider, aiming to meet the requirement. In this sense, each requirement was evaluated, and a final score was assigned based on Equation 4.1, which quantitatively measures the level of each requirement fulfillment by the implemented solution.

Adaptability

Table 4.7 shows the assessment of the adaptability requirement, along with the assigned final score representing the level of requirement fulfillment by the implemented solution.

Table 4.7: Assessment for the adaptability requirement.

Requirement	**Requirement description**				
Adaptability 1	Adaptability refers to the ability of industrial systems to learn from and respond to changing conditions over time, improving their performance autonomously				
Sub-requirements	**Sub-requirement description** (The proposed solution should...)	**Score of fulfilment (β)**			
		N/A	**Low**	**Medium**	**High**
Adaptability 1.1	have the perception and action capabilities to effectively interact with and respond to different contexts or situations				✓
Adaptability 1.2	enable seamless collaboration among the system's components, allowing them to share information, make coordinated decisions, and work together to solve complex tasks				✓
Adaptability 1.3	be able to learn from new data and experiences to improve its performance and decision-making processes over time		✓		
Adaptability 1.4	be able to modify its behavior in response to new information or changing environments				✓
					Final score = 4.1 out of 5

The two case studies and scenarios show that the agent-based AAS can be developed to interact with and adapt to different contexts and situations. The agent-based AAS can interface with the asset to perceive the environment, e.g., through sensors, and to take action in the environment, e.g., through actuators. Furthermore, by utilizing standardized submodels, the agent-based AASs can acquire the necessary knowledge to interface with assets employing different communication protocols, particularly for managing their skills (**Adaptability 1.1**).

The agent-based AAS approach decentralizes intelligence through autonomous, intelligent, and collaborative components. These components can employ multiple communication protocols to facilitate different types of interactions with each other, allowing for a distributed problem-solving method. By leveraging the capabilities of autonomous agents that can act and make decisions independently, this approach enables more efficient handling of complex and dynamic environments. Each agent, with its specific knowledge and skills, contributes to the collective intelligence of the system, facilitating a cooperative effort toward achieving common goals (**Adaptability 1.2**).

The implemented agent-based AAS approach uses the submodels as knowledge representation. Each agent-based AAS can interpret the information from its respective asset's submodel to make decisions over time, which involves processing and using information effectively. Even though the current implementation does not involve advanced learning algorithms, the capability of agents to interpret and act upon information from the submodels demonstrates a level of decision-making ability. This modular approach also provides the flexibility to integrate advanced learning algorithms, potentially enhancing the system's decision-making capabilities (**Adaptability 1.3**).

As demonstrated in both case studies (Sections 4.4 and 4.5), different behaviors can be designed to allow the agent-based AAS to adapt to different contexts. For instance, specific behaviors were developed in each scenario to meet its unique requirements. Although these behaviors were tested in isolation within each scenario, they can function in parallel. This capability enables the agent-based AAS to be equipped with multiple behaviors, allowing it to adapt and manage various conditions simultaneously (**Adaptability 1.4**).

Pluggability

Table 4.8 shows the assessment of the pluggability requirement, along with the assigned final score representing the level of requirement fulfillment by the implemented solution.

Upon integrating an agent-based AAS into the system, it becomes open to automatic detection and configuration. Within this framework, agent-based AASs can readily identify and communicate with the newly incorporated component. For instance, JADE's Yellow Page services allow agent-based AASs to discover the services offered by other agent-based AASs (**Pluggability 1.1**). Furthermore, the submodels (related to production parameters) can be reconfigured during the operational phase based on production requirements (**Pluggability 1.2**). This submodel configuration process does not necessitate halting and restarting the system.

As demonstrated in the two case studies, the agent-based AAS approach involves encapsulating an asset and enabling seamless communication and collaboration to carry out specific tasks. In both case studies, agent-based AASs were created for different assets,

Table 4.8: Assessment for the pluggability requirement.

Requirement	**Requirement description**				
Pluggability 1	Pluggability refers to the ability of machines, devices, and components to automatically recognize and self-configure to work together seamlessly without the need for manual intervention or extensive setup				
Sub-requirements	**Sub-requirement description** (The proposed solution should...)	**Score of fulfilment (β)**			
		N/A	**Low**	**Medium**	**High**
Pluggability 1.1	be capable of automatically detecting and configuring connected devices, minimizing the need for manual setup and intervention				✓
Pluggability 1.2	be able to quickly reconfigure manufacturing systems to accommodate changes in product specifications or production requirements				✓
Pluggability 1.3	enable components from different manufacturers to seamlessly communicate and collaborate to perform complex tasks without requiring extensive customization or integration effort			✓	
Pluggability 1.4	be modular and scalable to easily incorporate additional features or modules as needed				✓
Pluggability 1.5	be able to perform the capability assessment. This involve interpreting the asset description and matching the production requirements to the asset capabilities		✓		
					Final score = 3.8 out of 5

and they could communicate seamlessly and perform collaborative tasks using the communication structures of the agent-based AAS approach. Effective communication can be easily achieved if the components adhere to a standardized communication structure. This work used FIPA ACL, but a more comprehensive approach would involve adopting the VDI/VDE 2193 - I4.0 language (**Pluggability 1.3**).

The agent-based AAS is structured in a way that different modules (behaviors, communication protocols, submodels) can be developed and integrated independently of each other. This modularity allows easier maintenance, updates, and customization to suit different requirements. Depending on the specific case study or scenario, certain behaviors, communication protocols, and submodels were added, modified, or replaced to meet the requirements (**Pluggability 1.4**).

In this work, the capability assessment of the agent-based AAS approach involves an algorithm designed to evaluate whether the product requirements align with the technical specifications of the resource. This algorithm is tailored for each specific case, necessitating significant programming effort to ensure accurate feasibility analysis. However, once created, the algorithm can be reused for assets offering similar capabilities. Considering a more advanced approach to replace this one would be beneficial. For example, integrating LLM could be a possibility to comprehend and align product requirements

and technical specifications presented in complex technical language, providing meaningful insights from submodel data. This could reduce the need for highly customized algorithms (**Pluggability 1.5**).

Robustness

Table 4.9 shows the assessment of the robustness requirement, along with the assigned final score representing the level of requirement fulfillment by the implemented solution.

Table 4.9: Assessment for the robustness requirement.

Requirement	**Requirement description**				
Robustness 1	Robustness refers to the system's ability to maintain its performance and stability in the face of potential external and internal disturbances				
Sub-requirements	**Sub-requirement description** (The proposed solution should...)	**Score of fulfilment (β)**			
		N/A	**Low**	**Medium**	**High**
Robustness 1.1	continue functioning, potentially at a reduced level, despite component failures				✓
Robustness 1.2	continue functioning normally as new components are added to the system				✓
Robustness 1.3	anticipate potential failures before they occur, allowing for timely maintenance actions that prevent downtime			✓	
Robustness 1.4	automatically identify issues and initiate corrective actions without human intervention			✓	
Robustness 1.5	ensure that data remains accurate, consistent, and unaltered during transmission and storage (e.g., using cryptographic techniques and secure protocols), as well as protect data from unauthorized access and attacks	✓			
					Final score = 3 out of 5

The unexpected event scenario provided an opportunity to analyze the robustness aspects of the implemented solution. This was evident when, at a specific time, one of the assigned resources for carrying out one of the operations experienced a failure, requiring the products to reschedule their production plans in order to complete the production process. In this context, the production process continues to function despite component failures. Despite this condition being possible thanks to the redundancy of resources, the fact that the system can autonomously reorganize to continue operating demonstrates the system's robustness (**Robustness 1.1**).

Although Robustness 1.2 may appear to be a requirement related to pluggability, its focus is not solely on the ease of introducing new components but also on the system's stability when these new components are introduced. This has been observed when multiple products were added to the system. However, it was especially noticeable in a

separate test, where new resources were added to the system and continued to function normally (**Robustness 1.2**).

In this scenario, a simple rule-based algorithm was used to monitor the health condition of the asset as a proof of concept. Although less advanced than an AI algorithm, it can anticipate potential failures before they happen. This algorithm is integrated as an agent behavior, making it modular and easily replaceable with a more advanced version (**Robustness 1.3**).

This monitoring behavior allows for the identification of issues and the activation of corrective actions without human intervention. The corrective action is based on simply reorganizing the product production plan by finding alternative resources to perform the required task. However, other advanced approaches based on self-healing were not addressed (**Robustness 1.4**).

This work does not address data integrity and security mechanisms. It assumes data remains accurate, consistent, and unaltered during transmission and storage. However, these mechanisms could be included in the proposed approach but are not the focus of the present work (**Robustness 1.5**).

Interoperability

Table 4.10 shows the assessment of the interoperability requirement, along with the assigned final score representing the level of requirement fulfillment by the implemented solution.

As an AAS-based solution, the proposed approach reflects the asset in the information world in the five upper layers of the RAMI4.0 model. This approach enhances traditional AAS capabilities through integration with agents. The agent implements the intelligent part of the approach and is supported by standards, namely IEEE 2660.1, VDI/VDE 2193, and FIPA. The information part of the proposed approach comprises several submodels that structure the asset information in a standardized manner. Most submodels used in this work are official submodels defined by the IDTA, which ensures greater interoperability. However, this work also outlines some submodels that are not yet officially available but comply with the AAS metamodel specification (**Interoperability 1.1 and 1.2**).

This work used the FIPA ACL for message exchange between agent-based AASs, taking advantage of specific classes for FIPA message structures provided by JADE, which made the proof of concept easier. However, the proposed approach can also be adjusted to use the I4.0 language (VDI/VDE 2193 standards) as discussed in Section 3.5. The I4.0 language is recommended for developing I4.0-compliant solutions, especially AAS Type 3 solutions, providing standardized communication between I4.0 components. Essentially, the I4.0 language is based on the FIPA ACL and shares many similarities. In this regard,

Table 4.10: Assessment for the interoperability requirement.

Requirement	**Requirement description**				
Interoperability 1	Interoperability refers to the ability of different systems, machines, and software applications to connect, communicate, and work together seamlessly, ensuring they can understand and exchange information effectively, regardless of their manufacturer or the technology they use				
Sub-requirements	**Sub-requirement description** (The proposed solution should...)	**Score of fulfilment (β)**			
		N/A	**Low**	**Medium**	**High**
Interoperability 1.1	be aligned with reference architectures and standards to support the design of such systems				✓
Interoperability 1.2	adopt standardized information models to structure information				✓
Interoperability 1.3	adopt standardized communication protocols, data formats, and structures to enable seamless exchange of information between systems			✓	
Interoperability 1.4	adopt standardized semantics to describe asset properties in an unambiguous, machine-readable, and industry-independent manner	✓			
Interoperability 1.5	enable the integration of data from different sources (vertically and horizontally), allowing for comprehensive analysis and decision-making			✓	
					Final score = 3 out of 5

it is reasonable to assume that the proposed approach could also be implemented to communicate using the I4.0 language (**Interoperability 1.3**).

This work does not cover semantics at any level despite their importance in achieving interoperability. For example, ECLASS and IEC CDD are two dictionaries of standardized semantics based on the IEC 61360 standard, which could be used to describe asset properties in an unambiguous, machine-readable, and industry-independent manner. It is important to note that this work focuses on designing an agent-based AAS solution that can easily introduce semantic aspects. However, this work does not explore the semantic aspect itself (**Interoperability 1.4**).

The agent-based AAS can be designed for assets placed in different hierarchy levels, such as the shop floor and business levels. While this work presented implementation examples at the shop floor level, specifically focusing on the horizontal collaboration between products and resources, the proposed approach can also facilitate vertical collaboration. This is due to the agent-based AASs use standardized information models to structure information (submodels) and standardized communication protocols, data formats, and structures to facilitate seamless information exchange between systems. In this scenario, an agent-based AAS representing a MES or ERP system could enable exchanging information with shop floor devices (**Interoperability 1.5**).

(Re)usability

Table 4.11 shows the assessment of the (re)usability requirement, along with the assigned final score representing the level of requirement fulfillment by the implemented solution.

Table 4.11: Assessment for the (re)usability requirement.

Requirement	**Requirement description**				
(Re)usability 1	(Re)usability refers to the ability to efficiently (re)use resources, components, technologies, and processes across different applications, systems, and stages of production within the industrial environment				
Sub-requirements	**Sub-requirement description** (The proposed solution should...)	**Score of fulfilment (β)**			
		N/A	**Low**	**Medium**	**High**
(Re)usability 1.1	include generic models, templates, and guidelines to facilitate its design and implementation				✓
(Re)usability 1.2	be compatible with existing hardware and software infrastructures, enabling easy deployment in different use-cases without extensive modifications				✓
(Re)usability 1.3	allow for the efficient (re)use of modules, services, and classes in both small-scale and large-scale applications				✓
(Re)usability 1.4	include mechanisms or a platform for managing the system, such as state and configuration, as well as interfaces that are accessible to users with diverse backgrounds, abilities, and preferences		✓		
					Final score = 4.1 out of 5

This work provided a generic specification for designing an agent-based AAS that implements an AAS Type 3. This specification provides guidelines for modeling fundamental submodels, interfaces to get information from submodels, integration patterns with physical assets, and interaction patterns between the AASs. Based on this specification, AAS Type 3 solutions can be designed for different use cases, as demonstrated by designing and implementing the agent-based AAS approach in two distinct case studies and scenarios (**(Re)usability 1.1**).

The proposed approach can be implemented using several open-source development platforms to develop the intelligent (agent) and information (AAS submodels) parts of the agent-based AAS. For instance, in this work, the adopted platforms include JADE, AASX Package Explorer, and AASX Server due to their efficiency and reliability in building agents and AAS submodels. Most of these platforms, including those used in this work, are compatible with various operating systems and can be deployed on different devices, ranging from single-board computers like Raspberry Pi to more powerful industrial computers. As a result, the designed solution can be readily deployed in diverse scenarios involving hardware and software infrastructure variations (**(Re)usability 1.2**).

Through the process of design and implementation of the proposed approach in two distinct case studies and scenarios, it became evident that several classes and submodels could be reused. Most classes were adaptable to the different scenarios, with customization only being necessary for classes that incorporate interaction protocols or are specific to particular behaviors. However, integrating these tailored classes is straightforward and only requires minor adjustments. Regarding the submodels, only their content needed to be modified, while the structure remained unchanged. Furthermore, the proposed approach facilitates system scalability since a designed agent-based AAS can be reused for several assets. For instance, the template, consisting of a common set of classes and submodels developed for an agent-based AAS designed for a product, can also be effectively applied to other products. This reusability extends beyond products to encompass various types of assets within the system (**(Re)usability 1.3**).

The proposed approach does not address the mechanisms for managing the solution as a unified whole and the user interfaces for non-technical users. Without a user-friendly platform for managing the system, the proposed solution's usability is restricted to users with advanced knowledge of MAS and AAS, as well as the utilized development platforms. This does not impact the performance of the solution, but it does affect its usability. For example, a dashboard that offers an overview of the entire system, including status updates and options for managing different aspects of the agent-based AASs, could facilitate the use of the solution. While the proposed approach does not address these managing mechanisms, the adopted development platforms like JADE and AASX Server offer mechanisms for managing individual agents and submodels separately during the operational phase, which, to a certain extent, supports the monitoring of the solution partially (**(Re)usability 1.4**).

In conclusion, by adhering to the proposed specifications and applying the approach to two distinct case studies, it is clear that the proposed agent-based AAS approach successfully fulfills the criteria for an AAS Type 3. It demonstrates essential characteristics such as autonomy, intelligence, and collaboration. Additionally, the solution exhibits high levels of adaptability, pluggability, robustness, interoperability, and (re)usability, meeting nearly all identified requirements. The only exceptions are related to security and semantic aspects, which were not included in the scope of this work.

4.7 Summary

This chapter presented the implementation and the experiments conducted to assess the proposed approach, demonstrating its applicability and contributions to the design and development of AAS Type 3 solutions using an agent-based approach. In summary, this chapter described the work developed to validate the proposed approach as being an

AAS Type 3 solution and answer the second research question of this thesis: *How can the agent-based AAS approach enhance the digitization process in industrial environments?*

Section 4.1 outlined the assessment methodology for the agent-based AAS approach, defining the assessment scenarios, evaluation criteria, and case studies. Section 4.1.1 described scenarios that reflect different manufacturing aspects requiring collaboration between assets, including resource allocation, operation execution, and system reconfiguration due to unexpected events. Section 4.1.2 presented the evaluation criteria, focused on qualitative aspects such as adaptability, pluggability, robustness, interoperability, and (re)usability, which are key requirements for realizing I4.0-compliant solutions. Finally, Section 4.2 described the two case studies used to implement, test, and validate the proposed approach: the first based on a laboratory prototype (small-scale production system) and the second on an industrial automotive production line.

Section 4.3 detailed the process of implementing an individual agent-based AAS, particularly to implement the main classes described in the information metamodel. The information part was implemented using the Eclipse AASX Package Explorer tool, a C#-based viewer and editor for the development of AASs and their elements, specifically the submodels (Section 4.3.1). The intelligent part (agent) was implemented using the JADE framework, which implements an efficient agent platform and supports the development of MAS that complies with the FIPA specifications. In this context, JADE managed the software infrastructure for agent operations, while the information part (submodels) and the asset maintained their own management platforms (Section 4.3.2). The Eclipse AASX Server was adopted as an API-enabled approach, providing standard interfaces for the agents to obtain information from their corresponding AASs using the HTTP/REST communication protocol (Section 4.3.3). Finally, the IEEE 2660.1 recommendation method was applied to identify the best interface practice for interacting with the assets in the two case studies (Section 4.3.4).

Sections 4.4 and 4.5 presented the deployment of the agent-based AAS approach in both case studies to implement the defined scenarios. For each asset in the system, including both products and resources, an agent-based AAS was designed and implemented. The interactions between products and resources through AAS Type 3 followed the behavior models and interaction protocols based on FIPA and I4.0 language specified for each scenario, ensuring that the specific requirements of each scenario were addressed. Finally, Section 4.6 evaluated the proposed approach through a twofold assessment. First, the evaluation examined whether the proposed approach fulfills the requirements for an AAS to qualify as Type 3, which includes autonomy, intelligence, and collaboration. Second, it assessed the agent-based AAS based on the defined qualitative criteria. The experiments and this assessment demonstrated that the proposed agent-based approach successfully realizes an AAS Type 3, enabling assets to transition from passive components to truly I4.0 components with enhanced autonomy, intelligence, and collaborative capabilities.

Additionally, the approach addresses essential requirements for I4.0, namely adaptability, pluggability, robustness, interoperability, and (re)usability.

CHAPTER 5

Conclusions and Future Work

In today's globalized world, manufacturers are facing constant pressure to adapt quickly to the fast-changing market conditions. In this context, I4.0 emerges as a transformative paradigm, aiming to shift from traditional manufacturing systems to intelligent, interconnected, and adaptable systems based on CPS. However, this transition remains far from fully realized more than a decade after its introduction. Many companies encounter significant barriers, including challenges with technological integration, high implementation costs, and a lack of standardized solutions that ensure seamless interoperability.

In this context, the AAS serves as a standardized digital representation of an asset, encapsulating all asset-relevant information throughout its lifecycle, facilitating interoperability and efficient data exchange across the entire production network. At this stage, traditional AAS solutions, namely Type 1 and Type 2, may be sufficient to resolve the problem of data silos and address basic interoperability issues. However, the demands of dynamic and complex industrial environments go beyond these capabilities. Such environments often require a higher level of autonomy and intelligence from I4.0 components to enable decision-making and collaboration. This requires the evolution towards more sophisticated solutions, such as AAS Type 3, which incorporates enhanced capabilities to meet these emerging challenges.

Motivated by these aspects and supported by the gaps identified through a comprehensive literature review, this thesis proposed an agent-based AAS approach for designing AAS Type 3. This approach addresses the lack of official documentation and the limited research on AAS Type 3 design while identifying MAS as a key enabling technology for realizing AAS Type 3. The proposed solution bridges these gaps and provides a practical approach to enhance the autonomy, intelligence, and collaborative capabilities required for advanced

industrial applications. Additionally, it maintains compatibility with the official AAS specifications provided by IDTA and aligns with industrial standards.

The remainder of this chapter is structured as follows:

- Section 5.1 provides an overview of the conclusions derived from this work, reflecting how the objectives of the thesis were achieved, the research questions addressed, and the proposed hypothesis validated.
- Section 5.2 outlines potential directions for future research, presenting promising opportunities and exploring how these could further enhance and extend the work developed in this thesis.

5.1 Conclusions

At the very beginning of this thesis, an initial perception was formed, widely accepted in general understanding within the context of I4.0, that MAS could serve as an appropriate approach for implementing AAS Type 3 due to their inherent characteristics and diverse applications in the CPS context. In this sense, the following hypothesis was defined:

> *An agent-based approach can enable the design and development of an Asset Administration Shell Type 3 solution. This allows the transition of assets from passive components to truly Industry 4.0 components (dynamic and active participants in the Industry 4.0 ecosystem), equipped with enhanced management, interoperability, autonomy, intelligence, and collaborative capabilities.*

While this hypothesis appeared intuitive, given the alignment between MAS characteristics and AAS Type 3 requirements, it first required a robust theoretical foundation and a comprehensive analysis of existing literature to confirm its validity, as well as to assess the degree of innovation and contribution it could bring to the field. This groundwork was essential for subsequently proposing a solution to address the identified problem and for validating the hypothesis through a specification and experimental implementation.

Chapter 2, in particular, presented research works that address several aspects related to AAS Type 3. This review highlighted two critical gaps: the lack of a comprehensive specification for designing AAS Type 3 solutions and the MAS as a key enabling technology for its implementation. It was concluded that while the literature includes agent-based AAS approaches, these must be adapted to a higher level of abstraction to ensure applicability across diverse scenarios, ensuring they are sufficiently generic, independent of specific use cases, and capable of supporting various collaboration strategies. In addition, an analysis of several MAS-based CPS approaches in the literature and R&D projects, demonstrated the potential benefits of MAS in industrial environments, reinforcing its

role as a key enabling technology for implementing AAS Type 3. Based on this literature review, Chapter 2 concludes by examining the parallels between MAS, AAS Type 3, and RAMI4.0, offering insights for developing the specification to design and develop an AAS Type 3 solution using MAS.

From the gaps and research opportunities identified through the literature review, Chapter 3 presented the work developed to answer the first research question of this thesis: **(RQ1)** *What aspects an agent-based approach must consider to design and develop an Asset Administration Shell Type 3 solution?* To address this question, an information metamodel was proposed as a solution, offering a specification for designing AAS Type 3 through an agent-based approach while considering several key aspects. Firstly, the approach maintains compatibility with the official AAS metamodel, allowing novel classes or modules to be added without altering the core structure. This ensures adherence to AAS specifications and preserves interoperability with existing AAS Type 1 and 2 solutions, with these additions functioning as a higher-layer extension. Secondly, this higher-layer extension, represented by the agents, should not only handle intelligent and communication tasks but also interface directly with the asset, e.g., to enable rapid response for continuous monitoring and control, interpret data contained within the AAS, and use this information to drive actions and optimizations. Taking into consideration, the information metamodel itself consists of two parts: the information part and the intelligent part. The information part includes the primary classes necessary for describing asset information within the submodels, while the intelligent part encompasses the main classes designed to enhance the AAS with autonomy, intelligence, and collaborative capabilities provided by the agent-based approach. The functioning of each of these classes was discussed individually, presenting supporting standards for their development, as well as illustrative examples to demonstrate their application.

Based on the development of the proposed agent-based AAS approach to guide the design of AAS Type 3 solutions using MAS, Chapter 4 described the work developed to validate the proposed approach as being an AAS Type 3 solution and answer the second research question of this thesis: **(RQ2)** *How can the agent-based AAS approach enhance the digitization process in industrial environments?* To answer this question, the proposed approach was implemented and tested in two case studies. The first case study involved a laboratory prototype representing a small-scale production system, while the second focused on an industrial automotive production line. For both case studies, agent-based AASs were designed and implemented to enhance the existing systems with autonomy, intelligence, and collaborative capabilities. These implementations demonstrated how the proposed approach could be applied in diverse industrial scenarios, highlighting its adaptability, pluggability, robustness, interoperability, and (re)usability, which are important I4.0 requirements. The results from this experimental implementation provided valuable practical insights into the advantages of integrating MAS to realize AAS Type 3.

Furthermore, they validated the feasibility and effectiveness of the proposed solution in advancing the digitization process in industrial environments.

As discussed in the answers to the research questions above, the results obtained in this work contributed to achieving the objectives of this thesis and confirming the defined hypothesis. By addressing the identified gaps and developing the proposed agent-based AAS approach, this work demonstrated the feasibility, applicability, and benefits of using an agent-based approach to enable the design and development of an AAS Type 3 solution. Through theoretical development and experimental implementation in two case studies, the findings affirmed the initial hypothesis and contributed to advancing the state-of-the-art in this domain.

As a final remark, this research has significantly contributed to the scientific community through several publications in relevant international conferences and journal articles:

1. L. Sakurada, F. De la Prieta, and P. Leitao. "An Agent-Based Approach for Implementing Asset Administration Shell Type 3," *IEEE Open Journal of the Industrial Electronics Society*, 2025 (submitted for publication).

2. L. Sakurada, F. De la Prieta, and P. Leitao. "Ten Years of Asset Administration Shell: Developments, Research Opportunities, and Adoption Challenges," *IEEE Access*, 2025. (Impact Factor: 3.6; Quartile: Q1)

3. L. Sakurada, F. De la Prieta, and P. Leitao. "The Role of Multi-Agent Systems in Realizing Asset Administration Shell Type 3," *Future Internet*, 2025. (Impact Factor: 3.6; Quartile: Q2)

4. L. Sakurada, F. De la Prieta, and P. Leitao, "Digitization of Industrial Environments Through an Industry 4.0 Compliant Approach," in *49th Annual Conference of the IEEE Industrial Electronics Society*, Singapore, Singapore, 2023, pp. 1-6.

5. L. Sakurada, F. De la Prieta, and P. Leitao, "A Methodology for Integrating Asset Administration Shells and Multi-agent Systems," in *32nd International Symposium on Industrial Electronics (ISIE)*, Helsinki, Finland, 2023, pp. 1-6.

6. L. Sakurada, P. Leitao, and F. De la Prieta, "Engineering a Multi-agent Systems Approach for Realizing Collaborative Asset Administration Shells," in *International Conference on Industrial Technology (ICIT)*, Shanghai, China, 2022, pp. 1-6.

7. L. Sakurada, P. Leitao, and F. De la Prieta, "Agent-based asset administration shell approach for digitizing Industrial assets," in *14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022*, Tel-Aviv, Israel, 2022, 55, 193-198. (**Best paper award**)

8. L. Sakurada, P. Leitao, and F. De la Prieta, "Towards the Digitization using Asset Administration Shells," in *47th Annual Conference of the IEEE Industrial Electronics Society*, Toronto, ON, Canada, 2021, pp. 1-6.

9. L. Sakurada and P. Leitao, "Multi-Agent Systems to Implement Industry 4.0 Components," in *IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, Tampere, Finland, 2020, pp. 21-26.

10. A. Lopez, L. Sakurada, P. Leitao, O. Casquero, E. Estevez, F. De la Prieta, and M. Marcos, "Technology-Independent Demonstrator for Testing Industry 4.0 Solutions," in *IEEE 20th International Conference on Industrial Informatics (INDIN)*, Perth, Australia, 2022, pp. 21-26.

11. P. Leitao, J. Queiroz, and L. Sakurada, "Collective Intelligence in Self-Organized Industrial Cyber-Physical Systems," *Electronics*, 2022, 11, 3213.

5.2 Future Work

During the development of this work, some research opportunities have been identified for future research. In this regard, this section discusses possible research directions to extend the developed work.

General Improvements

Some improvements could be made regarding the testing and validating the proposed approach. While these enhancements do not invalidate the contributions of this thesis, they could offer new perspectives and further strengthen the robustness of the findings. Expanding the scope of experimental implementation, including more diverse case studies and real-world industrial scenarios, would provide deeper insights into the practical applicability of the developed work. Some key aspects can be highlighted in this regard:

- The primary objective of this thesis was to specify a generic approach for designing an AAS Type 3 using MAS. The case studies allowed for an evaluation of autonomy, intelligence, and collaborative aspects, as well as some I4.0 requirements, in selected scenarios. However, further testing in additional scenarios could provide deeper insights into the adaptability and effectiveness of the approach.

- Instantiating the proposed approach for other specific industrial challenges, such as real-time monitoring, process optimization, and fault tolerance, particularly using more advanced AI techniques, would provide a more comprehensive assessment of its capabilities in practical applications.

- Another crucial area for future work involves integrating the proposed approach with existing industrial systems, such as MES and ERP. As mentioned earlier, the AAS Type 3 is not intended to replace these established systems but to enhance their

functionality by providing a more autonomous, intelligent, and collaborative layer that can work seamlessly alongside them.

Deployment Strategies and Low-Cost Solutions

About a decade after the term "Industry 4.0" was coined, its full realization remains a distant goal. Beyond the inherent complexity of developing I4.0 solutions, the cost associated with transitioning from I3.0 to I4.0 systems is particularly high, especially for Small and Medium-sized Enterprises (SMEs). In this context, it becomes crucial to define a strategy for implementing the proposed solution as a low-cost approach, taking into account both hardware and software elements. Such a strategy would focus on leveraging affordable, scalable, and open-source technologies, reducing upfront investments while maintaining compatibility with existing systems. Additionally, modularity and interoperability in the design would ensure that SMEs can adopt the solution incrementally, mitigating risks and managing costs more effectively.

Human-Centered Approach

The currently renewed attention to human-centered approaches is a topic not further explored in this thesis, although the proposed approach can be designed to address this aspect. This thesis discussed the design and implementation of agent-based AASs for products and physical resources. However, many industrial processes still require human operators for tasks such as supervision, decision-making, or complementary manual operations. Human operators can indeed be considered resources in the production process, as they contribute their unique skills and expertise to achieve operational objectives. However, unlike machines, human operators have distinct needs and requirements that must be respected to ensure their quality of work and overall well-being. These include considerations such as physical and cognitive workload, preferences, and availability, as well as factors that impact their health, safety, and satisfaction. In this regard, the operator can be represented by an agent-based AAS, which not only contains detailed information about the operator, such as skills, preferences, availability, and physical or cognitive conditions but also acts as an active entity within the system. This approach should facilitate democratic collaboration between human operators and workstation assets, ensuring that operators are dynamically and intelligently allocated to workstations, considering process performance, and respecting operator preferences and conditions. Furthermore, it extends the RAMI4.0 hierarchy level to incorporate the human operator as an essential element, recognizing their fundamental role within the industrial ecosystem.

Advancing AAS with AI

While agent-based AAS can already incorporate AI techniques, recent advancements in LLMs, as discussed in Section 3.6, have the potential to significantly enhance their intelligence capabilities. In this sense, these LLM agent-based AASs could be able to interpret and process complex information stored in submodels, facilitate communication with humans and other agents, and dynamically adapt to different scenarios by leveraging their ability to learn from vast datasets. This integration can further enhance the autonomy and collaboration capabilities of AAS Type 3, particularly combining with standard message structures and interaction protocols (e.g., I4.0 language), advancing their potential for industrial applications.

References

[1] International Society of Automation, *ISA-95 Series of Standards*. [Online]. Available: https://www.isa.org/standards-and-publications/isa-standards/isa-95-standard.

[2] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution", *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 6–16, 2017.

[3] DIN, *Reference Architecture Model Industrie 4.0 (RAMI4.0) English translation of DIN SPEC 91345:2016-04*, 2016.

[4] Plattform Industrie 4.0, *Details of the Asset Administration Shell - from idea to implementation*, 2018.

[5] L. Sakurada, F. De La Prieta, and P. Leitao, "Ten Years of Asset Administration Shell: Developments, Research Opportunities, and Adoption Challenges", *IEEE Access*, 2025.

[6] M. Wooldridge, *An Introduction to MultiAgent Systems: Second Edition*. John Wiley & Sons, 2009.

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

[8] S. Karnouskos, P. Leitão, L. Ribeiro, and A. W. Colombo, "Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems: Multiagent Systems Entering Industry 4.0", *IEEE Ind. Electron. Mag.*, vol. 14, no. 3, pp. 18–32, 2020.

[9] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research", *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[10] H. Kagermann, W. Wahlster, and J. Helbig, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0*, 2013.

[11] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues", *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[12] European Commission, *Industry 5.0 – Towards a sustainable, human-centric and resilient European industry*. Publications Office of the European Union, 2021.

[13] European Commission, *ERA industrial technologies roadmap on human-centric research and innovation for the manufacturing sector*. Publications Office of the European Union, 2024.

[14] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and Industry 5.0—Inception, conception and perception", *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021.

[15] Research Council Industrie 4.0 and the Plattform Industrie 4.0, *The fourth industrial revolution and the term "Industry 5.0"' – a critical perspective*, 2024.

[16] M. D. Rodriguez, B. Langefeld, and O. S. Eguibar, *The state of Industry 4.0*, 2023.

[17] M. Rüßmann *et al.*, *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries*, 2015.

[18] G. Schuh, R. Anderl, R. Dumitrescu, A. Krüger, and M. ten Hompel, *Using the Industrie 4.0 Maturity Index in Industry. Current challenges, case studies and trends*, 2020.

[19] E. Oztemel and S. Gursev, "Literature review of Industry 4.0 and related technologies", *Journal of Intelligent Manufacturing*, vol. 31, pp. 127–182, 2018.

[20] P. Leitão, J. Queiroz, and L. Sakurada, "Collective Intelligence in Self-Organized Industrial Cyber-Physical Systems", *Electronics*, vol. 11, no. 19, 2022.

[21] Industry IoT Consortium, *The Industrial Internet Reference Architecture*, 2022.

[22] International Electrotechnical Commission, *IEC 62890 - Industrial-process measurement, control and automation - Life-cycle-management for systems and components*. [Online]. Available: https://webstore.iec.ch/en/publication/30583.

[23] R. Heidel, M. Hoffmeister, M. Hankel, and U. Döbrich, *The Reference Architecture Model RAMI 4.0 and the Industrie 4.0 component*. Beuth Verlag, 2019.

[24] VDI/VDE and ZVEI, *GMA Status Report: Reference Architecture Model Industrie 4.0 (RAMI4.0)*, 2015.

[25] ISO Standards, *IEC 62264 - Enterprise-control system integration*. [Online]. Available: https://www.iso.org/standard/57308.html.

[26] International Electrotechnical Commission, *IEC 61512 - Batch control*. [Online]. Available: https://webstore.iec.ch/en/publication/5528.

[27] Industrial Digital Twin Association, *Specification of the Asset Administration Shell - Part 1: Metamodel (version 3.0)*, 2023.

[28] Industrial Digital Twin Association, *Specification of the Asset Administration Shell - Part 2: Application Programming Interfaces (version 3.0)*, 2023.

[29] Industrial Digital Twin Association, *IDTA 02017-1-0 Asset Interfaces Description*, 2024.

[30] Industrial Digital Twin Association, *AAS Submodel Templates*. [Online]. Available: https://industrialdigitaltwin.org/en/content-hub/submodels.

[31] Plattform Industrie 4.0, *Verwaltungsschale in der Praxis*, 2020.

[32] Plattform Industrie 4.0, *Functional View of the Asset Administration Shell in an Industrie 4.0 System Environment*, 2021.

[33] Plattform Industrie 4.0, *What is the Asset Administration Shell from a technical perspective?*, 2021.

[34] X. Ye, W. S. Song, S. H. Hong, Y. C. Kim, and N. H. Yoo, "Toward data interoperability of enterprise and control applications via the industry 4.0 asset administration shell", *IEEE Access*, vol. 10, pp. 35 795–35 803, 2022.

[35] M. Stolze, A. Belyaev, C. Kosel, C. Diedrich, and A. Barnard, "Realizing Automated Production Planning via Proactive AAS and Business Process Models", in *IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024, pp. 1–8.

[36] A. López, "Industrial agents for resilient manufacturing systems - An I4.0 plantform for the manufacturing domain", Ph.D. thesis, University of the Basque Country (UPV/EHU), 2023.

[37] S. Grunau, M. Redeker, D. Göllner, and L. Wisniewski, "The implementation of proactive asset administration shells: Evaluation of possibilities and realization in an order driven production", in *Kommunikation und Bildverarbeitung in der Automation*, J. Jasperneite and V. Lohweg, Eds., Springer Berlin Heidelberg, 2022, pp. 131–144.

[38] B. Vogel-Heuser, F. Ocker, and T. Scheuer, "An approach for leveraging Digital Twins in agent-based production systems", *at - Automatisierungstechnik*, vol. 69, no. 12, pp. 1026–1039, 2021.

[39] VDI/VDE, *VDI/VDE 2193 Part 1 - Language for I4.0 Components - Structure of messages*, 2020.

[40] VDI/VDE, *VDI/VDE 2193 Part 2 - Language for I4.0 components - Interaction protocol for bidding procedures*, 2020.

[41] Siemens, *Nine partners demonstrate an interoperable digital twin based on the Asset Administration Shell*. [Online]. Available: https://press.siemens.com/global/en/pressrelease/nine-partners-demonstrate-interoperable-digital-twin-based-asset-administration-shell.

[42] X. Ye and S. H. Hong, "Toward Industry 4.0 Components: Insights Into and Implementation of Asset Administration Shells", *IEEE Industrial Electronics Magazine*, vol. 13, no. 1, pp. 13–25, 2019.

[43] X. Ye, J. Jiang, C. Lee, N. Kim, M. Yu, and S. H. Hong, "Toward the Plug-and-Produce Capability for Industry 4.0: An Asset Administration Shell Approach", *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, pp. 146–157, 2020.

[44] X. Ye, S. H. Hong, W. S. Song, Y. C. Kim, and X. Zhang, "An industry 4.0 asset administration shell-enabled digital solution for robot-based manufacturing systems", *IEEE Access*, vol. 9, pp. 154 448–154 459, 2021.

[45] S. Cavalieri and M. G. Salafia, "A Model for Predictive Maintenance Based on Asset Administration Shell", *Sensors*, vol. 20, no. 21, 2020.

[46] A. Sidorenko, M. Volkmann, W. Motsch, A. Wagner, and M. Ruskowski, "An OPC UA Model of the Skill Execution Interaction Protocol for the Active Asset Administration Shell", *Procedia Manufacturing*, vol. 55, pp. 191–199, 2021, FAIM 2021, ISSN: 2351-9789.

[47] F. Ocker, C. Urban, B. Vogel-Heuser, and C. Diedrich, "Leveraging the asset administration shell for agent-based production systems", *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 837–844, 2021, 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021, ISSN: 2405-8963.

[48] J. Arm *et al.*, "Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0", *Sensors*, vol. 21, no. 6, 2021.

[49] Q. Zhou, Y. Wu, C. Gu, W. Meng, S. He, and Z. Shi, "AASPMP: Design and Implementation of Production Management Platform Based on AAS", in *2022 IEEE 20th International Conf. on Industrial Informatics (INDIN)*, 2022, pp. 515–520.

[50] S. Jungbluth *et al.*, "Dynamic Replanning using Multi-Agent Systems and Asset Administration Shells", in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Stuttgart, Germany, 2022, pp. 1–8.

[51] M. Simon, S. Jungbluth, A. Farrukh, M. Volkmann, J. Hermann, and M. Ruskowski, "Implementation of Asset Administration Shells in a Shared Production Scenario with Gaia-X", in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8.

[52] A. López, O. Casquero, and M. Marcos, "Design patterns for the implementation of Industrial Agent-based AASs", in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, Victoria, BC, Canada, 2021, pp. 213–218.

[53] L. Sakurada, P. Leitão, and F. De la Prieta, "Agent-Based Asset Administration Shell Approach for Digitizing Industrial Asset", in *14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022*, Tel-Aviv, Israel, 2022, pp. 193–198.

[54] K. Kanaan, J. Wermann, M. A. Bär, and A. W. Colombo, "Industry 4.0-compliant Digitalization of a Re-configurable and Flexible Laser Cutter Module within a Digital Factory", in *2023 IEEE International Conference on Industrial Technology (ICIT)*, 2023, pp. 1–7.

[55] A. Sidorenko, W. Motsch, M. van Bekkum, N. Nikolakis, K. Alexopoulos, and A. Wagner, "The MAS4AI framework for human-centered agile and smart manufacturing", *Frontiers in Artificial Intelligence*, vol. 6, 2023.

[56] Y. Xia, M. Shenoy, N. Jazdi, and M. Weyrich, "Towards autonomous system: flexible modular production system enhanced with large language model agents", in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8.

[57] S.-J. Shin and J. Um, "Deploying data analytics models in asset administration shells: Energy prediction in manufacturing", *Engineering Applications of Artificial Intelligence*, vol. 138, p. 109 269, 2024.

[58] A. T. Bernhard *et al.*, "I4.0 holonic multi-agent testbed enabling shared production", in Cham: Springer Nature Switzerland, 2024, pp. 231–250.

[59] S. Karnouskos, L. Ribeiro, P. Leitão, A. Lüder, and B. Vogel-Heuser, "Key Directions for Industrial Agent Based Cyber-Physical Production Systems", in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, Taipei, Taiwan, 2019, pp. 17–22.

[60] B. Vogel-Heuser, M. Seitz, L. A. C. Salazar, F. Gehlhoff, A. Dogan, and A. Fay, "Multi-agent systems to enable Industry 4.0", *at - Automatisierungstechnik*, vol. 68, no. 6, pp. 445–458, 2020.

[61] P. Leitão, "An Agile and Adaptive Holonic Architecture for Manufacturing Control", Ph.D. thesis, Faculty of Engineering of University of Porto, 2004.

[62] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice", *The Knowledge Engineering Review*, vol. 10, 1995.

[63] R. Unland, "Chapter 1 - Software Agent Systems", in *Industrial Agents*, P. Leitão and S. Karnouskos, Eds., Boston: Morgan Kaufmann, 2015, pp. 3–22.

[64] R. Unland, "Chapter 2 - Industrial Agents", in *Industrial Agents*, P. Leitão and S. Karnouskos, Eds., Boston: Morgan Kaufmann, 2015, pp. 23–44.

[65] IEEE, "IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions", *IEEE Std 2660.1-2020*, pp. 1–43, 2021.

[66] P. Leitão, N. Rodrigues, C. Turrin, A. Pagani, and P. Petrali, "GRACE ontology inteGrating pRocess and quAlity Control", in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 4348–4353.

[67] F. Bergenti, M.-P. Gleizes, and F. Zambonelli, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*. Jan. 2004, p. 506.

[68] S. Bussmann, N. Jennings, and M. Wooldridge, "Multiagent systems for manufacturing control: a design methodology", Jan. 2004.

[69] A. Lüder, A. Calá, J. Zawisza, and R. Rosendahl, "Design pattern for agent based production system control — A survey", in *13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017, pp. 717–722.

[70] L. Cruz Salazar, D. Ryashentseva, A. Lüder, and B. Vogel-Heuser, "Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns", *The International Journal of Advanced Manufacturing Technology*, vol. 105, Dec. 2019.

[71] L. Monostori, J. Váncza, and S. Kumara, "Agent-Based Systems for Manufacturing", *CIRP Annals*, vol. 55, no. 2, pp. 697–720, 2006.

[72] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey", *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, 2009.

[73] P. Leitão, V. Mařík, and P. Vrba, "Past, Present, and Future of Industrial Agent Applications", *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2360–2372, 2013.

[74] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: PROSA", *Computers in Industry*, vol. 37, no. 3, pp. 255–274, 1998, ISSN: 0166-3615.

[75] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution", *Computers in Industry*, vol. 66, pp. 99–111, 2015.

[76] D. Ryashentseva, "Agents and SCT based self* control architecture for production systems", Ph.D. thesis, Otto-von-Guericke-Universität Magdeburg, 2016.

[77] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: A self-organized multi-agent system with big data based feedback and coordination", *Computer Networks*, vol. 101, pp. 158–168, 2016, ISSN: 1389-1286.

[78] H. Tang, D. Li, S. Wang, and Z. Dong, "CASOA: An Architecture for Agent-Based Manufacturing System in the Context of Industry 4.0", *IEEE Access*, vol. 6, pp. 12 746–12 754, 2018.

[79] N. Rodrigues, "Agent-based Service Reconfiguration for Dynamic and Evolvable Systems", Ph.D. thesis, Faculty of Engineering of University of Porto, 2019.

[80] P. Leitão, J. Barbosa, G. Funchal, and V. Melo, "Self-organized cyber-physical conveyor system using multi-agent systems", *International Journal of Artificial Intelligence*, vol. 18, no. 2, pp. 171–185, 2020, ISSN: 0974-0635.

[81] J. Queiroz, "An Agent-based Approach to Design Decentralized Cloud-Edge Data Analysis in Cyber-Physical System", Ph.D. thesis, Faculty of Engineering of University of Porto, 2022.

[82] L. Cruz Salazar and B. Vogel-Heuser, "A CPPS-architecture and workflow for bringing agent-based technologies as a form of artificial intelligence into practice", *at - Automatisierungstechnik*, vol. 70, no. 6, pp. 580–598, 2022.

[83] VDI/VDE, *VDI/VDE 2653 Blatt 4 - Multi-agent systems in industrial automation - Selected patterns for field level control and energy systems*, 2022.

[84] P. Leitão, N. Rodrigues, C. Turrin, and A. Pagani, "Multiagent system integrating process and quality control in a factory producing laundry washing machines", *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 879–886, 2015.

[85] P. Leitão, J. Barbosa, P. Vrba, P. Skobelev, A. Tsarev, and D. Kazanskaia, "Multi-agent System Approach for the Strategic Planning in Ramp-Up Production of Small Lots", in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 4743–4748.

[86] M. Onori, N. Lohse, J. Barata, and C. Hanisch, "The IDEAS project: Plug & produce at shop-floor level", *Assembly Automation*, vol. 32, pp. 124–134, Apr. 2012.

[87] N. Antzoulatos, E. Castro, D. Scrimieri, and S. Ratchev, "A multi-agent architecture for plug and produce on an industrial assembly platform", *Production Engineering*, vol. 8, Dec. 2014.

[88] J. Queiroz, P. Leitão, J. Barbosa, E. Oliveira, and G. Garcia, "Agent-Based Distributed Data Analysis in Industrial Cyber-Physical Systems", *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, vol. 3, no. 1, pp. 5–12, 2022.

[89] A. Köcher *et al.*, "A reference model for common understanding of capabilities and skills in manufacturing", *at - Automatisierungstechnik*, vol. 71, no. 2, pp. 94–104, 2023.

[90] M. Schleipen and R. Drath, "Three-view-concept for modeling process or manufacturing plants with AutomationML", in *IEEE Conference on Emerging Technologies and Factory Automation*, 2009, pp. 1–4.

[91] Industrial Digital Twin Association, *Repository submodel templates - Capability (Version 1.0)*. [Online]. Available: https://github.com/admin-shell-io/submodel-templates/tree/main/development/Capability/1/0.

[92] Plattform Industrie 4.0 and ECLASS, *Modelling the Semantics of Data of an Asset Administration Shell with Elements of ECLASS*, 2021.

[93] International Electrotechnical Commission, *IEC 61360-4 - IEC/SC 3D - Common Data Dictionary (CDD - V2.0018.0001)*. [Online]. Available: https://cdd.iec.ch/cdd/iec61360/iec61360.nsf.

[94] International Electrotechnical Commission, *IEC 61360 - Standard data element types with associated classification scheme*. [Online]. Available: https://webstore.iec.ch/en/publication/28560.

[95] International Electrotechnical Commission, *IEC 61131 - Programmable controllers*. [Online]. Available: https://webstore.iec.ch/en/publication/4550.

[96] International Electrotechnical Commission, *IEC 61499 - Function blocks*. [Online]. Available: https://webstore.iec.ch/en/publication/5506.

[97] Industrial Digital Twin Association, *IDTA 02015-1-0 - Control Component Type*, 2023.

[98] Industrial Digital Twin Association, *IDTA 02016-1-0 - Control Component Instance*, 2023.

[99] World Wide Web Consortium, *Web of Things (WoT) Thing Description 1.1*. [Online]. Available: https://www.w3.org/TR/wot-thing-description11/.

[100] Industrial Digital Twin Association, *IDTA 02058-1-0 Artificial Intelligence Dataset*, 2025.

[101] Industrial Digital Twin Association, *IDTA 02060-1-0 Artificial Intelligence Model Nameplate*, 2025.

[102] Industrial Digital Twin Association, *IDTA 02059-1-0 Artificial Intelligence Deployment*, 2025.

[103] Data Mining Group, *PMML 4.4.1 - General Structure*. [Online]. Available: https://dmg.org/pmml/v4-4-1/GeneralStructure.html.

[104] Data Mining Group, *Portable Format for Analytics (PFA)*. [Online]. Available: https://dmg.org/pfa/.

[105] ONNX, *Open Neural Network Exchange - The open standard for machine learning interoperability*. [Online]. Available: https://onnx.ai/.

[106] Industrial Digital Twin Association, *IDTA 02008-1-1 Time Series Data*, 2023.

[107] Swagger hub, *DotAAS Part 2 — HTTP/REST — Asset Administration Shell Repository Service Specification*. [Online]. Available: https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.0.1_SSP-001.

[108] Swagger hub, *DotAAS Part 2 — HTTP/REST — Asset Adminstration Shell Service Specification*. [Online]. Available: https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellServiceSpecification/V3.0.1_SSP-001.

[109] OPC Foundation, *OPC 30270: Industry 4.0 Asset Administration Shell*. [Online]. Available: https://reference.opcfoundation.org/I4AAS/v100/docs/.

[110] P. Leitão, S. Karnouskos, L. Ribeiro, P. Moutis, J. Barbosa, and T. I. Strasser, "Integration patterns for interfacing software agents with industrial automation systems", in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 2908–2913.

[111] P. Leitão, T. I. Strasser, S. Karnouskos, L. Ribeiro, J. Barbosa, and V. Huang, "Recommendation of best practices for industrial agent systems based on the ieee 2660.1 standard", in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, 2021, pp. 1157–1162.

[112] H. K. Pakala, A. Belyaev, and C. Diedrich, "Middleware Architecture for Application Layer Interoperability of Standardized Digital Representations", in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.

[113] Foundation for Intelligent Physical Agents, *FIPA ACL Message Structure Specification*, 2002.

[114] Foundation for Intelligent Physical Agents, *FIPA Communicative Act Library Specification*, 2002.

[115] Plattform Industrie 4.0, *Aspects of the Research Roadmap in Application Scenarios*, 2016.

[116] Foundation for Intelligent Physical Agents, *FIPA Request Interaction Protocol Specification*, 2002.

[117] Foundation for Intelligent Physical Agents, *FIPA Subscribe Interaction Protocol Specification*, 2002.

[118] Foundation for Intelligent Physical Agents, *FIPA Propose Interaction Protocol Specification*, 2002.

[119] Foundation for Intelligent Physical Agents, *FIPA Query Interaction Protocol Specification*, 2002.

[120] Foundation for Intelligent Physical Agents, *FIPA Request When Interaction Protocol Specification*, 2002.

[121] L. S. Nelson, "The Shewhart Control Chart - Tests for Special Causes", *Journal of Quality Technology*, vol. 16, no. 4, pp. 237–239, 1984.

[122] T. Murata, "Petri nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

ANNEX A

Spanish Summary of the Thesis

This annex contains the summary version of this thesis in Spanish, in accordance with Article 14.3 of the Doctoral Regulations of the University of Salamanca.

"14.3. La redacción de la tesis doctoral se hará en castellano o en una de las lenguas habituales para la comunicación científica en su campo de conocimiento. Si la tesis doctoral está redactada en un idioma diferente al castellano, se acompañará de un documento, avalado por el Director de la misma, en el que consten el título, el índice, la introducción, un resumen significativo y las conclusiones de la tesis doctoral en castellano."

In this regard, this annex includes the title page, table of contents, introduction, comprehensive summary, and conclusions in Spanish.

VNiVERSiDAD D SALAMANCA

TESIS DOCTORAL

Un Enfoque Basado en Agentes Industriales para el Diseño de *Asset Administration Shell* Tipo 3

Doctorando:
Lucas Hiroaki Sakurada

Directores:
Fernando De la Prieta Pintado
Paulo Jorge Pinto Leitao

Tesis presentada para el cumplimiento de los requisitos del

Doctorado en Ingeniería Informática

en la

Escuela de Doctorado "Studii Salamantini" de la Universidad de Salamanca

2025

A.1 Índice

A continuación se presenta la versión resumida del índice de la tesis doctoral:

A.2 Introducción

Los enfoques tradicionales de control se basan en estructuras centralizadas y jerárquicas, siguiendo típicamente la conocida pirámide de automatización ISA-95 [1]. Aunque estos enfoques han demostrado ser eficaces para optimizar los procesos de producción, a menudo carecen de la agilidad y adaptabilidad necesarias para responder rápidamente a un mercado en constante cambio, caracterizado por productos altamente personalizados y de alta calidad. Sin embargo, con la llegada de la Industria 4.0 (I4.0), se está produciendo un cambio de paradigma hacia estructuras de control descentralizadas, con el fin de abordar estas limitaciones mediante la implementación de Sistemas Ciberfísicos (CPS) [2]. Los CPS desempeñan un papel importante en esta visión, facilitando el diseño de sistemas a gran escala dotados de funciones inteligentes, altamente automatizados y adaptables, basados en un conjunto de entidades distribuidas y autónomas que combinan componentes cibernéticos y físicos, los cuales interactúan y colaboran para alcanzar los objetivos del sistema.

Un desafío clave en la implementación de CPS es garantizar una integración y comunicación fluida entre estos componentes ciberfísicos, lo que a menudo conduce al desarrollo de soluciones *ad hoc*. Aunque estas soluciones pueden abordar parcialmente los problemas de integración, impiden el establecimiento de sistemas estandarizados, escalables e interoperables, lo que conlleva problemas de compatibilidad, mayor complejidad y costos de mantenimiento elevados, impidiendo en última instancia la adopción generalizada y la sostenibilidad a largo plazo de los sistemas basados en CPS. Para ayudar a abordar estos desafíos, se ha introducido el Modelo de Arquitectura de Referencia para la Industria 4.0 (RAMI4.0) [3]. RAMI4.0 es un modelo tridimensional que formaliza todos los aspectos esenciales relacionados con la digitalización de activos industriales, con el objetivo de proporcionar directrices comunes entre los participantes en el desarrollo de soluciones compatibles con la I4.0, basadas en estándares industriales. En el centro de RAMI4.0 se

encuentra el concepto del componente I4.0, una categoría específica de CPS que abarca un activo y su contraparte digital, el *Asset Administration Shell* (AAS) [4].

El AAS, introducido en 2015, funciona como una representación digital estandarizada de un activo físico o lógico, encapsulando toda la información relevante a lo largo del ciclo de vida del activo en submodelos estructurados [4]. Aproximadamente una década después, el AAS se ha convertido en un habilitador clave para la interoperabilidad en los entornos de I4.0, donde la integración y comunicación fluida entre diversos sistemas, dispositivos y plataformas es esencial. Actualmente, la *Industrial Digital Twin Association* (IDTA) es el organismo supervisor, proporcionando especificaciones esenciales y plantillas de submodelos estandarizadas para la digitalización de entornos industriales.

Según publicaciones recientes de la IDTA y trabajos de investigación existentes en la literatura [5], el enfoque sigue siendo consolidar y perfeccionar las soluciones tradicionales del AAS, es decir, las soluciones de Tipo 1 y Tipo 2, y proporcionar plantillas de submodelos estandarizadas. A pesar de que estas soluciones ofrecen información estandarizada sobre el activo e interfaces compatibles con I4.0 para garantizar la interoperabilidad fluida e intercambio de datos en el entorno industrial, es necesario mejorarlas para enfrentar escenarios industriales dinámicos y complejos que exigen un mayor nivel de autonomía e inteligencia por parte de los componentes I4.0 para la toma de decisiones y la colaboración.

De cara al futuro, la visión del AAS contempla el desarrollo y estandarización del AAS Tipo 3. En resumen, el AAS Tipo 3, en comparación con otros tipos, no solo actúa como un facilitador esencial de la interoperabilidad en los entornos I4.0, permitiendo la integración y comunicación fluida entre diferentes sistemas, dispositivos y plataformas, sino que también ofrece un mayor grado de autonomía, inteligencia y capacidades colaborativas. Estas características avanzadas son indispensables para los componentes I4.0 y desempeñan un papel crucial en la implemententación completa de la I4.0: ecosistemas industriales totalmente conectados, inteligentes, resilientes y adaptativos. Sin embargo, el avance hacia el AAS Tipo 3 se ve actualmente obstaculizado por la falta de especificaciones detalladas y documentación oficial. Esta carencia representa un desafío significativo para los desarrolladores e ingenieros en el diseño de tales componentes.

En este contexto, tecnologías como los Sistemas Multiagente (MAS) [6] representan un enfoque adecuado para hacer realidad el AAS Tipo 3, proporcionando la autonomía, inteligencia y capacidades colaborativas requeridas, atributos inherentes a los agentes software [6]. Los MAS pueden definirse como una sociedad de agentes inteligentes, autónomos y cooperativos que representan los objetos físicos o lógicos de un sistema. En estos sistemas, el comportamiento global emerge de la interacción entre los agentes individuales mediante la implementación de estrategias de interacción, como por ejemplo, colaboración y negociación. Diversas soluciones basadas en MAS [7] han demostrado los beneficios de su uso en entornos industriales, ofreciendo una alternativa para diseñar sistemas complejos y a gran escala mediante la descentralización de responsabilidades. Esto permite

superar los problemas típicos de los enfoques tradicionales de control centralizado, que no pueden proporcionar la flexibilidad, robustez y capacidad de reconfiguración requeridas por los sistemas industriales actuales [8].

Teniendo esto en cuenta, esta tesis se basa en dos aspectos clave que apuntan a un objetivo común: **(i)** abordar la falta de documentación oficial y la limitada investigación sobre el diseño del AAS Tipo 3, y **(ii)** proponer un enfoque de AAS basado en agentes para apoyar el diseño y desarrollo de soluciones de AAS Tipo 3.

En este contexto, esta tesis aborda el desafío del diseño y desarrollo del AAS Tipo 3, el cual va más allá de las funcionalidades convencionales del AAS tradicional al incorporar niveles más altos de autonomía, inteligencia y capacidades colaborativas para los componentes de la I4.0. Con base en ello, esta tesis plantea la siguiente hipótesis:

> Un enfoque basado en agentes puede permitir el diseño y desarrollo de una solución AAS Tipo 3. Esto posibilita la transición de los activos desde componentes pasivos hacia verdaderos componentes de la I4.0 (participantes dinámicos y activos del ecosistema industrial), equipados con capacidades mejoradas de gestión, interoperabilidad, autonomía, inteligencia y colaboración.

Con el fin de verificar esta hipótesis y alcanzar los objetivos de este trabajo, la tesis define dos preguntas de investigación (PI) que abarcan tanto el diseño teórico (PI1) como la aplicación práctica (PI2). Este enfoque dual, centrado en el diseño del sistema y en su funcionalidad operativa, asegura que la tesis contribuya tanto a la teoría académica como a la práctica industrial.

PI1: ¿Qué aspectos debe considerar un enfoque basado en agentes para diseñar y desarrollar una solución AAS Tipo 3?

Un enfoque basado en agentes para el diseño y desarrollo de una solución AAS Tipo 3 debe considerar varios aspectos clave para garantizar que, más allá de las funcionalidades comunes del AAS, se incorporen otras como autonomía, inteligencia y colaboración. En primer lugar, el enfoque debe mantener compatibilidad con el metamodelo oficial del AAS, permitiendo la incorporación de nuevas clases o módulos sin alterar la estructura central. Esto asegura el cumplimiento de las especificaciones del AAS y preserva la interoperabilidad con las soluciones existentes de Tipo 1 y Tipo 2, configurándose dichas extensiones como una capa superior. En segundo lugar, esta capa superior, representada por los agentes, no solo debe encargarse de tareas inteligentes y de comunicación, sino también interactuar directamente con el activo. Por ejemplo, debe permitir respuestas rápidas para la supervisión y el control continuo, interpretar los datos contenidos en el AAS y utilizar esta información para impulsar acciones y optimizaciones. Además, la adhesión a los estándares es crucial para garantizar una comunicación e integración fluidas en entornos industriales diversos.

PI2: ¿Cómo puede el enfoque AAS basado en agentes mejorar el proceso de digitalización en entornos industriales?

Aunque el enfoque propuesto tiene como objetivo guiar el diseño y desarrollo de una solución AAS Tipo 3, con niveles más altos de autonomía, inteligencia y capacidades colaborativas para los componentes de I4.0, primero debe entenderse como una solución holística para la I4.0. Esto significa que debe aportar valor en áreas fundamentales de I4.0, como son la interoperabilidad, la adaptabilidad, la conectividad y la robustez. A diferencia de la PI1, que se centra exclusivamente en el diseño del AAS Tipo 3, la PI2 profundiza en la funcionalidad operativa, la eficacia y los beneficios potenciales en diversos escenarios. Al investigar estos aspectos, la PI2 busca ofrecer una comprensión más profunda de cómo este enfoque puede mejorar el proceso de digitalización en los entornos industriales, particularmente en el contexto de la I4.0.

A.3 Resumen Significativos

A.3.1 Fundamentos Teóricos y Revisión de la Literatura

Este capítulo revisó la literatura existente e introdujo conceptos clave para proporcionar a los lectores el marco teórico necesario. La revisión sirvió como base para alcanzar el objetivo principal de este trabajo: desarrollar un enfoque basado en agentes para la implementación del AAS Tipo 3. Al explorar los avances actuales e identificar áreas de mejora en la literatura, este capítulo estableció el contexto, subrayó la relevancia y reveló el impacto potencial de integrar MAS para mejorar las capacidades del AAS.

La Sección 2.1 contextualizó y posicionó la base del trabajo mediante la exploración del concepto de I4.0 y los CPS. Su objetivo fue establecer una comprensión clara de la visión y los objetivos de I4.0, enfocándose en cómo impulsa la transformación digital en sector manufacturero y cómo los CPS pueden contribuir a la realización de dicha visión. La Sección 2.2 describió el papel de RAMI 4.0 en la provisión de un marco común entre los participantes en el desarrollo de soluciones compatibles con I4.0. Los puntos clave planteados en estas secciones incluyen: **(i)** a pesar del surgimiento de I5.0, este trabajo se mantiene en el contexto de I4.0, ya que sigue siendo una revolución en curso y aún está lejos de materializarse completamente; **(ii)** los CPS son un elemento crítico para alcanzar los objetivos y la visión de I4.0; **(iii)** los CPS requieren un enfoque estandarizado e interoperable para garantizar una integración y comunicación fluidas entre los componentes ciberfísicos, por ejemplo, a través del componente I4.0, una categoría de CPS alineada con el modelo RAMI4.0.

La Sección 2.3 realizó una caracterización y análisis de contenido mediante una búsqueda bibliográfica para examinar el AAS, particularmente en relación con las prácticas de

diseño e implementación del AAS Tipo 3. Este análisis ofreció una visión integral de las soluciones AAS existentes en la literatura, centrándose en la identificación de tecnologías habilitadoras clave, desafíos y oportunidades de investigación. De este proceso de revisión se obtuvieron varios hallazgos: **(i)** aunque el AAS es un tema relativamente reciente, está ganando rápidamente importancia tanto en la comunidad científica como en la industria, como un habilitador clave de interoperabilidad en I4.0; **(ii)** existe una fuerte conexión entre el AAS y el concepto de Gemelo Digital (DT), lo que con frecuencia genera debates sobre si el AAS constituye un DT o viceversa. Desde la perspectiva del autor, tales debates deben evitarse, enfatizando en cambio, la importancia de definir claramente los roles y funcionalidades de las soluciones propuestas de AAS o DT para prevenir interpretaciones erroneas; **(iii)** el AAS Tipo 3 sigue siendo un área poco explorada, pero investigaciones emergentes lo vinculan con MAS, en tanto en cuento éstos aportan autonomía, inteligencia y colaboración a las soluciones AAS; **(iv)** si bien los beneficios de las soluciones AAS basadas en agentes existentes son evidentes, es necesario abstraerlas a un nivel superior para generalizar más allá de sus aplicaciones iniciales. Esto puede implicar el desarrollo de una especificación detallada que guíe el diseño del AAS Tipo 3, asegurando que estos sistemas sean suficientemente genéricos, independientes de casos de uso específicos y capaces de apoyar diversas estrategias de colaboración.

La Sección 2.4 amplió los hallazgos de la búsqueda bibliográfica sobre el AAS, la cual identificó a los MAS como una tecnología clave para la implementación del AAS Tipo 3. Los MAS surgen como un habilitador fundamental para mejorar la autonomía, inteligencia y capacidades colaborativas de las soluciones AAS. Su capacidad para apoyar la toma de decisiones descentralizada e interacciones dinámicas se alinea estrechamente con la visión y los requisitos del AAS Tipo 3, convirtiendo a los MAS en un enfoque fundamental para el desarrollo de estos componentes. Desde las primeras arquitecturas holónicas hasta los modernos sistemas basados en agentes, los MAS han evolucionado significativamente, especialmente con la llegada de I4.0 y los CPS. Las soluciones MAS revisadas en la literatura [7, 8, 59, 60, 71–73] comparten características comunes, a saber: adaptabilidad, robustez, escalabilidad, reconfigurabilidad, reutilización, modularidad, capacidad de respuesta y diseño de sistemas CPS complejos a gran escala. Además, los MAS han sido respaldados por comités técnicos internacionales que buscan fomentar la aplicación de agentes industriales en escenarios reales, por ejemplo, a través de iniciativas de estandarización, promoviendo así la adopción generalizada de soluciones basadas en MAS en entornos industriales.

La Sección 2.5 destacó las principales conclusiones obtenidas del capítulo, sirviendo como punto de partida para la especificación del enfoque propuesto. En este contexto, esta sección adoptó dos perspectivas complementarias para esclarecer las relaciones entre MAS, AAS Tipo 3 y RAMI4.0. La primera perspectiva comparó los agentes y el AAS Tipo 3, destacando la sinergia entre ambos conceptos. La segunda perspectiva se alineó más estrechamente con RAMI4.0, enfatizando cómo los agentes podrían potenciar el proceso de

digitalización a lo largo de las capas definidas por RAMI4.0, las cuales están directamente relacionadas con el AAS.

A.3.2 Propuesta Basada en Agentes para el Diseño de AAS Tipo 3

Este capítulo presentó la especificación del enfoque AAS basado en agentes para el diseño de soluciones AAS Tipo 3. Este enfoque permite la transición de componentes pasivos a verdaderos componentes de I4.0, con capacidades mejoradas de gestión, interoperabilidad, autonomía, inteligencia y colaboración.

La falta de especificaciones y la limitada investigación relacionada con el diseño del AAS Tipo 3 representa un desafío en el desarrollo de dichos componentes, que van más allá de las funcionalidades convencionales de las soluciones AAS Tipo 1 y 2, al incorporar características más sofisticadas que permiten a los componentes I4.0 operar con mayor autonomía, inteligencia y capacidades colaborativas. En este contexto, se propuso un metamodelo de información (Sección 3.1) para abordar este problema mediante una especificación para el diseño del AAS Tipo 3 a través de un enfoque basado en agentes. El metamodelo de información consta de dos partes: la parte informativa y la parte inteligente. La parte informativa incluye las clases principales para describir la información del activo descritos en los submodelos, y la parte inteligente incluye las clases principales para dotar al AAS de autonomía, inteligencia y capacidades colaborativas proporcionadas por el agente.

La Sección 3.2 presentó el modelado de la parte informativa del AAS basado en agentes, describiendo algunos submodelos fundamentales que permiten a este tipo de AAS gestionar e interactuar eficazmente con sus respectivos activos en el entorno industrial, así como colaborar entre sí. Estos submodelos cumplen con las especificaciones proporcionadas por la IDTA para garantizar una mayor interoperabilidad, y están principalmente basados en los conceptos de capacidades y habilidades, que se especifican formalmente en el modelo de referencia Capacidad-Habilidad-Servicio (CSS) [89].

Las secciones restantes se centraron en la parte inteligente del AAS basado en agentes. La Sección 3.3 presentó la integración entre la parte inteligente y la parte informativa, destacando dos enfoques. El primer enfoque se basa en la parte 2 de la especificación AAS (versión 3.0), que define APIs para acceder a la información del activo almacenada en submodelos. El segundo enfoque se basa en el estándar "OPC 30270: Industry 4.0 Asset Administration Shell", que proporciona directrices para mapear el metamodelo AAS al modelo de información OPC UA. La Sección 3.4 describió los patrones de integración para la conexión de AAS basados en agentes con activos físicos, utilizando las prácticas de interfaz genérica proporcionadas por el estándar IEEE 2660.1, así como la aplicabilidad del método de recomendación IEEE 2660.1 en el enfoque propuesto. La Sección 3.5 presentó el vocabulario y la estructura de los mensajes, junto con algunos ejemplos de patrones de

interacción entre AAS basados en agentes, basados en los estándares VDI/VDE 2193 y FIPA. Finalmente, la Sección 3.6 abordó algunos ejemplos y un enfoque para implementar inteligencia en AAS individuales basados en agentes.

A.3.3 Implementación Experimental y Validación

Este capítulo presentó la implementación y los experimentos realizados para evaluar el enfoque propuesto, demostrando su aplicabilidad y sus contribuciones al diseño y desarrollo de soluciones AAS Tipo 3 mediante un enfoque basado en agentes.

La Sección 4.1 describió la metodología de evaluación del enfoque AAS basado en agentes, definiendo los escenarios de evaluación, los criterios de evaluación y los casos de estudio. La Sección 4.1.1 describió escenarios que reflejan distintos aspectos de la manufactura que requieren colaboración entre activos, incluyendo la asignación de recursos, la ejecución de operaciones y la reconfiguración del sistema debido a eventos inesperados. La Sección 4.1.2 presentó los criterios de evaluación, centrados en aspectos cualitativos como la adaptabilidad, conectividad, robustez, interoperabilidad y reutilización, los cuales son requisitos clave para la implementación de soluciones compatibles con I4.0. La Sección 4.2 describió los dos casos de estudio utilizados para implementar, probar y validar el enfoque propuesto: el primero basado en un prototipo de laboratorio (sistema de producción a pequeña escala) y el segundo en una línea de producción automotriz industrial.

La Sección 4.3 detalló el proceso de implementación de un AAS individual basado en agentes, en particular para implementar las clases principales descritas en el metamodelo de información. La parte informativa fue implementada utilizando la herramienta Eclipse AASX Package Explorer, un visor y editor basado en C# para el desarrollo de AAS y sus elementos, específicamente los submodelos (Sección 4.3.1). La parte inteligente (agente) fue implementada utilizando el framework JADE, que ofrece una plataforma eficiente de agentes y admite el desarrollo de MAS conforme a las especificaciones FIPA. En este contexto, JADE gestionó la infraestructura de software para las operaciones de los agentes, mientras que la parte informativa (submodelos) y el activo mantuvieron sus propias plataformas de gestión (Sección 4.3.2). Se adoptó el servidor Eclipse AASX como un enfoque habilitado por API, proporcionando interfaces estándar para que los agentes obtengan información de sus respectivos AAS mediante el protocolo de comunicación HTTP/REST (Sección 4.3.3). Finalmente, se aplicó el método de recomendación IEEE 2660.1 para identificar la mejor práctica de interfaz para interactuar con los activos en los dos casos de estudio (Sección 4.3.4).

Las Secciones 4.4 y 4.5 presentaron el despliegue del enfoque AAS basado en agentes en ambos casos de estudio para implementar los escenarios definidos. Para cada activo en el sistema, incluidos tanto productos como recursos, se diseñó e implementó un AAS basado en agentes. Las interacciones entre productos y recursos a través del AAS Tipo 3

siguieron los modelos de comportamiento y los protocolos de interacción basados en FIPA y el lenguaje I4.0 especificado para cada escenario, garantizando que se cumplieran los requisitos específicos de cada uno.

Finalmente, la Sección 4.6 evaluó el enfoque propuesto mediante una doble aproximación. Primero, se examinó si el enfoque propuesto cumple con los requisitos para que un AAS se considere de Tipo 3, lo cual incluye autonomía, inteligencia y colaboración. Segundo, se evaluó el AAS basado en agentes según los criterios cualitativos definidos. Los experimentos y esta evaluación demostraron que el enfoque basado en agentes propuesto logra con éxito la implementación de un AAS Tipo 3, permitiendo que los activos pasen de ser componentes pasivos a verdaderos componentes I4.0 con capacidades mejoradas de autonomía, inteligencia y colaboración. Además, el enfoque aborda los requisitos esenciales para I4.0, como la adaptabilidad, la conectividad, la robustez, la interoperabilidad y la (re)usabilidad.

A.4 Conclusiones

Al comienzo de esta tesis, se formó una percepción inicial, ampliamente aceptada en la comprensión general dentro del contexto de la I4.0, de que los MAS podrían servir como un enfoque adecuado para implementar la AAS Tipo 3, debido a sus características inherentes y sus diversas aplicaciones en el contexto de los CPS. En este sentido, se definió la siguiente hipótesis: Un enfoque basado en agentes puede permitir el diseño y desarrollo de una solución AAS Tipo 3. Esto posibilita la transición de los activos desde componentes pasivos hacia verdaderos componentes de la I4.0 (participantes dinámicos y activos del ecosistema industrial), equipados con capacidades mejoradas de gestión, interoperabilidad, autonomía, inteligencia y colaboración.

Si bien esta hipótesis parecía intuitiva, dado el solapamiento entre las características de los MAS y los requisitos del AAS Tipo 3, primero requería una base teórica sólida y un análisis exhaustivo de la literatura existente para confirmar su validez, así como para evaluar el grado de innovación y la contribución que podría aportar al campo. Este fundamento era esencial para proponer posteriormente una solución al problema identificado y para validar la hipótesis mediante una especificación e implementación experimental.

El Capítulo 2, en particular, presentó investigaciones que abordan varios aspectos relacionados con el AAS Tipo 3. Esta revisión destacó dos brechas críticas: la falta de una especificación integral para el diseño de soluciones AAS Tipo 3 y los MAS como una tecnología clave habilitadora para su implementación. Se concluyó que, si bien la literatura incluye enfoques AAS basados en agentes, estos deben adaptarse a un nivel más alto de abstracción para garantizar su aplicabilidad en diversos escenarios, asegurando que sean suficientemente genéricos, independientes de casos de uso específicos y capaces de

soportar diversas estrategias de colaboración. Además, un análisis de varios enfoques CPS basados en MAS en la literatura y en proyectos de I+D demostró los beneficios potenciales de los MAS en entornos industriales, reforzando su papel como tecnología habilitadora clave para la implementación del AAS Tipo 3. Con base en esta revisión de literatura, el Capítulo 2 concluyó examinando los paralelismos entre los MAS, el AAS Tipo 3 y el modelo RAMI4.0, ofreciendo ideas para el desarrollo de la especificación del diseño y desarrollo de una solución AAS Tipo 3 utilizando MAS.

A partir de las oportunidades de investigación identificadas en la revisión bibliográfica, el Capítulo 3 presentó el trabajo desarrollado para responder a la primera pregunta de investigación de esta tesis: **(PI1)** *¿Qué aspectos debe considerar un enfoque basado en agentes para diseñar y desarrollar una solución de AAS Tipo 3?* Para abordar esta pregunta, se propuso un metamodelo de información como solución, ofreciendo una especificación para diseñar AAS Tipo 3 mediante un enfoque basado en agentes, considerando varios aspectos clave. En primer lugar, el enfoque mantiene la compatibilidad con el metamodelo oficial del AAS, permitiendo la adición de nuevas clases o módulos sin alterar la estructura principal. Esto asegura la conformidad con las especificaciones del AAS y preserva la interoperabilidad con las soluciones existentes AAS Tipo 1 y 2, funcionando estas adiciones como una extensión en una capa superior. En segundo lugar, esta extensión en forma de capa superior, representada por los agentes, no solo debe manejar tareas inteligentes y de comunicación, sino también interactuar directamente con el activo, por ejemplo, para permitir una respuesta rápida en la monitorización y control continuo, interpretar los datos contenidos en el AAS y utilizar esta información para impulsar acciones y optimizaciones.

Teniendo esto en cuenta, el metamodelo de información consta de dos partes: la parte informativa y la parte inteligente. La parte informativa incluye las clases principales necesarias para describir la información del activo dentro de los submodelos, mientras que la parte inteligente abarca las principales clases diseñadas para dotar al AAS de capacidades de autonomía, inteligencia y colaboración proporcionadas por el enfoque basado en agentes. El funcionamiento de cada una de estas clases fue discutido individualmente, presentando estándares de apoyo para su desarrollo, así como ejemplos ilustrativos para demostrar su aplicación.

Con base en el desarrollo del enfoque AAS basado en agentes propuesto para guiar el diseño de soluciones AAS Tipo 3 utilizando MAS, el Capítulo 4 describió el trabajo desarrollado para validar el enfoque propuesto como una solución AAS Tipo 3 y responder a la segunda pregunta de investigación de esta tesis: **(PI2)** *¿Cómo puede el enfoque AAS basado en agentes mejorar el proceso de digitalización en entornos industriales?* Para responder a ello, se implementó y probó el enfoque propuesto en dos casos de estudio. El primer caso de estudio involucró un prototipo de laboratorio que representaba un sistema de producción a pequeña escala, mientras que el segundo se centró en una línea de producción automotriz industrial. Para ambos casos de estudio, se diseñaron e implementaron AAS basados en

agentes para mejorar los sistemas existentes con capacidades de autonomía, inteligencia y colaboración. Estas implementaciones demostraron cómo el enfoque propuesto puede aplicarse en diversos escenarios industriales, destacando su adaptabilidad, conectividad, robustez, interoperabilidad y reutilización, que son requisitos importantes de I4.0. Los resultados de esta implementación experimental proporcionaron valiosos conocimientos prácticos sobre las ventajas de integrar MAS para implementar el AAS Tipo 3. Además, validaron la viabilidad y efectividad de la solución propuesta para avanzar en el proceso de digitalización en entornos industriales.

Como se discutió en las respuestas a las preguntas de investigación anteriores, los resultados obtenidos en este trabajo contribuyeron al cumplimiento de los objetivos de esta tesis y a la confirmación de la hipótesis definida. Al abordar las áreas susceptiobles de mejora identificadas y desarrollar el enfoque AAS basado en agentes propuesto, este trabajo demostró la viabilidad, aplicabilidad y beneficios de utilizar un enfoque basado en agentes para permitir el diseño y desarrollo de una solución AAS Tipo 3. A través del desarrollo teórico y la implementación experimental en dos estudios de caso, los hallazgos afirmaron la hipótesis inicial y contribuyeron al avance del estado del arte en este dominio.