NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF ELECTRICAL AND COMPUTER ENGINEERING

RAFAEL VALÉRIO TRAQUINO FERREIRA

BSc in Eletrical and Computer Engineering

DYNAMIC SCHEDULING FOLLOWING RAMI 4.0 REFERENCE ARCHITECTURE

MASTER IN Eletrical and Computer Engineering

NOVA University Lisbon

September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENT OF
ELETRICAL AND COMPUTER ENGINEERING

DYNAMIC SCHEDULING FOLLOWING RAMI4.0 REFERENCE ARCHITECTURE

RAFAEL VALÉRIO TRAQUINO FERREIRA

Bsc in Eletrotechnical and Computer Engineering

Adviser: André Dionísio Rocha
Full Professor, NOVA University Lisbon

Co-advisers: Duarte Alemão
MSc, NOVA University Lisbon

MASTER IN <Eletrical and Computer Engineering>

NOVA University Lisbon
September, 2024

To Patrícia, my parents and everyone that made this possible

ACKNOWLEDGMENTS

In this section, I would like to express my heartfelt gratitude to those who have supported and guided me throughout this journey.

First of all I would like to thank my supervisor, PhD Professor André Rocha, for proposing this project and giving me the opportunity to work on it. His support and guidance provided me with not only a rewarding challenge but also the best workplace experience I have ever had.

I am also grateful for my co-supervisor, MSc Duarte Alemão. For always being available whenever I found myself a bit lost, for his enormous patience, and constant feedback, without him this thesis and its success would not have been possible.

To my friends, Mauro, Chia, Luís, Alex and André who were also "struggling" through their own theses and responsibilities alongside me — we made it! Your camaraderie and encouragement were a source of strength during the most challenging times. It's been an incredible journey, and I'm grateful to have experienced it with all of you.

I also want to thank my family, mainly my grandparents for always being present. I know you were always worried about me and the progress of my thesis, and your dedication and care did not go unnoticed. I am profoundly grateful for everything you've done for me.

I am indebted to you Patrícia. This could not have been done without you. Your unwavering support made it possible to get here and made me who I am today. You were my constant source of strength, always believing in me, even when I doubted myself. Thank you.

To the best parents I could have asked for. Your love and encouragement laid the foundation for everything I have ever accomplished to this day. I owe all of my achievements to the values you instilled in me and the endless love you've shown me. Words cannot express how grateful I am for everything you've done for me.

I owe everything to you

Thank you

RESUMO

A Indústria 4.0 transformou a indústria de manufactura de sistemas lineares para sistemas altamente interconectados, através da digitalização e integração de tecnologias avançadas. Esta transformação resultou num aumento da eficiência, na redução de desperdícios e na possibilidade de produzir produtos personalizados em escala. No entanto, a integração destas tecnologias apresenta desafios, como a interoperabilidade fluida, a gestão de grandes quantidades de dados gerados por dispositivos IoT, cibersegurança e o desenvolvimento de novas competências da força de trabalho.

Outro desafio da indústria tem sido a alocação de tarefas para uma otimização do processo de produção. Esse desafio é ainda mais acentuado na I4.0, uma vez que os métodos tradicionais têm dificuldade em lidar com a natureza dinâmica e imprevisível da manufactura moderna. Esta tese propõe um sistema para a execução e monitorização dinâmica de tarefas, seguindo as práticas RAMI 4.0. O sistema inclui um algoritmo para uma alocação eficiente de tarefas, uma camada middleware para a comunicação e troca de dados, e Asset Administration Shells (AAS) para fornecer dados detalhados sobre capacidades, requisitos e tempos de processamento. A Base de Dados AAS consolida esta informação e encaminha-a para o Escalonador, que depois gera cronogramas de produção otimizados, aumentando a eficiência, flexibilidade e adaptabilidade nas operações de manufatura. Esta tese pretende contribuir para o conhecimento sobre programação dinâmica na Indústria 4.0 e fornecer uma solução prática para que os fabricantes possam navegar pelas complexidades dos processos de manufatura modernos.

Keywords: Alocação de tarefas, Indústria 4.0, Asset Administration Shells, Cronogramas de produção otimizados, RAMI 4.0

ABSTRACT

Industry 4.0, the digitization and integration of advanced technologies, has transformed the manufacturing industry from linear to highly interconnected systems. This has led to increased efficiency, reduced waste, and customized products at scale. However, the integration of these technologies presents challenges such as seamless interoperability, managing vast amounts of data generated by IoT devices, cybersecurity, and workforce upskilling.

Another challenge in the industry has been the allocation of tasks for optimizing the production process. This challenge is even more pronounced in Industry 4.0, as traditional methods struggle to cope with the dynamic and unpredictable nature of modern manufacturing. This thesis proposes a system for dynamically executing and monitoring tasks, following RAMI 4.0 practices. The system includes an algorithm for efficient task allocation, a middleware layer for communication and data exchange, and Asset Administration Shells (AAS) for detailed data on capabilities, requirements, and processing times. The AAS Database consolidates this information,forwards it to the Scheduler that then generates optimized production schedules, enhancing efficiency, flexibility, and adaptability in manufacturing operations. This thesis aims to contribute to the knowledge on dynamic scheduling within Industry 4.0 and provide a practical solution for manufacturers to navigate the complexities of modern manufacturing processes.

Keywords: Scheduling, Industry 4.0, Asset Administration Shells, Optimized production schedules, RAMI 4.0

Contents

LIST OF FIGURES

List of Tables

ACRONYMS

AAS	Asset Administration Shell.
AI	Artificial Intelligence.
CPS	Cyber-physical Systems.
CRM	Customer Relationship Management.
DER	Distributed Energy Resources.
DJSSP	Dynamic Job-Shop Scheduling Problems.
ERP	Enterprise Resource Planning.
I4.0	Industry 4.0.
ICS	Industry Control Systems.
IIC	Industrial Internet Consortium.
IIoT	Industrial Internet of Things.
IIRA	Industrial Internet Reference Architecture.
IoE	Internet of Everything.
IoT	Internet of Things.
JSSP	Job-Shop Scheduling Problem.
RAMI 4.0	Reference Architecture Model for Industry 4.0.
SGAM	Smart Grid Architecture Model.

1

INTRODUCTION

Over the past century, the manufacturing industry has transformed significantly, evolving from simple, linear production processes to highly interconnected, dynamic systems. This evolution has been driven by the advent of Industry 4.0, a concept introduced at the Hannover Messe Industrie Fair in 2011, which represents the digitization and integration of advanced technologies within the manufacturing sector. These technologies include artificial intelligence (AI), the Internet of Things (IoT), and cloud computing, all of which contribute to creating a "smart factory" environment where real-time data exchange, autonomous decision-making, and flexible production capabilities are possible.

Industry 4.0 enables manufacturers to achieve unprecedented levels of efficiency, reduce waste, and deliver highly customized products at scale. One of the core benefits of Industry 4.0 is its ability to adapt both software and hardware systems to meet fluctuating market demands, allowing for mass customization and even the production of unique, individualized items. However, the integration of these advanced technologies into manufacturing processes is not without its challenges.

Among the most pressing challenges is the need for seamless interoperability between diverse systems and machines, ensuring secure and reliable data exchange across various platforms. Additionally, there is the challenge of managing vast amounts of data generated by IoT devices and sensors, requiring robust data analytics and storage capabilities. Cybersecurity also becomes critical as increased connectivity exposes manufacturing networks to potential threats. Furthermore, there are significant workforce implications, including the need to upskill employees to work alongside advanced technologies, adapt to new roles, and collaborate in increasingly automated environments.

Another crucial challenge arises in the area of scheduling. Scheduling, in the context of manufacturing, involves the allocation of resources—such as machines, materials, and labor to specific tasks within a given timeframe to optimize production efficiency. Traditional scheduling methods, often static and based on predefined rules, struggle to cope with the dynamic and unpredictable nature of modern manufacturing. These methods are ill-suited to the real-time adjustments and rapid decision-making required in an Industry 4.0 setting, where production lines must be flexible enough to handle frequent changes in product type, volume, and custom specifications.

1.1 Main Approach

To address these challenges, there is a need for a more advanced approach to dynamic scheduling that can effectively manage the intricate interplay of machines, materials, and human labor on the shop floor. This thesis proposes the design and implementation of a system for dynamically executing and monitoring tasks, following the principles outlined in the RAMI 4.0 practices. The system integrates several advanced components to enhance scheduling capabilities.

At its core, the scheduler employs a straightforward algorithm to assign products from the product order to appropriate machines efficiently. This algorithm optimizes the allocation of tasks by considering machine capabilities and minimizing production time. The scheduler manages resource allocation over time, optimizing task sequences while accounting for constraints such as priority levels and dependencies. It integrates seamlessly with a middleware layer, which facilitates communication and data exchange between the scheduler, external sources, and databases. This middleware ensures real-time updates and dynamic adjustments to production schedules based on the latest information.

Machines, products, and the scheduler on the shop floor are represented by Asset Administration Shells (AAS), which provide detailed data on capabilities, requirements, and processing times. The AAS Database consolidates this information, serving as a central repository that supports real-time updates and scalable integration. By leveraging this comprehensive data, the scheduler generates optimized production schedules, enhancing efficiency, flexibility, and adaptability in manufacturing operations.

1.2 Research Question and Hypothesis

The study will explore how the integration of RAMI 4.0 and advanced scheduling practices can enhance operational efficiency and flexibility in the context of Industry 4.0. The research question for this thesis is:

How can the integration of RAMI 4.0 and advanced dynamic scheduling practices enhance operational efficiency and flexibility in Industry 4.0 manufacturing environments?

By developing a system, integrating a scheduling algorithm, a real-time data-exchange middleware, and digital Asset Administration Shells (AAS) for machines and products, the system enables adaptive task allocation based on machine capabilities and shop floor conditions, ensuring flexible, optimized production that adjusts instantly to changes in demand.

Ultimately, this thesis aims to contribute to the body of knowledge on dynamic scheduling within Industry 4.0, proposing a practical solution to help manufacturers navigate the complexities of modern manufacturing processes.

1.3 Dissertation Structure

Chapter 1 introduces the scope of the work, highlighting the problem to be addressed and outlines the proposed solution.

Chapter 2 reviews the state of the art, outlining the problem addressed in the thesis and highlighting previous research in the area. It includes definitions such as Industry 4.0, RAMI 4.0, and Administration Asset Shell, which are central to the work. These concepts provide the necessary context and foundation for the research.

Chapter 3 presents the architecture of the developed system. It explains each of the system's components along with an overview of the whole scheduling process.

Chapter 4 corresponds to the implementation of the scheduling system. This chapter outlines the development of the scheduling system components and their interactions, and also offers an insightful explanation of the scheduling algorithm developed for this work.

Chapter 5 is dedicated to the tests and results. It presents the various tests conducted, the results obtained, and the conclusions drawn from them.

Chapter 6 outlines the conclusions derived from analyzing the results and offers recommendations for additional future work to improve the system.

2

STATE OF ART

Over the past century, the manufacturing industry has evolved significantly. Early manufacturing relied on manual labor and simple machinery, producing goods in small batches. With the advent of electricity, production lines emerged, allowing for mass production and greater efficiency by dedicating specific lines to individual product types. Although the introduction of programmable machines brought some flexibility to the process, the range of products remained limited. The true revolution came with the advent of Industry 4.0 technologies, which integrated automation, data exchange, and smart systems, enabling highly flexible and efficient production environments capable of adapting to diverse and complex product demands.

The first revolution (1760-1840) shifted manufacturing from manual labor to mechanization, driven by steam engines and innovations like the spinning jenny, and marked the rise of the factory system, sparking urbanization and significant societal changes. The second revolution (1870-1914) brought electric power and mass production, with breakthroughs such as the internal combustion engine and assembly lines, which advanced industries like steel and automobiles, fostering economic growth and the rise of large corporations. The third revolution (1969-2015) ushered in the digital age, driven by the invention of transistors, microprocessors, and the internet, which transformed data processing, communication, and global connectivity, giving rise to the digital economy and highlighting the digital divide (Armenta A., 2021; George & George, 2020; Kumar et al., 2020; Pilevari & Yavari, 2021).

Today, Industry 4.0 empowers systems to self-adapt hardware and software components to meet evolving market demands in terms of product type and quantity,

enabling mass customization down to individual product specifications (George, 2024; Patnaik A., 2023a; Shen & Norrie, 1999).

Figure 1 showcases the Industrial revolutions necessary for the evolution of manufacturing.

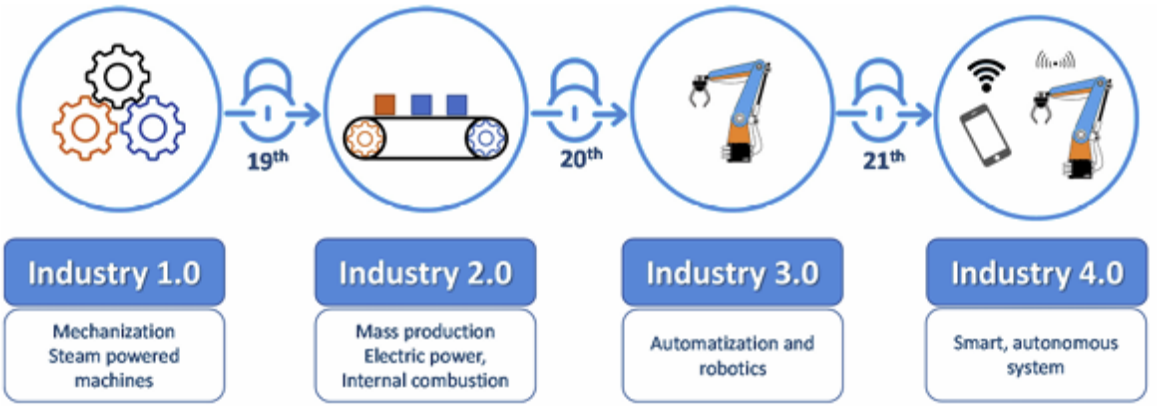

Figure 1 - The evolution of manufacturing as per (Fiedor & Ortyl, 2020).

Industry 4.0 transforms manufacturing by seamlessly integrating various aspects of the process. It connects different levels within a facility, from sensors on machines to overarching management systems, allowing real-time data flow and more informed decision-making. This connectivity also extends beyond a single plant, linking multiple factories, suppliers, and distribution networks, resulting in a highly coordinated and efficient supply chain.

Smart production processes use advanced technologies like IoT and AI to automate and optimize operations. Machines can monitor production in real time, adjust operations to increase efficiency, predict maintenance needs, and respond dynamically to changes. Intelligent products, embedded with sensors and connected to the network, communicate with production systems, providing data on their condition, use, and performance. This continuous feedback loop not only improves the manufacturing process but also helps tailor products to meet evolving customer needs, creating a more adaptive and efficient production environment.

Horizontal integration involves linking various systems to facilitate communication across all stages of the supply chain. This encompasses the interconnection between logistics, production, and design, both within individual companies and across different organizations.

Conversely, vertical integration pertains to the coordination of systems across different hierarchical levels within an organization, ranging from production to management (Benitez et al., 2019).

2.1 Industry 4.0

The term "Industrie 4.0," coined at the Hannover Messe Industrie Fair in 2011, represents a pivotal initiative by the German government to drive the digitization of the manufacturing sector (Kagermann et al., 2011; Smit et al., 2016). At its heart, Industry 4.0 introduces a new paradigm in industrial automation by integrating advanced technologies like cyber-physical systems (CPS), the Internet of Things (IoT), cloud computing, and Digital Twins. These technologies collectively enable the development of smart factories, where machines, systems, and humans interact seamlessly, fundamentally transforming traditional manufacturing processes and creating new business models (Wang et al., 2016; Zhong et al., 2017)(Roda-Sanchez et al., 2018; Wilkesmann & Wilkesmann, 2018; Yao et al., 2019).

The core of Industry 4.0 lies in the structuring of manufacturing processes around technologies that allow autonomous communication and interaction throughout the entire value chain models (Wang et al., 2016; Zhong et al., 2017). In this vision of the "smart factory," computer-controlled systems closely monitor physical processes, create a Digital Twin of the physical environment, and make decentralized decisions using self-organizing mechanisms. This enables a highly responsive, agile, and efficient production environment, transforming manufacturing systems to be both vertically integrated with internal business operations and horizontally linked across geographically distributed networks (George, 2024).

Central to this transformation is the IoT, which connects various devices and systems within the manufacturing ecosystem, facilitating real-time data exchange and communication. This connectivity enables continuous monitoring and control of manufacturing processes, resulting in improved decision-making and operational efficiency (Saqlain et al., 2019; H. Wu et al., 2021). When combined with cloud computing, IoT significantly enhances this capability by providing scalable data storage and processing power, which is essential for managing the vast amounts of data generated in smart factories (Ivkic et al., 2017). This integration is a key driver in the development of CPS, systems that merge physical processes with computational elements, allowing for autonomous operation and real-time responsiveness (Dai et al., 2019; Yao et al., 2019).

The concept of the Digital Twin is another crucial element of Industry 4.0. A Digital Twin serves as a virtual replica of a physical entity, enabling manufacturers to simulate, analyze, and optimize production processes without interrupting actual operations (Azangoo et al., 2021; Lu et al., 2020). This technology supports predictive maintenance and enhances the adaptability of manufacturing systems to rapidly changing market demands, allowing

companies to achieve higher efficiency, reduce downtime, and maintain a competitive edge (Kalsoom et al., 2020; Lu et al., 2020; Qi et al., 2018; Resman et al., 2021).

Moreover, the convergence of IoT, CPS, and cloud computing under Industry 4.0 blurs the lines between traditional goods and services, giving rise to hybrid products that combine both. This fusion promotes new business models centered on real-time data analytics and service-oriented architectures, driving the digital transformation of industries (Saqlain et al., 2019; Seiger et al., 2020). As a result, organizations are not only enhancing their operational capabilities but also redefining their value propositions, fostering innovation and competitiveness in a rapidly evolving market landscape (Dave et al., 2023; Quan Chong et al., 2018).

In essence, Industry 4.0 represents a comprehensive reimagining of manufacturing, leveraging the integration of IoT, CPS, cloud computing, and Digital Twins to enable smart factories to operate with unparalleled levels of efficiency, flexibility, and innovation. This transformation is fostering new business models and bolstering competitiveness in the global market, reflecting a fundamental shift towards a more connected and intelligent industrial ecosystem axes(Kagermann et al., 2011; Roda-Sanchez et al., 2018; Smit et al., 2016; Wilkesmann & Wilkesmann, 2018; Yao et al., 2019).

2.1.1 Design principles of Industry 4.0

Central to this transformation are specific design principles that guide the development and implementation of Industry 4.0 strategies. These principles have been defined and categorized in various ways across scientific literature, reflecting different emphases and perspectives. This section explores two widely cited categorizations of Industry 4.0 design principles and discusses their implications and applications in contemporary research (Axelsson et al., 2019; Hall et al., 2022).

One prominent categorization of Industry 4.0 (Hermann et al., 2016) design principles identifies four key principles as shown in Table 1:

Principle	Description
Interconnectivity	This principle refers to the seamless connectivity between entities and systems through the IoT (Giusto et al., 2010). This connectivity enables the exchange of information and the execution of improvements based on that information.
Information	Integrating physical and virtual contexts creates a virtual replica of

Transparency	the physical world, enabling novel information transparency. This fusion provides context-aware information essential for knowledge workers, enhancing task management, collaboration, and decision-making with real-time, process-critical data.
Decentralized decisions	Decentralized decision-making in an Internet of Everything (IoE) ecosystem combines interconnectedness and autonomy, enabling entities to use both local and global information for more effective decisions. This approach allows participants to operate autonomously, fostering a dynamic system that adapts to changing conditions.
Technical Assistance	As production processes become more complex, the need for advanced support systems to assist humans is growing. These systems play a critical role in aggregating and visualizing information, making it easier for human operators to understand and manage intricate production environments.

Table 1 - Industry 4.0 design principles as per (Hermann et al., 2016)

Another categorization of the design principles of Industry 4.0 details six principles as described in Table 2 (Hermann et al., 2015; Smit et al., 2016):

Principle	Description
Interoperability	Refers to the ability of systems, devices, and applications to connect and communicate seamlessly. In Industrial IoT (IIoT), interoperability is essential for a cohesive and efficient environment, allowing devices to exchange data and enhance operational efficiency, sustainability, and decision-making.
Virtualisation	Involves creating digital twins real-time virtual models of physical entities like products or factories. These digital twins bridge the physical and digital worlds, allowing businesses to simulate, analyze, and optimize processes virtually before applying changes in the real world.
Decentralisation	In CPS, this capability enhances autonomous decision-making and task execution, improving efficiency and flexibility in production. It is vital for achieving Industry 4.0 goals, which focus on integrating digital and physical elements in manufacturing.

Real-Time Capability	This is the ability to gather and analyze data instantaneously, which is vital for immediate decision-making and quick resolution of issues.
Service Orientation	This refers to the availability of on-demand services (for humans or machines) within the Industry 4.0 framework.
Modularity	This allows for the swift adaptation of smart factories to changing needs by modifying or expanding individual modules.

Table 2 - Industry 4.0 design principles as per (Hermann et al., 2015)

In summary, while both sets of principles promote connectivity and decentralized operations, the framework shown in Table 1 is more focused on supporting human operators and enhancing transparency, whereas the framework presented in Table 2 places a greater emphasis on technological flexibility, real-time responsiveness, and service-centric operations. Together, these principles offer a holistic approach to achieving the goals of Industry 4.0, balancing human and technological needs in a dynamic industrial landscape.

2.2 Reference Architectures for Industry 4.0

In the rapidly evolving landscape of Industry 4.0, the need for standardized frameworks that facilitate seamless integration and interoperability among various technologies has become paramount. This section delves into the concept of Reference Architectures within the realm of Industry 4.0, highlighting their significance in shaping the future of manufacturing and production.

Reference architectures serve as blueprints or templates that guide the design, development, and implementation of systems and solutions tailored to the demands of Industry 4.0. They play a crucial role in ensuring that the complex interplay between physical and digital elements, commonly referred to as CPS, is harmoniously integrated. This integration is vital for achieving the full potential of the fourth industrial revolution, characterized by its emphasis on automation, data exchange, machine learning, and IoT.

Below is a general overview of these reference architectures:

IIRA (Industrial Internet Reference Architecture) is a standards-based architectural template and methodology designed by the Industrial Internet Consortium (IIC), assisting system architects to design their own industry systems (Leitão et al., 2023).

SGAM (Smart Grid Architecture Model) is a three-dimensional framework that allows the representation of all possible Smart Grid use cases and their actors in the grid and market domains. It was developed by the CEN-CENELEC-ETSI Smart Grid Coordination Group (Leitão et al., 2023).

RAMI 4.0 (Reference Architecture Model for Industry 4.0) is a three-dimensional map showing how to approach the issue of Industry 4.0 in a structured manner. It was developed by the German Electrical and Electronic Manufacturers' Association (ZVEI) to support Industry 4.0 initiatives (Leitão et al., 2023; Patnaik A., 2023b)

2.2.1 IIRA Overview

The Industrial Internet of Things (IIoT) has emerged as a transformative technology, reshaping industries by integrating physical devices with digital connectivity. As IIoT systems become increasingly complex, managing their development requires a structured approach to address various aspects of the system lifecycle. To this end, the Industry Integration Reference Architecture (IIRA) provides a comprehensive framework for designing IIoT systems. The IIRA methodology divides the design considerations into four critical viewpoints, as seen in figure 2 each focusing on a unique aspect of the system's development. This structured approach ensures that all necessary elements are considered, facilitating the creation of robust and efficient IIoT solutions.

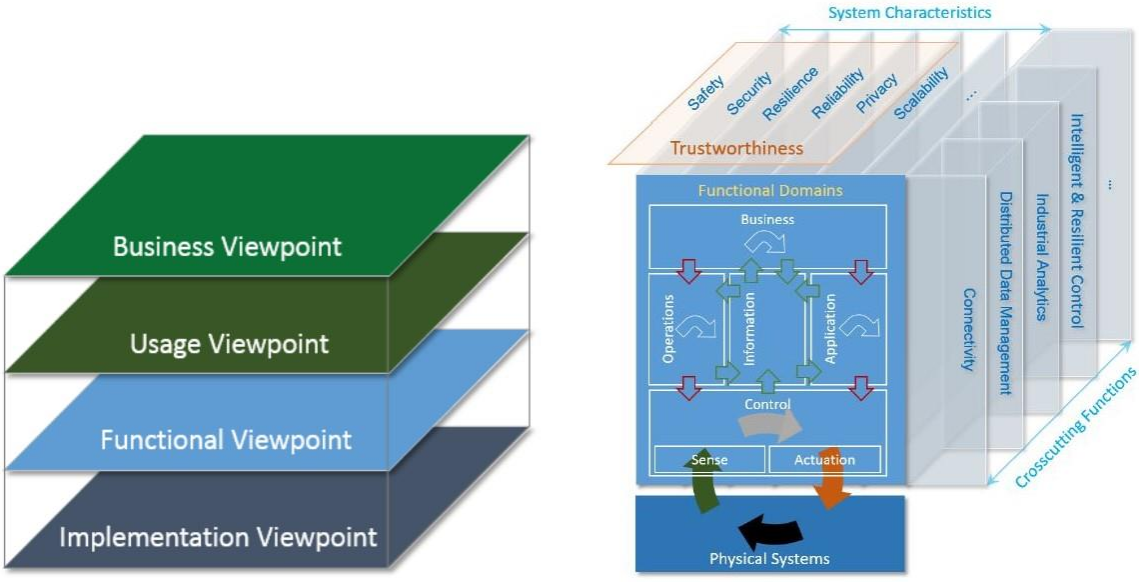

Figure 2 - IIRA viewpoints and functional domains (*The Industrial Internet of Things Volume G1: Reference Architecture*, 2019).

The **Business Viewpoint** in IIoT systems focuses on outlining the business goals, requirements, and objectives. By defining these parameters, organizations can make strategic decisions aimed at enhancing overall business performance. This viewpoint helps ensure that the IIoT system aligns with the company's broader goals and drives improvements in efficiency and effectiveness (Leitão et al., 2023; *The Industrial Internet of Things Volume G1: Reference Architecture*, 2019).

The **Usage Viewpoint** describes how the IIoT system is anticipated to be utilized. Understanding its intended use allows organizations to make informed decisions about system design, ensuring that the system is tailored to meet operational needs and user requirements effectively (Leitão et al., 2023; *The Industrial Internet of Things Volume G1: Reference Architecture*, 2019).

The **Functional Viewpoint** details the functional components of the IIoT system, organizing them into a connected structure. This viewpoint analyzes how these components interact and work together to support the system's functionalities. It is divided into five functional domains: control, operation, information, application, and business, each contributing to the system's overall performance and integration (Leitão et al., 2023; *The Industrial Internet of Things Volume G1: Reference Architecture*, 2019).

The **Implementation Viewpoint** addresses the technical aspects of IIoT system deployment, including the communication schemes required for data exchange between systems. These schemes are crucial for ensuring seamless communication within the IIoT infrastructure. The choice of technologies and implementation strategies is guided by the needs outlined in the Business Viewpoint (Leitão et al., 2023; *The Industrial Internet of Things Volume G1: Reference Architecture*, 2019).

2.2.2 SGAM Overview

The Smart Grid Architecture Model (SGAM) emerges as a pivotal framework within the realm of power and energy systems, offering a comprehensive three-dimensional architectural approach to understand and model the intricate interactions within these systems. Originating from the M/490 standardization mandate by the European Commission, aimed at CEN-CENELEC-ETSI, SGAM was designed to streamline the identification and resolution of gaps in the European smart grid standardization landscape. Its inception marked a significant tride towards harmonizing the architecture of smart grids, ensuring a coherent and integrated approach to the development and deployment of smart grid technology (CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012; Leitão et al., 2023).

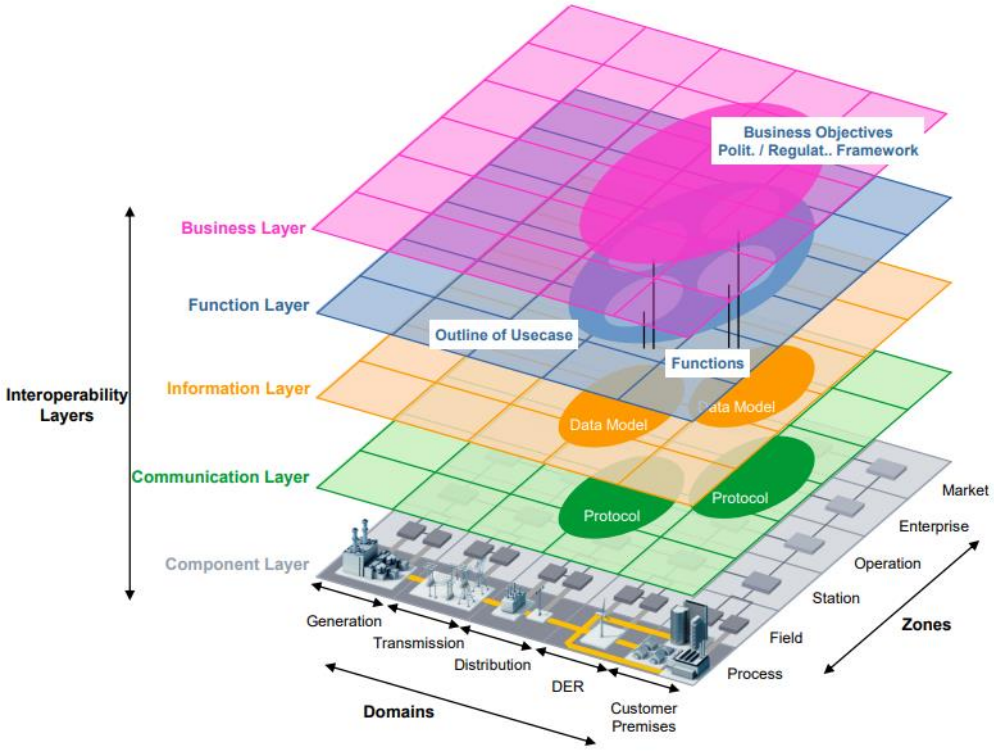

Figure 3 - SGAM architectural model (CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012).

This approach was initially elaborated to identify gaps and inconsistencies in the European smart grid standardization, and it provides a layered concept for the architecture development. It consists in three dimensions (axes); Figure 3:

At the heart of the Smart Grid Architecture Model (SGAM) is the **Domains' Dimension**, which represents the traditional structure of power system infrastructure. This dimension is divided into five distinct domains: Generation, Transmission, Distribution of Electricity, Distributed Energy Resources (DER), and Customer Premises. Each domain is crucial, from electricity generation to its distribution and final consumption, underscoring the interconnected nature of modern power systems (CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012).

Building on this foundation, the **Zones' Dimension** introduces a hierarchical view of power system management. It is organized into six zones: Market, Enterprise, Operation, Station, Field, and Process. These zones cover various operational levels, from high-level market strategies to detailed operations at individual stations and processes. Together with the Domains' Dimension, these zones form the Component Layer, or the SGAM Plane, which serves as the base of the architectural model, offering a clear visualization of the system's structural components (CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012).

Above the Component Layer, the **Layers' Dimension** unfolds, introducing five interoperability layers that build upon the foundational domains and zones. These layers include the Communication, Information, Function, and Business-related Services layers, each successive layer adding complexity and functionality to the model. This dimension emphasizes the importance of interoperability and communication in achieving a cohesive and efficient smart grid architecture, ensuring that all components of the system can interact seamlessly to support the delivery of reliable, sustainable, and flexible power solutions (CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012).

2.2.3 RAMI4.0

As it was previously explained, Industry 4.0 represents a shift in manufacturing, which enables the production of high-quality and personalized goods while taking into account maximum efficiency and time management. This is made possible by the seamless connectivity between different technologies and standards (Gheewala D., 2022). However, integrating these diverse aspects poses a challenge due to their different domains. This led to the development of standards for cross-domain integration, leveraging existing approaches like OPC-UA for information exchange in automation. Establishing a unified architecture model, RAMI 4.0, served as a template for Industry 4.0's concrete architecture.

RAMI 4.0 was developed by the German Electrical and Electronic Manufacturers´ Association (Leitão et al., 2023). RAMI 4.0 offers a structured approach to organizing and visualizing the elements of Industry 4.0 (Leitão et al., 2023), enabling manufacturers to effectively transition from traditional manufacturing practices to a more interconnected and data-driven ecosystem.

This three-dimensional model serves as a reference point for understanding the various layers and domains of I4.0, from the physical level of machines and sensors to the business level of strategic planning and decision-making. It provides a clear taxonomy for classifying I4.0 technologies and identifying the specific requirements and configurations needed for different applications.

In essence, RAMI 4.0 acts as a roadmap for manufacturers, guiding them through the complexities of Industry 4.0 and enabling them to use its transformative potential, in order to, enhance their operations, optimize production processes, and gain a competitive edge (Patnaik A., 2023b).

2.2.3.1 RAMI4.0 Dimensions

This reference architecture model's three-dimensional layout categorizes components not just by their organizational hierarchy, but also by their internal layers and life cycle stages. This section will provide an insight into the different layers of this reference model.

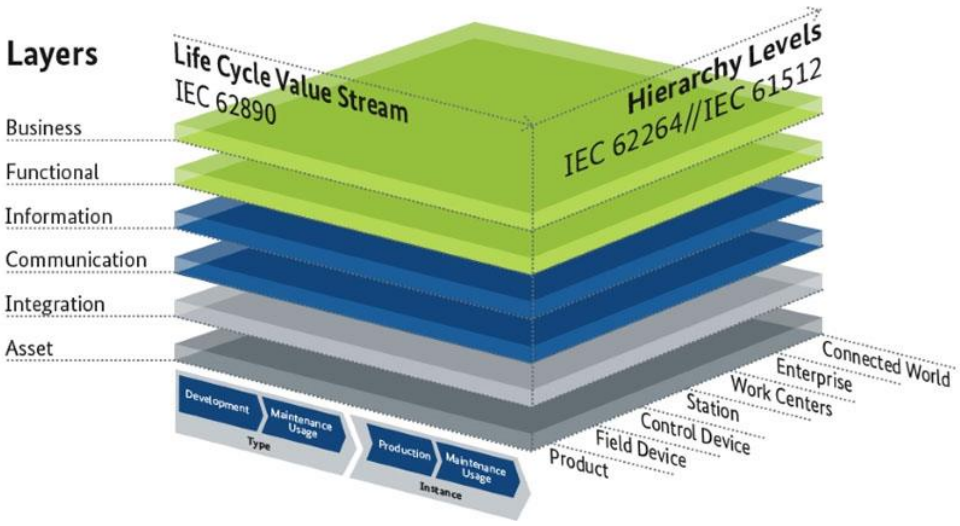

Figure 4 - RAMI4.0 and its dimensions

Layers Axis

The layers are used along a vertical axis to represent various perspectives such as data maps, functional descriptions, communication behavior, hardware/assets, and business processes. (ZVEI, 2016). The vertical dimension delves into the architecture of components within a manufacturing facility, in terms of their properties, functions and system structures, defining six layers per component (DIN, 2016):

- Asset Layer
- Integration Layer
- Communication Layer
- Information Layer
- Functional Layer
- Business Layer

The **Asset layer** corresponds to the real world, represents both the physical (Alemao et al., 2022) and virtual components of a system. It comprises elements such as a mechanical structure's dimensions, weight and material composition (Alemao et al., 2022; Patnaik A., 2023b). In the simulation of a system, taking into account the physical properties is just as

important as the information and communication functionality. Although they are not able to communicate, the physical properties' information needs to be available as a virtual representation. On this account, this layer enables a thorough description of the machines, components and factories as well as other physical elements (Alemao et al., 2022; ZVEI, 2016).

The **Integration layer** allows a transition from the real (physical) world to the virtual world. In other words, it allows the creation of a digital twin. As it was previously stated A digital twin consists in creating a digital representation of a physical component, thus allowing its integration with others (Alemao et al., 2022; Patnaik A., 2023b).

Because every significant event in the physical world generates a corresponding event in the virtual world, if there is a change in the former the integration layer is notified (DIN, 2016).

The **Communication layer** is the backbone of Industry 4.0, enabling seamless data exchange between various system components by standardizing communication formats and protocols and detailing data usage and distribution (Alemao et al., 2022; DIN, 2016; Patnaik A., 2023b). The choice of protocols—such as OPC-UA, Modbus, MQTT, Sparkplug, and HTTP—is crucial for optimizing data exchange, ensuring security, and enabling real-time updates. OPC-UA offers robust security and supports complex data models, ideal for environments requiring integration across diverse systems, though it can be complex and potentially slow. Modbus is simple and widely compatible, suitable for basic device communication, but lacks advanced security and scalability. MQTT provides lightweight, efficient publish-subscribe messaging with JSON data formats, suitable for real-time data dissemination, but has limited security and support for complex data structures. Sparkplug, an MQTT extension, enhances efficiency and data integrity, ideal for IIoT applications needing fast data exchange, but requires a deep understanding of its specifications. HTTP is widely used for data retrieval between servers and clients, offering built-in security and support for complex data structures, though it may be inefficient for real-time data transfer. These protocols are vital for ensuring interoperability between different devices (Patnaik A., 2023b).

The **Information layer** is responsible for organizing, interpreting, and managing the vast amount of data generated by Industry 4.0 systems, ensuring its interoperability (DIN, 2016). It encompasses the standards, technologies and processes that guarantee the integrity, consistency, and accessibility of information throughout the system. This layer describes the data that each asset has used, modified or generated (Alemao et al., 2022).

The **Functional layer** acts as a central hub for defining, organizing, and executing the core functionalities and services provided by the components.

This layer also includes logical functions that help in decision making in case of an anomaly.
In the context of Industry 4.0, the functional layer plays a crucial role in defining the capabilities of a manufacturing component. This layer describes the functionality of the component, such as its operational capabilities, error handling mechanisms, and communication protocols. This information is critical for integrating the component into the broader manufacturing system and ensuring that it can interact effectively with other components.

In essence, the functional layer in RAMI 4.0 provides a detailed description of the capabilities of a manufacturing component, enabling it to be integrated effectively into the broader manufacturing system and function optimally within it (Alemao et al., 2022; DIN, 2016; Patnaik A., 2023b).

The **Business Layer** in RAMI 4.0 is a crucial component that defines the link between the manufacturing component and the broader manufacturing business processes, legal conditions, and regulations. It represents the organization and its processes and deals with promotions, offers, target locations, advertisements, Customer Relationship Management (CRM), budget and pricing models, and manufacturing and cost analysis (Patnaik A., 2023b; Zaheer M., 2017a).

This layer is designed to ensure that the manufacturing component operates in accordance with the company's strategies, goals, and regulations. For instance, a manufacturing order for a specific product can be linked to a production machine in the business layer, ensuring that the right product is being manufactured to meet customer demand. Legal and regulatory conditions, such as safety standards, are also defined in this layer (Patnaik A., 2023b).

Fundamentally, the business layer in RAMI 4.0 acts as a bridge between the manufacturing component and the wider organizational context, ensuring that the manufacturing processes align with the business objectives and comply with legal and regulatory requirements.

Life cycle

The Life Cycle & Value Stream axis in RAMI 4.0 represents the life cycle of facilities and products. As we can see in Figure 4 this axis distinguishes sections (Lydon, 2019):

- Type
- Instance

A "type" represents the design and prototype stage of a product or component, while an "instance" refers to the actual product or component that is produced and used. The transition from a "type" to an "instance" occurs when design and prototyping are completed, and the actual product is being manufactured (Lydon, 2019).

The "type" stage is further divided into development and maintenance/usage. During the development stage, activities such as design, construction, computer simulation, and prototype creation take place. Once the development phase is complete, the "type" enters the maintenance/usage stage, where activities such as software updates, instruction manual creation, and maintenance cycles occur (Zaheer M., 2017b).

On the other hand, the "instance" stage is divided into production and maintenance/usage. The production stage involves the actual manufacturing of the product, while the maintenance/usage stage involves activities such as service provision, maintenance, recycling, and disposal of the product (Zaheer M., 2017a).

Hierarchy axis

The Hierarchy Levels axis in RAMI 4.0 represents the different functionalities within factories or facilities, based on the IEC 62264 international standards series for enterprise IT and control systems. This axis includes various levels, such as Product, Field Device, Control Device, Station, Work Centers, Enterprise, and Connected World (Lydon, 2019).

Each of these levels represents a different aspect of the manufacturing process. For example, the "Product" level represents the final output of the manufacturing process, while the "Field Device" level represents the machinery or equipment used in the production process. The "Control Device" level represents the devices that control these machines, the "Station" level represents the workstation where the work is performed, and the "Enterprise" level represents the entire manufacturing facility. The "Connected World" level represents the connection of all these components to the outside world, such as suppliers, customers, and the internet (Lydon, 2019).

The Hierarchy Levels axis in RAMI 4.0 allows for a comprehensive view of the manufacturing process, from the smallest individual components to the largest organizational structures. This enables a holistic and adaptive approach to Industry 4.0, allowing for the integration of all components and systems and facilitating effective communication and coordination across different levels (Schuck T., 2022).

2.2.3.2 Asset Administration Shell

The Asset Administration Shell (AAS) (Axelsson et al., 2019; DIN, 2016) is a pivotal concept within the Industry 4.0 framework, serving as a standardized digital representation of physical assets. This digital shell encapsulates essential information about an asset, including its properties, current state, and operational capabilities, while also providing metadata that describes the asset's semantics. By structuring data into submodels, as seen in Figure 5, the AAS facilitates modular and scalable data management, which is crucial for modern manufacturing environments where rapid technological advancements necessitate flexible asset management solutions (Bradac et al., 2019; Cavalieri & Salafia, 2020; Dhouib et al., 2023).

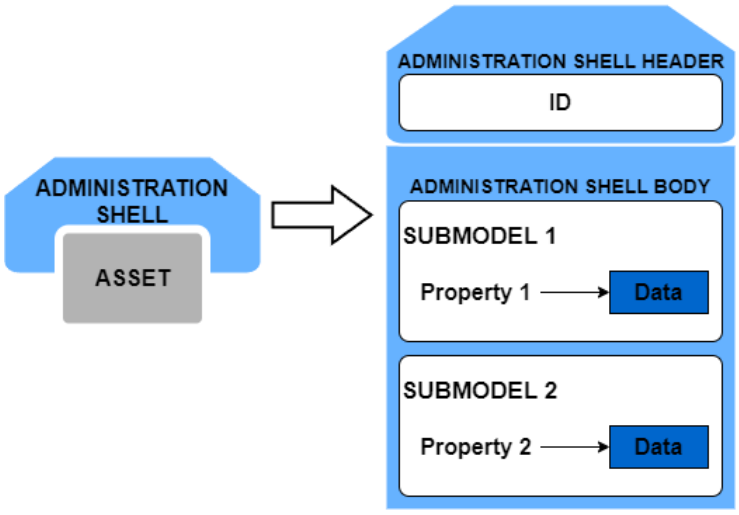

Figure 5 - AAS representation derived from (Bundesministerium für Wirtschaft und Energie (BMWi), 2016)

One of the primary advantages of utilizing AAS in manufacturing is its ability to establish a common language and structure for asset data. This standardization promotes seamless integration and communication among various technologies within the manufacturing ecosystem. As a result, the implementation of diverse technologies becomes more straightforward, enabling efficient coordination of operations across the shop floor (Bradac et al., 2019; Cavalieri & Salafia, 2020; G. Wu et al., 2022). Furthermore, the AAS enhances interoperability by adhering to established communication protocols, such as OPC UA, which allows for vertical and horizontal communication between different components and management levels (Arm et al., 2021; Ye et al., 2021).

Scalability is another significant benefit provided by the AAS. The modular nature of the AAS allows for easy updates and modifications to asset information as new machines are introduced, or existing ones are upgraded. This adaptability is essential in contemporary

manufacturing settings, where production demands can change rapidly due to market dynamics (Ambarita et al., 2023; Deuter & Imort, 2021; Siatras et al., 2023). The AAS not only supports the integration of new technologies but also ensures that asset management remains agile and responsive to evolving operational requirements (Ambarita et al., 2023; Veiga et al., 2021).

Moreover, the AAS promotes transparency by offering a unified view of asset status and historical performance data. This visibility is critical for enhancing decision-making processes, as it provides comprehensive insights into asset utilization and performance(Al Assadi et al., 2021; Deuter & Imort, 2021). With detailed asset data readily accessible, manufacturing teams can optimize processes, troubleshoot issues promptly, and improve overall operational efficiency (Kim et al., 2022; Siatras et al., 2023). The ability to monitor and analyze asset performance in real-time contributes significantly to the optimization of manufacturing processes and the reduction of downtime.

In the context of Industry 4.0, the AAS is instrumental in transforming traditional manufacturing into smart factories. By supporting automated processes through standardized data and lifecycle management—from design to decommissioning—the AAS facilitates the seamless integration of physical and digital technologies (Bradac et al., 2019; Rauh et al., 2022; G. Wu et al., 2022). This integration not only drives operational efficiency but also fosters innovation and competitiveness within the manufacturing sector, enabling organizations to adapt to the challenges of the digital age (Deuter & Imort, 2021; Dhouib et al., 2023; Lang et al., 2018).

In conclusion, the Asset Administration Shell serves as a foundational element in the realization of Industry 4.0 principles, providing a robust framework for asset management that enhances interoperability (Tantik & Anderl, 2017), scalability, and transparency. Its role in facilitating the digital transformation of manufacturing processes underscores its significance in the ongoing evolution of industrial practices.

2.3 Scheduling in Industry 4.0

In the era of Industry 4.0, scheduling has become a more dynamic and adaptive process that leverages technologies such as IoT, cloud computing, and artificial intelligence (Li et al., 2021) to optimize production efficiency and respond quickly to customer demands. The Manufacturing Execution System (MES) plays a crucial role in enhancing scheduling by providing real-time data and control over manufacturing operations on the shop floor. As a

software platform, MES connects the Enterprise Resource Planning (ERP) system, which manages business functions like planning, logistics, and finance, to the physical machinery involved in production. This integration allows for more precise scheduling by aligning business-level planning with actual production activities, ensuring that resources are allocated efficiently, quality is maintained, and production timelines are met. By combining MES and ERP, organizations achieve a seamless flow of information that improves scheduling accuracy and agility, allowing them to adapt quickly to changes in demand and optimize their overall manufacturing processes, as illustrated in the hierarchy of manufacturing systems in Figure 6.

Figure 6 - Hierarchy of Manufacturing Systems based on (D' et al., 2017)

2.3.1 Job-Shop Scheduling

The Job-Shop Scheduling Problem (JSSP) is a complex combinatorial optimization issue that arises in production planning and control. It involves scheduling a set of jobs through a set of machines, where each job consists of several operations that need to be executed in a specific order. (Karimi et al., 2017).

In scheduling problems, particularly those involving single-machine and parallel-machine scenarios, the main objective is often to minimize total completion time while adhering to various constraints. A key aspect of these problems is ensuring that no two operations of a job can be processed simultaneously, and once an operation starts, it must run uninterrupted. This aligns with mixed integer programming formulations for scheduling, where minimizing completion times in both single and parallel machine settings is central (Ying et al., 2019).

The constraints inherent in scheduling, such as the need for jobs to wait for machine availability, significantly impact overall efficiency. In the context of unrelated parallel machines, minimizing total completion time while managing the number of tardy jobs adds complexity to the scheduling process (Yin et al., 2019). A common assumption is that jobs can be processed flexibly as long as these constraints are respected. Machines can only handle one operation at a time, a well-established principle in scheduling theory. In single-machine scheduling, minimizing total weighted completion time is a prominent problem directly linked to non-preemption and single-operation processing (Geismar, 2011).

The assumption of negligible setup times simplifies scheduling by focusing on optimizing the sequence of job processing without adding complexity from setup delays. However, the literature does emphasize that jobs must wait for machine availability, a frequent scenario in both theoretical and practical settings. In environments where machines may be idle, careful job sequencing becomes crucial to minimize completion times under specific constraints (Morsy & Pesch, 2015).

In summary, the scheduling of jobs on machines, governed by strict operational constraints, is a well-explored field in computer science and operations management. The framework presented in the literature provides a comprehensive understanding of how to minimize total completion time while ensuring all constraints are respected.

Figure 7 depicts a schedule with three jobs that require three different operations processed by three different machines.

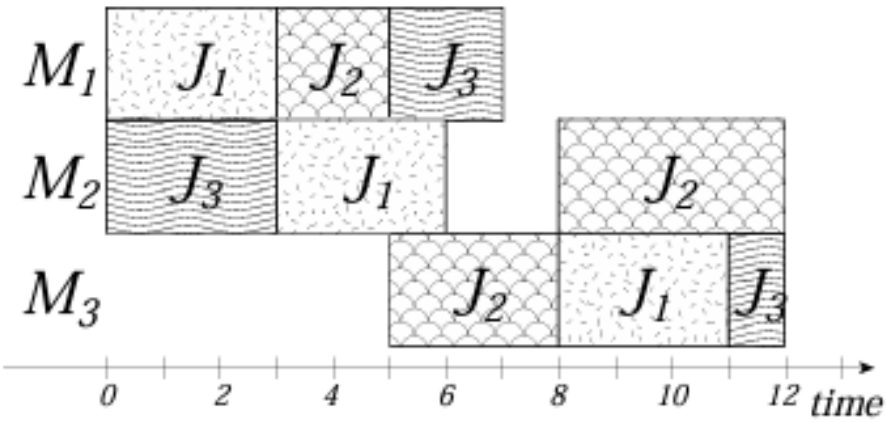

Figure 7 - Gantt chart depicting a schedule (Yamada & Nakano, 2000)

Dynamic Job-Shop Scheduling Problems (DJSSP) represent an evolution beyond traditional deterministic JSSP (Bierwirth & Mattfeld, 1999; Mattfeld, 1996; Werner, 2011) by incorporating real-world uncertainties that affect scheduling efficiency. While JSSP assumes

fixed and predetermined task durations and constraints (Zhang et al., 2023), DJSSP addresses the unpredictable nature of job arrivals, variable processing times, and potential machine failures or maintenance issues (Chen et al., 2022). This dynamic environment requires more advanced scheduling approaches, as static methods often struggle to adapt to changing conditions (Han et al., 2023).

In DJSSP scenarios, jobs may arrive at any time, and their processing times can vary due to factors such as machine breakdowns and unexpected maintenance. This unpredictability complicates scheduling, requiring real-time adjustments to maintain optimal production flow. An efficient metaheuristic algorithm was proposed to facilitate decision-making in dynamic environments, effectively scheduling orders while responding to dynamic events to optimize production systems (Zhang et al., 2023). Accounting for machine failure rates in flexible job shop scheduling also plays a crucial role in managing unexpected disruptions on the shop floor (Han et al., 2023).

DJSSP has gained importance in research due to its relevance in real-world manufacturing. Many earlier studies on flexible job shop scheduling assumed that jobs were available at the start of the scheduling horizon, a condition rarely met in practice(Nie et al., 2013). This highlights the need for models that can handle the continuous arrival of jobs and adapt to shifting conditions.

Recent advancements, such as reinforcement learning and genetic algorithms, offer promising solutions to the complexities of DJSSP. Deep learning methods have been applied to address DJSSP, providing adaptive solutions that can respond to dynamic events like machine breakdowns and job rework (Chen et al., 2022). Hybrid approaches combining genetic algorithms with tabu search techniques have also improved scheduling performance in dynamic environments (Zhang et al., 2023). These developments indicate a shift towards more flexible and responsive scheduling systems capable of managing the inherent uncertainties in modern production.

DJSSP aims to adapt to ongoing changes by making real-time adjustments. It considers factors, which JSSP overlooks, like new job arrivals, machine breakdowns, variations in execution times, and the addition or removal of machines (Kundakcı & Kulak, 2016; Scrimieri et al., 2015; Sharma & Jain, 2015; Xiong et al., 2017).

2.3.2 Scheduling Benefits

Effective scheduling provides a range of benefits essential for optimizing production processes. It significantly enhances efficiency by optimizing the use of resources, sequencing

tasks, and timing, which in turn reduces production time and boosts overall productivity. By ensuring that resources are used optimally and that tasks are performed in the most effective order, scheduling helps to streamline operations and minimize delays.

Accurate scheduling plays a crucial role in improving demand prediction and resource allocation. It allows organizations to forecast needs more precisely, allocate resources effectively (Mourtzis, 2022), and adapt to customer expectations, thereby enhancing service delivery. This precision in scheduling also contributes to reducing waste, as it ensures that resources are utilized efficiently and minimizes the likelihood of surplus or underuse.

Responsiveness is another key advantage of effective scheduling. It enables manufacturers to quickly adapt to changes in production demands or unexpected issues by adjusting schedules in real time. This flexibility helps in optimizing production schedules to address new challenges or shifts in demand promptly (Deo & Molnar, 2019).

Moreover, scheduling enhances product quality by aligning tasks with the specific skills of workers and the capabilities of machines. This alignment ensures that products are produced to the highest standards, leveraging the strengths of both human and machine resources effectively.

Finally, by ensuring that products are delivered on time and meet quality standards, effective scheduling leads to improved customer satisfaction. Timely delivery and consistent product quality contribute to better customer service and increased satisfaction, reinforcing the importance of a well-managed scheduling system in achieving strategic business goals.

2.3.3 Scheduling Challenges

Scheduling in Industry 4.0 offers significant potential benefits but also faces several challenges. One major issue is the integration of data from diverse sources and formats, which requires advanced techniques to ensure accurate and seamless data flow. Additionally, scheduling systems must be capable of making decisions in real time to adapt to rapidly changing conditions. This necessitates the development of sophisticated and responsive algorithms (Smith & Sturrock, 2018).

Another challenge lies in managing resource constraints such as time, materials, and space. Effective scheduling demands careful consideration of resource availability and efficient task allocation to navigate these constraints. The growing complexity of manufacturing systems further complicates scheduling processes. Addressing this complexity involves developing efficient algorithms that can handle system variations, finite capacity

scheduling, and intricate mathematical optimization models (Mourtzis, 2022; Smith & Sturrock, 2018).

Unplanned events, such as machine failures and unpredictable downtimes, pose additional difficulties. These variations often lead to overly optimistic plans, as the system may not account for such disruptions, ultimately causing the scheduled plans to become infeasible and affecting overall performance (Smith & Sturrock, 2018).

2.3.4 Dynamic Scheduling Approach

Integrating dynamic scheduling into the RAMI 4.0 framework involves leveraging the model's layers and levels to optimize production processes and adapt to real-time changes. RAMI 4.0 offers a structured approach to manufacturing systems but lacks detailed guidelines for specific functionalities like dynamic scheduling. This chapter proposes a framework to address this gap, focusing on how dynamic scheduling can be effectively incorporated within the RAMI 4.0 architecture.

As it was previously addressed RAMI 4.0 is organized into several layers which collectively represent the manufacturing ecosystem. Dynamic scheduling, crucial for adapting production schedules in real time, needs to be integrated thoughtfully into each of these layers (Alemao et al., 2022).

Business Layer: Dynamic scheduling should support business rules and goals, maintaining the integrity of the value stream. It should be designed to integrate seamlessly with business processes, adapting to changes in demand and production requirements.

Functional Layer: Dynamic scheduling requires capabilities such as real-time scheduling adjustments, data analysis, and simulation. These functionalities must be designed to handle varying production scenarios and ensure optimal scheduling outcomes.

Information Layer: For dynamic scheduling to be effective, the system must ensure data integrity and consistency. The information layer should also handle the rules and models that govern scheduling processes, allowing for real-time updates and adjustments.

Communication Layer: The scheduling system must use standardized communication protocols like OPC-UA, MQTT, or HTTP to ensure seamless interaction between scheduling components and other system elements. Real-time communication is critical for dynamic scheduling, enabling the system to respond to changes promptly.

Integration Layer: For dynamic scheduling, middleware must be developed to facilitate the integration of real-time data from sensors, RFID readers, and HMIs. The AAS

concept is useful here, as it provides a standardized digital representation of physical assets, ensuring that data from the shop floor is accurately integrated into the scheduling system.

Asset Layer: Dynamic scheduling must be able to adapt to changes in these assets, such as machine breakdowns or shifts in production capacity. Effective management of these components is crucial for maintaining an optimized production schedule.

In terms of hierarchy levels, scheduling at the station field level involves local scheduling algorithms that can adjust based on conditions at individual stations. Coordination between different stations is also necessary to ensure smooth workflow and optimal resource utilization.

The development stage of the RAMI 4.0 lifecycle is where scheduling systems are designed and implemented. Guidelines for this stage include defining system requirements, designing the architecture, developing scheduling algorithms, integrating systems, and conducting thorough testing and validation. This ensures that the scheduling system meets both functional and performance requirements and can handle real-world scenarios effectively.

In conclusion, integrating dynamic scheduling into the RAMI 4.0 framework requires a comprehensive approach that considers the interplay between architectural layers and levels. By following the proposed framework, manufacturers can develop robust scheduling systems that enhance production efficiency, adaptability, and overall system performance. This integration will contribute to the advancement of Smart Manufacturing and support the ongoing evolution of manufacturing technologies.

2.4 Solution Overview

To tackle the challenges of production scheduling and resource management, a comprehensive system was developed that integrates the MES, Scheduler, Middleware, Machines, and Products into a unified architecture.

Optimized Resource Allocation: The Scheduler uses real-time data to allocate machines and resources efficiently, reducing idle times and production delays.

Seamless Communication: Middleware ensures smooth data flow between all components, leveraging standardized AAS for interoperability, allowing dynamic adjustments to schedules based on current information.

Flexibility and Scalability: The system adapts to changes in production needs and integrates machines and products from different manufacturers, ensuring scalability.

Error Prevention: The system allows for schedule review and error handling by operators, reducing production mistakes.

Data-Driven Decisions: Real-time data enables advanced analytics, supporting better decision-making and operational efficiency.

Industry 4.0 Alignment: The architecture supports digital transformation with the implementation of AAS, by connecting and automating systems, positioning the manufacturer for future growth.

In essence, the architecture enhances production efficiency, adaptability, and decision-making, while minimizing errors and optimizing resource use.

3

ARCHITECTURE

As it was previously described in Chapter 2 the evolution of the market and its requirements has been noticeable, making it necessary for manufacturing environments to adapt. This means the whole shop-floor needs to be more flexible, ensuring seamless processing throughout all the systems that comprise it.

In order to achieve this goal, the proposed architecture encompasses a comprehensive design that ensures uninterrupted integration and communication between all the shop-floor systems and devices.

This chapter will outline the scheduling architecture aimed at optimizing production processes. This system is built around a Manufacturing Execution System (MES), a Scheduler, (these last two components will be combined into the scheduler), various stations represented by AAS, a data base, a middleware and the products manufactured. These entities are integral components of the proposed system, each playing a unique role in the overall operation and optimization of the manufacturing process and will all be explained in more detail.

The scheduling architecture implemented will allow to generate schedules taking into account the product requests and the capabilities of each machine. It is capable of communicating with all entities via a middleware. With all this information available it will be able to process it and generate a schedule that illustrates the allocation of operations with the goal of optimizing total processing time.

The architeture also has the ability of showing the schedules generated and saving them for error checking/handling done by human operators. In Figure 8 it is possible to see a representation of the proposed architecture.

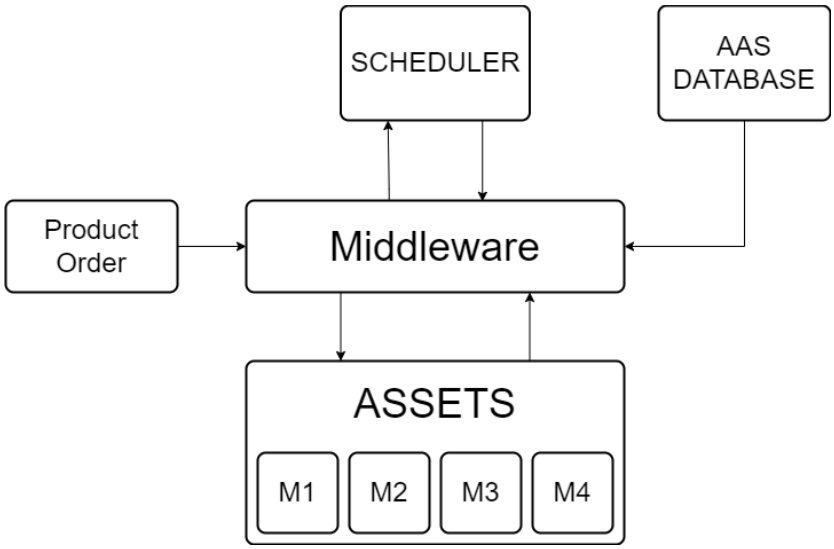

Figure 8 - Scheduling System Architecture

3.1 Main Components

Before delving into the proposed approach, it is crucial to understand the roles of the MES, Scheduler, Machines and Products, the Data Base and the Middleware.

As it was previously discussed the MES and Scheduler will be incorporated into a single unified system within the framework of this thesis. This consolidation aims to align efforts more efficiently, taking into consideration that the main objective of this work does not involve the development or optimization of the MES itself.

3.1.1 MES and ERP

In this thesis, the MES will be integrated into the Scheduler to create a unified system that aligns production scheduling with real-time execution data. This integration aims to streamline operations, improve task allocation, and optimize resource usage by minimizing delays. The focus is on enhancing scheduling and allocation rather than developing or optimizing the MES itself.

The ERP system, meanwhile, only handles initial product orders and does not interact with detailed scheduling or production processes. This approach allows for a clear focus on optimizing scheduling without needing extensive ERP modifications.

3.1.2 Scheduler

In the realm of business and technology, a Scheduler serves as a sophisticated system or software application that meticulously manages the allocation of resources across time. It orchestrates the sequence in which tasks are executed, taking into account various constraints such as priority levels and dependencies. Schedulers play a crucial role in optimizing workflows, ensuring efficient task completion, and adhering to predefined deadlines.

For manufacturing companies, shop floor scheduling is an important process aimed at optimizing manufacturing operations. This process involves breaking down work orders into individual operations and assigning theoretical start and completion times.

The scheduler developed in this project aims to streamline operations by maintaining continuous communication with a middleware system. This middleware system provides crucial information about the products scheduled for manufacturing and details about the machines and products represented by AAS stored in the database. Leveraging this comprehensive data, the scheduler can generate optimized schedules aimed at minimizing production time. Figure 9 demonstrates this process.

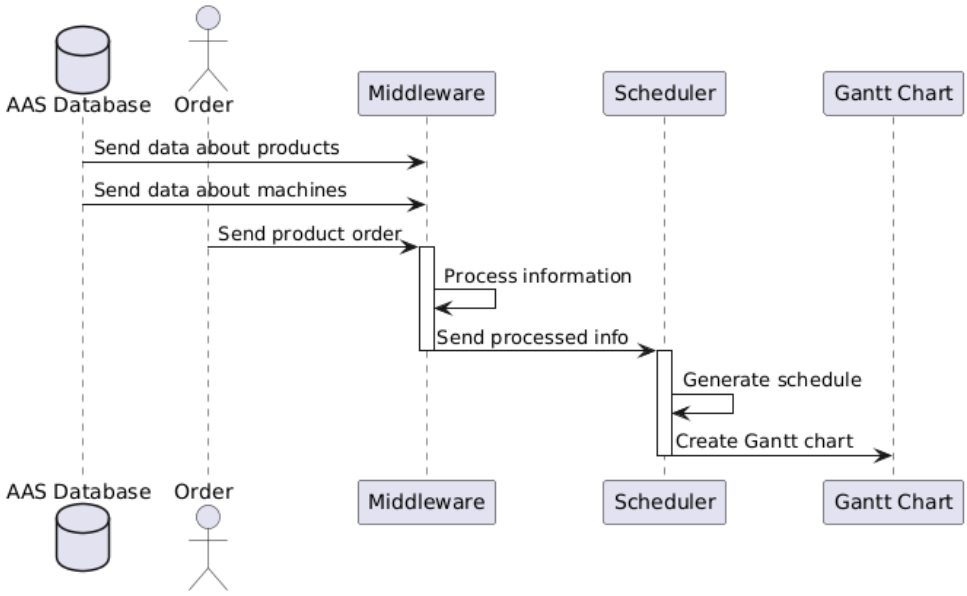

Figure 9 - Sequential Diagram for Information flow

In this proposed approach the scheduler presents noticeable advantages:

Seamless Integration and Communication: The scheduler developed has seamless integration capabilities, AAS. AAS standardizes all relevant data and services for sharing information, enabling efficient and secure communication between stakeholders and IT systems throughout the entire lifecycle of the asset. This integration ensures that your scheduler operates on a unified platform, supporting real-time data transmission, machine learning, and artificial intelligence.

Process Optimization and Effieciency: By utilizing AAS, your scheduler facilitates an increase in efficiency and transparency within operations. It replaces the heterogeneous data landscape with a unified and interoperable data structure, allowing for the integration of different IT systems. This leads to precise monitoring and controlling of production processes, increasing both profitability and competitiveness.

Innovative Business Models and Market Adaptability: The improved availability and usability of data through AAS promote the development of new approaches by providing a foundation for advanced analytics and data-driven decision-making. Additionally, the flexibility of AAS allows for quick adaptation to market changes, providing dynamic and scalable solutions for data integration and analysis, thus strengthening your company's market position.

High Flexibility and Scalability: Data standardization through AAS significantly reduces the resources required on the shop floor for planning, commissioning, and operation. It also ensures high flexibility and scalability in production, thanks to the ability to exchange devices with standardized models, regardless of manufacturer.

Digital Transformation: AAS serves as a key technology for transforming traditional operating models with a robust and secure data infrastructure. It enables a unified view of assets, asset types, and instances, and resources, promoting digital transformation.

The scheduler will also be represented by an Asset Administration Shell, which will be explored in more detail in the next sections. This ensures that the scheduler is integrated into the system with the same standardized approach, facilitating enhanced communication, efficiency, and adaptability

3.1.3 Middleware

In the context of a shop floor for Industry 4.0 the middleware refers to a software layer responsible for facilitating communication and data exchange between all the components. It serves as a bridge, allowing separate systems to interact and share information seamlessly.

The middleware enables integration of various components within the shop floor into a cohesive and, interconnected ecosystem.

The middleware handles several key pieces of information. It receives product orders from an external sender via a text file and retrieves data from a database containing information about the machines and products within the factory. After gathering this information, the middleware processes it to determine what products need to be manufactured and what resources are available.

Once the data is processed, the middleware communicates with the scheduler. It sends the necessary information (eg. machine ID and functions and product ID and requirements) to the scheduler, which then generates a production schedule. The middleware also receives the completed schedule back from the scheduler and uses this to initiate and manage the manufacturing process, allowing for dynamic adjustments as needed based on the latest data. The main advantageous aspects of the middleware include:

Interoperability: The middleware ensures that all the different systems and devices can communicate and work together, despite ahering to different standards and protocols. It is the most crucial aspect in order to guarantee the seamless operation and interconnectivity of all shop floor systems and devices.

Data Integration: Collecting, processing, and distributing data from multiple sources across the shop floor. This centralized data management allows for real-time monitoring and analysis of production processes, supporting decision-making and optimization efforts.

For the primary objective of this thesis, the middleware will also guarantee seamless communication between the various components of the architecture. Within the proposed architecture, the middleware is responsible for:

Product Order Reception: The middleware receives product orders which are provided by a sender via a text file. This communication is unidirectional, meaning the sender sends data but the middleware does not send any type of feedback or data back to the sender.

Database Interaction: The middleware also retrieves data from a database containing AAS. Similar to the product order reception, this interaction is unidirectional, with the middleware fetching data but not sending any commands or queries back to the database.

Data Processing: After receiving and processing the product orders and AAS data, the middleware prepares the information for dispatch to the scheduler. The Product orders contain information on what products are to be manufactured, while the AAS data contains information about the machines a factory is composed of, and also information about the products the manufacturing environment is able to produce.

Communication with Scheduler: Unlike the interactions described above, the communication between the middleware and the scheduler is bidirectional. After processing the data from the Product order and the Database the middleware will then send this parsed information to the Scheduler. After the schedule process is generated, the scheduler sends this information to the middleware which will then start the manufacturing process, allowing for dynamic adjustment and optimization of production schedules based on the latest data.

3.1.4 Machines and Products

The machines on the shop floor are integral to the manufacturing process, providing the physical means to transform raw materials into finished products. Within this architecture, each machine and product are represented by an AAS, which encapsulates all relevant information about the machine's capabilities, and their processing times and the product's requirements. Before delving into the outline of machines and products it is necessary to understand what are AAS and what their purpose is:

3.1.5 Asset Administration Shells

Three primary types of AAS were developed: Scheduler AAS, Machine AAS, and Product AAS. Each type contains specific information tailored to its role in the manufacturing process, enabling precise control and management of production activities.

The Scheduler AAS encapsulates the scheduling system's details. This AAS contains vital information such as the scheduler ID, and a submodel named "Information" that contains the product order and the tasks that need to be performed for that order. Another submodel containing information about the latest generated schedule is also present in the Scheduler AAS. This AAS enhances the integration and management of the scheduling system within the broader manufacturing framework.

The Machine AAS (Figure 10) captures essential data about the machines used in production. Important elements of the Machine AAS are a unique identifier for each machine, the machine name, descriptions of the machine's capabilities and the processes it can perform (e.g., placing, glueing, screwing) and the time required by the machine to complete specific tasks, essential for accurate scheduling.

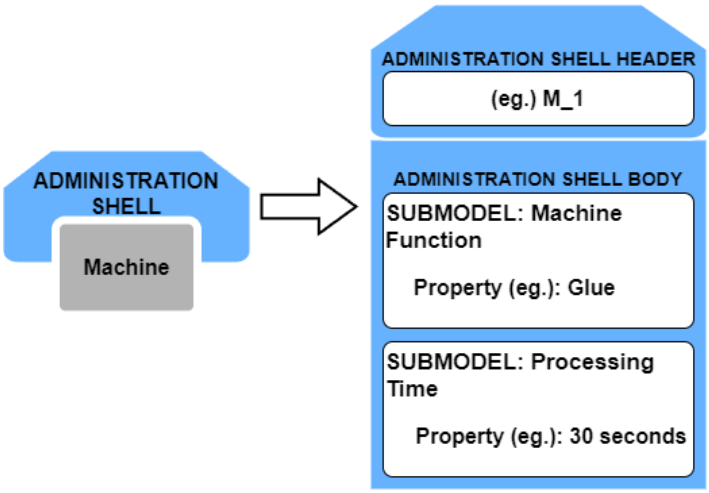

Figure 10 - Machine AAS Example

The Product AAS Figure 11 includes comprehensive details about the products in the system. Key components of the Product AAS are a unique identifier for each product, the product name, detailed requirements for producing the product such as necessary processing steps.

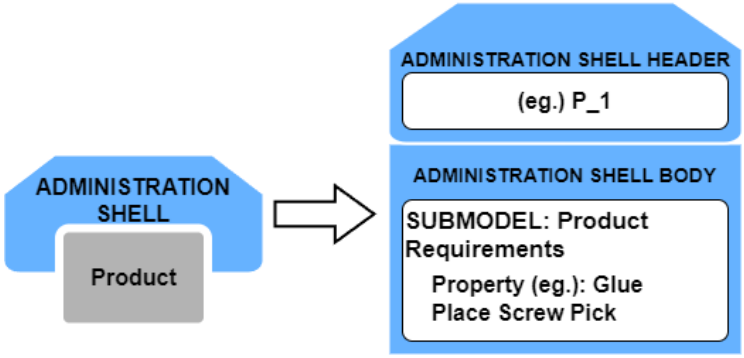

Figure 11 - Product AAS Example

Table 3, Table 4 and Table 5 provide a detailed overview of the information contained within each type of AAS. It illustrates the structure and key elements of the Scheduler AAS, Machine AAS, and Product AAS, respectively, along with additional information that can be added to these shells to enhance human interactions and ensure the completeness of AAS information.

Scheduler AAS			
Name	Explanation	Required	Example
Identification			
ID	Global unique ID of an asset, it	Mandatory	SCH_1

	can be read by both human and machine		
Name	Name consisting of one or more words to be easily identified by a human	Preferable	"Scheduling tool"
Description	Information that uniquely describes the meaning of the asset	Preferable	production scheduling software
Version	Number to distinguish the version of an asset	Preferable	001
Further Info	Additional info necessary to implement the solution	Optional	
Information Submodel			
Products_List	A collection of products to be allocated in the production schedule. Each entry refers to a Product AAS (containing the intrinsic information of the product). Each product contains at least one task to be allocated.	Mandatory	[P_1, P_2, …]
Tasks_List	A collection of tasks to be allocated in the production schedule.	Mandatory	[Glue, Pick, Place …]
Type	Indicates the type of schedule to be performed. E.g. production-oriented, energy consumption-oriented, etc. (For this thesis the main focus was the time efficiency schedule.)	Optional	"Full throughput"
Current Schedule Submodel			
ID	Unique ID of a generated schedule that uses the current time	Mandatory	"schedule_01062024"

Name	Name consisting of one or more words to be easily identified by a human	Optional	"Schedule generated for production order XYZ"
Generation Date	Date of generation	Preferable	01-06-2024 10:00:00
Description	Information that uniquely describes the meaning of the schedule	Optional	"Schedule generated for production order XYZ starting on June 10th, 2024"
StartTime	Refers to the schedule start time	Mandatory	10-06-2024 08:00:00
EndTime	Refers to the schedule end time	Mandatory	12-06-2024 20:00:00
QuantityProduced	Number of manufactured products	Optional	250
GanttChart	Visual description of the generated schedule	Preferable	It can be a PDF document, an image, a dynamic web-based chart, etc.

Table 3 - Scheduler AAS Details

Machine AAS			
Name	Explanation	Required	Example
Identification			
ID	Global unique ID of an asset, which can be read by both human and machine	Mandatory	M_1
Name	Name consisting of one or more words to be easily identified by a human	Preferable	"Glue Machine"
Description	Information that uniquely describes the meaning of the asset	Preferable	Machine responsible for glueing
Version	Number to distinguish	Preferable	001

	the version of an asset		
Further Info	Additional info necessary to implement the solution	Optional	
MachineData Submodel			
Function	Describes the main function of the machine	Mandatory	Glue
Processing time	Details the time the machine takes to finish its process. (This property also includes the time units as a concept description).	Mandatory	30 Minutes
ProductsAbleToManufactured	Identifies the type of products the machine is able to produce	Preferable	P_2, P_1…
KPIs	Describes some KPIs that may be relevant	Optional	
Maintenance	Defines information related to maintenance in the machine (more important if maintenance tasks should be considered)	Optional	
Documents Submodel			
Documentation	Documents related to the resource	Preferable	Manuals for installation and operation

Table 4 - Machine AAS Details

Product AAS			
Name	Explanation	Required	Example
Identification			
ID	Global unique ID of an	Mandatory	P_1

	asset, which can be read by both human and machine		
Name	Name consisting of one or more words to be easily identified by a human	Preferable	"Piece X"
Description	Information that uniquely describes the meaning of the asset	Preferable	Metal Part
Version	Number to distinguish the version of an asset	Preferable	001
Further Info	Additional info necessary to implement the solution	Optional	
ProductData Submodel			
ListOfTasks	Describes the processes needed for the manufacturing of the product.	Mandatory	Glue, Cut, Screw, Attach
RawMaterials	Raw materials necessary to manufacture the product	Optional	Screws, Metal Sheet

Table 5 - Product AAS Details

All of the AAS are stored in a database ensuring the scheduler has access to all necessary product and machine details. This comprehensive data enables the scheduler to match product requirements with machine capabilities, optimizing the production process and ensuring efficient use of resources.

The integration with the database acts as a centralized repository for all machines and products AAS ensuring the most current and accurate data is available to the middleware and scheduler. This integration allows for real-time updates on machine status and capabilities, facilitating dynamic and responsive production scheduling. The scheduler leverages this

detailed machine information to allocate resources efficiently, optimize the sequence of operations, reduce downtime, and improve throughput.

3.1.6 Asset Administration Shell Database

In the proposed architecture, the AAS Database is a central repository for all asset administration shells related to machines and products within the production environment. This server ensures that the most current and accurate information is accessible to the middleware and, subsequently, to the scheduler.

The AAS Database consolidates all relevant data about machines and products as seen in Figure 12, including technical specifications, operational parameters, historical performance data, and maintenance records. This centralized approach facilitates efficient data retrieval and management, reducing redundancy and ensuring consistency. By providing detailed and current information about both machines and products, the server plays a crucial role in effective decision-making and responsive production scheduling.

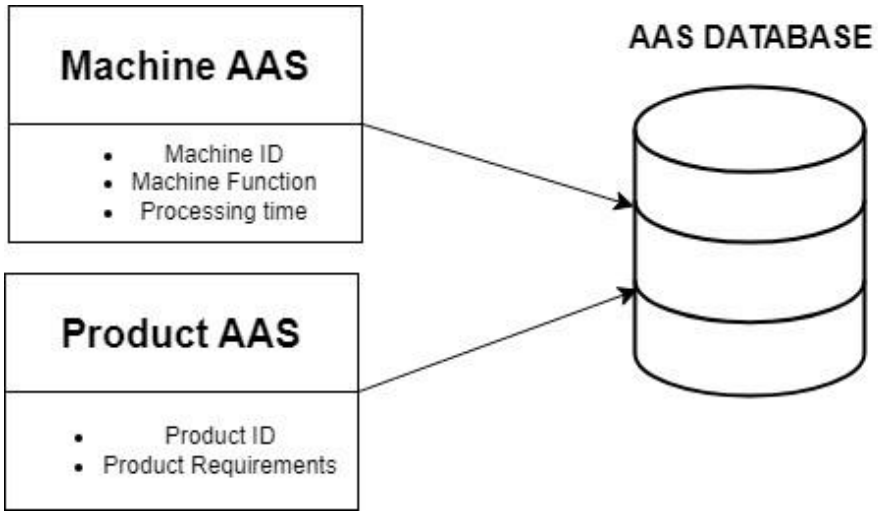

Figure 12 - AAS Database

As seen in Figure 13, the AAS Database communicates seamlessly with the scheduler through a middleware system, which processes data from the AAS Database and provides it to the scheduler in a usable format.

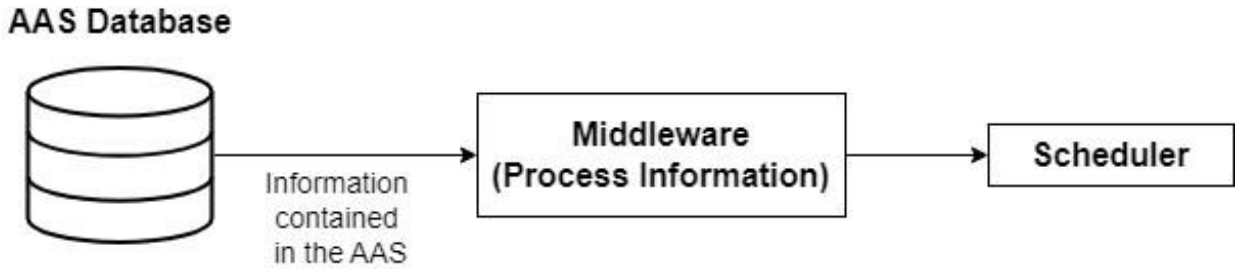

Figure 13 - Information Flow

This ensures the scheduler can generate optimized production schedules that minimize downtime and enhance throughput.

Designed with scalability in mind, the AAS Database can accommodate the addition of new machines and products as the manufacturing environment evolves. Human interactors can easily save new AAS data when a new machine is added to the shop floor, ensuring that the system remains up-to-date and accurate. This flexibility supports increased production capacity and new technological advancements.

The server's ability to integrate seamlessly with various IT systems and facilitate smooth information flow across the manufacturing process makes it a vital component of our architecture. By ensuring accurate and up-to-date information is readily available, the AAS Database significantly contributes to optimizing production processes and achieving operational excellence.

The following section provides a concise and detailed overview of the scheduling process, highlighting each step from start to finish.

3.2 Scheduling Process Overview

This section will offer an in-depth explanation of the entire scheduling process, presented in a clear and straightforward manner. Figure 14 illustrates a flowchart providing a general depiction of the scheduling process from start to finish, illustrating each step along the way.

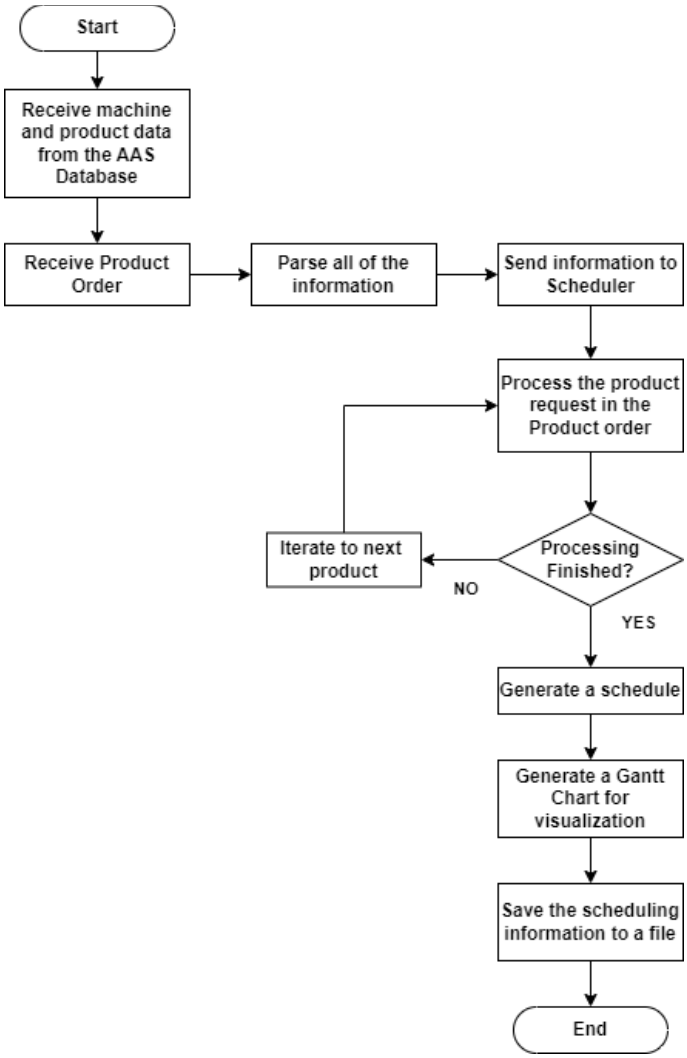

Figure 14 - Scheduling Process Overview

The Middleware acts as a pivotal component that listens for incoming product orders that are entered manually. Upon receiving a product order, the Middleware reads the text file and extracts the product IDs listed in the order. The Middleware parses the product order text file to identify individual product IDs. Each product ID corresponds to a specific product that needs to be manufactured. The product IDs are stored in a structured format for further processing.

This component is also responsible for receiving the AAS containing details about the machines (ID, capabilities, and processing times) and the products (ID and requirements). The retrieved machine and product data is stored and structured in a JSON format optimized for seamless access and processing by the scheduler. This ensures that the scheduler can efficiently utilize the data for production scheduling and optimization. With both machine

and product data retrieved, the Middleware aggregates both products' and machines' data and prepares it to be sent for the scheduler.

Figure 15 shows a flowchart depicting the receipt of data and processing done by the Scheduler. The Scheduler commences its operations by analyzing each product ID within the received order. It ensures that all pertinent requirements are thoroughly assessed before proceeding to the next phase of scheduling. With essential data derived from both the product order and the AAS database, encompassing detailed insights ìnto machine capabilities, processing times and specific manufacturing specifications, the scheduler systematically processes each ID listed in the order. Upon gathering and confirming all the necessary details for each product the scheduler transitions into the scheduling phase that optimizes the production times.

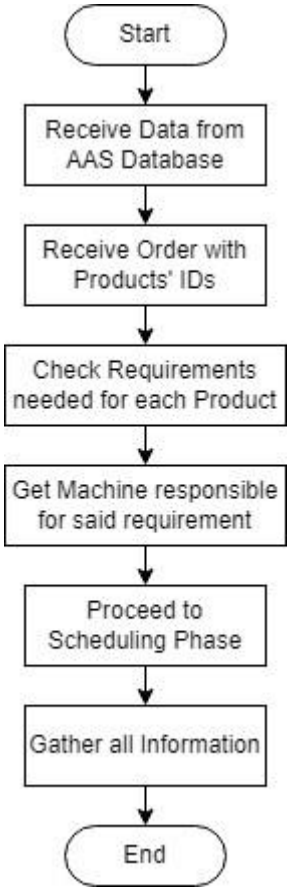

Figure 15 - Data Receipt and Processing

Using an algorithm the scheduler aims to optimize production schedules efficiently. It starts by analyzing the product order received from the Middleware, focusing on understanding the specific requirements and priorities of each product.

Once the product requirements are identified, the Scheduler proceeds to match these needs with the capabilities of the available machines. This involves evaluating factors such as machine capacity, processing times, and any specific requirements outlined in the product order.

A notable feature of the algorithm is its capability to facilitate parallel processing. This means that the Scheduler can schedule multiple products concurrently, even if they require different machine capabilities. By doing so, it optimizes overall production throughput and reduces idle time, thereby enhancing the efficiency of resource utilization on the production floor.

Furthermore, the algorithm treats each instance of the same product independently, ensuring that each product order is accurately represented in the production schedule. This approach takes into account the unique requirements of each instance, which is essential for precise scheduling of tasks. By treating each instance independently, the Scheduler maintains flexibility and responsiveness to varying production demands, thereby improving overall operational efficiency.

In addition to independent treatment of product instances, the algorithm calculates precise start and end times for each production task based on machine availability and operational sequences. This meticulous planning facilitates seamless coordination of production processes, minimizing bottlenecks and optimizing workflow throughout the manufacturing process.

Taking into account all the information provided in the previous step, visual representations of the production schedule are generated using Gantt charts. These charts provide clear timelines and task allocations, offering a visual overview of the production process by highlighting the sequence and duration of tasks and the products being manufactured. Each product is differentiated with distinct color labels in the Gantt chart, making it easy to identify and distinguish between different products, as illustrated in Figure 16.

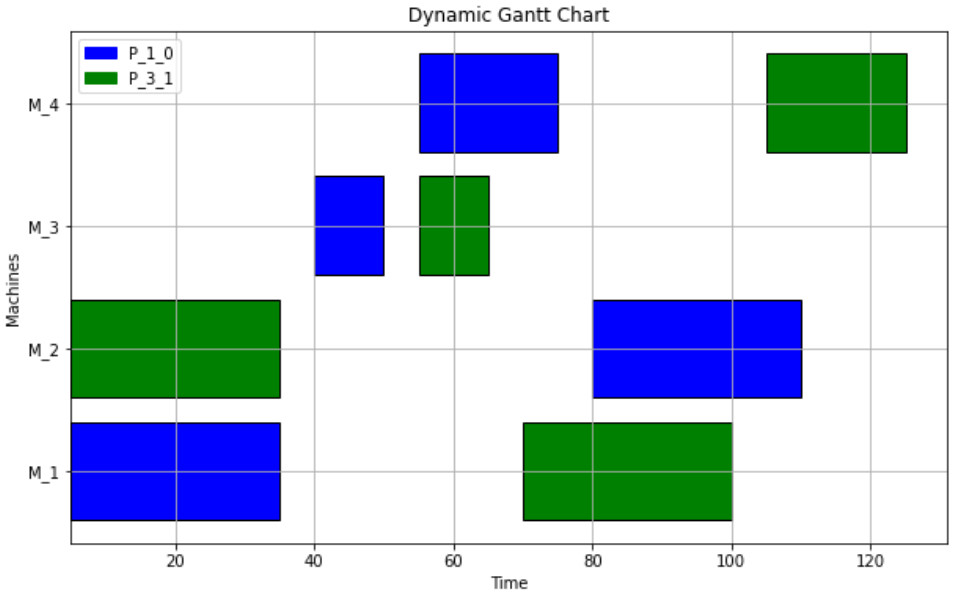

Figure 16- Gantt Chart Example

Detailed scheduling information, including task sequences, start times, and completion times, is saved for reference and analysis. This comprehensive record of the production schedule ensures that all necessary details are documented accurately.

To ensure traceability and ease of review, the file is saved with a timestamp indicating the date and time it was created. This information is then sent back to the Middleware, which subsequently forwards it to the machines, initiating the manufacturing process.

3.3 Communication

In the realm of Industry 4.0, the selection of communication protocols like OPC-UA, Modbus, MQTT, Sparkplug, and HTTP is crucial for optimizing data exchange, ensuring security, and facilitating real-time updates across manufacturing systems. Each protocol offers unique benefits tailored to specific communication needs, ranging from simple device interactions to complex data exchange and integration across heterogeneous environments.

Given the project's focus on optimizing data exchange efficiency, ensuring real-time updates, and meeting performance requirements within a shop floor architecture, MQTT and HTTP emerged as the most suitable choices. MQTT, with its lightweight and efficient design, is ideally suited for real-time updates on product orders and production schedules, ensuring timely and accurate data exchange without significant overhead. Meanwhile, HTTP was selected for its robust data retrieval capabilities, making it perfect for fetching updated AAS from the database to the Middleware.

Figure 17 depicts the different interactions across every component, the communication protocol used and what data is sent.

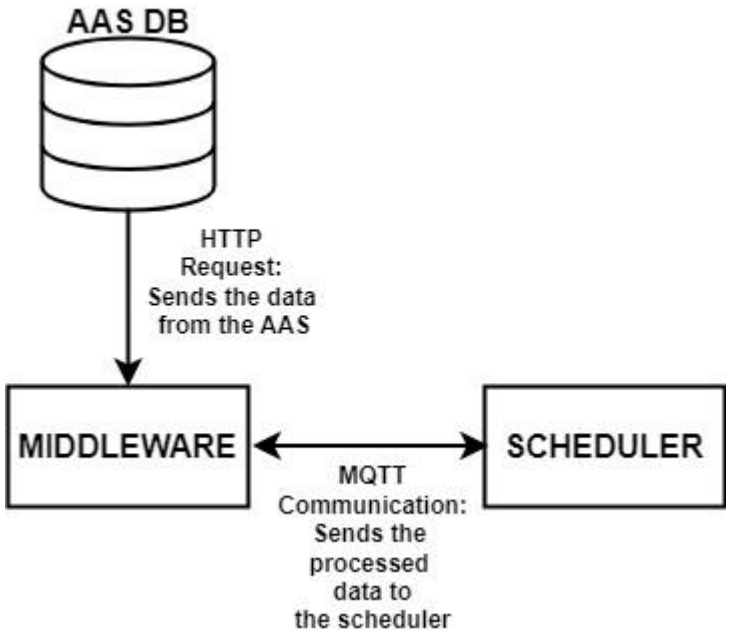

Figure 17 - Interactions between the system's entities

This selection reflects a comprehensive understanding of the operational dynamics of modern manufacturing environments and a commitment to leveraging cutting-edge technologies to drive operational improvements and competitiveness. By choosing MQTT and HTTP, the project aligns with the RAMI 4.0 framework's goals of enhancing operational efficiency, security, and real-time responsiveness, thereby contributing to the broader objectives of Industry 4.0.

3.4 Integration

In the dynamic environment of modern manufacturing, effective integration across all levels and entities is critical to maintaining efficiency and competitiveness. RAMI 4.0 provides a structured approach for achieving both vertical and horizontal integration, ensuring seamless communication and coordination throughout the enterprise and across the broader value network.

Vertical integration focuses on connecting all levels of the organization, from the shop floor to business management systems, to enable a seamless flow of information. Within the RAMI 4.0 framework, this involves integrating data from sensors, machines, and control systems with higher-level systems such as the MES and ERP.

The Middleware plays a crucial role in this integration by acting as a central hub that collects and processes data from various sources, such as machine statuses, production times, and operational parameters, which are critical for making informed decisions. This data is aggregated from the shop floor and made accessible to higher-level systems, allowing for real-time monitoring and control of production processes. Conversely, production orders, scheduling information, and business directives from ERP systems are transmitted downward through the Middleware to the Scheduler, which allocates tasks and resources on the shop floor. This bidirectional flow ensures alignment between strategic objectives and operational execution, enabling greater agility and responsiveness within the organization.

To visualize this, refer to Figure 18, which illustrates the RAMI 4.0 layers, where each component of the scheduling system is represented.

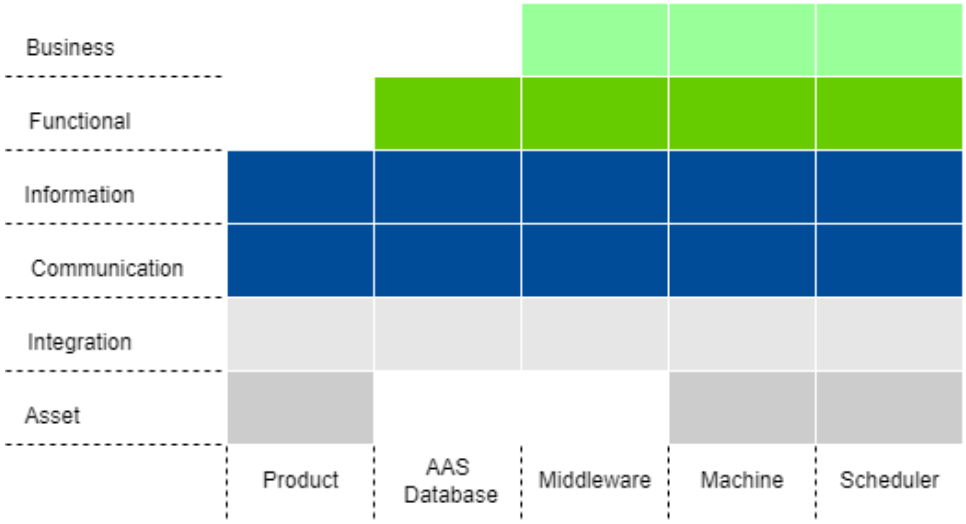

Figure 18 - System Components´ representation in the RAMI 4.0 layers

Horizontal integration is about creating a cohesive network of all components of the shop floor. However, in this case, there is not much to demonstrate for horizontal integration, as this was implemented within a single shop floor. RAMI 4.0 facilitates this integration by standardizing communication and data exchange, which promotes collaboration and transparency. The Middleware serves as the enabler, ensuring that data exchanged is consistent and standardized across different systems. Although the horizontal integration is limited to a singular environment, the Middleware still plays a key role in harmonizing information and facilitating efficient operations.

Together, vertical and horizontal integration through RAMI 4.0, supported by the Middleware, creates a fully integrated digital ecosystem. The Middleware serves as the

orchestrator, enabling real-time data flow and communication across all levels of the enterprise and with external partners. It ensures that the Scheduler has access to comprehensive and up-to-date information, allowing for optimized production planning and execution.

4

IMPLEMENTATION

This chapter details the design and implementation of the production scheduling system introduced in chapter 3. It highlights key components, data flow, and functionalities essential for dynamic scheduling. The Scheduler System section covers the MQTT broker, schedule and Gantt chart generation, and data storage. The Middleware section discusses using Node-RED for data integration and management. The AAS section explains digital asset representation for efficient scheduling and the AAS database implementation using Docker. An overview of the entire scheduling process is provided, ensuring optimized manufacturing workflows. Figure 19 illustrates the system architecture, communication protocols, and technologies used.

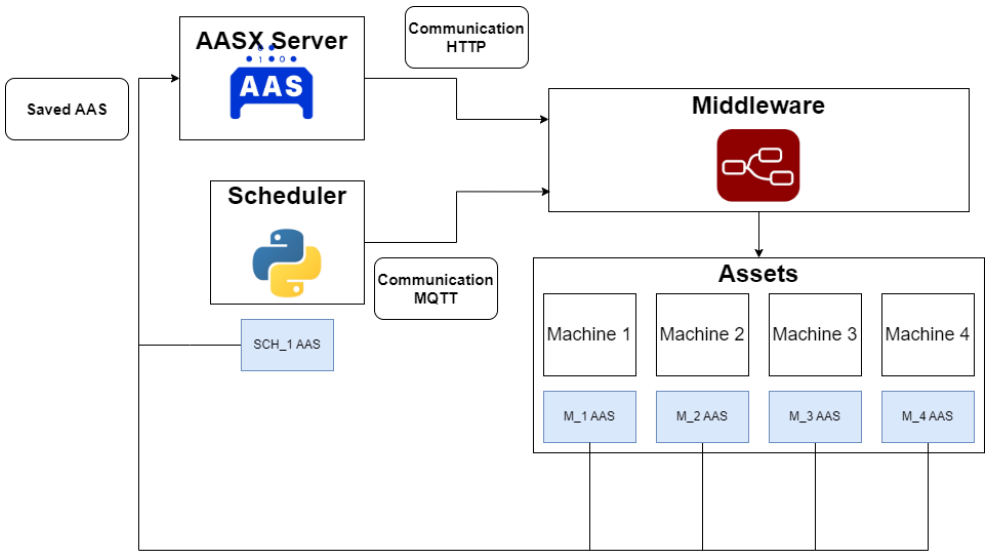

Figure 19 - System Architecture and technologies used

4.1 Scheduler System

The system comprises several key components, the scheduler itself, the Gantt chart generator, and data storage mechanisms. The MQTT broker serves as the communication hub, facilitating message exchange between the scheduler and the middleware. The scheduler, implemented as a Python script, subscribes to specific MQTT topics to receive data on machines, products, and orders. The Gantt chart generator uses matplotlib to visualize the production schedules, while the data storage component maintains records of product orders, machine availability, and usage times. Additionally, the scheduler is encapsulated within an administration asset shell, which acts as a digital twin for efficient management and interaction and will be explained in more detail in another section.

Figure 20 depicts a sequential diagram that showcases all the interactions within the Scheduling system that will be explained with more insight in this section.

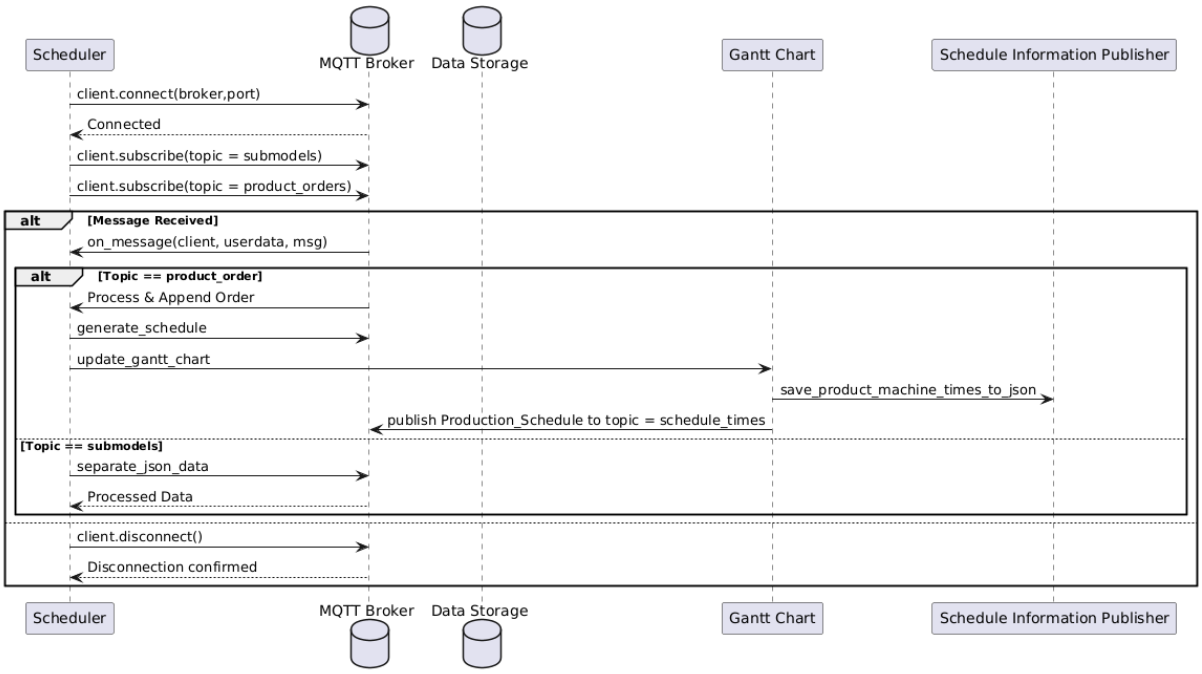

Figure 20 - Sequential Diagram detailing the Scheduler System

4.1.1 Data Flow and Processing

The *on_connect* function is invoked when the scheduler establishes a connection with the MQTT broker. The scheduler subscribes to two primary MQTT topics: *submodels* and *product_orders*. These topics provide data on machine and product details, as well as

incoming product orders. The scheduler processes messages from these topics using several callback functions. This real-time processing ensures that the scheduler can dynamically assign machines and update production schedules based on the latest available information.

The *on_message* function processes incoming messages based on their topics. When a message is received on the *product_orders* topic, the function decodes the message payload to extract the product order and adds it to the product_orders queue. It then calls *generate_schedule* to handle the new order immediately. Messages received on the submodels topic are expected to contain JSON data about machines and products, which are separated and stored in *machine_json* and *product_json* using the *separate_json_data* function.

4.1.2 Key Functionalities

The core functionality of the scheduler system is centered around the *assign_machines* function, which plays a crucial role in effectively managing the allocation of machines to process product orders. This function operates by evaluating the requirements of each product and determining the most suitable machines based on their current availability. The process begins by initializing data structures to track machine choices and assignment times. It also considers a wait time interval between the completion of one task and the start of another to account for transition periods made between machines.

The *assign_machines* function starts by transposing the product requirements, allowing it to process each requirement step-by-step across all products. For each step, the function iterates through the requirements, assessing available machines that meet the criteria. This is done through the *find_available_machines_for_requirement* function, which filters machines based on their capability to handle the specific requirement and their next available time. The function then selects the most suitable machine by finding the one that becomes available the soonest. If multiple machines are suitable, it picks the one with the earliest available time, ensuring that tasks are assigned as soon as possible.

The selected machine’s availability is updated to reflect the end time of the task, and the start and end times for each task are calculated. These times are determined by considering the machine's current availability, the processing time required for the task, and a defined wait time interval for transportation. The processing time is extracted from the machine JSON data, which details how long each machine takes to complete a task. The start time for a task is set to the maximum of the machine's next available time or the current time

for the product, plus the wait time interval. The end time is then calculated by adding the processing time to the start time.

The function maintains lists of start and end times for each machine assignment, ensuring that these are accurately tracked and updated throughout the scheduling process. If a suitable machine is not available for a particular requirement, the system logs this instance for further review, allowing for continuous improvement in the scheduling process.

Overall, the *assign_machines* function ensures that each product order is processed in an optimized manner, assigning tasks to machines based on their capabilities and availability while effectively managing the timing of each assignment to enhance overall production efficiency. Below a pseudo-code is shown for better understanding of the *assign_machines* function:

Find suitable machines for current requirement:
- Call find_available_machines_for_requirement(machine_json, individual_req.strip(), current_time)
If suitable machines found:
- Initialize variables to track selected machine and earliest end time
Iterate through suitable machines:
- Select machine available soonest
Calculate start and end times:
- Get processing time from extract_processing_times()
- Define start time with wait interval
- Calculate end time based on start time and processing time
Update machine choices, start times, and end times
Update machine availability
Update product time
Else:
Print message indicating no available machine found

4.1.3 Scheduling and Visualization

The scheduling and visualization process in this system involves allocating machine resources to meet production orders and displaying the results in a Gantt chart. The primary goal is to efficiently manage machine assignments while providing a clear visual representation of the production schedule.

The *update_gantt_chart* function is central to this process. It begins by initializing arrays to track the start and end times of machine tasks. For each product and machine, it

populates these arrays with relevant timing information, ensuring that each machine's schedule is accurately captured. This data is then used to generate a Gantt chart, which provides a visual timeline of machine assignments.

To create the Gantt chart, the function first determines unique product instances and assigns colors to them, ensuring that repeated products are distinguished by different hues if specified by user preferences. Each product is given a unique identifier, and a color map is created to facilitate clear visual differentiation.

Using *matplotlib*, the function then plots horizontal bars representing each machine's tasks on the Gantt chart. The start and end times for each task are translated into bar lengths, and colors are applied based on the product being processed. The Gantt chart is configured to include labels for machines and a legend for products, making it easy to interpret the schedule. Adjustments are made to ensure the legend fits well within the plot and the layout is optimized for readability.

Overall, the Gantt chart offers an intuitive and visually appealing representation of the production schedule, allowing stakeholders to quickly assess machine utilization and task timings.

4.1.4 Data Persistence and Publishing

The *save_product_machine_times_to_json* function ensures that the processed scheduling data is saved to a JSON file. This data includes the start and end times for each task, maintaining a record of the production schedule for future reference. Additionally, the function publishes the scheduling data to the *Production_Schedule* topic, allowing other systems or users to access the generated schedule and stay informed about the production plan.

Data persistence and publishing are crucial for maintaining a record of the scheduling decisions and making this information available for further use or integration with other systems. The *save_product_machine_times_to_json* function is responsible for this aspect of the system.

After generating the scheduling data, the function first creates a timestamped JSON file to store the details of machine assignments, including start times, process times, and end times for each product. This file is saved to a specified directory, ensuring that the data is securely stored and easily retrievable for future reference or analysis. The filename includes a timestamp to uniquely identify each scheduling instance.

Following the creation of the JSON file, the function publishes the scheduling information to an MQTT topic named *Production_schedule*. This allows other systems or stakeholders to access the latest scheduling data in real time. The published message includes the filename of the JSON file and the scheduling details, enabling recipients to retrieve the complete set of information from the specified file.

By combining local data persistence with real-time MQTT publishing, the system ensures that scheduling information is both archived and readily available for further use. This approach facilitates ongoing monitoring and integration with other production management tools, enhancing the overall efficiency and transparency of the scheduling process.

4.2 Middleware Implementation

The middleware for the production scheduling system is implemented using Node-RED, a powerful flow-based development tool. Node-RED allows for easy integration and management of data flows between various systems and components. This middleware is designed to retrieve and process data from a database, encode necessary information, construct specific URLs, and then publish the processed data to an MQTT topic.

A detailed representation of the Node-RED flow depicted by Figure 21 illustrates the sequence of nodes used for data retrieval, encoding, URL construction, HTTP requests, data aggregation, JSON processing, and MQTT publication. This flow diagram visually encapsulates the step-by-step process described below.

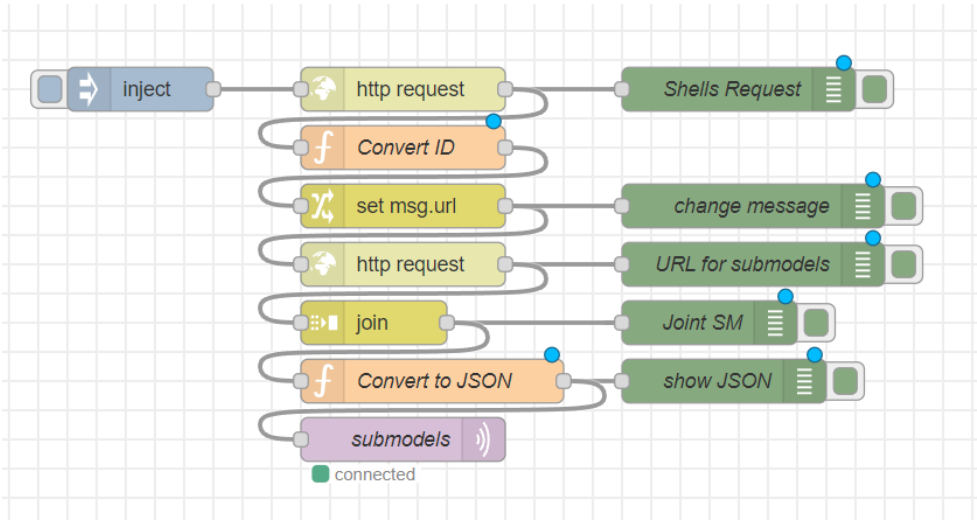

Figure 21 - Flow for AAS Data retrieving and processing

4.2.1 HTTP Request and Initial Data Retrieval

The process begins with an HTTP request node in Node-RED, which fetches all the shells (like the one present in Figure 22) from the database. This request is designed to retrieve comprehensive information on machines and products, encapsulated within a structured payload. Once the data is retrieved, it is passed to a function node for further processing.


```
{
  "idShort": "Machine1",
  "id": "M_1",
  "assetInformation": {
    "assetKind": "Instance",
    "globalAssetId": "https://example.com/ids/asset/0012_1161_5042_2243"
  },
  "submodels": [
    {
      "type": "ModelReference",
      "keys": [
        {
          "type": "Submodel",
          "value": "SM_M1"
        }
      ]
    }
  ],
  "modelType": "AssetAdministrationShell"
},
```


Figure 22 - Example of a shell taken by the HTTP request

4.2.2 Data Encoding and URL Construction

The function node is responsible for encoding specific parts of the data and constructing URLs. It first defines a *base64UrlEncode* function to convert strings to Base64 URL format, which replaces certain characters and removes padding to make the string URL-safe.

The function node then processes the message payload, iterating over each item in the retrieved result array. For each item, it extracts the *id* (show as M_1 in Figure 22) and the first submodel value (which will be interpreted as the submodel's own id). If either of these values is missing, it assigns a default value and logs a warning. These values are then encoded using the *base64UrlEncode* function. The encoded values are used to construct a URL, which includes the encoded *id* and *submodel* values as query parameters. This URL is then set as the payload of a new message, which is sent out independently.

4.2.3 Subsequent HTTP Requests and Data Aggregation

After encoding and URL construction, the middleware uses a set message URL node to set the constructed URLs as the payload. Another HTTP request node is used to send these URLs and retrieve the corresponding data from the target server. The responses from these HTTP requests are what is called a serialization, which can be interpreted as a JSON string, detailing all the necessary information needed for the scheduler as Figure 23 illustrates. These responses are then aggregated using a join node, which combines the multiple responses into a single message payload.

```
"assetAdministrationShells": [
  {
    "idShort": "Machine1",
    "id": "M_1",
    "assetInformation": {
      "assetKind": "Instance",
      "globalAssetId": "https://example.com/ids/asset/0012_1161_5042_2243"
    },
    "submodels": [
      {
        "type": "ModelReference",
        "keys": [
          {
            "type": "Submodel",
            "value": "SM_M1"
          }
        ]
      }
    ],
    "modelType": "AssetAdministrationShell"
  }
],
"submodels": [
  {
    "idShort": "MachineData",
    "id": "SM_M1",
    "kind": "Instance",
    "submodelElements": [
      {
        "category": "PARAMETER",
        "idShort": "MachineFunctions",
        "valueType": "xs:string",
        "value": "Glue ",
        "modelType": "Property"
      },
      {
        "category": "VARIABLE",
        "idShort": "MachineTime",
        "semanticId": {
          "type": "ExternalReference",
          "keys": [
            {
              "type": "GlobalReference",
              "value": "https://example.com/ids/cd/6162_1161_5042_9473"
            }
          ]
        },
        "valueType": "xs:integer",
        "value": "30",
        "modelType": "Property"
      }
    ],
    "modelType": "Submodel"
  }
]
}
```


Figure 23 - Example of a serialization taken by the HTTP request

4.2.4 JSON Processing and Validation

The aggregated data, now in the form of an array of JSON strings, is passed to another function node. This function checks whether the payload is a valid array and whether each string in the array is valid JSON. It converts each valid JSON string into a JSON object, filtering out any invalid entries. The resulting array of JSON objects is then set back to the message payload.

4.2.5 Publishing to MQTT

Finally, the processed data, now an array of JSON objects, is published to an MQTT topic named *submodels* using an MQTT node. This ensures that the updated machine and product details are made available to the production scheduling system, allowing it to dynamically adjust to the latest information. The seamless flow from data retrieval to processing and publishing ensures that the scheduler receives accurate and timely updates, facilitating efficient production management.

4.2.6 Handling Product Orders from a File

In addition to processing machine and product details, the middleware includes a separate flow to handle product orders as Figure 24 illustrates.

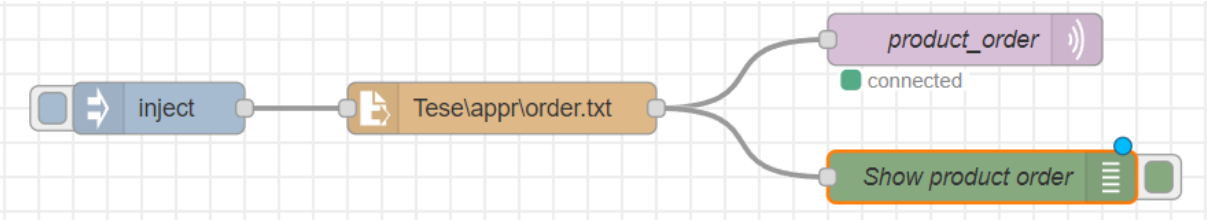

Figure 24 - Flow for Product Order retrieval

This flow begins with a read file node, which reads product order data from a specified file. The content of this file, representing new product orders, is then passed to an MQTT node. This node publishes the product order data to the *product_orders* topic, ensuring that the production scheduler is updated with new orders from external sources. This integration allows the system to process incoming orders efficiently and maintain an up-to-date production schedule based on real-time data inputs.

4.2.7 Updating Scheduler AAS

The process of updating the products list is depicted by Figure 25, which depicts a Node-RED flow that receives the product order via MQTT. Within this flow, a function node sets the message headers to accept all content types and specifies the content type as application/json. The message payload is then converted into a JSON string with a key of *Products_List* containing the received product order. Following this, a method node is set to *PATCH*, which updates the products_list property in the information submodel with the latest product order, and finally, an HTTP request node completes the update process.

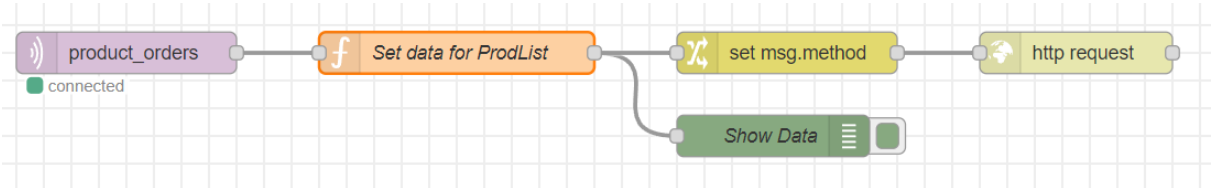

Figure 25 - Flow responsible for Updating the Product List in the Scheduler AAS

4.3 Asset Administration Shells Implementation

In this project, AAS were implemented to digitally represent the scheduler, the products and the machines within the manufacturing system. Using the AASX Explorer tool, the development of these AAS was made in order to capture detailed information necessary for efficient production scheduling. AASX Package Explorer is a tool with a graphical user interface designed for creating, viewing, and editing AAS packages, which are a key component of Industry 4.0 digital twin technologies. It includes features like an internal REST (Figure 26) and OPC UA server for interacting with AAS packages, making it suitable for both technical and non-technical users to experiment with and demonstrate the potential of digital twins. The tool aims to simplify the adoption and integration of Digital Twin technologies with minimal IT infrastructure investment. The AAS framework allows for easy retrieval and interaction with the data related to each asset, streamlining the scheduling process and enhancing interoperability between various components of the system.

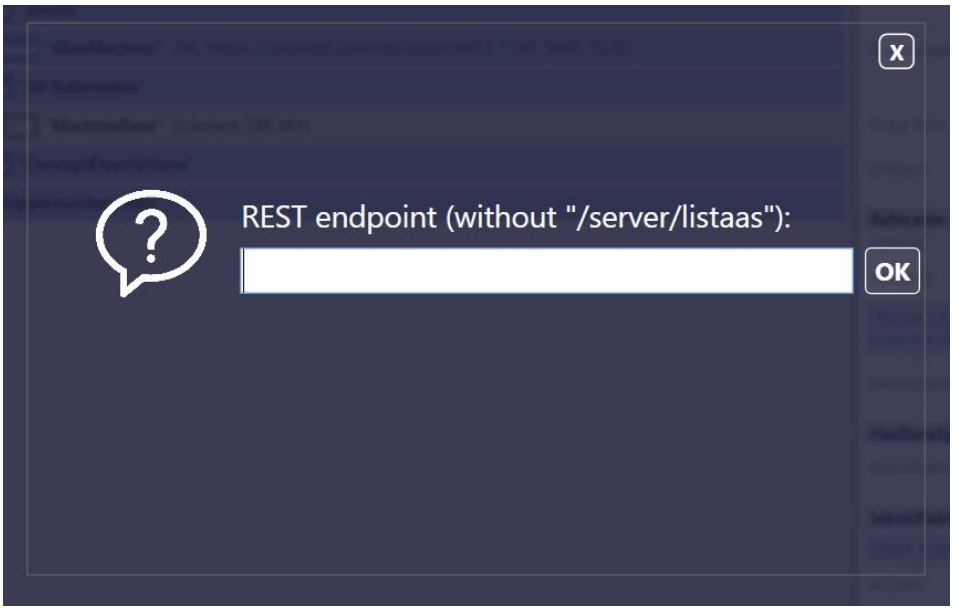

Figure 26 - Connection to REST repository

AASX Explorer was used to create the AAS for our manufacturing system. This tool helped organize asset data into structured, standardized formats by utilizing submodels and their submodel elements (properties). This ensured that both product and machine information could be seamlessly integrated into the production scheduling workflow. Furthermore, the submodels and their properties can be fully customized to suit specific requirements, providing flexibility in data management and representation. This customization is illustrated in Figure 27 and Figure 28.

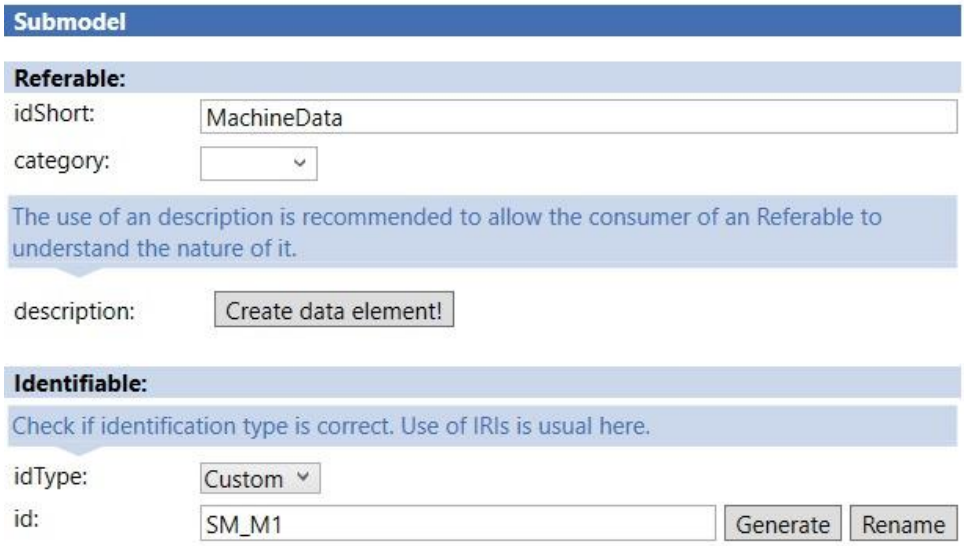

Figure 27 - Submodel creation in AASX Explorer

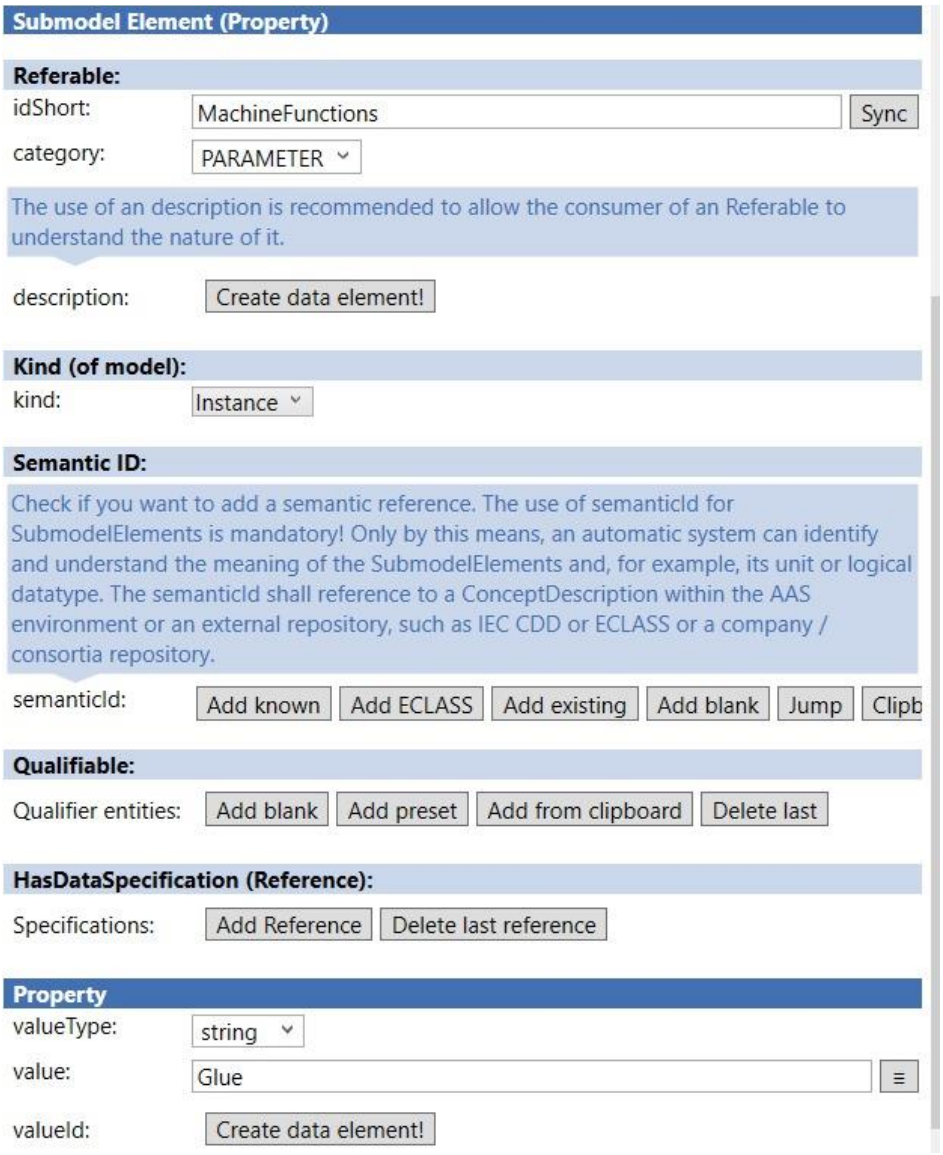

Figure 28 - Property Creation in AASX Package

Figure 29, Figure 30 and Figure 31 detail the AAS of products, machines and the scheduler in the AASX Package Explorer.

Figure 29 - Example of a Machine AAS shown by the AASX Package Explorer

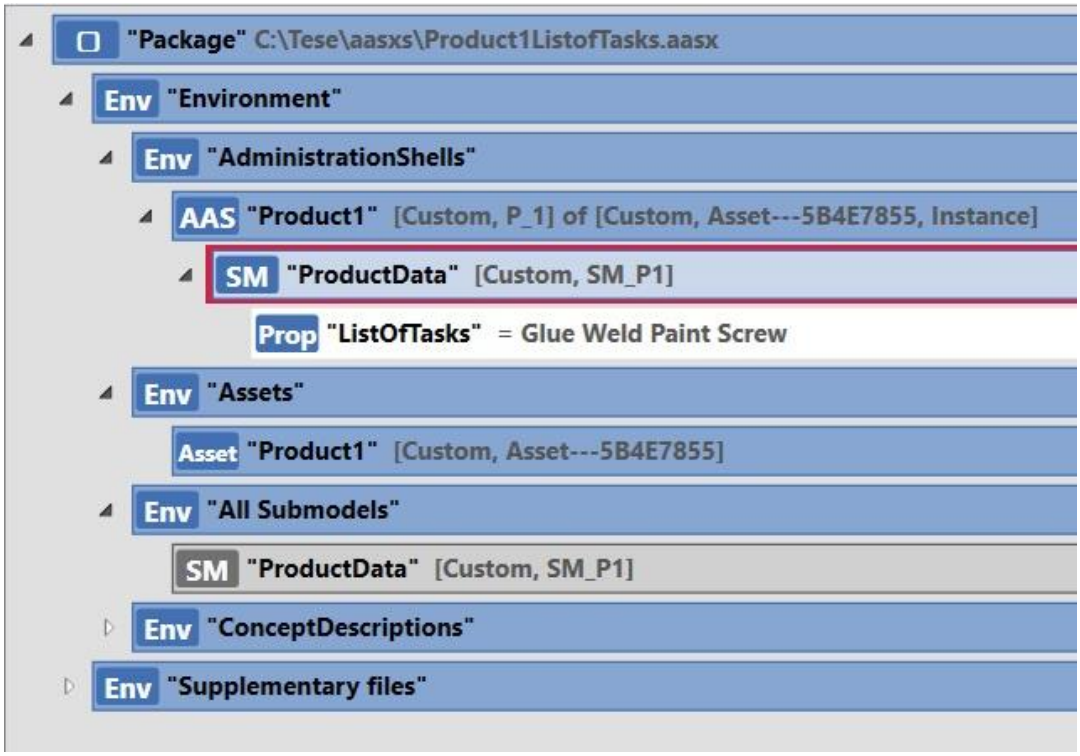

Figure 30 - Example of a Product AAS shown by the AASX Package Explorer

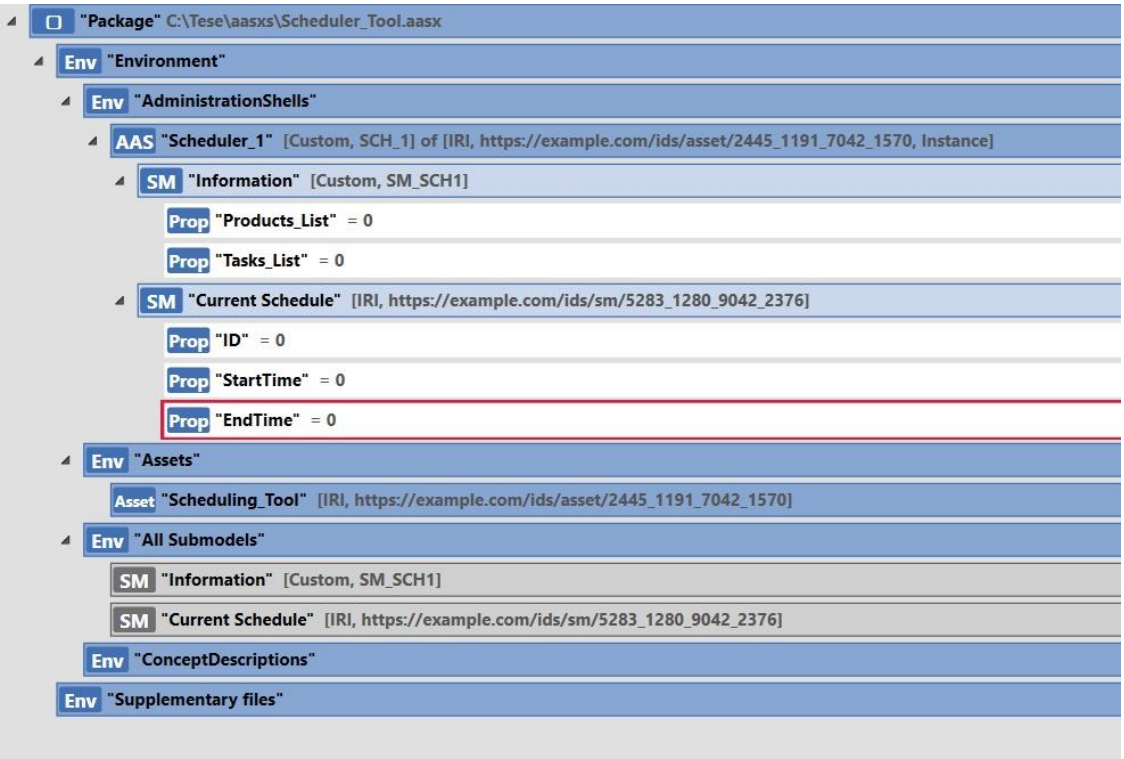

Figure 31 - Scheduler AAS shown by the AASX Package Explorer

4.4 AAS Database Implementation

In this project, the AAS database is managed using an AASX server, deployed through Docker. This setup provides a robust and scalable solution for hosting and managing AAS data, ensuring that it is readily accessible for various applications, including our production scheduler.

Docker is a platform that allows developers to package applications and their dependencies into a standardized unit called a container. Containers are lightweight, portable, and can run consistently across different computing environments. By using Docker, we can encapsulate the AASX server along with its configuration and dependencies, ensuring that it runs reliably regardless of where it is deployed.

The AASX server is launched with a Docker Compose configuration. In this setup, the server is set to restart automatically unless stopped manually, ensuring high availability. It exposes port 8000, which maps to the internal port 5001 of the container, making the server accessible via HTTP at *http://localhost:8000*. The server configuration specifies no security measures for simplicity and directs the server to use an external Blazor application hosted at the same URL.

Figure 32 depicts the AASX Server's interface.

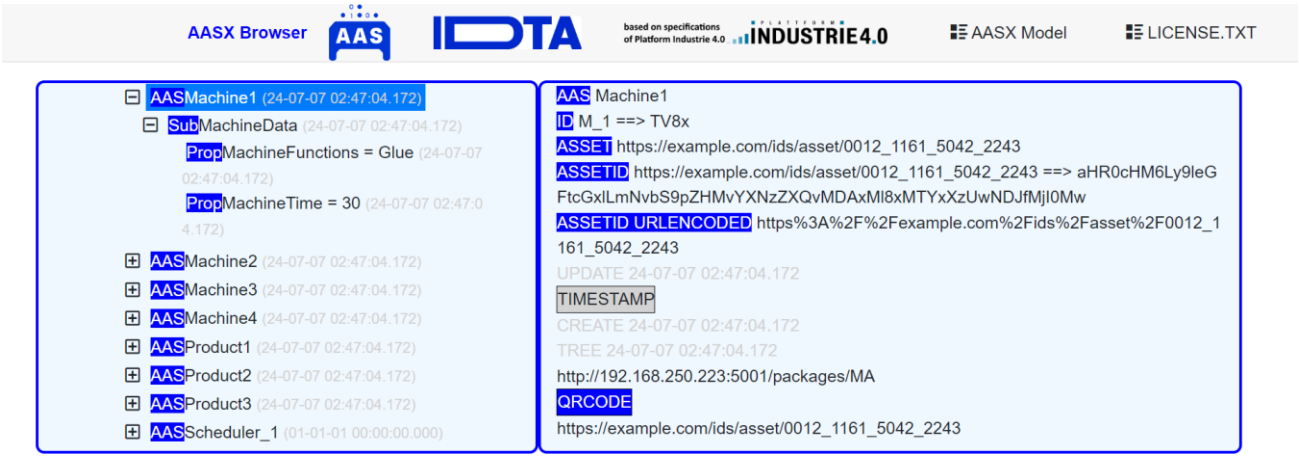

Figure 32 - AASX Server Overview

The volumes configuration in Docker allows the server to access and manage the AASX files stored on the host machine. This ensures that any changes made to the AAS data are persistent and survive container restarts or updates.

In addition to hosting the AAS data, the AASX server includes a Swagger API, which provides a standardized interface for interacting with the AAS data programmatically. The API requires data to be base64 encoded, ensuring that the data is transmitted safely and without corruption. This encoding is handled in the middleware, ensuring that all data exchanged with the API meets this requirement.

The use of Docker for the AASX server offers several advantages, including portability, consistency across environments, and ease of deployment. The inclusion of a Swagger API further enhances the server's functionality, allowing seamless programmatic access to the AAS data. This setup not only supports the efficient management of AAS data but also integrates smoothly with the production scheduling system, facilitating dynamic and optimized manufacturing processes.

4.5 Scheduling Process Overview

This section will outline the process of the dynamic production scheduling system, starting from data acquisition from the server and then proceeding through the steps that lead to schedule generation. A sequential diagram detailing the process is shown in Figure 33. For easier interpretation the Scheduler, Gantt chart and Data Storage are shown as different entities, but in reality these all exist within the scheduler system.

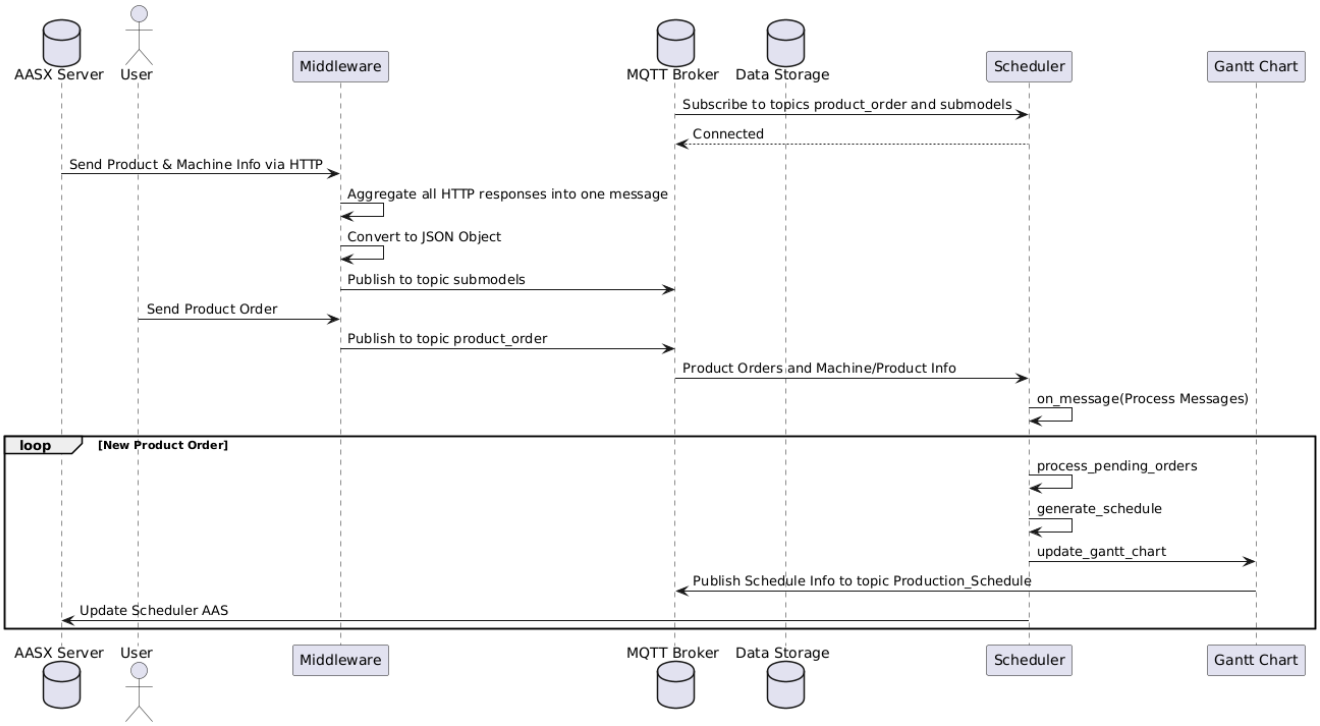

Figure 33 - Schedule Process Overview

4.5.1 Data Acquisition from AASX Server

The system initiates by retrieving comprehensive data from the AASX server using the Node-RED middleware. Through HTTP requests, detailed information about machines and products is fetched from the server, encapsulated in a structured payload. This data retrieval ensures othat the production scheduler has up-to-date insights into the capabilities and statuses of all relevant assets.

4.5.2 Middleware Processing and Data Integration

Upon receiving the data payload, Node-RED processes it in a function node. Here, specific components are base64 encoded to ensure secure transmission, and URLs are constructed using these encoded values. These URLs facilitate subsequent HTTP requests to fetch additional detailed data from the server, such as maintenance schedules or operational parameters for machines.

4.5.3 Aggregation and Validation of Data

The responses from these HTTP requests are aggregated into a unified message payload using a join node in Node-RED. This aggregation step consolidates all relevant information about machines and products into a single dataset. Subsequently, the aggregated data undergoes validation to ensure it adheres to JSON standards. Invalid entries are filtered out, leaving behind a refined array of JSON objects that accurately represent the current state of machines and products.

4.5.4 Publication to MQTT Topic (submodels)

Once validated, the refined JSON data is published to an MQTT topic named submodels. This MQTT topic serves as a communication channel through which the production scheduling system receives updated information about machine capabilities, product specifications, and other essential details. By subscribing to this topic, the scheduler ensures it always has the latest data to inform its decision-making process.

4.5.5 Real-time Processing of Product Orders

Simultaneously, the scheduler system listens on MQTT topics, including product_orders, where new product orders are received in real-time. When a new order arrives, the scheduler decodes the message payload to extract the product order details. These orders are then queued for processing through the process_pending_orders function, ensuring prompt handling and scheduling of production tasks.

4.5.6 Product Order Processing and Machine Assignment

The *process_pending_orders* function iterates through the queued product orders. For each order, it extracts product requirements and utilizes the assign_machines function to assign suitable machines based on their availability and capabilities. This assignment process matches product specifications with machine functionalities, ensuring optimal task allocation.

4.5.7 Dynamic Gantt Chart Visualization

Following machine assignment, the *update_gantt_chart* function generates dynamic Gantt charts using matplotlib. These charts visualize the production schedule by plotting tasks' start and end times on specific machines. Such visual representations facilitate the identification of

potential bottlenecks and aid in optimizing the scheduling process for enhanced efficiency and productivity.

4.5.8 Data Persistence and Publication for Transparency

Furthermore, the *save_product_machine_times* function ensures the persistence of processed scheduling data by saving it to a JSON file. This file serves as a historical record of the production schedule, facilitating traceability and analysis of past operations. Additionally, the scheduling data is published to a predefined MQTT topic, enabling other systems or stakeholders to access and stay informed about the current production plan.

4.5.9 Error Handling and System Reliability

Throughout the process, robust error handling mechanisms within the on_message function manage potential issues like JSON decoding errors or communication disruptions. These mechanisms ensure the system operates reliably in real-time, effectively responding to changes in machine statuses, incoming product orders, or other operational variables.

4.5.10 Integration with AASX Explorer for Detailed Representation

The system leverages the AASX Explorer tool to digitally represent machines and products within the manufacturing environment. This tool utilizes AAS to capture detailed information essential for efficient production scheduling. The Product AAS includes specifics such as unique identifiers, production requirements, and design specifications, while the Machine AAS details capabilities, maintenance schedules, and operational parameters. These representations enhance the system's ability to manage and optimize production activities seamlessly.

5

TESTS AND RESULTS

This chapter details the evaluation of the production scheduling system through various tests, aimed at examining its efficiency and capability under different conditions. The tests explore the system's performance with varying numbers of products and machines, providing insights into its scalability and effectiveness in optimizing production schedules.

To evaluate the system's performance accurately, time measurements were conducted from the moment a product order was received until the completion of the scheduling process. This approach focused on capturing the time taken by the system to generate a production schedule, excluding the actual production time. The use of the *time.perf_counter()* function ensured high-resolution timing to capture even minor fluctuations in execution time, providing precise insight into the system's scheduling efficiency. For each scenario, the execution time was measured three times, and the average value was calculated from those measurements.

To explore the system's performance, multiple test scenarios were created, each involving different numbers of products and machines. The primary aim was to understand how the system responds to changes in workload and resource availability, providing insights into its ability to manage diverse production requirements.

For the purpose of these tests, it was assumed that each machine performs a specific function, with these functions assigned to the machines in a hypothetical manner (i.e., not based on real-world constraints or actual machine capabilities). Specifically, in scenarios involving eight machines, each pair of machines was set to perform the same function to simulate increased production capacity:

- M_1 and M_5 perform the glue function, bonding components together. This operation takes 30 minutes.

- **M_2** and **M_6** perform the **screw** function, fastening components securely. This operation takes 30 minutes.
- **M_3** and **M_7** perform the **weld** function, joining metal parts through welding. This operation takes 10 minutes
- **M_4** and **M_8** perform the **painting** function, applying paint or protective coatings to the finished products. This operation takes 20 minutes

This configuration was chosen to evaluate the potential benefits of parallel processing when additional machines are available. By doubling the number of machines assigned to each function, the system's ability to handle more tasks simultaneously was tested, highlighting its capacity to reduce idle times and improve overall production efficiency.

To simplify chart evaluation, an update to the *update_gantt_chart* function was created that assigns different colors to repeated products in the product order. If a product appears multiple times, it is distinguished by appending an index to its label. For instance, the first occurrence of "P_4" will be labeled "P_4_0", and the second as "P_4_1", and so on.

The tests were conducted across several scenarios to assess the system's performance with different configurations:

1. **Test 1**: Four machines handling a production order of five products.
2. **Test 2**: Eight machines handling the same production order of five products.
3. **Test 3**: Four machines handling a production order of sixty products.
4. **Test 4**: Eight machines handling a production order of sixty products.
5. **Test 5**: Four machines handling a production order of one hundred and twenty products.
6. **Test 6**: Eight machines handling a production order of one hundred and twenty products.
7. **Test 7:** Four machines handling a production order of four products in a physical production simulation kit.

5.1 Production order of five products

In Test 1, the system was tasked with scheduling five products across four machines. The Gantt chart, presented in Figure 34, demonstrates a well-organized schedule with effective task allocation. The scheduling process took on average 0.34 seconds from the moment the product order was received until the completion of the scheduling process. This test confirmed that the system is capable of handling a straightforward scheduling scenario with

a manageable number of machines and products, laying a solid foundation for more complex scenarios.

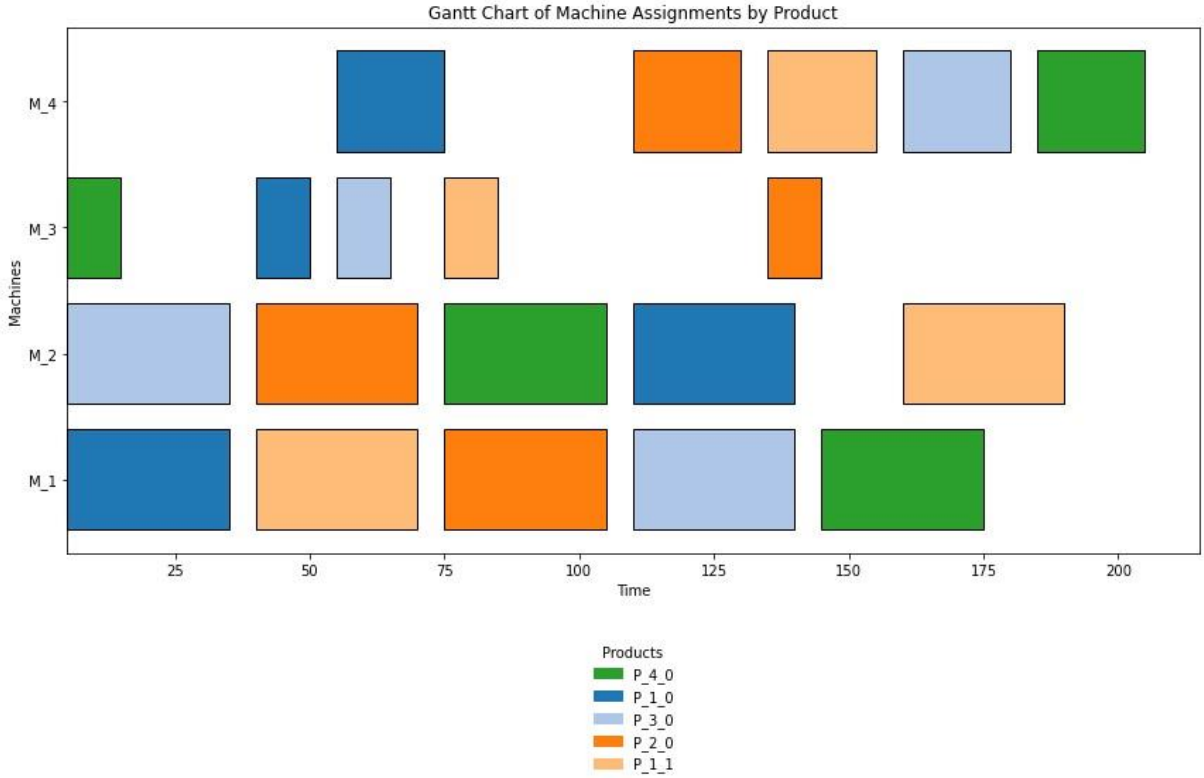

Figure 34 - Gantt Chart for five products and four machines

The production time was reasonable, and the system efficiently managed idle periods. This test confirmed that the system is capable of handling a straightforward scheduling scenario with a manageable number of machines and products, laying a solid foundation for more complex scenarios.

Expanding the machine count to eight while keeping the product count at five provided a chance to assess the benefits of additional machines. The Gantt chart for this scenario, shown in Figure 35, illustrates a more streamlined schedule with reduced idle times.

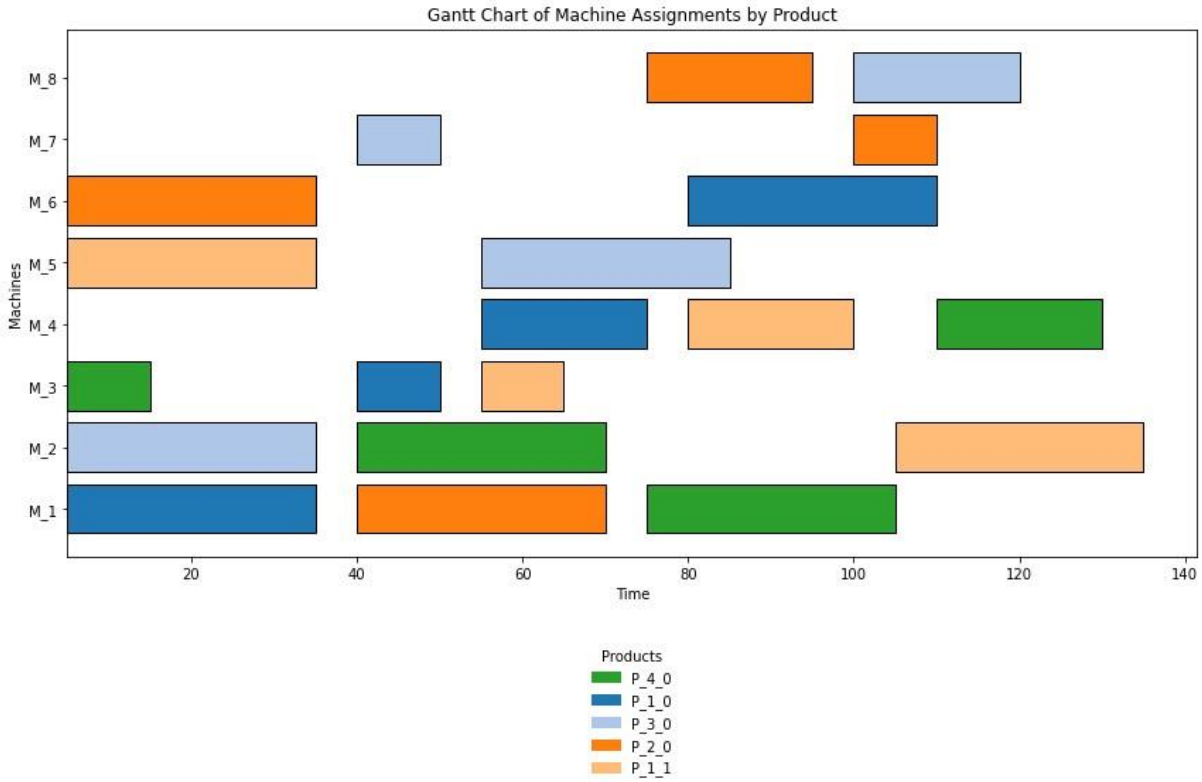

Figure 35 - Gantt Chart for five products and eight machines

The scheduling time slightly increased to 0.39 seconds. Despite this minor increase, the system utilized the extra machines to process tasks in parallel more effectively, leading to a decrease in the total production time compared to Test 1 as it would be expected. This test highlighted the advantage of having more machines in enhancing scheduling efficiency even with a low product count.

5.2 Production order of sixty products

In this test, the system was challenged with scheduling sixty products across four machines. The Gantt chart, depicted in Figure 36, reveals a more complex schedule with increased idle times and longer production durations. The scheduling process, on average, took 1.21 seconds.

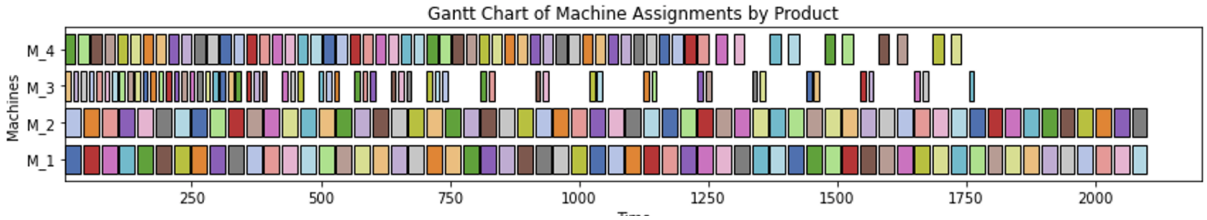

Figure 36 - Gantt Chart for sixty products and four machines

The limited number of machines led to longer scheduling times as the system struggled to allocate tasks efficiently across the available resources. This test underscored the limitations of having fewer machines when dealing with a large volume of products, showing that the system's performance could be constrained by the number of available machines.

The scenario of scheduling sixty products with eight machines aimed to test the benefits of additional machines for large orders. The Gantt chart for this test, shown in Figure 37, illustrates a more efficient schedule with reduced idle times when compared to the previous test, though the scheduling time slightly increased to 1.24 seconds.

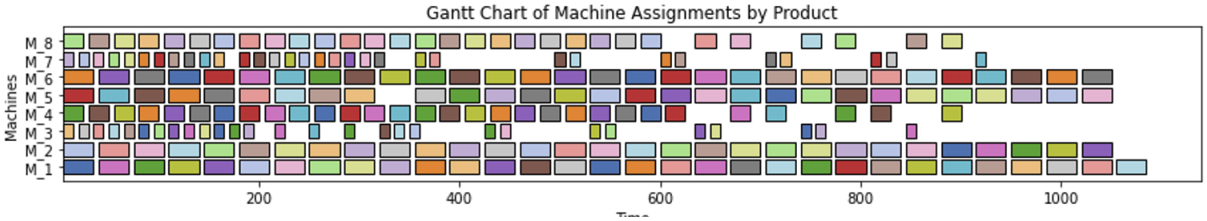

Figure 37 - Gantt Chart for sixty products and eight machines

The increased number of machines allowed for better parallel processing and task distribution, improving the overall scheduling process. This test highlighted the system's ability to handle large product orders effectively when ample resources are available.

5.3 Production order of one hundred and twenty products

The challenge in this test was scheduling one hundred and twenty products using only four machines. The Gantt chart, shown in Figure 38, reflects a highly congested schedule with extended production times and increased idle periods. The scheduling process, on average, took 2.40 seconds.

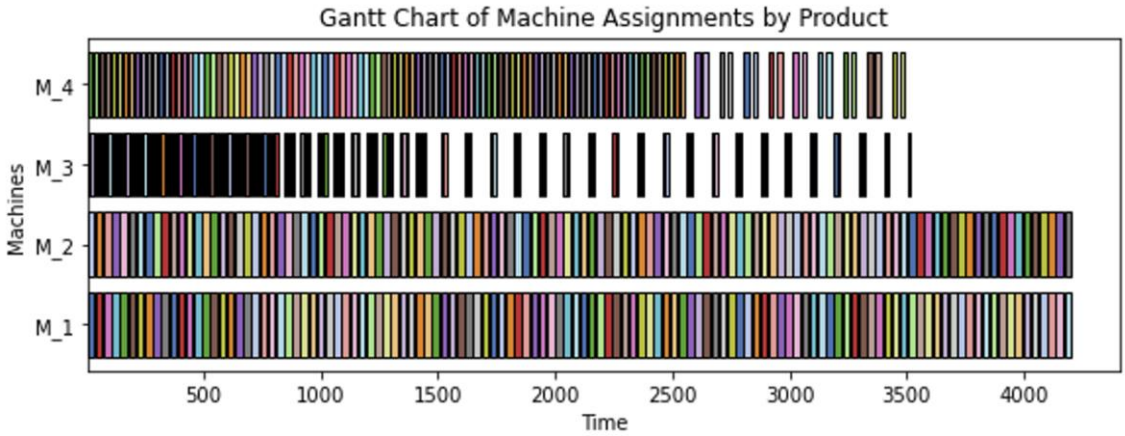

Figure 38 - Gantt Chart for one hundred and twenty products and four machines

The limited number of machines struggled to keep up with the high product volume, leading to less efficient task management and a longer scheduling process. This test demonstrated the limitations of the system when operating under resource constraints with very large orders.

In this final test, the system managed one hundred and twenty products across eight machines. The Gantt chart for this scenario, presented in Figure 39, shows a significant improvement in production time with a comparable scheduling time, but with better task distribution and reduced idle time. The scheduling process took 2.45 seconds

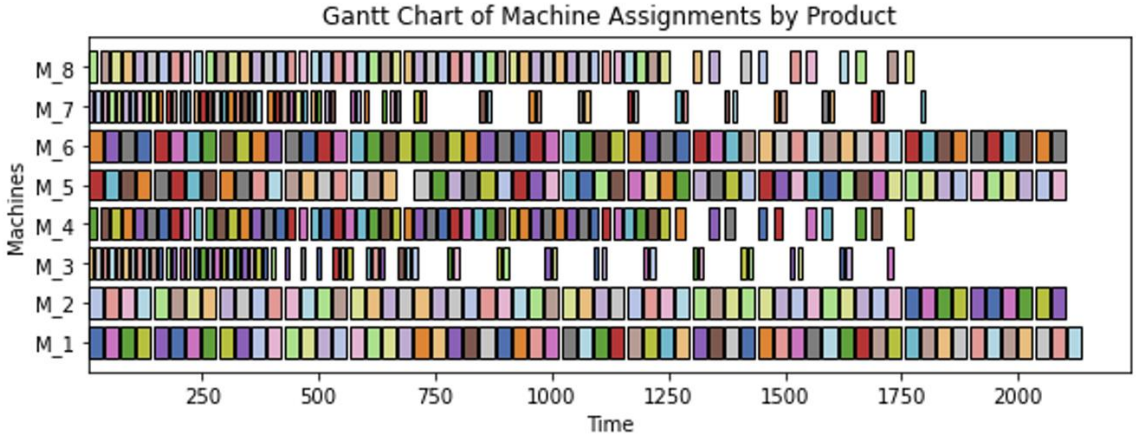

Figure 39 - Gantt Chart for one hundred and twenty products and eight machines

The increased number of machines allowed for effective parallelism, reducing idle times and shortening the overall production duration. The well-organized schedule highlights the system's capability to scale effectively with both product volume and machine count, providing a compact and efficient production process.

5.4 Production order of Four Products in a Physical Production Simulation Kit

To further evaluate the production scheduling system's performance in a real-world context, an additional test was conducted using a physical production simulation kit. This test involved four machines handling a production order of four products, simulating an environment with actual production constraints and machine interactions. The aim was to assess the system's capability to manage scheduling in a tangible setting, where variables such as machine response times, communication delays, and unexpected events could impact performance.

The physical simulation kit shown in Figure 40 comprises four machines (M_1, M_2, M_3, M_4), whose functions are the same as in the previous sections.

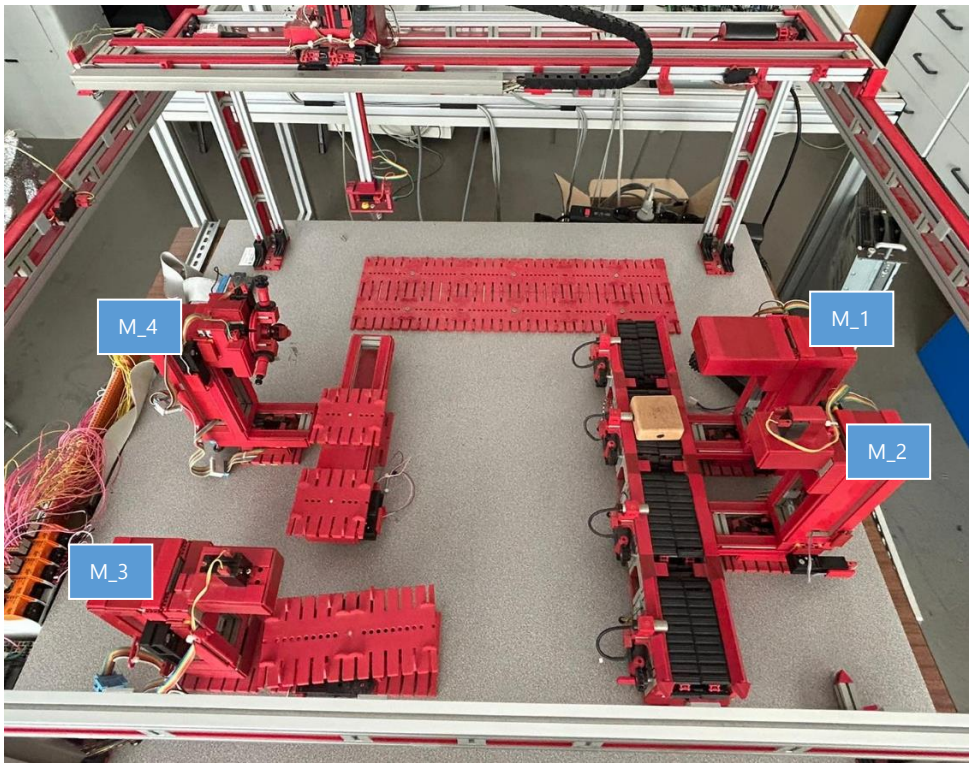

Figure 40 - Physical Simulation Kit with corresponding machine IDs

To manage the execution of the production schedule in this setup, a monitoring function was employed to iterate through the JSON object published on the *Production_Schedule* MQTT topic. This function continuously checked the machine IDs and their corresponding start times. When the scheduled start time for a machine arrived, the function sent a value of 1 to an MQTT topic specific to that machine (e.g., machine_1, machine_2, etc.). This message

signaled the middleware to initiate the machine's operation. Below an example of this JSON object is shown:


```
{
    "product": "P_1_0",
    "machine": "M_5",
    "start_time": 5,
    "process_time": 30,
    "end_time": 35
},
{
    "product": "P_3_1",
    "machine": "M_6",
    "start_time": 5,
    "process_time": 30,
    "end_time": 35
},
{
    "product": "P_2_2",
    "machine": "M_6",
    "start_time": 40,
    "process_time": 30,
    "end_time": 70
},
```


Once a machine completed its assigned task, it sent a "done" message back to the middleware. This acknowledgment enabled the system to track the entire production process in real time, ensuring that the completion of one machine's task could trigger the next scheduled task without delay. If a machine's "done" message was not received within an expected time window, the system would detect this issue and automatically regenerate the production schedule to account for the delay or machine malfunction. This feature ensured that the scheduling system could adapt to unexpected disruptions and continue operations with minimal impact on overall production flow.

The physical simulation test effectively demonstrated the production scheduling system's ability to manage real-world constraints and dynamic interactions between machines. The system's monitoring and real-time tracking functions ensured smooth coordination, while its adaptability to disruptions, such as missed machine acknowledgments, highlighted its

robustness in handling unexpected events. Overall, the test validated the system's capability to optimize production scheduling in a tangible setting, confirming its readiness for more complex and larger-scale production environments.

5.5 Final Results and Insights

The results across the various tests provide several key insights:

Machine Count Impact: Increasing the number of machines consistently improves scheduling efficiency. This benefit is particularly notable when managing larger product orders, as additional machines help reduce idle times and overall production durations. The Gantt charts for larger orders with more machines show better task distribution and reduced production times, validating the effectiveness of resource scaling.

Product Volume Management: The system demonstrates its ability to handle both small and large product orders. However, performance is significantly enhanced with more machines. Larger orders with fewer machines lead to increased idle times and longer production cycles, emphasizing the importance of balancing machine resources with product volume. The longer processing times for larger orders further highlight the need for seamless communication among system components to manage complexity.

Scalability: The tests validate the system's scalability, showing that it can effectively manage varying loads and machine counts. The efficiency gains observed in scenarios with more machines underscore the system's robustness and adaptability in diverse manufacturing environments. The improvement in scheduling efficiency with more machines confirms that the AASX, middleware, and scheduler are working together effectively to achieve optimized production schedules

Seamless Integration: The successful management of diverse scheduling scenarios and the observed efficiency improvements validate that the AASX, middleware, and scheduler are communicating seamlessly. This integration, which was the main goal of this thesis, ensures that the system functions cohesively and effectively across different conditions, meeting the anticipated objectives of the project.

Overall, the evaluation confirms that the production scheduling system performs effectively across different conditions, optimizing manufacturing workflows and demonstrating improved performance with increased resources. The results also affirm that the system's components are seamlessly integrated, validating the primary goal of this thesis.

6

CONCLUSION AND FUTURE WORK

This thesis explored the development and evaluation of a production scheduling system designed to dynamically allocate resources, optimize production schedules, and provide visual insights through Gantt charts. The system was built on a foundation of robust technologies, including MQTT, Node-RED middleware, and AAS-based digital representations. These components, along with the system's adherence to RAMI 4.0, facilitated seamless communication across different system layers and enabled real-time monitoring and control. By following the RAMI 4.0 model, the system ensured interoperability, scalability and alignment with Industry 4.0 standards.
While the system was not implemented in modern Industrial environments, it’s architecture and capabilities demonstrate potential for future applications In such settings.

The production scheduling system proved effective in several key areas, demonstrating its ability to dynamically allocate resources, optimize production schedules, and provide clear visualizations for monitoring and decision-making. The system managed to allocate resources dynamically, responding promptly to changing conditions in the production environment, a crucial capability for modern manufacturing where flexibility and adaptability are essential for maintaining a competitive edge. It also optimized production schedules by minimizing machine idle time, reducing production lead times, and enhancing overall throughput, all while adhering to constraints related to machine capabilities and job priorities.

The inclusion of a real-life simulation kit with four machines provided practical insights into the system’s performance in a controlled, yet realistic, production environment. This test demonstrated the system's ability to handle actual physical machines, proving that

its scheduling algorithms and communication protocols function effectively in real-world conditions. The system efficiently managed the coordination and execution of tasks, demonstrating its robustness and reliability when applied to a tangible production scenario. This real-world validation underscores the system's applicability beyond theoretical models, confirming its readiness for deployment in real manufacturing settings.

The integration of Gantt charts provided clear and intuitive visualizations of production schedules, allowing stakeholders to quickly assess the status of ongoing jobs, identify bottlenecks, and make informed decisions. This visualization capability also facilitated better communication and collaboration among different teams involved in the production process. Moreover, the system's scalability and robustness were ensured through the use of MQTT for communication and Node-RED as a middleware platform, allowing for easy integration with additional sensors, machines, and external systems, thus paving the way for future expansions and adaptations.

However, the evaluation of the system also revealed several areas for improvement. While the system performed well under normal operating conditions, scenarios involving highly constrained resources exposed limitations in the current resource allocation strategies, indicating the need for further refinement to handle situations where machine availability is severely restricted. The scheduling algorithms, although effective, could benefit from the integration of advanced optimization techniques such as machine learning and artificial intelligence. These methods would enhance the system's ability to predict and respond to dynamic changes in the production environment, further optimizing resource allocation and scheduling decisions.

Additionally, it is necessary to pointed out the need for a more user-friendly interface, particularly for non-technical stakeholders. Future developments could focus on improving the usability and accessibility of the system, ensuring that it caters to the needs of all users, regardless of their technical background. Overall, while the production scheduling system demonstrated significant strengths in optimizing and managing production workflows, there are clear opportunities for enhancement, particularly in resource allocation strategies, the incorporation of advanced optimization techniques, and the design of the user interface to create a more efficient, adaptable, and inclusive solution.

6.1 Future Work

Building on the current achievements, several avenues for future research and development have been identified to further enhance the system's capabilities. One promising direction involves integrating machine learning algorithms, which would enable the system to gain predictive capabilities, allowing it to anticipate and respond to potential disruptions before they occur. This would lead to greater efficiency and resilience in production schedules by making the system more adaptive to changing conditions in real-time.

Another opportunity for improvement lies in developing hybrid optimization approaches. By combining multiple optimization techniques, such as genetic algorithms, neural networks, and linear programming, the system could employ more sophisticated scheduling strategies capable of handling a wider range of production scenarios. This would enhance its ability to address complex scheduling problems and improve overall production efficiency.

Improving the system's interoperability is also crucial. Enhancing its integration with other digital platforms and industrial control systems (ICS) could enable a more holistic approach to production management, facilitating better communication and coordination across different layers of the production environment. This would help bridge gaps between various systems and improve overall operational efficiency.

Furthermore, there is significant potential in expanding data analytics and reporting tools. By incorporating more advanced data analytics capabilities, the system could provide deeper insights into production performance, identifying trends and patterns that inform continuous improvement efforts. This would allow for more strategic decision-making and ultimately contribute to more efficient and effective production processes.

These areas for future development aim to build on the system’s current strengths, addressing its limitations while unlocking new possibilities for optimization and adaptability in dynamic production environments.

6.2 Final Thoughts

The production scheduling system presented in this thesis represents a significant step toward more agile and efficient manufacturing processes. The evaluation through multiple test scenarios, including the real-life simulation using a physical production kit, demonstrated the system's effectiveness in optimizing scheduling tasks and managing resources efficiently.

The results validated that increasing the number of machines consistently improves scheduling efficiency, especially when dealing with larger product volumes. The tests also confirmed the system's scalability, adaptability, and robustness in diverse manufacturing environments.

The integration of modern digital technologies and intelligent algorithms has provided a flexible, scalable solution that addresses the challenges of dynamic production environments. The real-world testing reinforced the system's potential for practical deployment, providing confidence in its reliability and effectiveness in actual manufacturing scenarios. The insights gained from this research offer a foundation for future developments in the field, contributing to the broader goals of Industry 4.0 and the digital transformation of manufacturing.

The continued evolution of this system, guided by the recommendations outlined above, will further enhance its capabilities and impact. Incorporating advanced optimization techniques, expanding its predictive capabilities, improving interoperability, and enhancing usability are key steps toward creating a more efficient, sustainable, and competitive production process. The production scheduling system, therefore, represents not just a technological innovation but also a strategic tool for advancing modern manufacturing practices.

BIBLIOGRAPHY

Al Assadi, A., Waltersmann, L., Miehe, R., Fechter, M., & Sauer, A. (2021). *Automated Environmental Impact Assessment (EIA) via Asset Administration Shell* (pp. 45–52). https://doi.org/10.1007/978-3-662-62962-8_6

Alemao, D., Rocha, A. D., Nikghadam-Hojjati, S., & Barata, J. (2022). How to Design Scheduling Solutions for Smart Manufacturing Environments Using RAMI 4.0? *IEEE Access*, *10*, 71284–71298. https://doi.org/10.1109/ACCESS.2022.3187974

Ambarita, E. E., Karlsen, A., Scibilia, F., & Hasan, A. (2023). Industry 4.0 Digital Twins in Offshore Wind Farms. *Wind Energy Science Discussions*, *2023*, 1–34. https://doi.org/10.5194/wes-2023-108

Arm, J., Benesl, T., Marcon, P., Bradac, Z., Schröder, T., Belyaev, A., Werner, T., Braun, V., Kamensky, P., Zezulka, F., Diedrich, C., & Dohnal, P. (2021). Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0. *Sensors*, *21*(6), 2004. https://doi.org/10.3390/s21062004

Armenta A. (2021). *A History of Industrial Revolutions and How They've Impacted Manufacturing*.

Axelsson, J., Fröberg, J., & Eriksson, P. (2019). Architecting systems-of-systems and their constituents: A case study applying Industry 4.0 in the construction domain. *Systems Engineering*, *22*(6), 455–470. https://doi.org/10.1002/sys.21516

Azangoo, M., Taherkordi, A., Blech, J. O., & Vyatkin, V. (2021). Digital Twin-Assisted Controlling of AGVs in Flexible Manufacturing Environments. *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, 1–7. https://doi.org/10.1109/ISIE45552.2021.9576361

Benitez, G. B., Ferreira Lima, M. J. do R., Lerman, L. V., & Frank, A. G. (2019). Understanding Industry 4.0: Definitions and insights from a cognitive map analysis. *Brazilian Journal of*

Operations & Production Management, *16*(2), 192–200. https://doi.org/10.14488/BJOPM.2019.v16.n2.a3

Bierwirth, C., & Mattfeld, D. C. (1999). Production Scheduling and Rescheduling with Genetic Algorithms. *Evolutionary Computation*, *7*(1), 1–17. https://doi.org/10.1162/evco.1999.7.1.1

Bradac, Z., Marcon, P., Zezulka, F., Arm, J., & Benesl, T. (2019). Digital Twin and AAS in the Industry 4.0 Framework. *IOP Conference Series: Materials Science and Engineering*, *618*(1), 012001. https://doi.org/10.1088/1757-899X/618/1/012001

Bundesministerium für Wirtschaft und Energie (BMWi). (2016). *Structure of the Administration Shell*. https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/structure-of-the-administration-shell.pdf?__blob=publicationFile&v=5

Cavalieri, S., & Salafia, M. G. (2020). Asset Administration Shell for PLC Representation Based on IEC 61131–3. *IEEE Access*, *8*, 142606–142621. https://doi.org/10.1109/ACCESS.2020.3013890

CEN-CENELEC-ETSI Smart Grid Coordination Group. (2012). *Smart Grid Reference Architecture*.

Chen, S., Huang, Z., & Guo, H. (2022). An End-to-End Deep Learning Method for Dynamic Job Shop Scheduling Problem. *Machines*, *10*(7), 573. https://doi.org/10.3390/machines10070573

D', G., Antonio, N. A., Segonds, F., Laverne, F., Bedolla, J. S., & Chiabert, P. (2017). A framework for manufacturing execution system deployment in an advanced additive manufacturing process. *International Journal of Product Lifecycle Management*, *10*(1), 1. https://doi.org/10.1504/IJPLM.2017.082996

Dai, H.-N., Zheng, Z., & Zhang, Y. (2019). Blockchain for Internet of Things: A Survey. *IEEE Internet of Things Journal*, *6*(5), 8076–8094. https://doi.org/10.1109/JIOT.2019.2920987

Dave, S., Desai, D., Shah, S., Savalia, A., & Kiran, M. B. (2023, June 15). Smart Factories -An Important Component of Industry 4.0. *Proceedings of the International Conference on Industrial Engineering and Operations Management*. https://doi.org/10.46254/NA8.20230198

Deo, D., & Molnar, A. D. (2019). The application of simio scheduling in industry 4.0. *2019 Winter Simulation Conference (WSC)*, 3793–3801.

Deuter, A., & Imort, S. (2021). Product Lifecycle Management with the Asset Administration Shell. *Computers*, *10*(7), 84. https://doi.org/10.3390/computers10070084

Dhouib, S., Huang, Y., Smaoui, A., Bhanja, T., & Gezer, V. (2023). Papyrus4Manufacturing: A Model-Based Systems Engineering approach to AAS Digital Twins. *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8. https://doi.org/10.1109/ETFA54631.2023.10275523

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

Fiedor, P., & Ortyl, J. (2020). A New Approach to Micromachining: High-Precision and Innovative Additive Manufacturing Solutions Based on Photopolymerization Technology. *Materials*, *13*(13), 2951. https://doi.org/10.3390/ma13132951

Geismar, N. (2011). Single Machine Scheduling. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley. https://doi.org/10.1002/9780470400531.eorms0786

George, A. S. (2024). *The Fourth Industrial Revolution: A Primer on Industry 4.0 and its Transformative Impact. puirp. com*. https://doi.org/10.5281/zenodo.10671872

George, A. S., & George, A. s. (2020). INDUSTRIAL REVOLUTION 5.0: THE TRANSFORMATION OF THE MODERN MANUFACTURING PROCESS TO ENABLE MAN AND MACHINE TO WORK HAND IN HAND. *Seybold Report*, *15*, 214–234. https://doi.org/10.5281/zenodo.6548092

Gheewala D. (2022). *Reference Architecture Model Industry 4.0 (RAMI 4.0)*.

Giusto, D., Iera, A., Morabito, G., & Atzori, L. (2010). *The internet of things: 20th Tyrrhenian workshop on digital communications*. Springer Science & Business Media.

Hall, R., Schumacher, S., & Bildstein, A. (2022). Systematic Analysis of Industrie 4.0 Design Principles. *Procedia CIRP*, *107*, 440–445. https://doi.org/10.1016/j.procir.2022.05.005

Han, W., Feng, Z., & Cai, J. (2023). *Dynamic Scheduling Study of Flexible Job Shop Considering Machine Failure Rate* (pp. 42–47). https://doi.org/10.2991/978-2-38476-126-5_6

Hermann, M., Pentek, T., & Otto, B. (2015). *Design Principles for Industrie 4.0 Scenarios: A Literature Review*. https://doi.org/10.13140/RG.2.2.29269.22248

Hermann, M., Pentek, T., & Otto, B. (2016). Design Principles for Industrie 4.0 Scenarios. *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 3928–3937. https://doi.org/10.1109/HICSS.2016.488

Ivkic, I., Wolfauer, S., Oberhofer, T., & Tauber, M. G. (2017). On the cost of cyber security in smart business. *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 255–260. https://doi.org/10.23919/ICITST.2017.8356395

Kagermann, H., Lukas, W.-D., & Wahlster, W. (2011). Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. *VDI Nachrichten*, *13*(1), 2–3.

Kalsoom, T., Ramzan, N., Ahmed, S., & Ur-Rehman, M. (2020). Advances in Sensor Technologies in the Era of Smart Factory and Industry 4.0. *Sensors*, *20*(23), 6783. https://doi.org/10.3390/s20236783

Karimi, S., Ardalan, Z., Naderi, B., & Mohammadi, M. (2017). Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*, *41*, 667–682. https://doi.org/10.1016/j.apm.2016.09.022

Kim, B., Kim, S., Teijgeler, H., Lee, J., Lee, J. Y., Lim, D., Suh, H.-W., & Mun, D. (2022). Use of Asset Administration Shell Coupled with ISO 15926 to Facilitate the Exchange of Equipment Condition and Health Status Data of a Process Plant. *Processes*, *10*(10), 2155. https://doi.org/10.3390/pr10102155

Kumar, S., Suhaib, Mohd., & Asjad, M. (2020). Industry 4.0: complex, disruptive, but inevitable. *Management and Production Engineering Review*. https://doi.org/10.24425/mper.2020.132942

Kundakcı, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers & Industrial Engineering*, *96*, 31–51. https://doi.org/10.1016/j.cie.2016.03.011

Lang, D., Friesen, M., Ehrlich, M., Wisniewski, L., & Jasperneite, J. (2018). Pursuing the Vision of Industrie 4.0: Secure Plug-and-Produce by Means of the Asset Administration Shell and Blockchain Technology. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 1092–1097. https://doi.org/10.1109/INDIN.2018.8471939

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

Li, Y., Goga, K., Tadei, R., & Terzo, O. (2021). Production scheduling in Industry 4.0. *Complex, Intelligent and Software Intensive Systems: Proceedings of the 14th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2020)*, 355–364.

Lu, Y., Liu, C., Wang, K. I.-K., Huang, H., & Xu, X. (2020). Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics*

and Computer-Integrated Manufacturing, *61*, 101837. https://doi.org/10.1016/j.rcim.2019.101837

Lydon, B. (2019). RAMI 4.0 reference architectural model for Industrie 4.0. *InTech*, *66*(2).

Mattfeld, D. C. (1996). *Evolutionary Search and the Job Shop*. Physica-Verlag HD. https://doi.org/10.1007/978-3-662-11712-5

Morsy, E., & Pesch, E. (2015). Approximation algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, *18*(6), 645–653. https://doi.org/10.1007/s10951-015-0433-1

Mourtzis, D. (2022). Advances in Adaptive Scheduling in Industry 4.0. *Frontiers in Manufacturing Technology*, *2*. https://doi.org/10.3389/fmtec.2022.937889

Nie, L., Gao, L., Li, P., & Li, X. (2013). A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. *Journal of Intelligent Manufacturing*, *24*(4), 763–774. https://doi.org/10.1007/s10845-012-0626-9

Patnaik A. (2023a). Smart Manufacturing. *Industry 4.0: Synching the Future*.

Patnaik A. (2023b). Understanding RAMI 4.0. *Industry 4.0: Synching the Future*.

Pilevari, N., & Yavari, F. (2021). *Industry Revolutions Development from Industry 1.0 to Industry 5.0 in Manufacturing*. https://api.semanticscholar.org/CorpusID:234481332

Qi, Q., Tao, F., Zuo, Y., & Zhao, D. (2018). Digital Twin Service towards Smart Manufacturing. *Procedia CIRP*, *72*, 237–242. https://doi.org/10.1016/j.procir.2018.03.103

Quan Chong, Z., Yee Low, C., Mohammad, U., Abd Rahman, R., & Bahari Shaari, M. S. (2018). Conception of Logistics Management System for Smart Factory. *International Journal of Engineering & Technology*, *7*(4.27), 126. https://doi.org/10.14419/ijet.v7i4.27.22499

Rauh, L., Gärtner, S., Brandt, D., Oberle, M., Stock, D., & Bauernhansl, T. (2022). Towards AI Lifecycle Management in Manufacturing Using the Asset Administration Shell (AAS). *Procedia CIRP*, *107*, 576–581. https://doi.org/10.1016/j.procir.2022.05.028

Resman, M., Protner, J., Simic, M., & Herakovic, N. (2021). A Five-Step Approach to Planning Data-Driven Digital Twins for Discrete Manufacturing Systems. *Applied Sciences*, *11*(8), 3639. https://doi.org/10.3390/app11083639

Roda-Sanchez, L., Garrido-Hidalgo, C., Hortelano, D., Olivares, T., & Ruiz, M. C. (2018). OperaBLE: An IoT-Based Wearable to Improve Efficiency and Smart Worker Care Services in Industry 4.0. *Journal of Sensors*, *2018*, 1–12. https://doi.org/10.1155/2018/6272793

Saqlain, M., Piao, M., Shim, Y., & Lee, J. Y. (2019). Framework of an IoT-based Industrial Data Management for Smart Manufacturing. *Journal of Sensor and Actuator Networks*, *8*(2), 25. https://doi.org/10.3390/jsan8020025

Schuck T. (2022). *RAMI 4.0 - Axis 1 (Hierarchy Levels)*.

Scrimieri, D., Antzoulatos, N., Castro, E., & Ratchev, S. M. (2015). Automated Experience-Based Learning for Plug and Produce Assembly Systems. *IFAC-PapersOnLine*, *48*(3), 2083–2088. https://doi.org/10.1016/j.ifacol.2015.06.396

Seiger, R., Zerbato, F., Burattin, A., Garcia-Banuelos, L., & Weber, B. (2020). Towards IoT-driven Process Event Log Generation for Conformance Checking in Smart Factories. *2020 IEEE 24th International Enterprise Distributed Object Computing Workshop (EDOCW)*, 20–26. https://doi.org/10.1109/EDOCW49879.2020.00016

Sharma, P., & Jain, A. (2015). Performance analysis of dispatching rules in a stochastic dynamic job shop manufacturing system with sequence-dependent setup times: Simulation approach. *CIRP Journal of Manufacturing Science and Technology*, *10*, 110–119. https://doi.org/10.1016/j.cirpj.2015.03.003

Shen, W., & Norrie, D. H. (1999). Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems*, *1*(2), 129–156. https://doi.org/10.1007/BF03325096

Siatras, V., Bakopoulos, E., Mavrothalassitis, P., Nikolakis, N., & Alexopoulos, K. (2023). On the Use of Asset Administration Shell for Modeling and Deploying Production Scheduling Agents within a Multi-Agent System. *Applied Sciences*, *13*(17), 9540. https://doi.org/10.3390/app13179540

Smit, J., Kreutzer, S., Moeller, C., & Carlberg, M. (2016). *Industry 4.0. Study. European Parliament. Directorate General for Internal Policies Policy Department A: Economic and Scientific Policy. February 2016*.

Smith, J. S., & Sturrock, D. T. (2018). *Simio and simulation: modeling, analysis, applications*.

Tantik, E., & Anderl, R. (2017). Integrated Data Model and Structure for the Asset Administration Shell in Industrie 4.0. *Procedia CIRP*, *60*, 86–91. https://doi.org/10.1016/j.procir.2017.01.048

The Industrial Internet of Things Volume G1: Reference Architecture. (2019). https://api.semanticscholar.org/CorpusID:208051025

Veiga, J. T., Pessoa, M. A. O., Junqueira, F., Miyagi, P. E., & dos Santos Filho, D. J. (2021). A Systematic Modelling Procedure to Design Agent-Oriented Control to Coalition of

Capabilities—In the Context of I4.0 as Virtual Assets (AAS). *Computers*, *10*(12), 161. https://doi.org/10.3390/computers10120161

Wang, S., Wan, J., Zhang, D., Li, D., & Zhang, C. (2016). Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, *101*, 158–168. https://doi.org/10.1016/j.comnet.2015.12.017

Werner, F. (2011). GENETIC ALGORITHMS FOR SHOP SCHEDULING PROBLEMS: A SURVEY. *Preprint Series*, *11*, 1–66.

Wilkesmann, M., & Wilkesmann, U. (2018). Industry 4.0 – organizing routines or innovations? *VINE Journal of Information and Knowledge Management Systems*, *48*(2), 238–254. https://doi.org/10.1108/VJIKMS-04-2017-0019

Wu, G., Wang, C., Huang, X., & Lu, Q. (2022). Design of Smart Factory Based on Asset Administration Shell. *Digital Technologies Research and Applications*, *2*(1), 7. https://doi.org/10.54963/dtra.v2i1.75

Wu, H., He, J., Tömösközi, M., Xiang, Z., & Fitzek, F. H. P. (2021). In-Network Processing for Low-Latency Industrial Anomaly Detection in Softwarized Networks. *CoRR*, *abs/2112.03683*. https://arxiv.org/abs/2112.03683

Xiong, H., Fan, H., Jiang, G., & Li, G. (2017). A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *European Journal of Operational Research*, *257*(1), 13–24. https://doi.org/10.1016/j.ejor.2016.07.030

Yamada, T., & Nakano, R. (2000). *Job-Shop Scheduling*.

Yao, X., Zhou, J., Lin, Y., Li, Y., Yu, H., & Liu, Y. (2019). Smart manufacturing based on cyber-physical systems and beyond. *Journal of Intelligent Manufacturing*, *30*(8), 2805–2817. https://doi.org/10.1007/s10845-017-1384-5

Ye, X., Yu, M., Song, W. S., & Hong, S. H. (2021). An Asset Administration Shell Method for Data Exchange Between Manufacturing Software Applications. *IEEE Access*, *9*, 144171–144178. https://doi.org/10.1109/ACCESS.2021.3122175

Yin, Y., Chen, Y., Qin, K., & Wang, D. (2019). Two-agent scheduling on unrelated parallel machines with total completion time and weighted number of tardy jobs criteria. *Journal of Scheduling*, *22*(3), 315–333. https://doi.org/10.1007/s10951-018-0583-z

Ying, K.-C., Cheng, C.-Y., Lin, S.-W., & Hung, C.-Y. (2019). Comparative Analysis of Mixed Integer Programming Formulations for Single-Machine and Parallel-Machine Scheduling Problems. *IEEE Access*, *7*, 152998–153011. https://doi.org/10.1109/ACCESS.2019.2947685

Zaheer M. (2017a). *RAMI 4.0 (Part 1): Smart Electronic Industry 4.0 Architecture Layers*.

Zaheer M. (2017b). *RAMI 4.0 (Part 2): Smart Electronic Industry 4.0 Architecture Layers*.

Zhang, H., Buchmeister, B., Li, X., & Ojstersek, R. (2023). An Efficient Metaheuristic Algorithm for Job Shop Scheduling in a Dynamic Environment. *Mathematics*, *11*(10), 2336. https://doi.org/10.3390/math11102336

Zhong, R. Y., Xu, X., Klotz, E., & Newman, S. T. (2017). Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering*, *3*(5), 616–630. https://doi.org/10.1016/J.ENG.2017.05.015

ZVEI. (2016). *Implementation Strategy Industrie 4.0 Report on the results of the Industrie 4.0 Platform*. https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/Implementation_Strategy_Industrie_4.0_-_Report_on_the_results_of_Industrie_4.0_Platform/Implementation-Strategy-Industrie-40-ENG.pdf

DYNAMIC SCHEDULING FOLLOWING RAMI 4.0 REFERENCE AR

RAFAEL FERREIRA

⟨2024⟩