NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

Modelling Smart Manufacturing Assets Targeting Scheduling Optimisation

Duarte José Marques Alemão

Master in Electrical and Computer Engineering

DOCTORATE IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon

September, 2024

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

Modelling Smart Manufacturing Assets Targeting Scheduling Optimisation

Duarte José Marques Alemão

Master in Electrical and Computer Engineering

Adviser: José António Barata de Oliveira
Full Professor, NOVA School of Science and Technology - NOVA University Lisbon

Co-advisers: André Dionísio Bettencourt da Silva Parreira Rocha
Assistant Professor, NOVA School of Science and Technology - NOVA University Lisbon

Examination Committee:

Chair: Luís Manuel Camarinha de Matos,
Full Professor, NOVA School of Science and Technology - NOVA University Of Lisbon

Rapporteurs: Carlos Baptista Cardeira,
Associate Professor, Instituto Superior Técnico - Universidade de Lisboa

Paulo Jorge Pinto Leitão,
Principal Coordinator Professor, Escola Superior de Tecnologia e Gestão - Polytechnic Institute of Bragança

Adviser: José António Barata de Oliveira,
Full Professor, NOVA School of Science and Technology - NOVA University Of Lisbon

Members: Luís Manuel Camarinha de Matos,
Full Professor, NOVA School of Science and Technology - NOVA University Of Lisbon

Ricardo Luís Rosa Jardim Gonçalves,
Full Professor, NOVA School of Science and Technology - NOVA University Of Lisbon

DOCTORATE IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

Modelling Smart Manufacturing Assets Targeting Scheduling Optimisation

To Catarina, my family and all those who supported me
on this chapter of my life

ACKNOWLEDGMENTS

In this section I want to express my gratefulness to all those who supported me and contributed to the conclusion of this stage of my life.

First and foremost, I would like to thank my supervisor, Full Professor José Barata, for giving me the opportunity to keep working on this topic for which I became interested a few years ago, as well as the opportunity to participate in several national and international research projects along the way.

To my co-supervisor, Assistant Professor André Rocha. I cannot describe my gratitude for all these years. His support, dedication and patience were essential. It was a pleasure to learn so much from you, and I can only be very grateful. André had a great impact on the success of this work. I will never forget all the memories and discussions we went through for almost a decade.

I cannot forget Mafalda, who was involved in the beginning of this journey, always supported me when necessary, and who always provided fantastic discussions. I'm sure she will be proud of this milestone.

I would also like to thank my working colleagues, Fábio, Nelson, and Miguel, for so many good moments.

Furthermore, I want to thank my friends Afonso, Gonçalo e Tiago for being present for so long and always supporting me.

I also want to express my gratitude to all my family, mainly my grandparents and my uncles Celízia e Marianela.

To my amazing brother, who deserves all the best. I cannot describe my gratitude for always supporting and helping me. Thank you.

To my mom. For everything what she means to me. It was not possible without her support and dedication through all these years.

A huge thank you to Catarina. For keeping present, in my best and worst moments. For always encourage and try to motivate me. For all the support. For everything you mean to me. Thank you.

Finally, to my dad who always supported, encouraged and inspired me. Thanks for contributing for what I am today. I will always be grateful to you.

To all of you,
Thank You

"There is nothing so useless as doing efficiently that which should not be done at all."

- Peter Drucker

Abstract

The industry sector has evolved faster and faster over the past few decades, driven by increasingly complex market demands. Customers now hold greater decision power, and factories are pressured not only to deliver products fast but also to optimize production processes, reducing costs, inefficiencies, and delays. Companies must ensure they can meet customer expectations without compromising operational efficiency.

Thus, modern manufacturing systems must be robust and agile, capable of reacting smoothly to external events, and adaptable to unexpected changes. In this world that is becoming more and more connected, the rise of smart factories, characterized by interconnected and autonomous entities, is transforming how production systems operate. These entities are becoming able to adapt to real-time events but also share critical data between them to optimize workflow, minimize downtime, and ensure continuous production.

To maintain production efficiency, meet KPIs like makespan, reduce downtimes, and improve energy efficiency, or be more prepared for unexpected disturbances in the system, it is important that companies are equipped with robust and adaptable manufacturing scheduling systems. While numerous solutions have been proposed over the years to implement scheduling systems, in many cases, those approaches focus on specific cases and do not fulfill the necessary requirements to be applied in industry.

This research addresses these limitations by providing a more generic, comprehensive, and adaptable approach for smart manufacturing environments. The design and implementation of scheduling solutions in smart manufacturing systems is not standardized and there is not a reference model to develop scheduling solutions that reflect real industrial environments, leading to a gap between reference architectures and scheduling systems. Therefore, the proposed research intends to study the main challenges related to manufacturing scheduling and

to model manufacturing components targeting the scheduling optimization based on one of the most prosperous reference architectures, RAMI 4.0.

Through an extensive literature review, both functional and non-functional requirements were identified and, after analyzing them, the design principles to develop a manufacturing scheduling system were established. Additionally, a methodology was proposed to serve as the foundation for designing scheduling solutions aligned with RAMI4.0, including the identification of the main assets and the development of their corresponding Asset Administration Shells, while addressing key design principles such as data uniformity, KPI harmonization, and automatic rescheduling. Finally, the proposed approach was applied to various use cases, including the KITT4SME and PERFoRM projects, to demonstrate its efficiency and adaptability.

This work aims to fill a critical gap in existing literature but also offers a practical roadmap for industry professionals aiming to fully integrate production scheduling into RAMI4.0, paving the way for smarter, more responsive manufacturing systems in the era of Industry 4.0.

Keywords: Manufacturing Scheduling, Scheduling Methodology, Industry 4.0, Cyber-Physical Production Systems, RAMI4.0, Assets Modelling

Resumo

O setor industrial tem evoluído a um ritmo cada vez mais acelerado nas últimas décadas, impulsionado por exigências de mercado cada vez mais complexas. Os clientes têm agora maior poder de decisão, e as fábricas estão sob pressão para não apenas entregar produtos rapidamente, mas também otimizar os processos de produção, reduzindo custos, ineficiências e atrasos. As empresas precisam de garantir que conseguem satisfazer as ordens dos clientes sem comprometer a eficiência operacional.

Assim, os sistemas de produção modernos precisam ser robustos e ágeis, capazes de reagir eficazmente a eventos externos e de se adaptarem a mudanças inesperadas. Num mundo cada vez mais conectado, o aparecimento de fábricas inteligentes, caracterizadas por entidades interconectadas e autónomas, está a transformar o funcionamento dos sistemas de produção. Estas entidades estão a tornar-se capazes de se adaptar a eventos em tempo real e também de partilhar dados críticos entre si para otimizar o fluxo de trabalho, minimizar tempos de inatividade e garantir uma produção contínua.

Para manter a eficiência da produção, cumprir indicadores de desempenho como o *makespan*, reduzir tempos de inatividade, melhorar a eficiência energética e estar mais bem preparado para perturbações inesperadas no sistema, é fundamental que as empresas possuam sistemas de escalonamento de produção robustos e adaptáveis. Embora ao longo dos anos tenham sido propostas várias soluções para implementar sistemas de escalonamento, muitas delas focam-se em casos específicos e não cumprem os requisitos necessários para serem aplicadas na indústria.

Este estudo aborda essas limitações, propondo uma abordagem mais genérica, abrangente e adaptável para ambientes de manufatura inteligente. O *design* e implementação de soluções de escalonamento em sistemas de manufatura inteligente não são padronizados e não existe um modelo de referência que reflita os ambientes industriais reais, o que cria uma

lacuna entre as arquiteturas de referência e os sistemas de escalonamento. Assim, o objetivo deste estudo é analisar os principais desafios relacionados com o escalonamento de produção e modelar os componentes de manufatura, visando a otimização do escalonamento com base numa das arquiteturas de referência mais promissoras, a RAMI 4.0.

Através de uma extensa revisão da literatura, foram identificados os requisitos funcionais e não funcionais e, depois de os analisar, foram estabelecidos os princípios de *design* para o desenvolvimento de um sistema de escalonamento de produção. Além disso, foi proposta uma metodologia que serve como base para a conceção de soluções de escalonamento alinhadas à RAMI 4.0, incluindo a identificação dos principais ativos e o desenvolvimento das suas correspondentes *Asset Administration Shells*, abordando princípios de design fundamentais como a uniformidade de dados, harmonização de *KPIs* e reescalonamento automático. Por fim, a abordagem proposta foi aplicada a vários casos de estudo, incluindo os projetos KITT4SME e PERFoRM, para demonstrar a sua eficiência e adaptabilidade.

Este trabalho tem por objetivo preencher uma lacuna crítica na literatura existente, enquanto oferece um guia prático para profissionais da indústria que desejam integrar totalmente o escalonamento de produção na RAMI 4.0, abrindo caminho para sistemas de manufatura mais inteligentes e responsivos na era da Indústria 4.0.

Palavras-chave: Escalonamento de Produção, Metodologia de Escalonamento, Indústria 4.0, Sistemas de Produção Ciber-Físicos, RAMI 4.0, Modelação de Ativos

CONTENTS

LIST OF FIGURES

List of Tables

ACRONYMS

AAS	Asset Administration Shell
ADACOR	Adaptive Holonic Control Architecture for Distributed Manufacturing Systems
AI	Artificial Intelligence
CIMOSA	Computer Integrated Manufacturing Open System Architecture
CPPS	Cyber-Physical Production Systems
CS	Computer Science
DM	Data Mining
DT	Digital Twin
ERP	Enterprise Resource Planning
I4.0	Industry 4.0
I5.0	Industry 5.0
ICT	Information and Communication Technologies
IDTA	Industrial Digital Twin Association
IIRA	Industrial Internet Reference Architecture
IoT	Internet of Things
JSSP	Job-Shop Scheduling Problem
MES	Manufacturing Execution System
ML	Machine Learning

RAMI4.0	Reference Architecture Model Industrie 4.0
SGAM	Smart Grid Architecture Model
SM	Smart Manufacturing
SME	Small and Medium Enterprises

1

INTRODUCTION

In this section is presented an introduction of the proposed research topic, as well as an overview of the whole document. Initially, a summary of the problem definition is described, which presents the challenges and needs that will be addressed in this thesis. Then, the objectives and motivations of this thesis are defined and explained, to expose the proposed research intentions in order to solve the presented problem.

1.1 Contextualization

Manufacturing has suffered deep changes during the past decades, mainly driven by market trends that forced companies to move from traditional mass production lines to more dynamic and flexible manufacturing systems. The increasing demand for highly customized products with several variants led to smaller lot sizes, which requires companies to quickly adapt and optimize production, costs, and energy usage to adjust to new market opportunities in order to thrive in a very competitive world. Therefore, it is crucial that manufacturers, especially Small and Medium Enterprises (SME), develop approaches that have more dynamism, flexibility, reconfigurability, and efficiency at the factory level. Manufacturing scheduling is critical as it enables to optimize the allocation of resources, reduce production costs, minimize energy usage, and adapt quickly to shifting market demands. Current solutions often fall short because they rely on rigid, centralized, and static methods that cannot handle the complexity and variability of modern manufacturing environments.

The product's life cycle is getting smaller and smaller, which leads companies not to adapt timely the production lines to new market opportunities, which takes time and high costs. In recent years, new production paradigms have been proposed in order to support, mainly, SMEs tackling this problem as well as improving production efficiency while pursuing

sustainable practices. These paradigms, such as lean production, agile manufacturing, Smart Manufacturing (SM), and now Industry 5.0 (I5.0), emphasize not only the integration of advanced technologies but also the importance of sustainability, addressing the problems of energy efficiency or human labor conditions (Ding, B., Ferràs Hernández, X., & Agell Jané, 2021; Sakurada et al., 2022). I5.0, in particular, focuses on human-centric, resilient, and sustainable manufacturing, further complicating the scheduling landscape by introducing the need to consider the circular economy, which includes processes like disassembly and remanufacturing (Voulgaridis et al., 2022).

These evolving needs in manufacturing have led to significant academic efforts aimed at addressing the complex challenges associated with scheduling. However, despite these advancements, a substantial gap still exists between academia and industry (Alemao et al., 2022). This gap highlights the need for stronger collaboration to achieve consistent and acceptable results. For instance, both parties should engage in a closer interaction, which will provide more prosperous advances. Academia needs to be made aware of specific manufacturing requirements, such as relevant production technical features, business environment, clients' preferences, or societal aspects. On the other hand, companies lacking a solid research department may not be aware of new technologies and processes being developed, which need to be strongly matured, mainly by the contribution of industrial partners.

The recent advances in Information and Communication Technologies (ICT) have enabled the development of new approaches, which hugely rely on Computer Science (CS). Concepts such as Artificial Intelligence (AI), Machine Learning (ML), Data Mining (DM) have been increasingly used and adopted on top of stable manufacturing technologies, and new concepts such as Cyber-Physical Production Systems (CPPS) emerged (Freitas et al., 2023; J. Zhang et al., 2019). These technologies are crucial for the development of manufacturing systems that are not only flexible and dynamic but also more cost-effective and efficient in several dimensions, such as production performance or sustainability.

One of the biggest challenges of humankind was always to maximize its productive work in an efficient and effective way. To do so, it is of huge importance to plan a well-structured schedule with detailed description of the tasks to execute, where they should be executed and when the task should be performed. This is applicable to areas like transportation services (Carosi et al., 2019), staff distribution (Bandi & Gupta, 2020), energy efficiency on smartphones (J. Wang et al., 2016) and, unquestionably, production systems (Alemão et al., 2021).

Now, more than ever, there is an opportunity to implement valid and efficient schedule solutions, not only on the shop-floor but even along the value chain, since more information

is available than ever. However, there is a big challenge in implementing scheduling solutions in real manufacturing systems, especially in the context of I5.0 and circular economy. Despite the required complexity of such implementation in the real world, there needs to be a reference guide in the context of SM to assist in implementing these solutions within smart and sustainable manufacturing environments.

Thus, this work proposes an approach to facilitate the development of efficient and sustainable scheduling solutions in SM environments. It will be suggested a methodology to support the design and implementation of manufacturing scheduling systems, so end users can start from a predefined basis instead of starting from zero when they want to develop scheduling systems, since some development stages may be common to different scenarios.

1.2 Motivation

In the last decades, plenty of research studies have been conducted to deal with the scheduling problem in manufacturing. While production systems have advanced significantly, the pace of market evolution, driven by increasing demand for highly customized products, shorter product life cycles, and more sustainable-related requirements, has surpassed the adaptability of many existing systems. Despite these advancements, the majority of scheduling solutions are not industry-oriented as they still fail to cope with so many industrial requirements and constraints, and, thus, they need to be more attractive for companies (Alemão et al., 2021).

So, it is important to understand that, although there has been a lot of research in this area, apparently most of the proposed scheduling solutions have had difficulty in being adopted by the industry. The fact that they are not being applied to real manufacturing systems may be related to the fact that the solutions do not transpose the conditions, requirements, and constraints of industrial environments. Consequently, scheduling solutions, that are not being directly applied to the industry, end up being too simple and completely far from reflecting what is happening in the factories and across the value chain.

However, it is possible to find in the literature some solutions applied to real scenarios, as well as interesting advances in scheduling approaches to cope with SM scheduling (K. Gao et al., 2019; Nikolakis et al., 2018; Zeng et al., 2018). There are even reference architectures to design and develop manufacturing systems, that deal with vertical integration, which involves integrating components across different stages of production within the factories, and horizontal integration, that combines companies or systems at the same level of the supply chain, such as RAMI4.0, or Industrial Internet Reference Architecture, detailed in section 2.2. But on

the other hand, there is not a consensus on what the main needs and requirements need to be considered when developing scheduling approaches in order to design solutions that can be applied to the industry.

As the demand for more diversified and highly customized products is growing a lot, the factories need to transform their production systems as the traditional ones are not capable of meeting these complex requirements. In medium and large manufacturing facilities, it is difficult to schedule a large number of different parts and products. This is an even a more complicated problem when there are disturbances in the production line when it is not possible to know exactly how much time a product will take to get finished by some machine when new products arrive at the factory during the production process or if a machine unexpectedly breaks down and it becomes necessary to wait or re-route product to another machine. Additionally, manufacturers must balance these scheduling difficulties with the need to optimize other critical objectives, including maximizing efficiency, minimizing costs, and ensuring sustainable practices, which are in so much demand nowadays. These competing demands make it even more difficult to maintain smooth and effective production operations.

This research aims to review existing manufacturing scheduling solutions and identify the main challenges and requirements for solving scheduling problems in real scenarios. By doing so, it seeks to develop a reference methodology to guide the creation of scheduling solutions applicable to SM systems. The goal is to bridge the gap between academic research and industrial practice, providing valuable insights and solutions for developing and implementing effective production scheduling systems.

1.3 Research Questions

Therefore, it is imperative to study and evaluate how to harmonize the main necessities of scheduling problems in SM environments. It is important to improve the process of designing and developing manufacturing scheduling solutions as much as possible. This leads to the following research questions:

Q1 – Which approach can be considered to develop systems capable of delivering dynamic scheduling solutions for SM, with the objective of optimizing these solutions for different Key Performance Indicators (KPIs)?

The preceding research question, Q1, can be addressed according to the following hypothesis:

H1 – If a methodology for the development of production scheduling uses, as a reference, an architecture aimed at Industry 4.0 (I4.0), such as Reference Architecture Model Industrie 4.0 (RAMI4.0), it will be possible to provide dynamic scheduling solutions for SM, with the objective of optimizing these solutions for different KPIs. Such KPIs may include traditional ones, as makespan, throughput, or idle time, but also more recent ones that want to reflect new challenges and opportunities in industry, such as energy efficiency, maintenance optimization, or dynamic rescheduling

However, H1 leads to another possible research question, related to how the assets can be modeled in SM environments:

Q2 – Which concepts and characteristics of RAMI 4.0 can be used to model the different functionalities and information present in a SM system for the development of dynamic scheduling solutions?

The preceding research question, Q2, can be addressed according to the following hypothesis:

H2 – By using the Asset Administration Shell (AAS) concept from RAMI4.0, which is used to describe an asset in a standardized way and exchange data between industrial assets, it is possible to model each of the functionalities and information from different sources, such as machines, human workers or software tools, to provide scheduling solutions capable to cope with the challenges arising from the smart industrial systems.

The use of a reference architecture, such as RAMI 4.0, which has increasingly been adopted, mainly at European level by big industrial players, will thus facilitate the introduction of these solutions, since it is a reference architecture that is already known and explored, and I4.0 will rely heavily on standards for the development of interoperability. Furthermore, it allows easy integration and interoperability with other industrial components and functionalities, which is one of the main features of SM.

Despite being explored for a long time in several areas, reference architectures have not been explored for the development of dynamic production scheduling solutions.

1.4 Approach and Contributions

In real world manufacturing systems, new parts arrive unexpectedly, precise processing times may not be known in advance and may change during the process, availability and interactions

between human workers and machines need to be considered to execute the production plans. There have been interesting advances related to scheduling solutions and algorithms; however, dealing with so much information highly increases the complexity of the problems, and the provided academic solutions often opt to simplify the problems. How to manage all this information that is sometimes skipped?

To help to close the gap between academia and industry, this work will do an in-depth study of the assets related to the scheduling process. Consequently, it aims to describe and model the assets in SM environments aligned with RAMI4.0 in order to optimize and standardize the scheduling process. RAMI4.0 is a reference architecture that provides a structured, three-dimensional layered approach to implementing I4.0 technologies. It integrates several aspects of industrial processes, combining the lifecycle of products and systems with hierarchical levels and architectural layers. This model helps standardize and guide the digital transformation of manufacturing and production systems. In turn, by clarifying which components should interact in the scheduling process and which data need to be available and flow within the system, scheduling developers may be better prepared, and companies may benefit from the developed scheduling solutions.

The main goal is to provide a structured module that includes relevant information about both the inputs of the scheduling process and the desired outputs. This module, the scheduling module, will serve as a guideline to help in the designing and development of scheduling systems by modeling stations and products that support the scheduling process. The scheduling and the modelling module will be the most relevant in this work. The modelling module will be where any other modules will be modelled, and the scheduling one since it will be the most detailed module in this research contribution. Several actors are present when dealing with scheduling. Figure 1.1 exemplifies the overall scheduling system and some of its possible modules. There is real-time information from the shop-floor related to machines, products, human workers, transportation, and more. New orders arriving and new data being analyzed can affect future production. Simulations being executed on time while the system is working can affect the remaining scheduling. Algorithms are used to build schedules for the next hours, days, or weeks.

It is relevant at this stage to make a distinction between module and entity, which will be often used throughout the document. The term module is used as referring to a self-contained unit within a larger system. On the other hand, entity refers to any distinct element or actor within the manufacturing ecosystem, physical or virtual, and includes everything from individual machines to production lines, software applications, or human operators. Modules often

consist of or interact with multiple entities. Entities can be part of one or more modules, depending on their role within the manufacturing process.

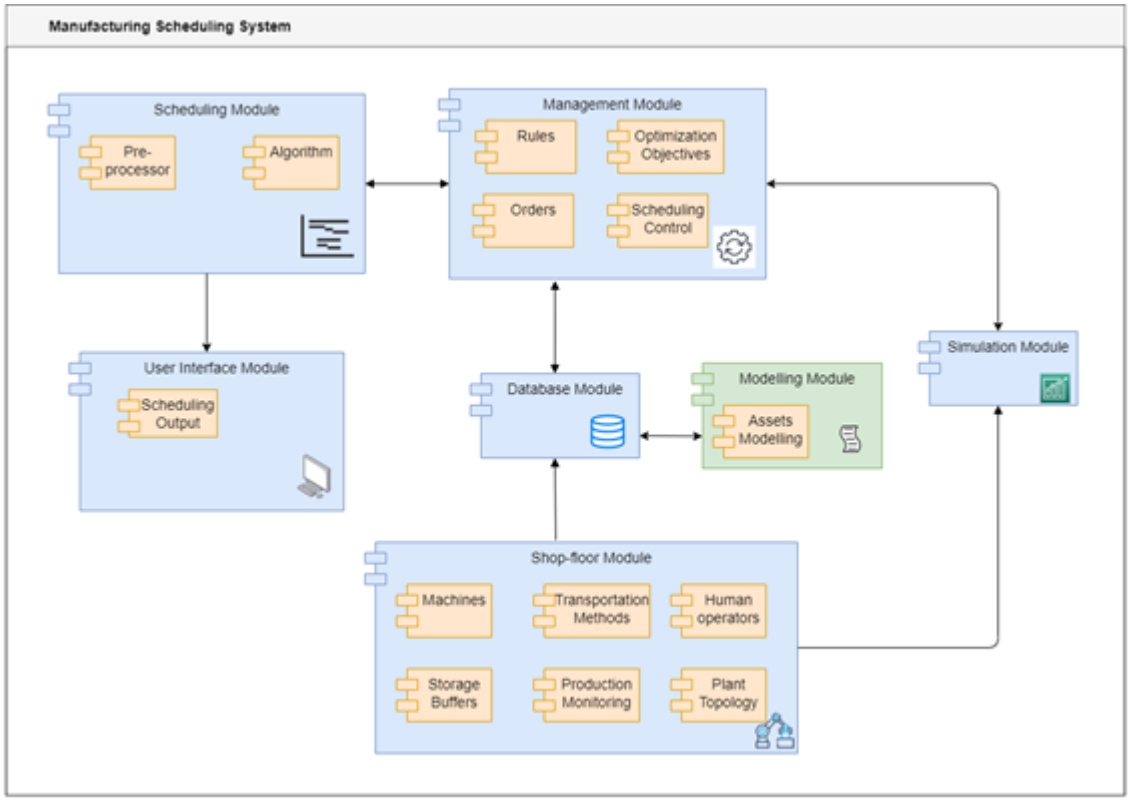

Figure 1.1 – Manufacturing scheduling system

Consequently, all these processes need to be connected in order to minimize major disturbances in the system. Thus, it is necessary to establish a methodology for development of scheduling solutions towards the SM, based on one of the most used reference architectures for I4.0, RAMI 4.0.

1.5 Research Methodology

This study was conducted by following the traditional research methodology, which encompasses seven different steps, as described in Figure 1.2. In a first stage the focus was on stages 1, 2 and 3, since an extended review of the literature was done, and the research question and respective hypothesis were formulated.

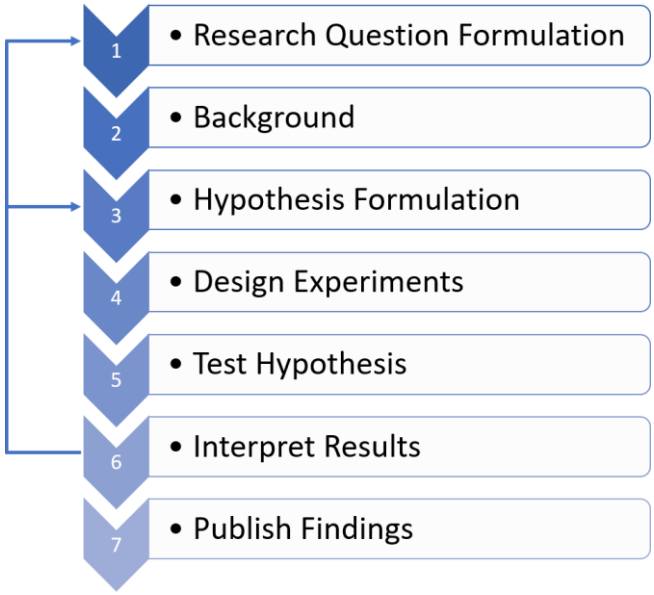

Figure 1.2 – Research Methodology (adapted from the handouts of the Scientific Research Methodologies and Techniques class by professor Camarinha-Matos)

It starts with the formulation of research questions, which was presented in the previous chapter.

After that, it was defined the background, where all the information relevant to the research and the state of the art is defined, and it is presented a thoroughly literature review. For the search for the literature three relevant indexed databases in manufacturing were used: Web of Science, Scopus and IEEE Xplore. It is mainly focused on documents published since 2013 to ensure that the information is up to date. Different terms were used individually or combined with others in order to come out with the current literature review. The most relevant terms are composed by "scheduling", "manufacturing scheduling", "production scheduling", "scheduling systems", "job shop scheduling", "dynamic scheduling", "industry 4.0", "industrie 4.0", "smart manufacturing", "cyber-physical systems", "internet of things", "cloud computing" "manufacturing architectures", "RAMI4.0". After finishing the literature analysis, a set of hypotheses is proposed. Normally, the number of hypotheses is the same as the number of research questions.

Later, the development stage was initiated with the definition of the experiments and the designing of the solution, which was then implemented. Afterwards the implementation tested and validated, and verified according to the hypothesis defined before. If the results are not the expected the process can return to the previous stages. Finally, the results were published in different scientific journals in order to increase the impact of the proposed work.

1.6 Integration with other research activities

On the one hand this thesis work is building upon the results of previous projects in the SM and CPPS area, such as H2020 PERFoRM (Leitão et al., 2016), that aimed at empowering legacy systems with intelligent capabilities associated with the I4.0 paradigm, and H2020 GO0DMAN (Leitao et al., 2022), which involved the development of an agent-based CPPS to perform quality control on multistage production systems based on Zero Defect Manufacturing paradigm. On the other hand, this work is aligned with the current (meanwhile finalized) projects of H2020 AVANGARD (Perlo et al., 2022), P2020 AdAM (Rocha et al., 2021), H2020 KITT4SME, and P2020 CESME (Alemão et al., 2020). AVANGARD addresses the integration of three novel processing units into an existing micro factory test bed conceived to produce urban electric vehicles, where UNINOVA is responsible for the design of a Cloud-based CPPS architecture and the implementation of a network of trustability using Blockchain technology and integrate it with the developed platform. The AdAM project aimed to demonstrate how various devices and subsystems can be integrated into an automated production system to support I4.0, particularly in quality control and energy consumption monitoring, making production systems more flexible with shorter development and integration times. KITT4SME Specifically targets European SMEs and mid-caps to provide them with scope-tailored and industry-ready hardware, software and organisational kits, delivered as modularly customisable digital platform, that seamlessly introduce artificial intelligence in their production systems. CESME aims to create a methodology to develop evolutionary complex systems, namely systems of CPPS.

1.7 Document Organization

This document is composed of seven different chapters. The first one is composed of an introduction to the work developed in this thesis, including contextualization and motivation, Identification of research questions and respective hypotheses, the contributions of the work, and the research methodology. Chapter two presents the revision of literature performed during this work, with a focus on the 4th industrial revolution, CPPS, reference architectures, and manufacturing scheduling. Chapter 3 identifies the functional requirements and proposes a set of design principles that serve as the foundation for the scheduling methodology. In Chapter 4 is presented the scheduling methodology, composed of the identification of the main assets to model when designing manufacturing scheduling systems, a data model, and how the design principles are aligned with the proposed methodology. The fifth chapter depicts how the

scheduling methodology can be adapted to an already finished use case. In Chapter 6, the methodology is applied to two different scenarios, one developed on a NOVA University of Lisbon laboratory kit, and another one to an industrial scenario. The document finishes with a conclusion of the developed work and identification of future steps to keep evolving this work.

2

LITERATURE REVIEW

In order to find the answers for the proposed research questions, a literature review was conducted towards creating a stable basis for modelling assets to achieve scheduling optimization. Therefore, this section presents literature on four key subjects. Initially, it is presented the current state of the art of the fourth industrial revolution, also known as I4.0 or SM. Then, reference architectures that are applied in I4.0 are presented. After that, is presented the notion of CPPS, which are of huge importance in this new industrial age. Finally, is presented a review of manufacturing scheduling, regarding its main developments, types, implementations and gaps.

2.1 4th Industrial Revolution

The recent developments and advances in technology as well as the market demand for highly customized and personalized products have been pushing manufacturing companies to develop new solutions to become more dynamic and flexible to face these emergent trends and the quickly changing markets.

Most of the existing production systems, are based on automated systems built to achieve high performance and high delivery rates, coming from the second and third industrial revolutions, but have no capability regarding autonomy, adaptation, and flexibility. Consequently, a group of expert technicians is needed to solve a problem each time a disturbance occurs on the production line, such as machine breakdowns or changes in production orders. Besides these restrictions, the emergence of new manufacturing paradigms, the appearance of new technologies and processes, the cheaper development of IT infrastructures, and the emerging possibility of digitization, among other factors led to a disruption in the industrial scene.

In Figure 2.1 are represented the four stages of the industrialization process. The evolution of the manufacturing industry started at the end of the 18th century when mechanical equipment began to be used in the production of goods. Then, around the beginning of the

20th century, a new industrial revolution started, led by the use of electrical power in mass production systems. After that, the third industrial revolution emerged at the beginning of the 1970s decade, and it was marked by the introduction of electronics and Information Technologies (IT) in order to achieve higher automation levels during the production processes, leading to a reduction of manual work. Afterward, at the beginning of the new century, a new revolution began.

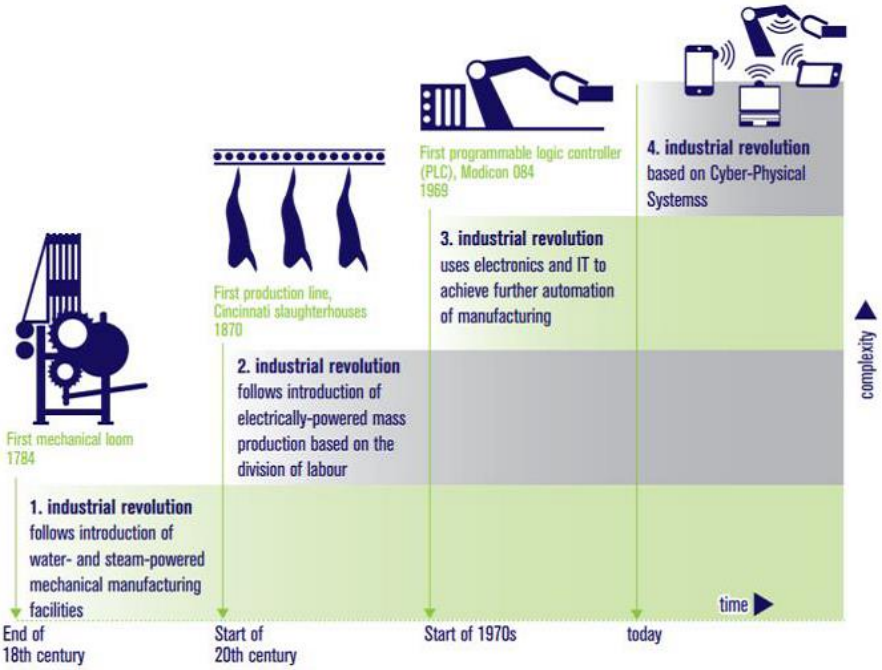

Figure 2.1 - The four phases of the industrial revolution, (Kagermann et al., 2013)

The 4th Industrial Revolution, under the nomination of I4.0 in Europe, Industrial Internet or SM in the USA, Smart Factory in South Korea and Made in China 2025 in China (Kagermann et al., 2013; Q. Li et al., 2018; Wuest et al., 2016), is happening now, but has started with the first steps several years ago. It makes use of different emerging technologies and paradigms such as AI, Cyber-Physical Systems (CPS), Internet of Things (IoT), Cloud Computing, Digital Twin (DT), among others, and is allowing to develop more dynamic and agile approaches to improve the efficiency of manufacturing systems (J. Zhang et al., 2019).

Figure 2.2 shows the evolution of data in manufacturing. It is possible to observe (in the bottom of the figure) that for many years manufacturing and information followed different paths. First, before the First Industrial Revolution, while the low complexity work in manufacturing was predominantly done by hand, the data generated in the process was limited and the information was transmitted verbally to the next generation or recorded on paper if available, nevertheless the data could be easily lost. With the appearance of the first steam machine and, consequently, the machine age, machines began to be used in the production processes, leading to a fast grow in manufacturing. Since there was a tight relation between workers and

machines, the data related to each one started to be collected, stored and analyzed manually. There were no significant changes compared to the handicraft age regarding how the data was handled. However, the emergence of computer and information systems led to the information or digital era, where IT was highly applied to production processes and the available manufacturing data grew exponentially. Data started to be stored in databases and processed by information systems, however systems were not able to communicate with other systems and there was not an efficient way to analyze unstructured data, which was keeping it difficult for companies to benefit from the huge value of data, particularly for SMEs. Nevertheless, the evolution of IT has continued and has been pushing manufacturing towards digitization. Lately, due to emergence of new paradigms such as CPS, IoT, Cloud Computing, Systems of Systems, Agents, Service-oriented approaches, AI and Big Data, manufacturing and information systems are more and more fused, since the huge amount of data from different sources being collected and processed will directly impact the production processes. The capability to collect, store and analyze data was hugely improved, which makes possible for manufacturers to better understand their equipment, their products, their manufacturing processes, their customers, their workers, and even their competitors, which increases the smartness degree in manufacturing systems, and leads to a better interconnection between the different systems.

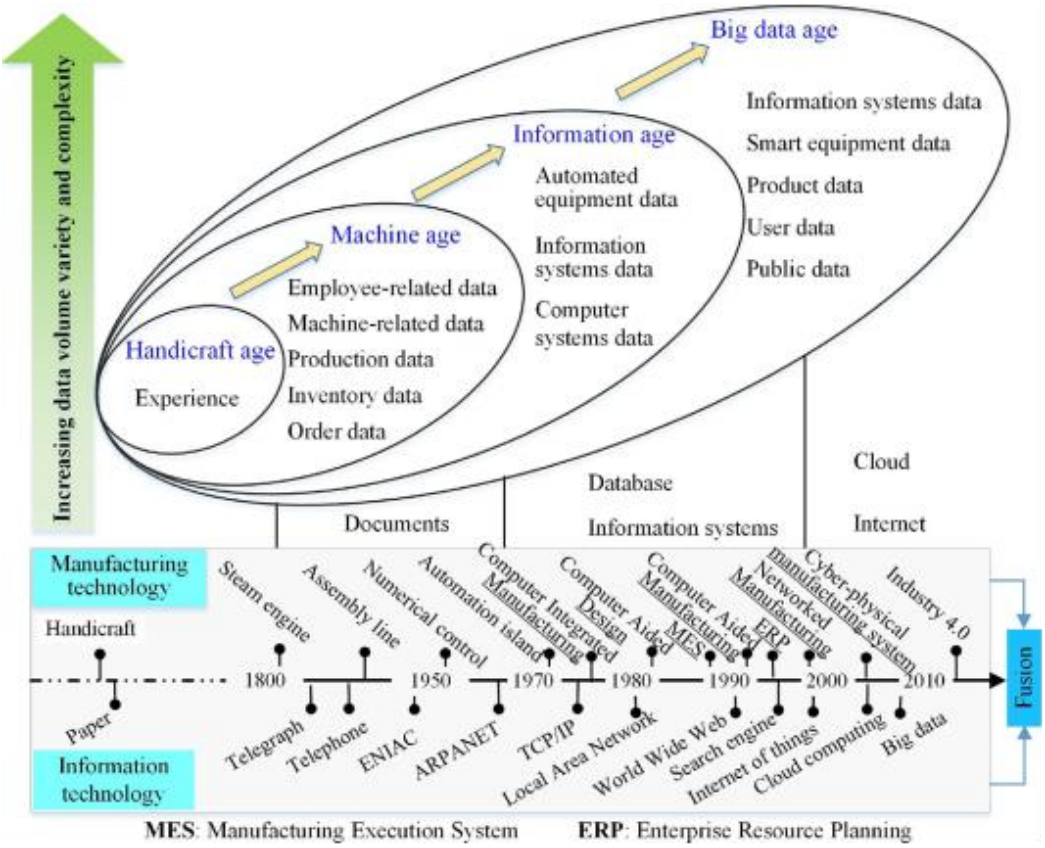

Figure 2.2 - Evolution of data in manufacturing, (Tao, Qi, et al., 2018)

I4.0 can be described as a collaborative system which reacts in real time to reach the demand of the factory and the customers (Kagermann et al., 2013; Kusiak, 2018). It makes use of concepts such as CPS, IoT, Cloud Computing, AI and data analysis merged with sensors, communication protocols, control and predictive engineering to build the manufacturing systems of today and tomorrow (Kagermann et al., 2013; Tao, Qi, et al., 2018). The concept of I4.0 covers three main characteristics (Stock & Seliger, 2016):

- **"Horizontal integration across the entire value creation network"** – the integration of the different players of the value chain during all the product life cycle allows to optimize the entire production process. If the suppliers, manufacturers, materials and logistics are connected, the entire ecosystem can be more easily coordinated and adapted. The digitization of data and processes allow to share, analyze and dynamically adapt the system in real-time during the manufacturing process.

  In this new manufacturing paradigm, the horizontal integration plays a very important goal, where all the value chain's parties need to be considered during the entire production process (Figure 2.3).

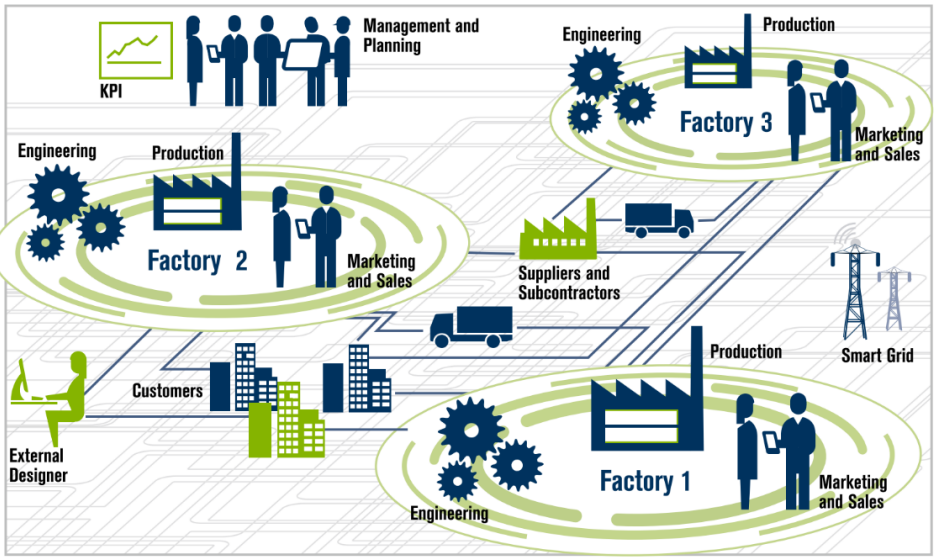

Figure 2.3 - Horizontal integration in Industry 4.0 context (Kagermann et al., 2013)

The manufacturing process is not only limited to the production phase and to what is happening within the factory environment. The giant grow of digitization is allowing to have intercommunications with manufacturing contributors from different stages, both within and outside the factory. Besides, these interactions are not only regarding the exchange of products and orders requests, but also about the change of data and valuable information for the different parts. This allows to achieve better and more efficient results, since more information is available to be collected, processed and analyzed, and the system will be able to adapt quickly to different circumstances. Then, not only the productivity inside the factory will be improved, but also the entire ecosystem,

including deliveries of new parts to the factory and delivery of finished products to the customers.

- **"End-to-end engineering across the entire product life cycle"** – the linking and digitization between the different phases of the product life-cycle, namely the acquisition of raw materials, product manufacturing, product use and the product end of life allow to collect, store and process data to acquire new knowledge and improve the entire production process, from design to recycle stage. In Figure 2.4 is representing the entire product lifecycle, since its conception going through all the phases until reaches its end of use or recycle phase.

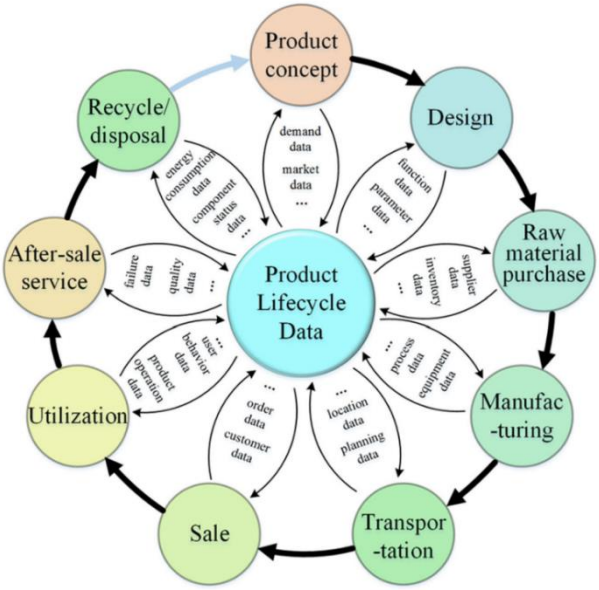

Figure 2.4 - Product life cycle, (Tao, Cheng, et al., 2018)

- **"Vertical integration and networked manufacturing systems"** – by combining different hierarchical levels in the production process, from working stations and human workers in the factories to technological innovations, such as Manufacturing Execution Systems (MES), and marketing activities through CPS, the optimization and integration with the above-mentioned characteristics can be achieved. The vertical integration brings up the concept of smart factory, which aims to interconnect all the resources and software application inside the factory. This characteristic allows the system to be more flexible and deal with unexpected disturbances more quickly and efficiently, by having real-time control from what is happening internally. The integration between all the components, will also allow to optimize the resources' parametrizations, raw materials, and consequently the entire factory ecosystem (Kagermann et al., 2013).

In order to harmonize all these principles that are associated to SM it is necessary to clearly identify the main concepts and characteristics of manufacturing systems to achieve this

smartness along all the value chain. According to Kusiak (Kusiak, 2018) there are 6 vital concepts for SM:

- Manufacturing technology and processes – new manufacturing technologies and processes will emerge to integrate and improve the previous technologies and enable smarter manufacturing systems;
- Materials – the search for new materials, as well as the recycling and recovering of existing materials will have an important role to achieve a sustainable manufacturing;
- Data – the importance of data in SM is undeniable. The use of more sensors and the progress in data analytics is leading to a massive collection of data that can be converted in useful information for production processes at all levels, from the acquisition of materials to products delivery to customers;
- Predictive engineering – predictive engineering is emerging as a solution to anticipate what is going to happen instead of use a reactive approach. It allows to develop high-fidelity models of the physical representations and explore situations to support future decisions.
- Sustainability – the aim of a sustainable world is of huge importance in this moment, and the effort in manufacturing for reducing wastes plays a critical role in this field. The reuse and reconditioning of materials will have a special attention in the next years;
- Resource sharing and networking – manufacturing will benefit from the share of resources to improve the production efficiency. Sharing manufacturing equipment, software and the workers' expertise will allow to build strong collaborative networks.

Building upon the advancements of I4.0, I5.0 introduces a more human-centric approach, emphasizing the collaboration between humans and machines rather than solely focusing on automation and connectivity. While I4.0 aimed at creating smart factories driven by data and artificial intelligence, I5.0 seeks to harness these technologies to enhance human capabilities and promote sustainable manufacturing. This paradigm shift addresses the limitations of its predecessor by prioritizing the personalization of production processes, fostering resilience, and integrating environmental and social considerations into the core of industrial practices. As a result, I5.0 represents a significant evolution, aiming not only to optimize manufacturing systems but also to create a more balanced and sustainable industrial ecosystem (Nahavandi, 2019; Xu et al., 2021).

Although there is a huge complexity involved to developing SM systems and there may be some restrictions regarding process and memory power, nowadays the main limitations are not limited to the hardware neither to the connection between entities. One of the main problems is to make the link between different types of data sources and how they need to cooperate in order to achieve better and more efficient performances. Consequently, it is necessary to go one step further and direct more efforts to modelling, optimization and standardization of manufacturing systems (Arm et al., 2021; Kusiak, 2018).

2.2 Cyber-Physical Production Systems

The benefits of integrating both cyber and physical components led to an increasing trend of developing new approaches where those are integrated in manufacturing systems. By using CPS, it is possible to develop a virtual representation of physical objects, contributing to the vertical integration referred to before. CPS not only have been applied to manufacturing and industry but also in many different areas, as far as the physical systems can be improved through the cooperation and sharing of information among different entities. They can be applied to increase the productivity in areas such as agriculture, energy, healthcare, society, transportation, and more (Monostori et al., 2016).

Several challenges are faced when developing CPS, mainly related with (Monostori et al., 2016):

- Cybersecurity – the systems and networks need to be protected against malicious digital attacks, aiming to access, modify or change sensitive information;
- Economics – the efficient development of CPS will allow companies and countries to grow economically;
- Interoperability – the interconnectivity and operability between the involved components, not only physical but also in the business level, is fundamental to achieve better performance levels when developing CPS, since this is one of the characteristics that can bring smartness to the systems;
- Privacy – it is necessary to protect data and manage which information may become publicly available to other entities and which needs to be kept private, this way protecting the access to different components;
- Safety and reliability – the security of the system is vital in CPS, mainly when there are real-time constraints. Operational questions need to be considered in order to achieve safety for the workers and the system;
- Socio-technical aspects of CPS – social aspects need to be considered in manufacturing systems in order to improve the system production. Social interactions may comprise machine-to-machine communication, human-machine interaction and human-human communication.

Recently, an evolution in the design of CPPS, which expands the concept and benefits of CPS to the manufacturing context, has been observed. CPPS rely on the integration of physical and virtual worlds through interaction interfaces, which are used to monitoring and control operations. The data generated by the different resources during the production process is analyzed and converted into useful knowledge to be used in different scales to continuously improve the production process. The interaction with external systems may also be considered in order to acquire external knowledge that can add value to improve CPPS operations (Ribeiro, 2017). CPPS can provide virtual capabilities to every physical component, high degree of

automation, reconfiguring capabilities, interaction between components at different scales and integration at different spatial and temporal scales (Kumar Khaitan Siddhartha & McCalley, 2015). Although these concepts are not completely new in manufacturing systems, only now, with the advances in ICT, they are becoming reliable.

In a CPPS environment, each physical component is abstracted by a digital (cyber) entity. This entity is able to process and analyze data in order to make decisions that influence the real system. Both parts work cooperatively, continuously sending and receiving feedback to one another, in order to monitor and control the physical resources, as show in Figure 2.5. This evolution, the virtualization of the physical world, allows the traditional systems to improve their capabilities by sharing information between different entities that before had no interactions. This fact allows to have an evolvable system that can not only focus on local goals but also on the global objectives (Monostori et al., 2016). Allied to the developments in IoT, CPPS can hugely improve the manufacturing systems, since IoT systems facilitate the data acquisition and monitoring of systems (J. Zhang et al., 2019). Consequently, CPPS can improve the manufacturing systems by delivering better capabilities of control, reconfiguration, monitoring, diagnosis, scheduling, and so on.

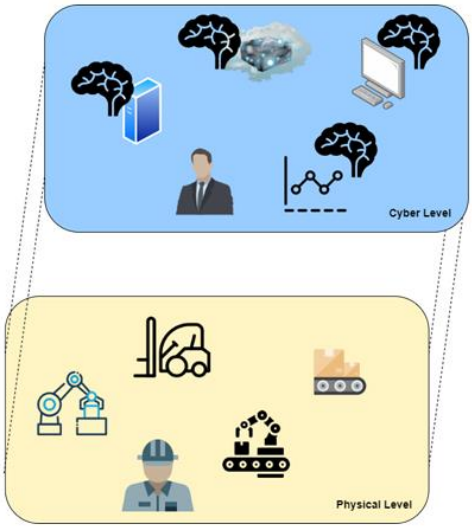

Figure 2.5 - Cyber-Physical Production System

The physical components collect the information in real-time from the real world, using sensors, and send it to the cyber layer to be analyzed by computational algorithms. After that, the found knowledge is sent back to the physical systems so it can be adjusted, allowing dynamic control (Alam & El Saddik, 2017). Consequently, they bring the capability of intelligence to the system, since this interaction between physical and cyber entities allows to monitor and control physical components in a more efficient, collaborative, and autonomous way, i.e. with less human interaction, which allows the system to react to changes in more quickly (Tao et al., 2019). Since the system is distributed, it is easier to integrate new components and

functionalities in runtime without the need to reformulate completely a specific layer. Thus, the system is able to adapt and reconfigure by itself locally, instead of spreading detected problems to other layers, and issue warnings if necessary. One of the main challenges of CPPS is exactly the need to develop robust solutions for scheduling in order to deal with the unpredictable events on distributed production processes (D. A. Rossit et al., 2019).

In the last years, some projects have been pushed to use modular approaches to deal with the need to have dynamic and flexible manufacturing approaches, deal with disturbances and adding and removing resources in runtime. The FP7 Self-Learning project developed self-learning solutions based on service-oriented architectures to enable the integration between control and secondary processes such as scheduling activities, maintenance or energy efficiency of manufacturing systems, allowing self-adaptations to the recurring changes (Scholze et al., 2013). The FP7 Arum project was developed to improve the planning and control systems for the manufacturing in small lot productions. Different scheduling and planning tools were implemented and their pluggability was enabled due to the exposition of their functionalities as services (Leitao et al., 2015). In H2020 PERFoRM a CPPS was developed to integrate the existent legacy systems with higher level tools, providing plug and produce capabilities to the system. It was developed a scheduling system able to schedule thousands of products in a shop-floor with dozens of working machines.

Also, several research articles have discussed and proposed manufacturing scheduling approaches under the umbrella of SM and CPPS (Fu et al., 2018; Ivanov et al., 2016; Mourtzis & Vlachou, 2018; D. Rossit & Tohmé, 2018; Shiue et al., 2018; J. Zhang et al., 2019; Yingfeng Zhang et al., 2018; Zhong et al., 2013).

2.3 Reference Architectures

In order to standardize the designing and development of manufacturing systems, some reference architectures have emerged during the last years. These architectures aim to assure a common comprehension, achieve standardization, enable semantic interoperability and to provide consistent operation models for the system.

Among these architectures, one of the oldest, and maybe the most known and used, is ISA-95. ISA-95 is composed by five levels, as shown in Figure 2.6. Level 0 for physical production process. Level 1 for production process sensing and manipulating. Level 2 is for manufacturing control, mainly monitoring, supervisory and automated control of the production. Level 3 is for operation management, establishing production schedules and optimizing the production process. Finally, Level 4 deals with business planning and logistics, dedicated to business-to-manufacturing transactions (ZVEI, 2011).

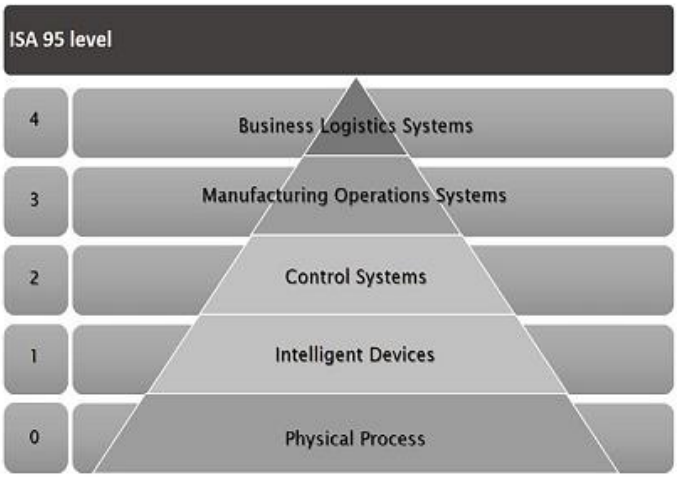

Figure 2.6 – ISA-95 pyramid architecture, (D. Rossit & Tohmé, 2018)

Also, Computer Integrated Manufacturing Open System Architecture (CIMOSA) is a comprehensive framework developed to provide a standardized approach to designing, integrating, and managing manufacturing systems. It was developed with the aim of improving the flexibility and efficiency of manufacturing enterprises through better integration of business processes and information systems. CIMOSA provides a robust framework for designing and managing complex manufacturing systems. By promoting modularity, standardization, and a process-oriented approach, it helps manufacturing enterprises achieve better integration of their business processes and information systems (Kosanke, 1995).

More recently, some architectures have emerged more oriented to the design and development of SM systems. Contrary to ISA-95 and CIMOSA, architectures such as the 5C architecture, Adaptive Holonic Control Architecture for Distributed Manufacturing Systems (ADACOR), the Smart Grid Architecture Model (SGAM), the Industrial Internet Reference Architecture (IIRA), or RAMI4.0, consider recent progresses, visions, and paradigms in manufacturing to design CPS and IoT systems. Although they are all different, several similarities can be found between them, even when comparing with the ISA-95.

The 5C level architecture was proposed for the realization of CPS, Figure 2.7, that is composed by hierarchical levels similar to ISA-95. It consists of five different levels (smart connection, data-to-information conversion, cyber, cognition and configuration levels), which defines how to develop a CPS from the initial data acquisition to the last value creation (J. Lee et al., 2015). More recently, (Jiang, 2018) extended this version by adding three additional levels (coalition, customer, content), which focus on both vertical and horizontal integration and on the entire product lifecycle.

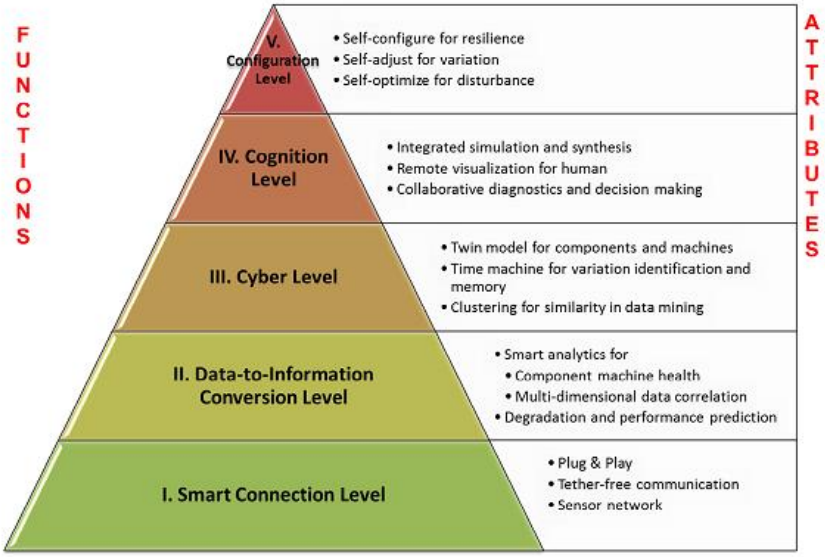

Figure 2.7 – 5C architecture for implementation of CPS, (J. Lee et al., 2015)

ADACIR is a control architecture designed to enhance the flexibility, agility, and adaptability of manufacturing systems. It is particularly well-suited for complex and dynamic production environments where traditional hierarchical control structures may be insufficient. It is based on the principles of holonic manufacturing systems, where a "holon" is an autonomous and cooperative building block of the manufacturing system, e.g. a product, a machine, or a control unit, each processing autonomous and cooperative characteristics. One of the primary features of ADACOR is its ability to adapt to changes in the manufacturing environment, as it dynamically adjusts its control strategies in response to disturbances, such as machine failures or changes in production demands, ensuring optimal performance. ADACOR also promotes a decentralized control structure where holons operate independently but coordinate with each other to achieve common production goals. This decentralization improves the system's robustness and scalability (Barbosa et al., 2015).

Entering in the three-dimensional level architectures, i.e. architectures covering three different characteristics, the similarities are even greater. The SGAM is a three-dimensional architecture that merges the concept of the interoperability layers (component, communication, information, function and business) in the smart grid plane, which results in a model that replicates the domain, interoperability and zone dimensions and allows the representation of entities and their relationships, as illustrated in Figure 2.8 (CEN et al., 2012).

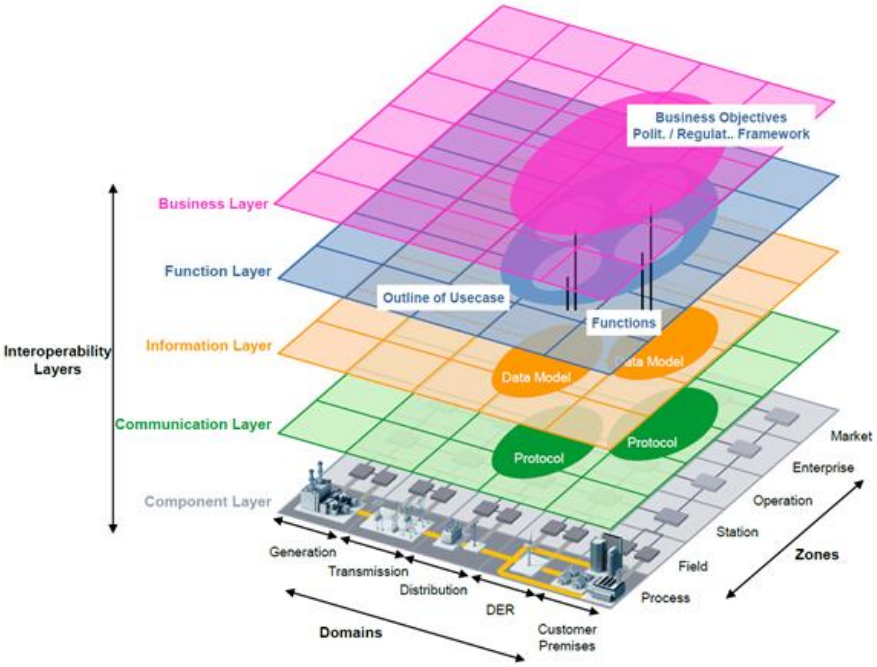

Figure 2.8 – SGAM architecture (CEN et al., 2012)

Alternatively, the IIRA is an architecture for Industrial IoT isystems, i.e. the IoT capabilities applied to industry. The IIRA relies on four viewpoints (layers), mainly business, usage, functional and implementation that cover the lifecycle process and different industrial sectors (Shi-Wan et al., 2017).

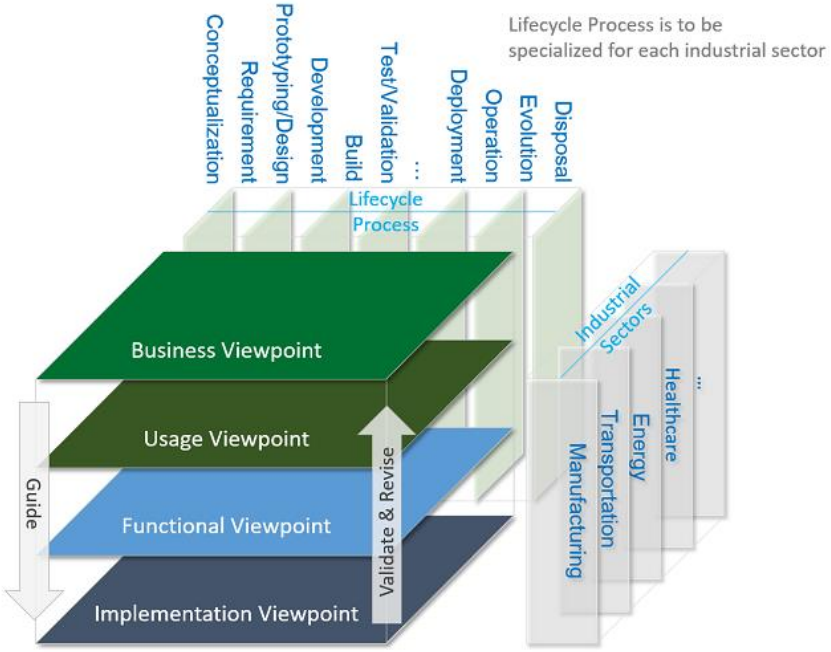

Figure 2.9 – Relationship amid IIRA viewpoints, application scope and lifecycle process (Shi-Wan et al., 2017)

When considering one of the most recent manufacturing paradigms, the I4.0, RAMI 4.0 is one of the most known and used architectures currently. The architecture puts together key

elements of I4.0 in a three-dimensional layer model, composed of six different layers, the life cycle value stream and the hierarchy levels.

RAMI4.0 is becoming a global standard and is, nowadays, one of the most used architectures and very supported by industry companies in Europe, such as ABB, Bosch, Festo, Siemens and SAP. So, it will be more thoroughly analyzed.

2.4 RAMI 4.0

In recent years, Digital Transformation in manufacturing has become a prominent topic of research and debate. This focus has led to the development and implementation of new technologies and tools designed to leverage the data generated by shop floor assets. However, this increased interest in data generation and management has also underscored a critical issue: the lack of standardized models and structures for data sharing and interoperability. Among the various concepts proposed to address this challenge, the concept of AAS, conceptualized under the scope of RAMI4.0 is gaining popularity, as it offers standardized and modular information about assets, processes and events, making it a promising solution for improving data interoperability (Quadrini et al., 2022).

RAMI4.0 was developed to accomplish a common understanding of which standards, models and use cases are vital for developing a SM. The architecture puts together key elements of I4.0 in a three-dimensional layer model. The procedures described in the standards allow smaller companies to adapt for I4.0, by implementing even partial standards which may allow to develop an early I4.0 application. RAMI 4.0 also enables the identification of standards required for relevant use cases (ZVEI, 2016). From this architecture two important requirements arise (Platform Industrie 4.0, 2018):

1. It must be possible to use, maintain or even extend the definitions and data in respect of an asset throughout its lifetime if the Use Case so requires.
2. It should be possible to preserve a link between “type” and “instance” definitions in respect of an asset throughout its lifetime.

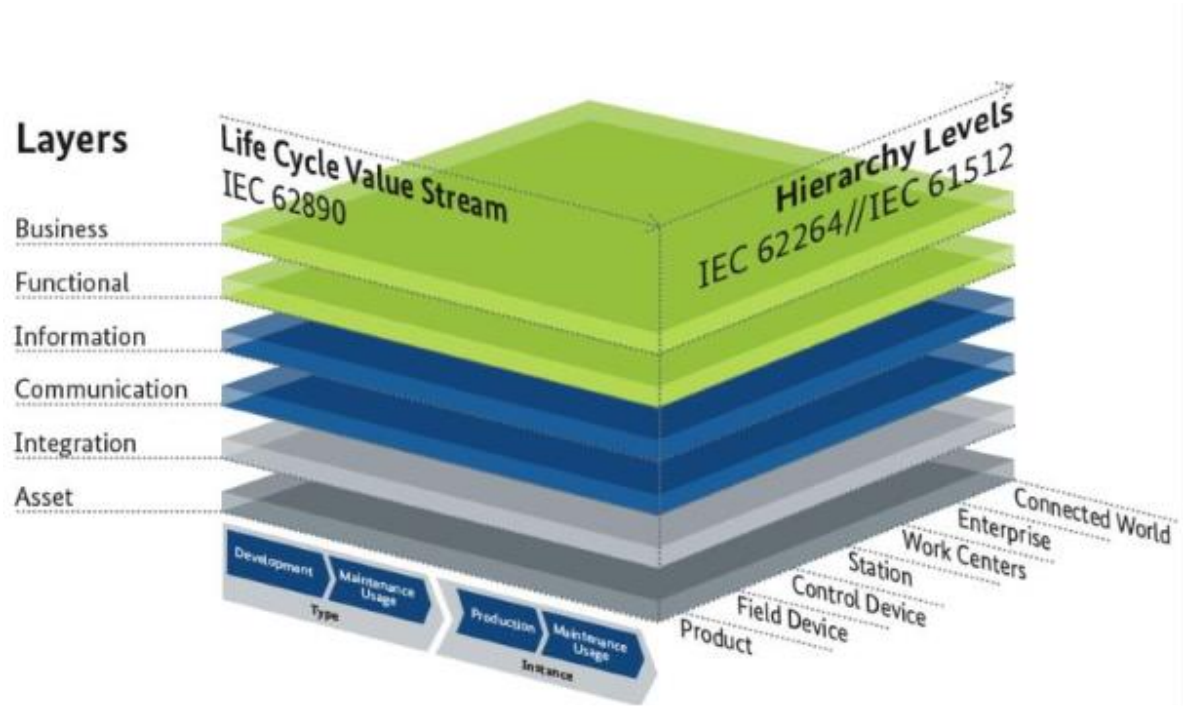

Figure 2.10 – RAMI4.0 architecture overview, (ZVEI, 2016)

It makes possible to easily handle the different phases and aspects of the system (Figure 2.10). RAMI 4.0 expands the Hierarchy Levels of IEC 62264 by adding the product level at the bottom and the connected world, which encompasses individual factory boundaries, at the top. The Life Cycle Value Stream of products or systems differentiates between type and instance. The structure of the IT representation of an I4.0 Component is represented in the Layers axis (Platform Industrie 4.0, 2018; ZVEI, 2016).

This new approach allows the description and development of higher flexibility solutions in the context of an I4.0 environment as well as the encapsulation of functionalities where appropriate, due to its three-dimensional model. Next, the three architecture axes are described.

2.4.1 Layers axis

The Layers axis (Figure 2.11) that represents the different perspectives such as data maps, functional descriptions, communications behavior, hardware/assets or business processes is divided into Asset, Integration, Communication, Information, Functional and Business (ZVEI, 2016).

Asset

The asset layer represents physical and virtual components, such as metal parts, software, documents, ideas, and more. In the simulation of a system, such as a machine, it is not only the information and communication functionality that is important. Its cables and its mechanical structure are also considered, although they are not able to communicate. Their information

needs to be available as a virtual representation. Thus, the asset layer at the bottom enables an improved description of the machines, components and factories. Essentially, it describes how to integrate physical components. The Asset is connected to the virtual world through the Integration layer (ZVEI, 2016).

Figure 2.11 – RAMI 4.0 Layers axis, (Schweichhart, 2016)

Integration

Above the asset layer is the integration layer, which allows digitization of the assets for virtual representation. This layer provides information on the assets that can be processed computationally. It also contains the elements connected with IT, such as sensors, RFID readers, HMI and so on. In the case of a software system, the entire asset is digitally represented in the digital world, while for physical objects such as resources (production components, tools, stations, human workers) and products, this may not be the case. The Integration layer tells which part of the assets will be digitally available in the network (ZVEI, 2016).

Communication

The communication layer is made-up to deal with protocols and the transmission of data and files. It should provide the services for control of the integration layer and a standardized communication between the Integration and the Information layers. The data should be handled using TCP-IP, HTTP, RESTFul services, MQTT and OPC-UA protocols, transmitted through LAN or WAN, and communicate through Bluetooth or Wi-Fi devices. It basically defines how to access the data (ZVEI, 2016).

Information

Linked to the communication layer is the information layer which comprehends relevant data about the system. The information layer is a runtime environment for (pre-) processing of events. Rules are applied here to one or more events to generate on or more further events which will initiate processing in the functional layer. The Information layer describes what data

the asset has to provide. Adopting AutomationML and JSON data format for storage and exchange of information will help in defining consistent data structures and semantics, while it eases the interconnection between different components (ZVEI, 2016).

Functional

The information layer contains all the essential functions. It should contain a formal description of the functions and should be a platform for horizontal integration of the various functions. At the same time, it should be a runtime and modelling environment for services which support business processes and for applications and technical functionality. Here are generated the rules and decision-making logic. Most exactly it defines what the asset is supposed to do and with what services (ZVEI, 2016).

Business

Finally, the business layer maps the relevant business processes. It ensures the integrity of function in the value stream, models the rules the system needs to follow, orchestrates the services in the functional layer, makes the link between diverse business processes and receives events for advancing of the business processes. It is in this layer that the rules the system has to follow need to be modelled. The Business layer explains what the end user needs are (ZVEI, 2016).

2.4.2 Life Cycle & Value Stream Axis

Along the horizontal axis is the product life cycle and its value streams, such as dependencies that can also be represented in the reference architecture model. The draft of IEC 62890 is a decent guideline for the life cycle considerations, which distinguish between *type* and *instance*. The *type* is created when a product comes to the development phase, which covers commissioning, development and testing up to the initial sample and prototype production. Here is where the type of product or machine is created. In the case where an error is reported to the manufacturer or improvements should be made, the type documents may be revised. The *instance* is represented by each manufactured product of a general type, and it is assigned a unique serial number. These instances are then purchased by customers and delivered to them. For the customer, the products, which are originally just types, only became instances once they are deployed in a specific system (ZVEI, 2016).

Digitization is a core element in SM paradigm. The digitization and linking of the value streams provide enormous potential for improvements in the system, once purchasing, order planning, assembly, logistics, maintenance, customers, suppliers and so on are all interconnected. The cross-linking between different areas is of huge importance, since it makes possible to see inventories in real time and know where are the necessary parts for production at

any time, at the same time as the customers are able to see the completion status of the production (ZVEI, 2016).

2.4.3 Hierarchy Levels

Finally, the Hierarchy Levels axis represents the vertical integration, specifically, the location of functionalities and responsibilities within factories and plants. In I4.0 there are smart products being operated in smart factories that are connected to an external world. This makes the systems more flexible, where functions are distributed through all the network and all the participants communicate and interact across hierarchy levels. The goal is to do a functional assignment that describes the functional classification of different stages of I4.0. It is based on ISA-95 and follows the IEC 62264 and IEC 61512 standards, the international standards series for enterprise IT and control systems and batch production processes respectively. The functionalities have been extended to include parts in production, i.e. Products, and also the connection between the system and the IoT and services as well as the link to other factories and external collaborations, i.e. Connected World. The Field Device was added below the Control Device in order to have consideration regarding a machine or system, and not only about control device. The Field Device represents the functional level of an intelligent field device, such as a smart sensor. The distinct terms Enterprise, Work Centers, Station, Control Device and Field Device are used to identify different working levels in the shop floor and then cover as many sectors as possible from process industry to factory automation (ZVEI, 2016).

Here are represented in a functional way the structure of technical assets such as products and stations organized into their different functionalities in the system as well as the external interconnected world.

2.4.4 Asset Administration Shell

The AAS allows to transform physical components, such as robots, machines or devices and, similarly, intangible assets like functions, plans or an entire network into I4.0 Components, which will then allow to standardize as much as possible solutions for engineering, operation and management and implement a heterogeneous communication structure in a SM-oriented system. More specifically, the AAS is a standardized information model used to describe technical assets in the context of I4.0, as it provides a digital representation of the structure, behavior, and capabilities of an asset, which may be physical or virtual, as well as relationships to other assets and systems in the industrial environment. It leverages several functions designed to make this technology a true enabler of interoperability in the manufacturing domain (Platform Industrie 4.0, 2022; ZVEI, 2019).

The AAS includes all the relevant information of the asset, being complaint with the layers' specification of the RAMI 4.0. It is the virtual representation of the asset, which encompasses all the information and technical functionalities of the asset and manages communication, collaboration, and interoperability with other I4.0 Components (ZVEI, 2019), as represented in Figure 2.12. Thus, the AAS makes it possible for different systems to be seamlessly integrated, reduces information asymmetry, and improves data-driven decision-making by adopting a common language and format for asset information.

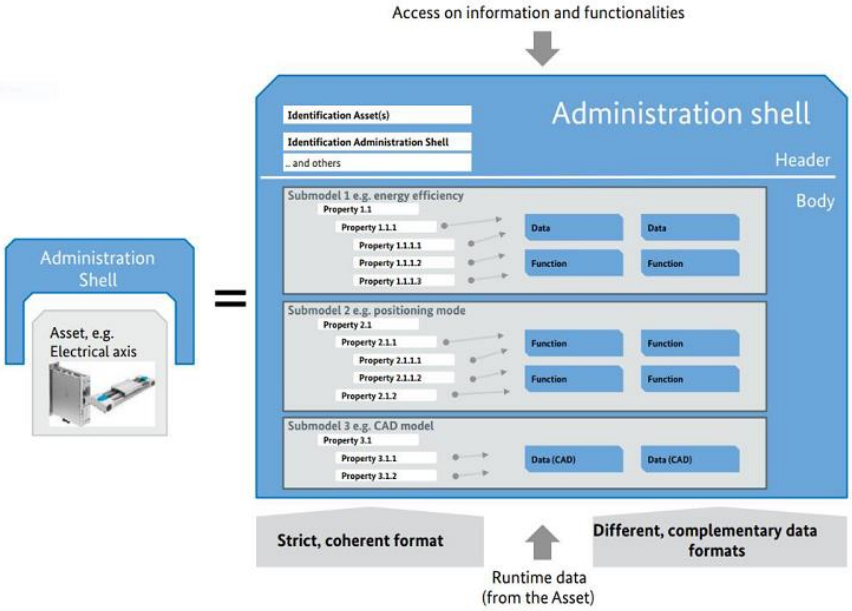

Figure 2.12 – General structure of an Administration Shell, (Platform Industrie 4.0, 2018)

The AAS is composed of a Digital Factory (DF) *Header* and a DF *Body*. Specifically, the DF *Header* contains the *Manifest* with the information about the identification of the AAS and the represented assets. On the other hand, the DF *Body* comprises the submodels that characterize the AAS itself, containing the *Manifest* with the meta-information of the submodels and the *Component Manager* that manages the services described in the submodels. The AAS comprises a variety of information about an asset, including its identification, configuration, status, and operational data. At the same time, it also provides a structure for defining the communication protocols and interfaces required for accessing and controlling the asset (Platform Industrie 4.0, 2022; Plattform Industrie 4.0, 2016).

Each AAS entity ensures interoperability through a series of functions. Firstly, the Reference function provides an ordered list of keys that define the represented element. The Kind function distinguishes the asset's level of embodiment, identifying it as either a type (e.g., a specific tool type) or an instance (a specific object on the shop floor). For referability and identifiability, AAS offers global identifiers that unambiguously identify an asset type, independent of context, and attributes such as *idShort*, category, description, and parent that make the

entity referable within a defined namespace and explicit in its relationship with parent entities. The Semantics function provides structured references to external entities, while the Data Specification function adds additional attributes that can be referenced against existing global templates. The *AssetAdministrationShell* is the core element of the metamodel, containing references to the Asset, any Submodel(s), and the concept Dictionary. The Asset is an identifiable entity containing all metadata about the related asset. The Submodel lists specific attributes of a subsystem of the metamodel, typically decomposing a complex object into its components (e.g., a machine into its subsystems). Being identifiable, the Submodel is often referred to as a specific AAS metamodel describing the subsystems, which can be further decomposed into submodels (e.g., describing its components in terms of sensors, actuators, structures). Submodels use their own Data Specification and Semantics to model the subsystem in terms of important properties (Quadrini et al., 2022).

All these characteristics contribute to the primary goal of interoperability. A metamodel, defined as a type, can be uniquely represented across various contexts. This promotes the standardization of machine bills of materials and production processes, and simplifies processes such as requesting spare parts, scheduling maintenance, and integrating new equipment into existing manufacturing systems. For instance, it enables seamless communication between different machines and systems, automates inventory management, and ensures that replacement parts are accurately identified and ordered, reducing downtime and improving overall efficiency.

Figure 2.13 shows some possible sub models and the respective standards used by each one. The sub models are used to standardize different skills or the different actions the asset may perform. Then, the system just need to know which capabilities are required and find the AAS that offers that skill or a submodel with the required properties As it is possible to observe in Figure 2.12, it is possible to externally access the data and functionality of each sub-model, for example, through an API.

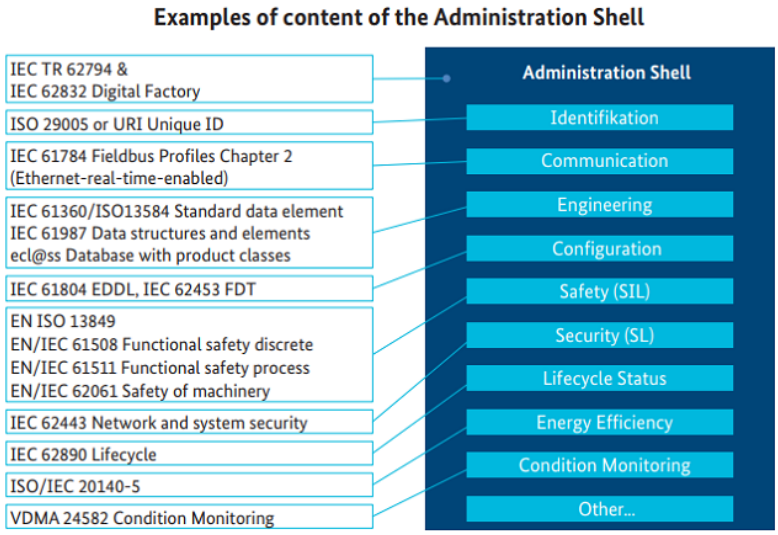

Figure 2.13 – Examples of content of the AAS, (Plattform Industrie 4.0, 2016)

By adopting the utilization of AAS, organizations are able to create a DT of their assets that can be used for optimization, simulation or monitoring of their systems. It can also be used to support the assets integration in broader manufacturing and supply chain systems, leading to greater visibility and control over the entire production process. However, well defined standards and communication protocols are essential to have everything working smoothly.

Due to different customs, unit definitions, naming conventions, and taxonomies across nationalities, adhering to a standard is essential. The IEC 61360 standard defines a dictionary schema for vocabularies, and the data specification templates are specified by IDTA in (Industrial Digital Twin Organization, 2023). Other dictionaries, such as eCl@ss, can be created based on this standard. While these dictionaries standardize property descriptions and metadata for submodel creation, they do not fully describe component features or property relationships. In I4.0, AAS components share data and functions, and while interaction protocols are defined in (Platform Industrie 4.0, 2022), a complete messaging standard is still needed. Currently, no standard defines the runtime framework for AAS. Options include server execution for maximum computational power or PLC implementation for distributive I4.0 compliance, though the latter is hardware-limited. AAS relies on OPC UA technology, and if a standard explicitly mapping AAS to OPC UA is established, OPC UA could serve as the general framework for AAS creation (Benešl et al., 2022).

As already mentioned, for communication OPC UA or MQTT protocols are recommended for the actual communication of individual devices or their AAS components.

2.4.5 I4.0 Component

The I4.0 Component is considered a specific case of a CPS in an I4.0 context, it can be a physical machine, a device, an application or even a function. These entities can be considered an I4.0 Component after being wrapped by an AAS, as described in Figure 2.14. The use of I4.0 Components is important because they provide not only a wide range of life cycles in relation to the various partners of a value-added network, but also the possibility of locating the I4.0 Component in RAMI4.0 and the possibility of operating I4.0-compliant communication equally for both active and passive connected assets (Plattform Industrie 4.0, 2016).

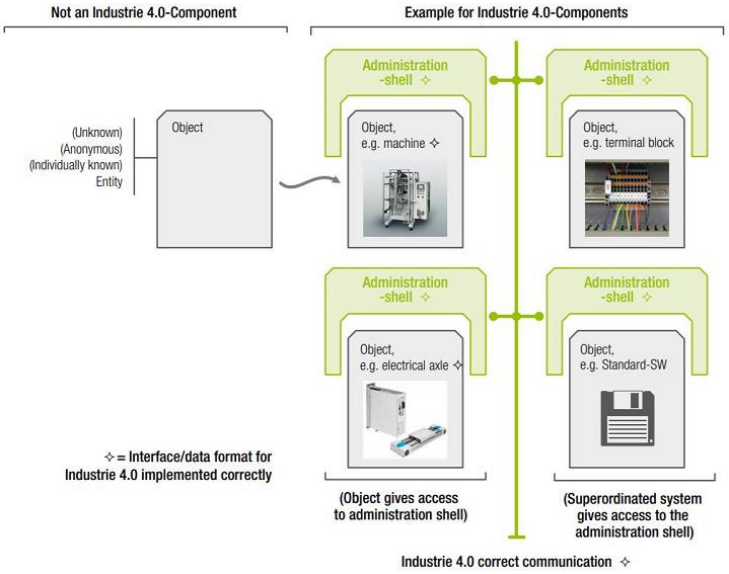

Figure 2.14 – Object becoming an I4.0 Component, (ZVEI, 2016)

The I4.0 Component, represented in Figure 2.15, is a combination of one or more assets with an Administration Shell. An asset may be physical, e.g. a motor or a working station, or logical, e.g. a protocol or software, and it is allowed to have several Administration Shells which may be appropriate for different reference frameworks (Plattform Industrie 4.0, 2016).

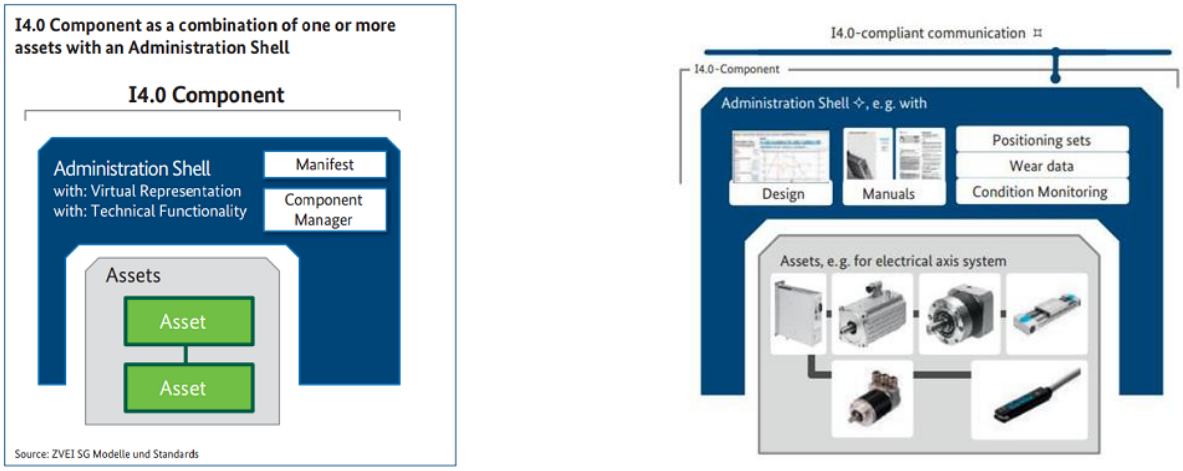

Figure 2.15 - Industrie 4.0 Component representation (left). Several assets within an Administration Shell (right), (Plattform Industrie 4.0, 2016)

Therefore, an I4.0 Component should also be able to incorporate other I4.0 Components, allowing the modularization of the system, which is crucial in SM systems, creating a nest of I4.0 Components. Here, the different objects are split and transformed into other I4.0 Components and thus, a higher-level object may have different I4.0 compliant communication interfaces, where there is a logical and physical separation between the higher and lower levels (Figure 2.16, No. 1). Another option is to have the I4.0 compliant communication physically unified at both top and bottom, but logically separated, where other I4.0 Components in level 1 would be able to access the bottom elements (Figure 2.16, No. 2) (ZVEI, 2016).

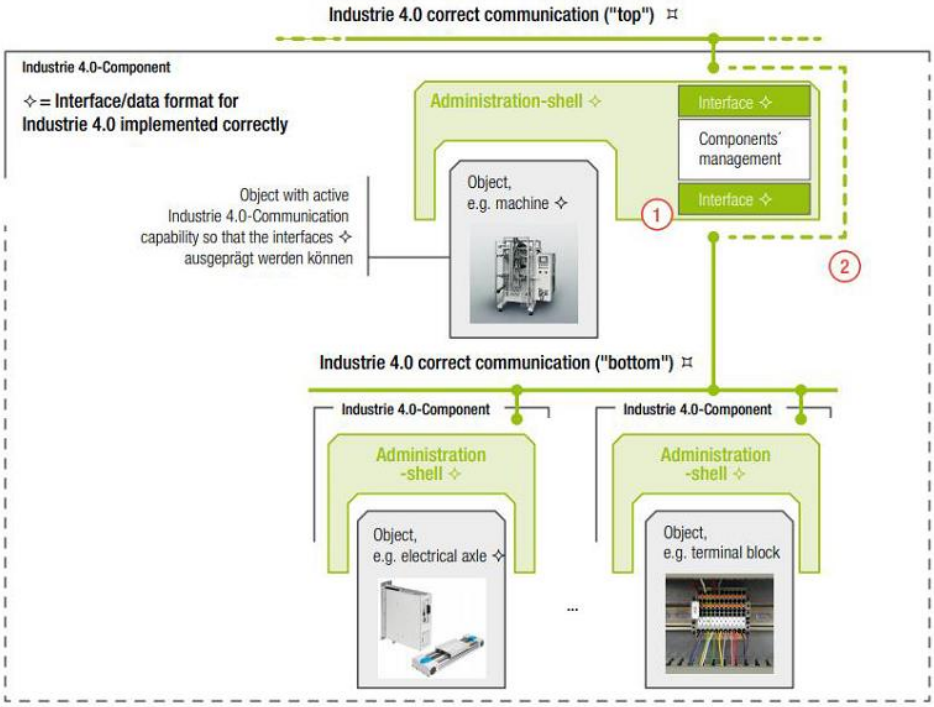

Figure 2.16 – Nestability of I4.0 Components, (ZVEI, 2016)

An I4.0 Component can be anything from an entire production system to an individual machine, going through a working center and an assembly line. Each of these I4.0 Components go through the life cycle of the factory in dynamic relevance to the office and shop floors, and in contact with central systems such as Product Life Cycle Management, Enterprise Resource Planning (ERP) and Industrial Control and Logistics systems (ZVEI, 2016).

The presented architectures are mainly implemented CPS-oriented since they integrate both physical and cyber capabilities. For instance, the I4.0 Component from RAMI4.0 can be an example of a CPS, a case of a physical component that is transformed in a cyber-physical entity.

2.5 Manufacturing Scheduling

Manufacturing processes can be very dynamic. Even in environments where the processes happening in the shop-floor are always the same and known in advance, they can be affected by

one or another disturbance that forces to stop all the production until the problem is solved. Although some years ago manufacturing systems were not ready for this change and were not efficient enough to deal with these disturbances, nowadays, manufacturing is becoming more adaptive, dynamic and highly flexible to meet market requirements and adjust to all and every change that may improve the process. This is even more important in the era of the emergence of mass individualization, where the disturbances on the production line can be even more significative. In order to minimize unexpected events and improve the overall production performance, one of the key challenges is to develop reliable and robust scheduling solutions. I4.0 scheduling approaches should be designed to deal with these smart and dynamic manufacturing systems and their new technologies.

Scheduling has been largely applied in many different areas such as energy consumption (C. Lu et al., 2017; J. Wang et al., 2016), transportation (Carosi et al., 2019), staff distribution (Moosavi et al., 2022), and manufacturing (Balin, 2011; X. Li & Gao, 2016), among other areas, to help the industries to plan their activities. However, specific algorithms and mathematical models should be developed for scheduling solutions in different areas of application.

In manufacturing, scheduling can be considered as a process of arranging, controlling and optimizing work in the shop-floor (Y. Liu, Wang, Wang, Xu, et al., 2018). Sometimes, parts need to wait too long on the shop-floor due to limited resources to manage them or to a weak planning of the system. Production scheduling aims to efficiently allocate the available resources and reach a predefined goal. Scheduling is a process of optimizing work and time. A scheduling problem may be described as an environment composed of one or more machines, with specific characteristics, and a set of jobs (products with one or more operations that will be processed by the machines). The goal is to optimize an objective or group of objectives by assigning each job to a specific machine in a specific time in order to be processed, while conflicts between operations are avoided (Nguyen et al., 2017; Serrano-Ruiz et al., 2022). In a succinct way, scheduling determines what is going to be done, where and with what resources. Dynamic scheduling is of extreme importance for SM operations, as it refers to the process of adjusting schedules in real-time based on changes in demand, machine availability, and other factors. In an industrial environment, this is critical for ensuring production stays on track and deadlines are met. Without dynamic scheduling, manufacturing operations would be much less efficient and flexible. For instance, if a machine breaks down unexpectedly, without dynamic scheduling, production would come to a halt until the machine is repaired or replaced. This could result in missed deadlines and lost revenue. On the other side, if a customer places a large order unexpectedly, it can help the manufacturer adjust production schedules to accommodate the order on the current production and ensure it is delivered on time.

Although has been studied for decades, complex and robust scheduling solutions are often disregarded in real manufacturing scenarios, where they are sometimes done manually or on simple software programs. These solutions lead many times to significant errors since they do not consider the current status of the shop-floor and are not adaptive to different scenarios. Though, more robust solution are not implemented mainly due to the complexity to implement them in large scale systems with real-time constraints, since it is considered to be a NP-hard combinatorial optimization problem which is quite difficult to reach an optimal solution with traditional optimization techniques (Çaliş & Bulkan, 2015). However, scheduling optimization has direct impacts on the production efficiency, sustainability and also on costs of manufacturing systems and must be developed to its full capabilities (Gahm et al., 2016; J. Zhang et al., 2019).

Whereas most researchers assume some constraints, such as that stations are always available or that the processing time of a job is known in advance and remains constant during the entire process, in real systems this is not always true.

Disturbances may occur during the production process, which leads to a rescheduling that should be carried out as fast as possible. These disturbances can be the arrival of new orders, cancelled orders, station breakdowns, which lead to station's unavailability, human errors or absences, unplanned maintenance, or some emergency event (Yoo & Lee, 2016; J. Zhang et al., 2019). Additionally, jobs' processing times may increase over time, which is a situation knowing as *deterioration of resources* in scheduling problems (Chung & Kim, 2016), or even decrease when there is a learning factor or the workload can be reinforced (Framinan et al., 2014). Consequently, in order to adapt to the manufacturing system, it is vital that the scheduling process can be dynamic and quick to avoid unnecessary system downtimes and costs.

In flexible and agile manufacturing environments, products can have several different feasible processing plans and most of the time it is very hard to find a good one for all the products. Production scheduling is a very important decision making in a factory and it can be a difficult problem depending on the number of calculations necessary to obtain a schedule that gives an optimal or near-optimal solution for the given objectives (Balin, 2011).

The high complexity and large instances of scheduling problems in real manufacturing systems make that traditional methods are not able to solve those problems in reasonable time. Moreover, the stochasticity and dynamics of modern systems, with a lot of changes during the production process, new job arrivals or machine breakdowns makes it impossible to use exact mathematical methods in the factories (Branke et al., 2016). So, the traditional scheduling solutions should be abandoned at the expense of smart and dynamic scheduling solutions.

The question about how to achieve high production efficiency, with low production costs and high product quality has attracted a lot of attention in the last decades (Zhuang et al., 2018). Today, the recent advances in IT and manufacturing technologies such as CPS, AI, IoT, Cloud Computing, Big Data, DT, and so on will help to optimize and dynamically adapt the scheduling solutions to the current needs of the manufacturing systems (Hermann et al., 2016; J. Zhang et al., 2019).

The production scheduling optimization problem may be decomposed in several categories, according to its type. In almost all of them, the essence is that several jobs (products with one or more operations to be processed) are assigned to a set of machines at specific times, satisfying some constraints, while trying to minimize the makespan, i.e. the time between the moment that the first job started until the moment that the last job is finished, or optimize some other objective, such as the production due dates (Çaliş & Bulkan, 2015; Chung & Kim, 2016; Karimi et al., 2016; Serrano-Ruiz et al., 2022).

In Figure 2.17 is represented a simple schedule composed by three jobs, each one with three different operations to be executed in three different machines.

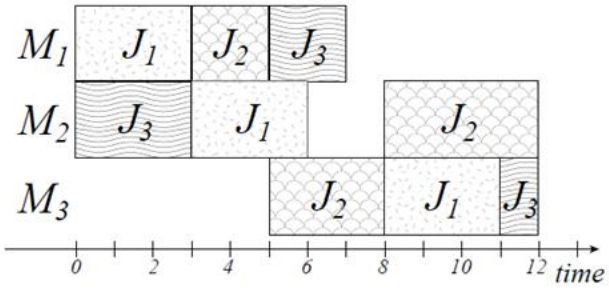

Figure 2.17 - Gantt chart representation of a scheduling problem, (Yamada & Nakano, 1997)

Moreover, to produce an optimized solution, restrictions regarding product parts, material availability, machines or work capacity, start and due dates, costs, distribution requirements or setup efficiency conditions must be known (Lin et al., 2012).

Employing DTs for advanced optimization is crucial. There is an urgent need to delve into advanced manufacturing, focusing on dynamic scheduling optimization, resource allocation, and process control to enhance production efficiency. By leveraging DT, manufacturers can simulate and analyze production processes in real-time, identify bottlenecks, predict maintenance needs, and optimize workflows (Xiang et al., 2023). This not only improves production efficiency but also enhances overall productivity, reduces downtime, and ensures a more agile and responsive manufacturing environment. Exploring the full potential of DT for advanced optimization is essential for staying competitive in the rapidly evolving manufacturing landscape.

2.5.1 Shop scheduling problem

There are several job scheduling environments depending on the machines' layout and the flow of the products. They can be divided in three big groups: flow-shop, job-shop, and open-shop.

2.5.1.1 Flow-shop

A flow-shop is composed of a set of machines disposed in series, one after another, where the products go in the same direction through all the machines. Thus, the execution order is the same for all the jobs. This kind of configuration is used in mass production lines, such as assembly lines or electronic and food industries (González-Neira et al., 2017). All jobs must follow the exact same route in the factory, going from machine to machine until reach the last one. The problem in this kind of configurations consists in determine the processing sequence of the jobs on each machine (Pinedo, 2016).

A variation of this problem is the flexible flow-shop. In this case, at least one stage of the shop-floor is composed of two or more machines in parallel, able to execute the same operation. Consequently, it is necessary to decide to which of the parallel machines the operation will be allocated (Pinedo, 2016). From this scenario may result problems related to storage between stations. If there is no buffer, or if it does not have enough capacity, the system may incur in a blocking situation.

2.5.1.2 Job-shop

The Job-Shop Scheduling Problem (JSSP) is one of the most famous scheduling problems and is a sub-problem of production scheduling. The traditional JSSP can be described as a set of m machines $\{M_1, \ldots, M_m\}$ that should process a set of n different jobs $\{J_1, \ldots, J_n\}$. Each job has a set of operations to be processed in a given order in the different machines. An operation O_{JM} of a job J requires the use of the machine M for an uninterrupted duration, dominated processing time, and each machine can only operate a job at a time. No job needs to go through every machine, neither a job needs to be processed twice by the same machine (H. Xiong et al., 2022). The main difference for flow-shop is that every job may have a different route.

A list of restrictions of the static model of JSSP is presented in (Mattfeld, 1996) and are still applied, as in (M. Zhang et al., 2021):

- No two operations of one job may be processed simultaneously.
- No preemption (i.e. process interruption) of operations is allowed.
- No job is processed twice on the same machine.
- Jobs may be started at any time.

- Jobs may be finished at any time.
- Jobs must wait for the next machine to be available.
- No machine may process more than one operation at a time.
- Machine setup times are negligible.
- There is only one of each type of machine.
- Machines may be idle within the schedule period.
- Machines are available at any time.
- The technological constraints are known in advance and are immutable.

Although some of those assumptions still be considered in academic studies when developing scheduling solutions nowadays (as seen in Table 1), they are not applicable for real-world scenarios. For example, jobs are often released at different times, which leads to a more dynamic environment. When the release times are known in advance, the job-shop can be classified as dynamic, but deterministic. On the other hand, when release times are not known in advance, the model is considered to be non-deterministic (Majdzik, 2022).

The JSSP is one of the most important combinatorial optimization problems. It is not only NP-hard but is one of the worst members in the class. This means that it is maybe impossible to find an optimal solution in the whole search space (Kundakcı & Kulak, 2016; Z. Zhang et al., 2023). Such complexity is due to larger product variety and constraints and because there is a factorial number of solutions that can result in a feasible schedule. This problem is even trickier when operations can be processed by more than one machine or if machines can execute more than one type of operation (Balin, 2011; Ying Liu et al., 2014).

Once technology has evolved over past decades, factories have started to use flexible machines, able to perform more than one type of operation, which is known as Flexible JSSP. This is very important in actual factory systems. However, there is an additional problem, since it is not only needed to get the sequence of operations on machines but it is also needed to assign operations to those machines that are able to change their operation mode (Arm et al., 2021; Karimi et al., 2016).

Other problems may arise, for example, when both machines and workers are necessary to execute the jobs and those cannot be processed if machines, workers or both are not available. This scenario is known as Dual Resource Constrained Job-Shop Scheduling Problem (DRCJSP). This situation presents additional challenges that must be considered on scheduling, such as the interaction between workers and stations (X. Li & Gao, 2016).

2.5.1.3 Open shop

Open shops are composed by machines that can perform all operations. Thus, there are no fixed routes for each job. Each job consists of unordered operations that do not have precedence constraints. It is assumed that each job has to be processed on any machine and it can visit the same machine several times (Asghar et al., 2019).

Unlike job-shops, here the routes may differ and are not defined according to the job. Therefore, the flow of the products is very dynamic in order to be adjusted to the schedule. An open shop is more used when a balance between production volume and variation of products is the objective (Abreu et al., 2020; He et al., 2022).

Open shops are also the most general and complex of the shop types. Both flow and job-shops may be considered as specific derivations of open shops.

2.5.2 Objective Functions

Objectives are used to find the best results to particular cases. They can be evaluated as single objectives, where only one parameter is considered to optimize the scheduling process, or multi-objective, where two or more objectives are considered, increasing the difficulty of the algorithm implementation. Furthermore, different weights may be assigned to different criteria, accordingly to the main objective.

Objective functions have been mainly based on production objectives. Most of research works focus on optimizing the schedule makespan, so the operations may be finished has soon as possible. In some cases, the objective is to optimize the flow time, i.e. the sum of completion times. Maximum or average variations of the previous cases can be applied. Another type of objectives is related with due dates, such as maximum or average tardiness and earliness, the number of tardy and early jobs or even the maximization of just-in-time jobs (Framinan et al., 2014; Pinedo, 2016).

Although a lot of importance have been given to production-related objectives, in recent years other type of objectives has emerged in SM systems, namely energy efficiency-related, sustainability, social and societal requirements, among others.

In order to optimize those objective functions, several methods have been utilized along the last decades. Exact methods (W. C. Lee et al., 2016), i.e. mathematical models, were commonly used, however they are not efficient to face the dynamic changes that happen in real environments. Other techniques are dispatching rules (Freitag & Hildebrandt, 2016), shifting bottleneck heuristic (Sobeyko & Mönch, 2016), Particle Swarm Optimization (Singh & Mahapatra, 2015; Zhao et al., 2014), Artificial Bee Colony (Madureira et al., 2013), Ant Colony

Optimization (Saidi-Mehrabad et al., 2015), Genetic Algorithms (Balin, 2011; Kundakcı & Kulak, 2016), Local Search algorithms (Asadzadeh, 2015; X. Li & Gao, 2016), and so on.

The previous topics lead to restrictions and requirements imposed by the type of factory or the objective that is being optimized in each solution. However, by analyzing the state of the art in manufacturing scheduling, it is possible to come up with several requirements that are intrinsic to a lot of scheduling systems but are not considered together in each approach.

2.5.3 Scheduling Non-Functional Requirements

In this sub-section are studied some requirements considered to be fundamental when designing and developing manufacturing scheduling solutions. They may not be required for every case but need at least to be considered.

Requirements play a crucial role when designing any kind of system, as they are the basis to start developing any component from software to hardware. The purpose of a requirement is to influence the process of systems development and establish a basis for further steps, such as communication, system integration and maintenance, system architecture, benefits optimization, improving employee satisfaction and so on. Requirements may be differentiated in many types, but mainly in two big groups: Functional Requirements (FRQ) and Non- Functional Requirements (NFRQ). FRQ refers to the actions that a system has to perform automatically and the interactions between the system and other systems or human users. It is a requirement regarding a behavior that shall be provided by a function or service and will be further explored in section 3.2 . NFRQ usually refer to every system's requirement that is not considered a FRQ, and can be divided but not limited to System requirements, Technological requirements, Networking requirements, Quality of Service requirements, Legal requirements and Constraints (Rupp, 2020).

Here are presented numerous articles found in the literature that focus on different aspects of manufacturing scheduling. Contrary to traditional approaches that mostly use centralized manufacturing systems, underneath the SM environment, most of the components are smart, autonomous, and dynamic, leading to a more intelligent and decentralized manufacturing system. Consequently, a lot of data and different information need to be available in order to model and develop robust scheduling solutions.

One of the main findings is related to the innumerous different NFRQ that are considered among different solutions. A preliminary analysis of the selected literature allowed to identify some requirements that may be crucial in real scenarios but are not always considered in the literature. The list of those requirements is presented next, and from here it was possible to build Table 1 and evaluate which studies consider each of these requirements.

Among these data, there are a lot of constraints and requirements that need to be considered before developing a production scheduling solution. Even though these requirements may be different for each particular case, some of them can be transversal to several real manufacturing scenarios. Although it is not possible to consider all of them, next are presented some relevant constraints and requirements.

2.5.3.1 Dynamic Environments

In most real-world manufacturing environments disturbances may occur over time, this can be known has dynamic environments, and so it is important to be prepared to deal with them. This may mean that the optimal solution for the problem may also change. These disturbances may include the arrival of new jobs, cancellation of jobs, changing of processing times, or machines availability, since machinery may be subject to maintenance operations or incur in breakdowns (Framinan et al., 2014; X. Wang et al., 2022). Although the simplest way to cope with this problem may be to reschedule all the remaining jobs, as done by (Tran et al., 2019), it may be unpractical from a temporal point of view. So, a possible solution is to use the previous search space to improve the search after a change by incorporating or removing the jobs in the previous schedule without affecting the other tasks. However, in some cases, reschedule could be the better option.

2.5.3.2 Flexibility

A flexible manufacturing system is able to produce different products by sharing tools. More factories are adopting flexible machines, able to perform more than a unique task, thus a flexible scheduling should be adopted to those manufacturing systems. In a flexible scheduling an operation can be executed in more than one machine (routing flexibility) or each machine can be able to perform more than one operation by sharing resources (machine flexibility) (Karimi et al., 2016; H. Lei et al., 2017; P.-H. Lu et al., 2018).

2.5.3.3 Processing times variation

Most of the times, in literature, tasks processing times are considered to be static, i.e. they are known in advance and do not change along the way. However, in real manufacturing systems, due to the most diverse situations, they may vary, mostly to increase over time. This situation can happen due to resources deterioration, a fault in the setup, or because of the surrounding conditions (Chung & Kim, 2016; Zarook & Abedi, 2014). But, processing times may also be reduced, for example, by assigning more work force to a task, which means that those time variations can sometimes be controlled (Framinan et al., 2014).

2.5.3.4 Setup times

Setup times encompass all the operations that are performed on the machines but are not related to the production process directly. It includes, for example, add or remove product parts, calibration, machine cleaning, tests, etc. Those operation can occur both before and/or after the processing of the task. Most of the times these processes are not considered or are considered as part of the processing times of the product. However, in real manufacturing systems, setup times can be a substantial part of the production time, and so, it should be managed wisely (Framinan et al., 2014).

2.5.3.5 Maintenance

Although often ignored in scheduling studies, maintenance activities play a crucial role in manufacturing systems, since they are a constant in real environments, either to prevent/avoid or to correct/recover failures. Thus, maintenance activities are an important element to be considered when developing scheduling approaches, in order to have a more robust solution and achieve a better performance of the system (Kaplanoğlu, 2014; Q. Liu et al., 2018; Yingfeng Zhang et al., 2018).

2.5.3.6 Precedence constraints

Although precedence constraints have been studied for a few decades, it is still a very explored topic inside the research community. Even though in literature it is usually assumed that every job or task to be processed is independent of any other, it is not always the case, since some jobs may be intermediates to other jobs (Framinan et al., 2014; H. Xiong et al., 2022). Thus, in the cases where there are precedence constraints between jobs, the first task of a certain job cannot start before all the tasks of its predecessors are finalized, as in the case of the assembly of two or more parts (H. Xiong et al., 2017). So, these constraints need to be known in advance.

2.5.3.7 Preemption

In some cases, it can be necessary or desired that operations in jobs can be continued after a pause. This is known as preemption. On the other hand, when jobs cannot be interrupted, preemption is not allowed. Although preemption is rarely considered in the literature, in several scenarios, it may be needed, such as the arrival of new jobs with more importance than the ones being processed, which requires to stop the operations; it can be beneficial to continue an operation in another machine or another time; unexpected cancellations by the clients might also require to stop the production; breakdowns in the machines as mentioned previously (Framinan et al., 2014).

2.5.3.8 Release and Due Dates

Release dates (the time from when the job is available to be processed) and due dates (the time when the job must be completed) are other type of requirements or constraints when developing scheduling solutions, although they can also be considered as part of the objectives of the scheduling. Sometimes, however, it may not be possible to complete all the jobs in the time interval between the release and due dates, but these times should not be disregarded (Framinan et al., 2014). It is important to respect the dates, since the products need to be ready for delivery at some time and it could be crucial to not overcome those dates. On the other hand, it could be important to do not finish the products too soon as well, since it can lead to some wear in the parts or involve storage costs. Thus, when those dates are not respected, it may be necessary to apply penalties, both for early and late finishes (Kuhpfahl & Bierwirth, 2016; Yazdani et al., 2017).

2.5.3.9 Transportation

Product parts need to be moved inside the factory from one machine to another, or to the storage zone. This process can involve the transportations through conveyors, robotic arms, automated guided vehicles as many other solutions. Which means that, first the product will not be available immediately in the next machine and a time is required to transport it, and second the number of transporters is limited, and so they must by synchronized with the scheduling process along the chain or the parts need to wait for an available one. However, transportation times are not often considered in the scheduling problems found in the literature (Framinan et al., 2014).

2.5.3.10 Storage

Another constraint that is often ignored or is considered to be unlimited, in literature, is the storage. Products may need to be placed in storage, both during and/or at the end of the production process. Obviously, this space is not infinite and having full or poorly managed storage zones may imply additional problems in the production. Thus, storage buffers may be considered when developing scheduling approaches in order to have more realistic solutions and reduce unexpected problems (Framinan et al., 2014).

2.5.3.11 Distributed Factories

Although scheduling systems are mostly associated to the scheduling process within a factory, the scheduling process of supply chains has been evolved. This arises even more the complexity of the problem, since the products may be assigned to different factories, which may be distant

from each other, and the transportation across those facilities needs to be considered (Chang & Liu, 2017; P.-H. Lu et al., 2018).

2.5.3.12 Environmental Issues

During the last years there has been an increase in concern about the negative environmental impact caused by the manufacturing environments. Since the population is increasing it is quite natural that energy consumption increases as well in order to respond to demand for any type of goods. Nevertheless, to achieve a sustainable development in order to reduce gas emissions or acidification, it is crucial to reduce the energy demand (Gahm et al., 2016).

As stated by Gahm (Gahm et al., 2016) in their research, "none of the IPCC (Intergovernmental Panel for Climate Change) reports identify scheduling as either a method or an instrument to improve energy efficiency [...] scheduling is rarely considered as a suitable instrument to improve sustainability either in general or with respect to energy efficiency in particular". However, scheduling can be an important tool in order to reduce the environmental impact and achieve sustainability, since it can inform what the best steps to improve and reduce the energetic consumption and costs during the manufacturing process are, such as machines' consumption or materials utilization (Helu et al., 2016; Salido et al., 2017).

Based on this, the next sub-section will analyze the literature considering some important requirements and restrictions that are important to consider when designing scheduling approaches to be applied in manufacturing systems.

2.5.4 Existing Approaches

In this section are presented several solutions found in literature that focus on different aspects of manufacturing scheduling.

Contrary to traditional approaches that are mainly centralized manufacturing systems, underneath the SM environment, most of the components are smart, autonomous and dynamic, leading to a more smart and decentralized manufacturing system (Iwamura & Sugimura, 2010). Thus, a lot of data and different information needs to be available in order to model and develop robust scheduling solutions. In the Table 1 are considered some requirements that contribute to the scheduling solutions and whose are considered in the literature.

The following notation was used:

AC – assembly cost
ATCT – adjustment of total completion times
CJ – completed jobs
CW – cost of workers
E – earliness
EC – energy consumption
FS – flow shop
JS – job shop
LB – load balance
M – makespan

MAPE – mean absolute percentage error
MC – manufacturing cost
MDO – maximize delivery of orders
MeanET – Mean earliness and tardiness
MEP – maximize early production
MET – sum of maximum earliness and tardiness
MFT – mean flow time
MOO – minimize overdue orders
MSA – maximize system availability
MtC – maintenance cost
MW – material wastage
P – productivity
PM – parallel machine
PT – processing time
RM – resources management
S – storage
SCT – sum of completion times
SM – single machine
Stab - stability
T – tardiness
TDR – tardiness delivery rate
TET – total earliness and tardiness
TFT – total flow time
TWM – total weighted makespan
TWT – total weight tardiness

Table 1 - Literature review of scheduling solutions

Ref.	Shop type	Industry-oriented	Dynamic Events	Flexible	Distributed factories	Transport Time	Earliness	Tardiness	Processing Times Variation	Setup Times	Mainte-nance	Precedence	Preemption	Release Date	Due Date	Storage buffer	Objective
(W. C. Lee et al., 2016)	PM														✓		TWM
(Chou et al., 2013)	JS		✓	✓													M/EC
(P.-H. Lu et al., 2018)	JS			✓	✓	✓											M
(Jia et al., 2007)	JS			✓	✓	✓		✓									M/T/ MC
(Bürgy & Bülbül, 2018)	JS						✓	✓						✓	✓	✓	M/TWT/E /T/S
(Shen et al., 2018)	JS			✓						✓							M
(Shahrabi et al., 2017)	JS		✓							✓							M
(Yazdani et al., 2017)	JS						✓	✓							✓		MET
(Kuhpfahl & Bierwirth, 2016)	FS							✓							✓		TWT
(Chang & Liu, 2017)	JS			✓	✓	✓											M
(Kaplanoğlu, 2014)	SM	✓	✓			✓				✓	✓						M
(Yingfeng Zhang et al., 2018)	FS		✓					✓			✓	✓			✓		AC/RM/E C/TDR
(Q. Liu et al., 2018)	SM		✓					✓			✓				✓		MC/T

Ref.	Shop type	Industry-oriented	Dynamic Events	Flexible	Distributed factories	Transport Time	Earliness	Tardiness	Processing Times Variation	Setup Times	Mainte-nance	Precedence	Preemption	Release Date	Due Date	Storage buffer	Objective
(Zeng et al., 2018)	FS/PM	✓		✓						✓							M/EC/MW
(Mou et al., 2018)	FS								✓								M/ATCT/PT
(Alotaibi et al., 2016)	JS		✓	✓				✓		✓					✓		T/EC
(L. Zhang et al., 2013)	JS		✓										✓				M/Stab
(Sobeyko & Mönch, 2016)	JS			✓				✓						✓	✓		TWT
(Yoo & Lee, 2016)	PM										✓				✓		M/SCT/T
(Ladj et al., 2016)	SM			✓							✓						MtC
(Zarook & Abedi, 2014)	PM						✓	✓	✓		✓				✓		E/T/MtC
(Liu, Wang, Wang, Wang, & Zhang, 2018)	JS		✓		✓			✓						✓	✓	✓	M
(Nikolakis et al., 2018)	JS	✓	✓										✓				MFT
(Qu et al., 2016)	FS		✓	✓				✓	✓	✓						✓	MDO/MOO/P/CW
(Freitag & Hildebrandt, 2016)	JS		✓				✓	✓		✓				✓	✓		E/T/PT

Ref.	Shop type	Industry-oriented	Dynamic Events	Flexible	Distributed factories	Transport Time	Earliness	Tardiness	Processing Times Variation	Setup Times	Mainte-nance	Precedence	Preemption	Release Date	Due Date	Storage buffer	Objective
(Ivanov et al., 2016)	FS		✓	✓				✓					✓				T/CJ/LB
(H. Xiong et al., 2017)	JS		✓					✓						✓	✓		T
(Nie et al., 2013)	JS		✓	✓				✓						✓	✓		M/MFT/T
(Mokhtari & Hasani, 2017)	JS			✓							✓						M/MSA/EC
(Silva et al., 2018)	PM	✓	✓	✓						✓					✓		T
(Nguyen et al., 2014)	JS		✓					✓						✓	✓		M/TWT/MAPE
(Afzalirad & Shafipour, 2015)	PM									✓							M
(Asadzadeh, 2015)	JS																M
(Mokhtari & Noroozi, 2018)	FS						✓	✓							✓		MET
(Han et al., 2015)	JS			✓		✓										✓	M
(D. Lei et al., 2018)	FS			✓				✓							✓		EC/T
(Azami et al., 2018)	FS	✓										✓			✓	✓	MEP

Ref.	Shop type	Industry-oriented	Dynamic Events	Flexible	Distributed factories	Transport Time	Earliness	Tardiness	Processing Times Variation	Setup Times	Mainte-nance	Precedence	Preemption	Release Date	Due Date	Storage buffer	Objective
(K. Z. Gao et al., 2016)	JS			✓			✓	✓							✓		M/ MeanET
(Jamili, 2016)	JS		✓														M
(Ahmadi et al., 2016)	JS		✓	✓													M/Stab
(K. Gao et al., 2019)	JS	✓	✓	✓													Stab/M/TFT/LB
(Marzouki et al., 2017)	JS			✓													M
(Parsa et al., 2017)	SM						✓	✓							✓		TET
(Tang et al., 2016)	FS		✓	✓													M/EC
(Gao, Suganthan, Pan, Tasgetiren, & Sadollah, 2016)	JS		✓	✓													M/LB
(De Giovanni & Pezzella, 2010)	JS			✓	✓												M
(Zhao et al., 2014)	JS						✓	✓						✓	✓		M/TET
(Costa et al., 2016)	PM																M
(Petrović et al., 2016)	JS			✓		✓				✓							M/LB/MFT

Ref.	Shop type	Industry-oriented	Dynamic Events	Flexible	Distributed factories	Transport Time	Earliness	Tardiness	Processing Times Variation	Setup Times	Mainte-nance	Precedence	Preemption	Release Date	Due Date	Storage buffer	Objective
(Chung & Kim, 2016)	SM								✓								M
(Balin, 2011)	PM																M
(Karimi et al., 2016)	JS			✓		✓											M
(Kundakcı & Kulak, 2016)	JS		✓										✓				M
(Salido et al., 2017)	JS		✓														M/EC
(Jung et al., 2017)	FS			✓						✓							M
(Kai Zhou Gao, Suganthan, Pan, Chua, et al., 2016)	JS	✓		✓					✓								M/LB
(L. Wang et al., 2017)	JS			✓													M
(W. Xiong & Fu, 2015)	JS			✓													M
(Chan et al., 2011)	JS			✓						✓				✓	✓		M
(Helo et al., 2018)	FS		✓						✓						✓	✓	M
(Fu et al., 2018)	FS							✓	✓						✓		M
(Uhlmann et al., 2022)	FS		✓							✓							M

Ref.	Shop type	Industry-oriented	Dynamic Events	Flexible	Distributed factories	Transport Time	Earliness	Tardiness	Processing Times Variation	Setup Times	Mainte-nance	Precedence	Preemption	Release Date	Due Date	Storage buffer	Objective
(Yi Zhang et al., 2022)	JS		✓	✓				✓				✓		✓	✓	✓	M
(Souza et al., 2022)	JS		✓								✓	✓					M/Stab
(G. G. Wang et al., 2022)	JS		✓					✓	✓								M/T/EC
(R. Li et al., 2022)	JS		✓	✓					✓								M/LB
(Delorme et al., 2023)	JS		✓	✓					✓								M

(Chaudhry & Khan, 2016) stated in their research survey study (focused on articles between 1990-2014) that most of the work was focused on testing the developed algorithms on benchmark instances, and just a fraction of the research has been applied on practical problem solutions as compared to pure research. From the analysed articles during this work (Table 1), it is possible to observe that only less than 10% tried to solve the scheduling problem in real scenarios. The focus on algorithm development is of huge importance and can contribute greatly to solve real problems. However, real scenarios have an entire set of conditions and circumstances that are not considered when algorithms are developed in laboratory. The current problem in obtaining feasible solutions for SM scenarios is not related with technology by itself (Ribeiro & Bjorkman, 2018), as technology has evolved a lot during the past decades, it is related with managing all the actors and the connections between them, and use them in order to improve the industry. Thus, more effort should be dedicated to solving industry-oriented problems.

Besides that, most researchers focus on trying to minimize makespan, nevertheless, although some of them use benchmarks for testing and comparison, it is not clear which techniques are superior to other for a specific problem, although they can be better than other in specific points. These leads to solution that are good at beating benchmarks but are not able to be deployed in real manufacturing systems (Chaudhry & Khan, 2016; Framinan et al., 2014).

In addition to this, some problems arise when unrealistic assumptions are made, since manufacturing environments are very dynamic and do not rely on static features. These assumption can rapidly lead to unfeasible scheduling solutions which can be costly to the companies (Framinan et al., 2014). The most common assumptions are: all jobs and machines are available at time 0 and release times are not considered; each machine can only execute one specific operation, which is not always true since some machines have the flexibility to perform more than one operating, by changing tools for example; the processing time of an operation is unchangeable, however times may changing according to the conditions in a specific moment; machines never break and they are always available; setup times of any operation is sequence independent and it is included in its processing time, which is not always true and may compromise the entire schedule; preemption is not allowed; storage buffers are unlimited.

Aligned to this this problem, in (Framinan et al., 2014) is referred that "…the intersection among three or more sets of constraints for any shop layout would return an empty set of references most of the time". Which means that only a few researches contemplate more than three constraints in their studies. From Table 1, less than a third of the research studies

considered in the same approach more than three of the analysed requirements, as demonstrated in Figure 2.18.

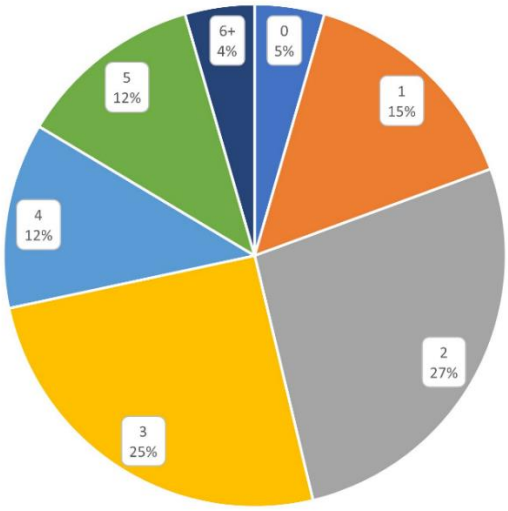

Figure 2.18 - Number of constraints considered per study

2.6 Gap analysis

Sometimes it can look like academic studies are going easy when trying to narrow the bridge with the industry, once the published studies are often too simple, vague and even with convenient restrictions and simplifications that do not reflect minimally the real industrial systems. This may even be true. But, on the other hand, both computational and real-world complexity can difficult the problems solving. Also, most of the times companies are not willing to provide information that can be valuable to develop better manufacturing systems. In these days, where data are becoming the core of industrial systems, it is crucial that data are provided to academia, so better solutions, not only for the shop-floor but to all value chain, may emerge. Nonetheless, academia should also make an effort to meet industry needs. Although there are plenty of scheduling studies in literature, the research dealing with real-world problems is very uncommon, even more while adopting reference architectures.

Thus, the main challenges, and likewise the main gaps, in manufacturing scheduling research for SM comes from the fact that most studies found in literature, as mentioned before, are based on assumptions related to the manufacturing environment that are, commonly, not true and even naïve in real manufacturing systems. Furthermore, in order to cope with more realistic scenarios, not only more constraints and requirements need to be considered, as they need to be considered simultaneously.

Based on the presented study, it is possible to identify several objectives and requirements, such as dynamic events, setup times, or release and due dates, and the tendency is for these to increase, which is the case of the human-related aspects. These are often overlooked in manufacturing scheduling solutions. However, there is a growing trend among companies and society to consider how SM solutions interact with and support human operators. this adds significant complexity to manufacturing scheduling studies because human behavior varies greatly. To address this challenge, interdisciplinary teams are necessary to bring together the expertise needed to explore these factors, which are often outside the scope of traditional manufacturing scheduling. Although this topic is vital for the future development of SM solutions, it is not covered in this dissertation.

Many manufacturers rely on traditional MES software to manage their production scheduling but often become frustrated when they still face order delays and cash flow issues. Some avoid implementing it altogether, believing the software cannot match the extensive hands-on knowledge their team has gained over the years. As a result, they resort to using whiteboards, spreadsheets, and other manual processes. The solution for manufacturers is not simply implementing better scheduling; it is about achieving the visibility needed on the shop floor to make informed decisions based on experience, knowledge, and real-time conditions once work begins. In this regard, seamless data and information sharing throughout operations are crucial. The status of all production resources, equipment, personnel, and materials—both direct and indirect—need to be known at all times and all data needs to be interconnected with proper context, enabling manufacturers to better plan their production lines and meet customer demands.

In the analyzed literature no study was found describing the components of manufacturing scheduling for reference architectures. Different approaches consider different constraints, and there is not a common and uniform way to develop scheduling solutions, even though there are common points in industrial systems that may be harmonized. Having a reference for scheduling designing and development can speed up the creation of scheduling solutions and make it easier to adapt these solutions to different scenarios. Knowing that scheduling can have a direct impact on production efficiency, sustainability and costs of manufacturing systems, it is of huge interest to have an in-depth work on how to model smart components to optimize scheduling approaches in SM systems. Production scheduling provides several advantages for manufacturing, including enhanced efficiency and productivity by optimizing the allocation of resources, reducing idle times, and minimizing production bottlenecks. It improves delivery performance and customer satisfaction by ensuring timely

completion of orders and better adherence to deadlines. Moreover, effective scheduling also aids in cost reduction by decreasing overtime, reducing inventory levels, and maximizing equipment utilization. Additionally, it provides better visibility and control over the production process, enabling quicker responses to changes in demand or unexpected disruptions, and fostering a more agile and resilient manufacturing environment.

RAMI 4.0, with its comprehensive and structured framework, can play a pivotal role in addressing these gaps. By providing a multi-dimensional model that integrates hierarchy levels, lifecycle stages, and functional layers, RAMI 4.0 facilitates the development of more realistic and holistic scheduling solutions. It promotes interoperability and data sharing across all aspects of manufacturing, ensuring that real-time conditions and human-related factors, such as operator fatigue and skill levels, may be incorporated into the scheduling process. This integration helps bridge the gap between theoretical models and practical applications, fostering interdisciplinary collaboration and enabling more sustainable, efficient, and human-centric manufacturing systems.

Throughout the document, the gap between RAMI4.0 and the development of scheduling systems is thoroughly analyzed. The focus is on identifying and addressing the lack of a standardized approach for aligning scheduling with the RAMI4.0 framework, providing a solution to bridge this gap.

3

FUNCTIONAL REQUIREMENTS AND DESIGN PRINCIPLES

In this section are identified the FRQs recognized as more relevant to this work, as well as are presented a set of Design Principles (DPs) to develop a scheduling system for manufacturing. Although NFRQs were already mentioned in the document, as they come directly from the literature analysis, the FRQs to implement a manufacturing scheduling solution are now proposed.

3.1 Toward a Unified Approach for Scheduling System Development in Manufacturing

As identified earlier, little attention has been given to providing comprehensive guidelines to develop manufacturing scheduling architectures, which impacts the system design and implementation, and may help decreasing the development process times and costs (Alemão et al., 2021). Moreover, some scheduling systems may have common requirements, since all of them must have a number of common functionalities. The reuse of a validated methodology will shorten the development cycle, ensure the main functionalities of the system are appropriately covered, and allows the re-utilization for forthcoming systems of part of the developed code, as far as the architecture is designed in terms of independent blocks or function-specific modules (Framinan & Ruiz, 2010). The generic and specific activities are illustrated in Figure 3.1, and both, generic and specific activities, should be considered when developing scheduling systems. Most of the requirements of a scheduling system are generic, with just a few

requirements being use-case specific. Even on the system design phase, part of the architecture may be generic, while some components are use-case specific.

Thus, it is possible to identify characteristics that are common to all systems, and it is on this aspect that this work will focus on - identifying and defining the requirements and DPs that allow for the development of any scheduling system for SM.

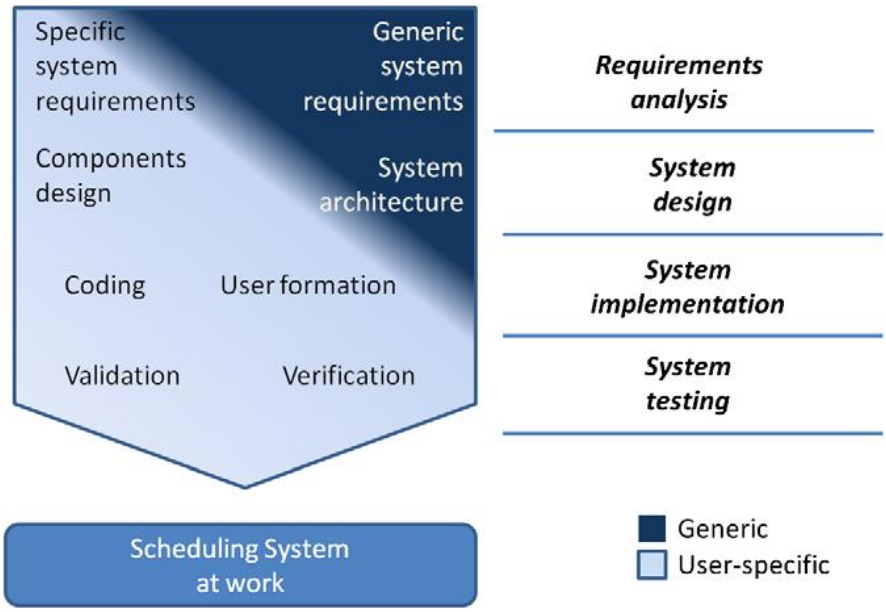

Figure 3.1 - Scheduling generic and specific activities in the development of a manufacturing scheduling system (Framinan & Ruiz, 2010).

3.2 Functional Requirements for Smart Manufacturing Scheduling

As stated before, requirements play a crucial role when designing and implementing any kind of systems, and it is not different for manufacturing scheduling. Here, the suggested FRQs for the development of a manufacturing scheduling system will be presented. The following keywords were adopted from (Rupp, 2020) as described to express the necessity of each requirement:

- "Shall" - Legally binding. It determines that it is mandatory to fulfill the requirement.
- "Should" - Not legally binding. It expresses an intention to implement the requirement under certain circumstances, but it does not need to be implemented.

The identified FRQs are based on real-world challenges that can be found in nowadays factories and are generic enough to be present in a lot of them. These requirements were

identified through the contact with several industry experts and the participation in some national and international projects, namely PERFoRM, GO0DMAN, AdAM, and KITT4SME, where it was possible to align the partners' needs with best practices for creating manufacturing systems. These requirements emerged during conversations and meetings with industrial partners, where the needs and challenges related to developing scheduling systems for SM were discussed. They are described as a tree format, where there is a main requirement that is subdivided into different ones. Then, each of the DPs identified in 3.3 is related with the FRQ previously defined. This is a simplification of the axiomatic DPs, where each FRQ is matched with one DP (Pourabbas et al., 2021). The process of defining FRQs is inherently hierarchical and operates in a tree-like structure. It begins with identifying a primary FRQ, which represents the overarching goal of the system. To ensure that this primary FRQ is met, it is necessary to decompose it into several subordinate FRQs, each one addressing a specific aspect required for the fulfillment of the higher-level requirement. This decomposition continues recursively, with each FRQ being broken down into more detailed and granular requirements, until all aspects of the system are sufficiently defined. This structured approach ensures that every requirement is systematically linked to the successful verification of its parent FRQ, ultimately leading to a comprehensive and coherent design methodology. The first proposed FRQ is:

- *FRQ1 Develop a dynamic scheduling system able to deal with several targets and specific production requirements in SM environments:* the main requirement identified in this work is to develop and provide a dynamic scheduling system that shall be able deal with several targets and requisites. These targets represent the production objectives, which may or not be defined a priori and may change during the process. On the other hand, the requisites define the working boundaries of the manufacturing system, necessary to achieve efficient production, that may change according to the circumstances.

Using FRQ1 as a reference, below, is a list of all the proposed requirements to design and develop manufacturing scheduling systems, in order to fulfill FRQ1:

- *FRQ1.1 The system shall be able to replicate the physical system into the logical system:* the system using recent technologies shall receive and interpret information regarding the physical system and based on that create a logical replica that is quickly accessible and ready to be tested with different purposes in a computational environment.

To achieve FRQ1.1, Two secondary requirements support the primary goal of FRQ1.1, by ensuring that the logical replica is both comprehensive and updated.

- *FRQ1.1.1 A factory topology replica shall always be held in the logical system:* independently of how the data are saved (ontology, datamodel, etc.) there shall always be a saved replica in the logical systems, which makes it easy to access data related to the shop-floor
- *FRQ1.1.2 The replica shall be updated every time the factory topology is changed:* this way it is guaranteed that there is always the most recent replica in the system and there is no missed information.

- *FRQ1.2 The system shall be able to interoperate with other systems:* the interoperability with other systems that may provide relevant information and knowledge for scheduling (ERP, MES, Data Analytics tools, etc.) will allow to develop more accurate and efficient solutions, that may be deployed faster.

To support this FRQ, the following three secondary requirements ensure the system's interoperability is thorough and helpful to the overall goal of efficient and accurate scheduling.

- *FRQ1.2.1 The system shall have the capability of integrating software tools independently of the technology in which those are developed:* The technology adopted should not be an obstacle to an easy and quick integration of different tools. In the current context of SM, it is crucial that different tools may be connected and exchange data between them.
- *FRQ1.2.2 The sharing of data between the different tools shall be harmonized:* In order to facilitate the exchange and interpretation of data, it is necessary to follow a common path, so the data sharing between tools shall be harmonized.
- *FRQ1.2.3 The tools to be developed shall have the ability to work collaboratively:* collaboration is a key requisite in current manufacturing paradigm in order to achieve success. Thus, it is essential that the newly developed tools have the ability to work collaboratively with other tools, therefore information among the system may be exchanged faster and more effectively.

- *FRQ1.3 The system shall be independent of the chosen KPIs and capable to deal with several combinations:* solutions not being KPI-dependent are of immense importance in smart systems where the surrounding conditions may change frequently. This will allow to deal with different objectives and restrictions in different times, giving more flexibility and adaptability to the scheduling system.

FRQ1.3 will be supported by the following FRQs, which will guarantee that the system's scheduling process is not only adaptive but also remains capable of delivering optimized solutions under varying conditions.

- *FRQ1.3.1 KPIs description shall be uniform for all of them:* In order to facilitate the system's understanding of KPIs, they need to be created uniformly, to avoid misinterpretations or reading errors.
- *FRQ1.3.2 The KPIs list shall be updated as the objectives are updated:* by updating the list of KPIs each time the objectives of the system change, it allows to always have the scheduling system working on up-to-date solutions.
- *FRQ1.3.3 The system shall allow the possibility to assign priorities to KPIs:* KPIs do not have all the same importance and impact in the system. With that in mind, it is important to differentiate and prioritize them to better execute optimization solutions.
- *FRQ1.3.4 The system shall allow to choose which KPIs to use in each moment:* During the production process, the user shall have the opportunity to choose which KPIs need to be considered in the next scheduling instance. This may vary according to the needs of the factory.

- *FRQ1.4 The system shall have the capability to perform the rescheduling of operations in real-time:* this means that, if there is the need to change plans during the execution, the system shall have the capability to adapt accordingly and suggest new schedules.

The following requirements were identified to support FRQ1.4 and ensure that the system is capable of detecting and responding to real-time changes to adapt dynamically to the changing conditions of a manufacturing environment.

- *FRQ1.4.1 Thresholds for deviations should be defined:* In order to perform rescheduling as necessary, the system needs to know when the rescheduling needs to be executed. Thus, it is essential to define boundaries that tell the system when the current schedule is not valid anymore.
- *FRQ1.4.2 The system shall have the ability to monitor its own performance in order to check for considerable deviations from the running scheduling*. After knowing the defined thresholds, the system needs, in a consistent manner, to check how it is behaving itself and if the current execution is still within the boundaries or not.

- *FRQ1.4.3 The system shall be able to notice whether rescheduling is necessary:* when new orders arrive to the system, and if they are urgent, the system shall be able to notice whether rescheduling is necessary (if there is a considerable deviation from execution) or whether orders can be inserted into the current scheduling. The arrival of new information may or may not affect the current production.
- *FRQ1.4.4 When there are considerable deviations, the system shall be able to automatically recalculate a new schedule*. This will limit human intervention and, consequently, improve the overall performance of the system, leading to faster deliveries of finished products and expenses reduction.

- *FRQ1.5 The system shall provide human-machine interfaces that allow a seamless interaction with the human operator*. The interaction between the human and the system needs to be easy and practical for any operator, and do not consume more time than necessary for the human operator.

To achieve FRQ1.5, the following requirements have been defined, thus human-machine interfaces are not only functional but also enhance the overall efficiency and effectiveness of the system, by creating a balanced interaction, where human feedback is integrated into the system's operation.

- *FRQ1.5.1 The application of new schedules at the factory shall be validated:* the new schedules shall be validated by the production manager (or equivalent) before being deployed on the production line. After each new scheduling suggestion, the production manager will validate the schedule to make sure that it fits the current needs.
- *FRQ1.5.2 The user should have the possibility to ask for new schedules at any time*. If the user finds it necessary, he/she may ask the system to suggest new schedules.
- *FRQ1.5.3 The Graphical User Interface (GUI) shall provide the choice of priorities and use of KPIs, as well as adding and removing KPIs*. When an operator decides to ask for a new schedule, he/she needs to define what are the priorities for that particular schedule and what KPIs are supposed to be met. Additionally, it is important that new KPIs may be added, or old ones removed, according to the process evolution.
- *FRQ1.5.4 The system shall allow interaction with the operator to receive feedback from the executions*. Thus, it is crucial that the human operators may give feedback about previous executions, which will allow to improve future suggestions.

- *FRQ1.6 The system should have the capability to use different optimization approaches that allow the best system's performance*. By allowing several optimization techniques, the system may find what is the best approach for different scenarios.

Finally, to support FRQ1.6, three dependent requirements were defined to make sure that the system is equipped with different optimization techniques tailored to different scenarios and objectives. By enabling multi-objective optimization, integrating comprehensive data sources, and grounding its processes in the current state of the physical system, the system can deliver highly effective and adaptable scheduling solutions.

  - *FRQ1.6.1 The system shall be able to optimize the scheduling process simultaneously for different KPIs and targets*. This means that the system will be able to focus on different indicators that may be requested and not only on one specific factor.
  - *FRQ1.6.2 The system should make use of data from both production line and external systems in order to provide more precise and robust scheduling solutions*. By receiving feedback from internal and external sources the system may adapt accordingly and improve overall performance.
  - *FRQ1.6.3 The scheduling starting point shall be the current state of the physical system*. Thus, the system is not supposed to make assumptions regarding scheduling process but shall consult the present state to use the correct necessary data.

These requirements are important to build a manufacturing scheduling system, although such systems may not be limited to the ones identified in this work. Nevertheless, the author believes that an important step is being given by presenting these requirements in order to achieve harmonization and standardization in scheduling systems designing and development for manufacturing. These are also the baseline for the definition of the DPs explored in the next sub-section.

Figure 3.2 summarizes the proposed functional requirements. They are distributed by three different levels and all of them derive from the first requirement, FRQ1. More requirements can be identified, however they stop being generic and start being specific for the desired solution.

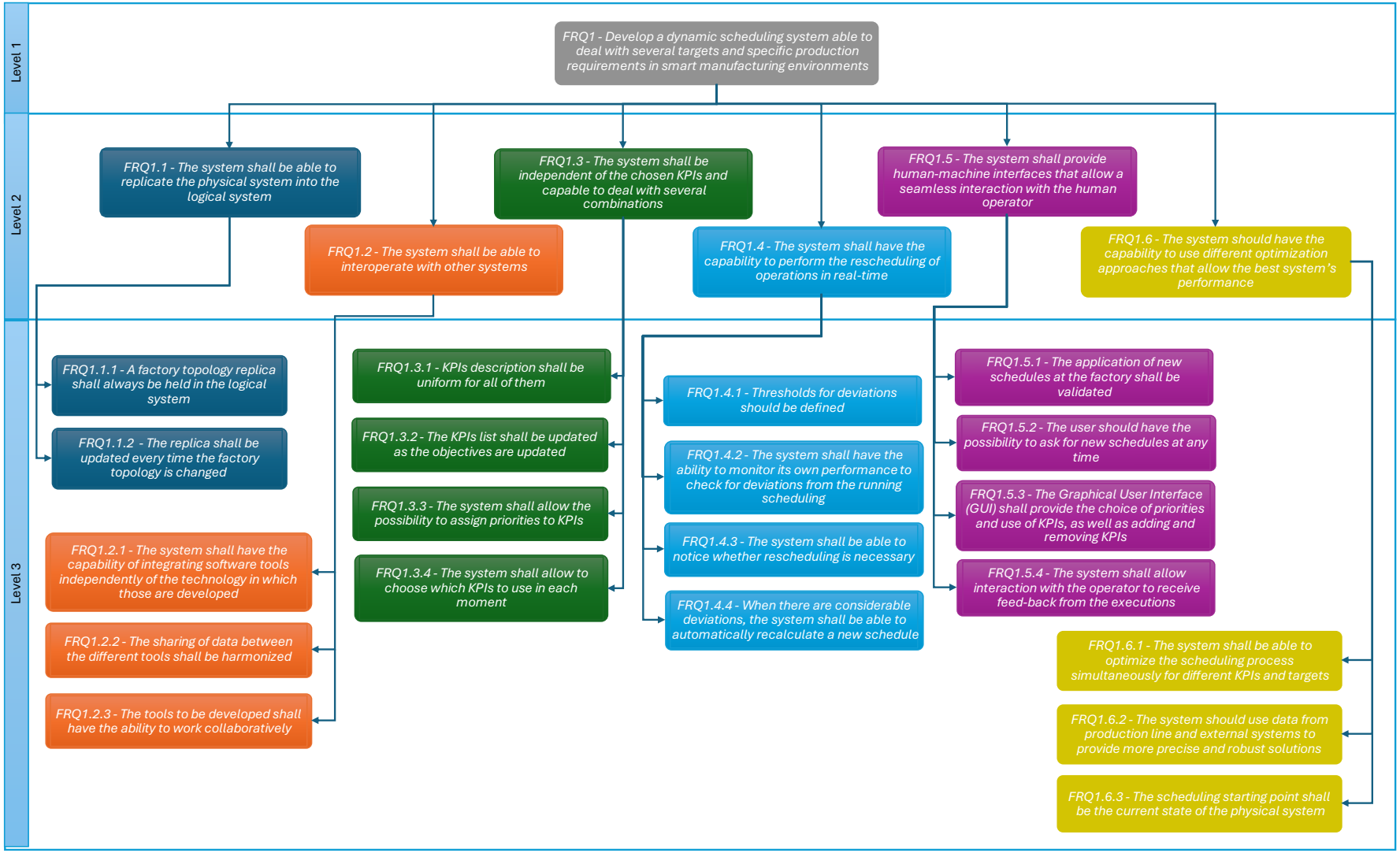

Figure 3.2 - Tree of functional requirements

3.3 Design Principles for Smart Manufacturing Scheduling

This sub-section presents the DPs to support practitioners in developing scheduling systems for manufacturing environments based on the FRQs.

The identified DP are the second step of the adaptation of the axiomatic DPs where each FRQ of the lowest level is matched with one DP (Pourabbas et al., 2021). Thus, each of the presented DP is related with the FRQ previously defined. The identified DPs will give an answer to each FRQ of the lowest level of the tree, thus ensuring that the FRQs of the upper levels are verified.

The first group of DPs refers to replication of the physical system into a cyber system, making it able to easily communicate with other systems and strive in SM environments.

- *DP1.1.1 Digital factory replica:* A replica of the factory may be held in the logical system through the adoption of AAS, which is the Digital Twin applied to I4.0. The AAS transforms the physical or intangible asset (entity) into an I4.0 Component, which is the way to describe thoroughly the properties of a CPS. By other words, the AAS is the digital representation of the real asset, containing all of its information and technical functionalities, and administering communications with other I4.0 Components. It is included in the Integration layer of RAMI 4.0.
- *DP1.1.2 Communication protocols:* In order to achieve this updatability, two factors must be considered. First, it needs to be defined how the data is accessed. For this, RAMI 4.0 suggests standard communication protocols to link the Integration and Information layers, such as HTTP, TCP-IP, MQTT, and OPC-UA protocols, which may depend on the device type and its capability to send data. Second, it is necessary to identify which parts of the asset will be available digitally and, consequently, what data will be provided by the asset and stored in the Information layer. The Information layer will include the several data models that may describe the asset. By adopting generic data models to describe the assets it will be possible to the digital factory updated.

The second set refers to the interoperation between the scheduling system and other systems relevant to it, such as ERP or MES. Seamless interoperability allows to develop more efficient systems.

- *DP1.2.1 Common interfaces:* In the context of I4.0, an asset may be any entity, from a hardware device to a software application. Thus, the AAS will allow the integration of the asset in the digital world, which can be done through HMI, switches, or sensor readers. According to RAMI 4.0, the communication between different tools shall be done through data mapping to XML, JSON, OPC UA, AML, RDF. Nowadays, there are so many technologies being adopted to develop different or similar tools that it is vital to keep them interacting smoothly, thus the adoption of common interfaces will allow tools to interact with each other without being reprogramed each time a new tool is connected. In this way, communication between different tools will be facilitated and technology independent.
- *DP1.2.2 Data model:* The transferring of data shall be harmonized through the use of a data model or ontology. The Information layer of RAMI 4.0 comprises not only the technical data of the assets, but also semantics as a common language, which means that the data models will be included in this layer. Consequently, by defining common data models or ontologies, the exchange of information will be harmonized, leading to faster and easier communication between different tools. As examples of data exchange formats recommended in RAMI 4.0 there are XML, JSON, or RDF (Bader et al., 2020).
- *DP1.2.3 Uniform data among components:* The adoption of standard communication protocols, as mentioned in DP1.1.2, and common I4.0- compliant interfaces and data models, will facilitate the collaboration between tools of different manufacturers. Thus, the infrastructure shall be designed to facilitate the integration between tools, which means tools shall be designed in a way to work together so they can provide functionalities that would not be able to by themselves. Consequently, data sharing and using needs to be uniformed between the different components.

Next are presented the DPs relevant for developing KPI-independent scheduling solutions. Different systems have different objectives and constrains, thus a generic scheduling solution cannot be dependent on what the KPIs are.

- *DP1.3.1 Harmonized KPIs description:* KPIs description need to follow the same rules of creation in order to be uniform. This may be realized by the definition of several fields that need to be fulfilled, when creating the data model, either by adopting XML, AML, JSON, or other standard format. The KPIs description shall be part of the Information layer.

- *DP1.3.2 Save production objectives:* Save system's objectives in Information layer (data model). KPIs can be updated by the user, that should have their own digital replica for user. Then, it will be easy to establish the communication between both parts through standard communication protocols. KPIs must be saved in a generic data model, which will facilitate the update by the user during the process, if it is necessary to update the production objectives. These objectives may vary and comprise, for example, energetic consumption, waste reduction or throughput increase.
- *DP1.3.3 Editable KPIs priorities:* An editable field, at the moment of KPI's creation or during the process, needs to be available in order to assign or change KPI's priorities. This data will be stored in the Information layer and updated if there are changes during the process.
- *DP1.3.4 Considering different KPIs for schedule generation:* During the production process, and before starting a new schedule execution, the system shall provide the possibility to choose (through the GUI) which KPIs will be considered in the next execution, since not all of them may be relevant at each time. The user may then choose which KPIs are relevant for the next scheduling instantiation. This can be done through selection boxes, for instance.

The fourth group of DPs is related to the capability of the system to execute a rescheduling in real-time, if there is a plan changing during the production execution.

- *DP1.4.1 Define boundaries:* Boundaries need to be specified when the KPIs/restrictions are defined. Thus, maximum and minimum levels of acceptability need to be identified. These boundaries shall be defined when the KPIs are created, i.e. when the associated asset is instantiated, and stored in the Information layer of RAMI 4.0, which is where the data models can be found. For example, if a working center can only manufacture products of type X, it shall be specified during the instantiation of the digital replica of that working center. On the other hand, if the energy consumption of a work center must be below some threshold, the scheduling system may consider that KPI when allocating production tasks.
- *DP1.4.2 Evaluate production process evolution:* Through the acquisition of real-time data from the shop-floor, the system needs to evaluate how the production process is evolving. This data can be acquired from working stations, human workers, digital readers, or other sources of information that are able to communicate with the digital world. This process can be done in different ways, for example, periodically (where the

information is being verified in regular time intervals) or event-driven (where, each time there is a new event, i.e. a change in the system's state, that information is communicated to the higher layers). Then, an algorithm shall perform the calculations in order to verify if there are deviations that affect the current scheduling execution and, if necessary, suggest new alternatives.

- *DP1.4.3 Analyze new orders:* An algorithm should be implemented to verify if it is possible to fit the new order within the current schedule without going over the thresholds, for instance by placing new orders in idle time positions, or if it is necessary to perform a reschedule and change the actual production process.
- *DP1.4.4 Automatic rescheduling:* An algorithm should be implemented to automatically perform a rescheduling when there are significant deviations regarding the original schedule. It may or may not be the same algorithm that performs the initial scheduling, depending on the necessary conditions for each particular case.

A human-machine interface should be developed in order to ease the process between operators and the shop-floor. This HMI will make the link between the asset (human worker) and the digital world. The interface should be a friendly and intuitive GUI and contain all the necessary features for the worker to operate with a scheduling system.

- *DP1.5.1 Schedule validation:* The GUI should allow the production manager to analyze and validate the suggested schedule. Preferably, the interface will show all the important metrics that the responsible worker needs to analyze and, eventually, a graphical output of the schedule suggestion, such as a Gantt chart. Then, the user will be able to accept, refuse or ask for a new solution.
- *DP1.5.2 New schedules request:* The GUI should allow users to ask for new schedules in a simple way. Thus, new schedules may be generated not only when deviations occur, but also when the production manager considers it is necessary. As mentioned in DP1.3.4, this GUI should display the available KPIs to choose for the next schedule.
- *DP1.5.3 Choose KPIs:* The GUI needs to contain checkable boxes that allow to quickly choose which KPIs must be taken into account in the next scheduling generation, as well as which are the priorities for each KPI. Additionally, the GUI should allow the addition and removal of KPIs from the system.
- *DP1.5.4 Human worker inputs:* The GUI should support inputs from human workers, so feedback about previous executions may be registered and further analyzed. This can be done by allowing text insertion, which needs to be analyzed more carefully, or having

predefined options that can be selected in order to identify recurrent situations. This may comprise, for example, production, maintenance, or process details.

Finally, in order for the system to have the capability to use different optimization techniques to find the best approach for different scenarios are presented the next DPs.

- *DP1.6.1 Implementation of different optimization techniques:* Different optimization techniques should be implemented in order to reach a better performance of the system and, at the same time, allow to deal with multi-objectives problems. Heuristic optimization techniques are usually not optimal but more flexible and faster to find solutions. May be considered Genetic Algorithms (Alemão, Parreira-Rocha, et al., 2019), Particle Swarm Optimization (Jamrus et al., 2018), Ant Colony Optimization (Tran et al., 2019), Multi-agent Systems (X. Wang et al., 2022), among many other approaches.
- *DP1.6.2 Communication with external systems:* The scheduling system should be able to communicate with external systems to collect the necessary data to be used in the scheduling process. Thus, it will be able to adapt to the challenges in real-time. This can be done through the use of common interfaces and data models (DP1.2.3). Moreover, I4.0-compliant tools shall be wrapped in an AAS, hence the scheduling tool will be able to communicate easily with those tools.
- *DP1.6.3 Start schedule not before the current state:* Unless otherwise stated, the starting point for a new scheduling generation shall be the current state of the physical system, but never before the current state. Exceptions may occur if the schedule is supposed to start in a posterior defined date. Consequently, the scheduling system needs to be up to date.

The analysis presented in Figure 3.3 aims to identify where the application of the proposed DPs can have the most significant impact on the various layers of the RAMI 4.0 layers axis. It is important to clarify that the DPs do not directly fit as specific components or implementations within the RAMI 4.0 layers. Instead, the DPs serve as guidelines that steer the development and implementation of systems aligned with the RAMI 4.0 reference architecture. By conducting this mapping exercise, the goal was to highlight the areas where adopting the DPs could positively influence the implementation of solutions consistent with RAMI 4.0. Although the DPs do not correspond directly to the RAMI layers, their application can be crucial in ensuring that emerging practices and technologies are effectively integrated. The table presented reflects an interpretation of how the DPs might impact different aspects of system development within the RAMI 4.0 framework, with the understanding that they provide strategic

guidance for achieving a robust and interoperable implementation aligned with the fundamental principles of I4.0. Note that this work does not intend to depict business processes and value chain interactions, as so, the business layer is not covered in this document.

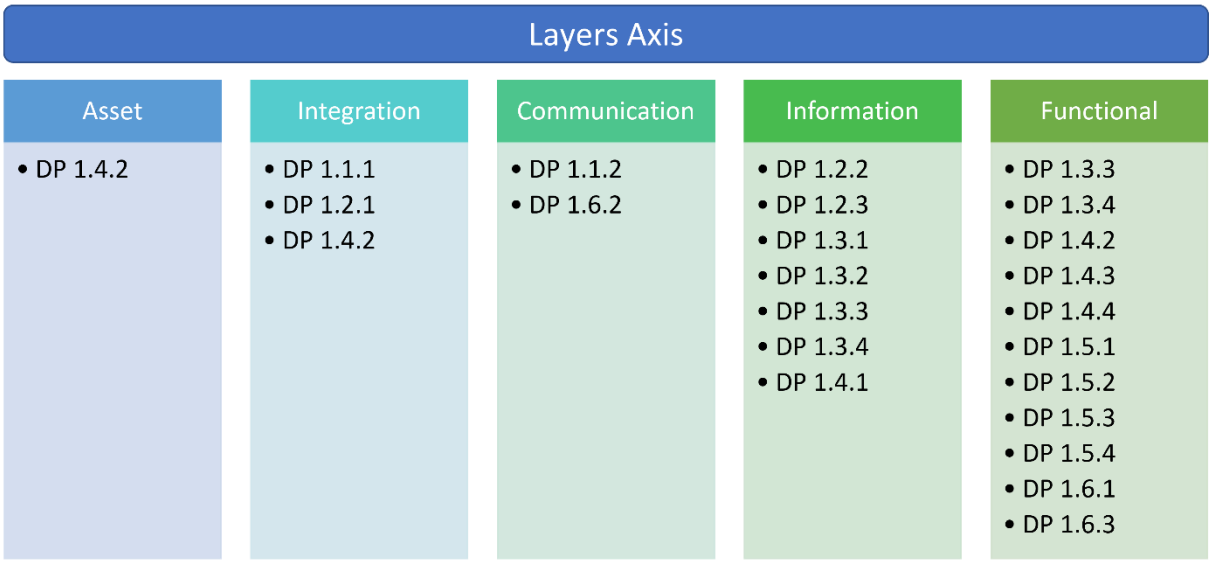

Figure 3.3 - Identification of design principles in each layer of RAMI 4.0.

Table 2 illustrates the impact of each DP on the design and development of generic scheduling solutions for SM environments, based on the extent to which each DP should be considered. It is important to note that the impact of each DP may vary depending on specific use cases, and the degree of importance might shift accordingly. However, for the purposes of this analysis, the impact levels are categorized as follows:

- High: indicating that the solution may not be successfully implemented without the inclusion of these principles, as they are critical to the system's success.
- Medium: representing an intermediate level of importance, where the DP is not critical but still plays a significant role in ensuring the effectiveness and efficiency of the solution.
- Low: meaning that while the solution could still be developed without the implementation of these principles, incorporating them could facilitate the development process and improve usability.

This categorization provides an understanding on how each DP influences the overall development process, allowing developers to prioritize their efforts according to the relative importance of each principle in the context of SM scheduling solutions.

Table 2 - Impact of each design principle for smart manufacturing scheduling (Low/Medium/High)

Design Principle	Impact for scheduling	Design Principle	Impact for scheduling
DP 1.1.1	High	DP 1.4.2	High
DP 1.1.2	High	DP 1.4.3	High
DP 1.2.1	High	DP 1.4.4	Medium
DP 1.2.2	High	DP 1.5.1	Medium
DP 1.2.3	Medium	DP 1.5.2	Low
DP 1.3.1	High	DP 1.5.3	Low
DP 1.3.2	High	DP 1.5.4	Low
DP 1.3.3	Medium	DP 1.6.1	High
DP 1.3.4	Low	DP 1.6.2	Medium
DP 1.4.1	High	DP 1.6.3	Low

Based on the identified DPs, in the next section will be presented the methodology for the development of manufacturing scheduling systems.

4

SCHEDULING METHODOLOGY

This section describes the methodology to support the design and development of manufacturing scheduling systems. Here is proposed part of the generic manufacturing scheduling system in order to facilitate the use and adoption of this solution to more complex solutions. For this, are identified the main assets composing the system, that are common to any kind of system of this nature, as well as a suggestion for the most common and important parameters of each one. Then it is provided a data model for supporting the implementation of the system, and finally the DPs identified in the previous chapter are connected to the proposed methodology.

4.1 Generic Smart Manufacturing Scheduling Methodology

As stated by Cavalieri and Salafia, "standards are the pillars of interoperability in the context of Industry 4.0 because their adoption creates the basis for the interworking between partners of a value-chain network" (Cavalieri & Salafia, 2020). So, here is presented a methodology comprising the first steps into standardization and homogenization of the way data are represented and structured in scheduling solutions.

The solution is based on AAS and RAMI4.0 best practices. An essential difference between existing solutions and the AAS concept is the manufacturer-independent standardization of the management shell meta model. In the domain of manufacturing scheduling there are currently no standards in this respect that provide such a self-description of systems and devices in a technology-neutral and manufacturer-independent way. Although a DT, as a digital representation of an entity, is sufficient to meet the requirements of a set of use cases, the lack of interoperability between the DTs of different companies and between the DT and the hardware it is abstracting hinders use cases that require information exchange between different organizations (Platenius-Mohr et al., 2020). The adoption of AAS will bring benefits such as time and cost savings on design and production phases, simple data transfer throughout the

product life cycle, manufacturer independency, or integration of the cross-company value chain.

Interoperability can be studied from different aspects, such as technical interoperability regarding communication protocols and data formats, semantic interoperability concerning ontologies and standards, business interoperability about policy and governance, human interoperability regarding user interfaces, and many more. In this work, it is focused on the interoperability of CPS for data formats and information exchange. Thus, the focus of this work is on information models (instead of mathematical, analytical, and simulation models).

A methodology for implementing SM scheduling solutions based on RAMI4.0, specifically for the generic activities is proposed. This methodology is based on the NFRQs identified in 2.5.3. Thus, this work will contribute with the guidelines to design a scheduling system in the context of SM, focusing on the main aspects of RAMI4.0. Consequently, the following aspects will be considered:

- Architectural layers: focusing on Functional, Information, Communication and Integration;
- Hierarchy Levels: focusing on Station field;
- Lifecycle & Value Stream: the guidelines for manufacturing scheduling will be concentrated in the Development stage of Type stage.

Hereafter, the term resource will be used when referring to any kind of resource, such as machinery, human labor, or working tools; station will be used to identify, specifically, machinery in the system; while work center will be used to refer to a place where can be several resources. Thus, it directs this work more towards RAMI4.0 definitions.

The smart scheduling system represented in the layers of the RAMI 4.0 framework consists of building a generic foundation for developing manufacturing scheduling systems for SM, interconnecting the most high-level functions of scheduling to the shop-floor where are the physical assets that will execute the schedule.

As referred before, in Industry 4.0, an asset refers to a physical or virtual entity that requires a connection for the I4.0 solution, including tangible entities such as machines, components, materials, products, or even intangible entities such as software systems, diagrams, data, contracts, and production orders (Platform Industrie 4.0, 2022; Ye & Hong, 2019). All these different assets must be identifiable, i.e., other devices and systems within the I4.0 network must be able to read and understand the asset's type, operational and technical data, status, and other specific information, which can be achieved by transforming assets in I4.0

Components, by adopting the AAS. To design and create an AAS it is necessary to follow some generic steps, according to RAMI 4.0 reference architecture (Dr. Heinz et al., 2019; Platform Industrie 4.0, 2020, 2022):

1. Define the AAS purpose and structure: The AAS structure outlines the metadata required to describe the properties, capabilities, and relationships of an asset. It includes the AAS ID, the AAS version, and the AAS components, such as the Asset, Submodel, and Concept Description.
2. Modeling the asset information: The asset information provides a description of the asset's physical and functional properties (such as its name, type, and location), capabilities (the services it provides and the data it produces), and behavior. A standard data modeling language, such as OPC-UA, UML, or RDF/OWL, must be used to model this data. It is possible to use existing data models or create one that comprises the specific asset requirements.
3. Create the AAS instance: After defining the AAS structure and modeling the asset information, the AAS instance can be created by instantiating the AAS components, assigning them the relevant data and information, and integrating the AAS with other components (this can be done using software development kits (SDKs) such as the AAS Java SDK or BaSyx.)
4. Register and publish the AAS: Subsequently, it is necessary to register it in a central AAS directory, such as an AAS Marketplace. Thus, other systems and stakeholders can find and access the AAS, and the asset information related to it in the AAS directory. To enable interoperability among different systems and applications, it is possible to publish the AAS instance in several formats, such as XML or JSON.
5. Test and deploy: Finally, it is necessary to test the communication protocols, verifying that the AAS is able to manage the asset information, and deploying the AAS in the manufacturing system, which includes integrating it with other systems, training users on how to use the AAS and provide support.

The first step is to define the AAS kind. It is the most important step in the design phase. The AAS kind defines the type of asset one wants to create an AAS instance for. It is necessary to specify the asset's properties, behaviors, and relationships with other assets. After defining the AAS kind and modeling the asset information, it is possible to create an instance of the AAS by instantiating it with specific values for its properties. It involves creating a digital representation of the physical asset that includes the relevant information about its configuration, status, and operational data. Then, it is necessary to register the AAS in a registry service that

provides a centralized location for storing and accessing AAS instances. Finally, it is necessary to interconnect the AAS instance with other assets in the manufacturing environment, by defining the communication protocols and interfaces necessary to access and control the AAS. Once the AAS is deployed, it can be used to monitor the status of the machines, products, or software on the production line, track production data, and perform predictive maintenance, among others. For example, if the AAS detects that its asset, e.g. a welding machine, is running at a higher temperature than usual, it can trigger a maintenance request to prevent a potential breakdown.

The proposed methodology's main goal is to ease the asset modelling, targeting scheduling systems designing and development for SM environments. Therefore, the methodology targets not only the acquisition of data at different granularity levels, as well as the asset modelling and asset allocation optimization through the execution of scheduling approaches. This optimization process outputs possible scheduling solutions to be deployed in the factory. With the help of a monitoring module, it is possible to get feedback from the shop-floor in real-time and perform data analytics techniques to monitor critical parameters and trigger self-adjustment mechanisms, if necessary. Thus, the methodology relays on a framework composed by three fundamental parts, as demonstrated in Figure 4.1.

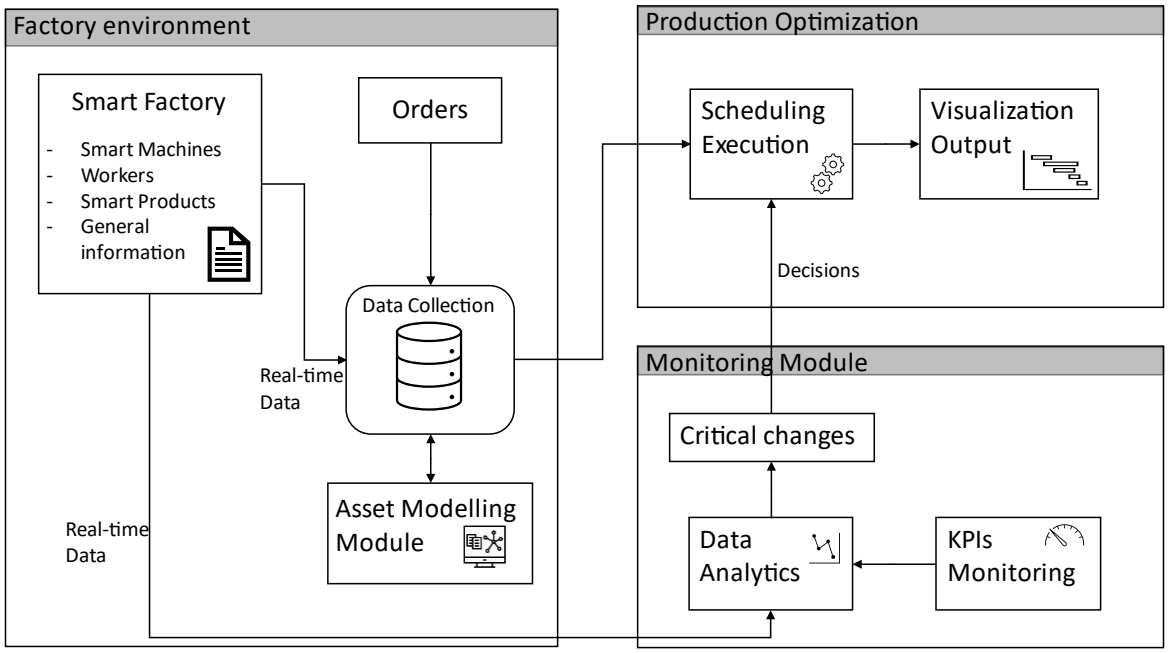

Figure 4.1 -Manufacturing scheduling generic framework

The factory environment represents the assets in the system and the integration between the shop-floor and the software components, that will support the scheduling system. Thus, when the system receives an order to execute a new schedule, it will redirect to the scheduling

tool, which will then execute an algorithm and provide an output with the result. At the same time, the system does not only send the output orders to the shop-floor for execution, as it is receiving data from the shop-floor, and providing that data to the monitoring module. If some critical changes are detected, new decisions may be made in order to update the schedule solution.

- Physical and software components integration - The cyber-physical connection is a driving force for SM environments that want to stay competitive in modern market conditions. The emergence of CPPS is allowing different entities in the shop-floor to be virtualized and connected together, which in turn facilitates the real-time monitoring of the shop-floor components as well as the controlling of those components in the production line. Consequently, it is possible to integrate external tools regarding supply chain and data analysis with the shop-floor. Based on the concept of RAMI's I4.0 Component, this part of the methodology aims to connect the different hardware and software entities among the system. Namely, within the factory it is important to represent, virtually, entities such as machines and working stations, parts and products going through the factory, human workers, and general information about the factory as working hours, shifts, and any other relevant information. Furthermore, it is crucial to register the orders coming from a higher level. All these assets, either physical or not, must be modeled, so they can be represented in a uniform and standard way, allowing an easy understanding and integration with other elements and even external assets. To conclude, all the previous information can be stored in a database, where it can be easily accessed by any module, such as production optimization and shop-floor monitoring.
- Production optimization - This is a core part of this methodology. Here is where the production scheduling is performed. The scheduling tool needs to gather all the necessary data from the shop-floor in order to allocate the orders accordingly. Here different approaches may be adopted, although the final goal is to optimize the production processes in the most convenient and needed way. This can be to reduce the overall execution time, reduce average idle the of working stations, balancing human workers effort, finish the products just in time, or any other relevant metric for each specific use case. In order to achieve these objectives, optimization techniques need to be adopted, which will be completely independent from the framework itself, and use case-oriented. Last, the final result should be delivered, to whom it may concern, in an easy-to-understand way, preferably through a visual output.
- Monitoring - The monitoring module will allow to monitor in real-time what is happening in the factory. The emergence of smart working stations and sensors in recent years

has allowed more data collected from the shop-floor. The adoption of data analytics techniques will allow to understand these data and turn them into useful knowledge for the company. By defining KPIs to trace the working conditions, it is possible to verify how well the objectives are being achieved. If critical deviations are verified, feedback may be sent to the scheduling execution in order to adapt accordingly.

In summary, the figure represents a comprehensive manufacturing scheduling framework within the context of Industry 4.0, illustrating the integration of key components and modules essential for modelling the system. At the heart of the system is the Smart Factory, which encompasses machines, workers, products, and the general information needed for operation. The Production Optimization module includes a component for executing scheduling tasks, and providing visualization outputs, such as Gantt charts, to monitor progress. The Monitoring Module plays a crucial role in tracking critical changes, performing data analytics, and monitoring KPIs to ensure the production process runs efficiently. Additionally, the Asset Modelling Module maintains digital representations of factory assets, which are essential for simulations and continuous optimization. Together, these components create a robust framework for smart manufacturing, enabling efficient scheduling and real-time responsiveness to dynamic production conditions.

To bring this entire complex ecosystem to life, a structured methodology is proposed to guide the development process. Given the intricate nature of integrating multiple components and ensuring seamless operation, this methodology serves as a roadmap for implementation. Together, these elements provide a comprehensive approach to successfully managing and optimizing the smart manufacturing environment. The proposed methodology to develop a manufacturing scheduling solution involves:

1. Define the industrial scenario: Although different scenarios may share some components, it is crucial to define the problem to be solved and the context in which it occurs. By doing so, it will be simpler to create a reliable and effective scheduling solution. It provides a thorough understanding of the limitations imposed on the production process. Information on a range of variables, including machine capabilities, production volumes, raw material availability, and human workers availability, is needed for manufacturing scheduling solutions. Depending on the industrial scenario, these variables can vary significantly. Additionally, when it comes to scheduling, different industries have different needs and goals. For instance, maximizing the use of machinery may be an aim in the automotive industry, whereas ensuring food safety may be a top priority in the food industry. The scheduling solution can be tailored to match the unique requirements and constraints of an industrial scenario by precisely specifying it. As a

result, there is a rise in production, decreased downtime, enhanced efficiency, and eventually, more profitability.

2. Identify assets: Physical assets can include machines, tools, equipment, and personnel, and each of them has unique capabilities, limitations, and availability. On the other hand, digital assets, such as MES, ERP, or Scheduling and Data Analytics tools, are equally relevant, as they provide further information about the shop-floor and production processes, and support in system's optimization and decision making, by identifying the assets present in the system, scheduling solutions can determine the production capacity of each asset and the optimal sequence of operations. This information can be used to minimize idle time and increase throughput, leading to improved efficiency and profitability.
3. Describe assets: In manufacturing scheduling solutions, describing assets is crucial since it aids in precisely determining the availability and utilization of stations required for production. Additionally, physical asset information is essential for maintenance scheduling. By tracking asset usage and maintenance requirements, scheduling solutions can plan for preventative maintenance, reducing the likelihood of equipment failure and unplanned downtime. Furthermore, asset information can aid in inventory management. By understanding the capacity and constraints of each physical asset and integrating it with relevant information gathered from digital assets, scheduling solutions can optimize material flow, reduce waste, and ensure that there are sufficient materials to support production.
4. Define objectives and KPIs: it helps to establish clear priorities and objectives that can guide the scheduling process. Making effective decisions on complex manufacturing scheduling solutions that usually involve many variables and constraints can be challenging without a clear understanding of what is important. Different manufacturing environments have different priorities, and what is important for one may not be as critical for another. For example, in a high-volume production environment, reducing setup time and increasing machine utilization may be the top priorities, while in a low-volume environment, the focus may be on minimizing changeover time and ensuring product quality. Scheduling solutions can be customized to match certain needs and goals by determining what is important. This can help to ensure that the scheduling solution is aligned with the overall goals of the organization and can help to optimize production efficiency, reduce downtime, and increase profitability. Moreover, identifying what is important can help to establish performance metrics and KPIs that can be used to measure the success of the scheduling solution. This can help to identify areas for improvement and refine the scheduling process over time.
5. Modelling assets: The assets identified in manufacturing scheduling solutions should be modeled according to RAMI 4.0, since it provides a standardized approach to describing and integrating the different components of a manufacturing system. RAMI 4.0 is a framework that defines a layered architecture for I4.0, with each layer representing a different aspect of the manufacturing process. RAMI 4.0 provides a structured way to model the assets, including their interfaces and interactions, which can help to

ensure interoperability and compatibility between different systems and components. By modeling assets according to RAMI 4.0, it becomes easier to integrate them into a larger system and to manage and monitor their performance. Furthermore, RAMI 4.0 provides a standardized language for communication and data exchange between different components of a manufacturing system. This can help to reduce the risk of errors and improve efficiency by automating data transfer and analysis. In addition, modeling assets according to RAMI4.0 can facilitate the use of advanced manufacturing technologies such as artificial intelligence, machine learning, and predictive analytics. By having a standardized approach to data modeling and integration, it becomes easier to develop and deploy these technologies in manufacturing environments.

6. System execution: After developing and testing the manufacturing scheduling solution, it must be deployed within the actual production environment. This involves integrating the solution with physical assets, configuring data inputs and outputs, and ensuring seamless operation. System's continuous monitoring is essential to track scheduling performance against the defined KPIs and ensure that the system is functioning as intended. A feedback loop should be established to refine and optimize the scheduling solution based on real-time data, adapting to changes in production demand, asset availability, and other dynamic factors. Regular maintenance, updates, and scalability considerations are also crucial to ensure the solution remains effective and can be extended or integrated with other systems, ultimately creating a more efficient and adaptable manufacturing environment.

The proposed methodology and its corresponding steps are represented in Figure 4.2, for a better understanding. It works in an iterative way, starting with step 1, but where each step can be revisited any time.

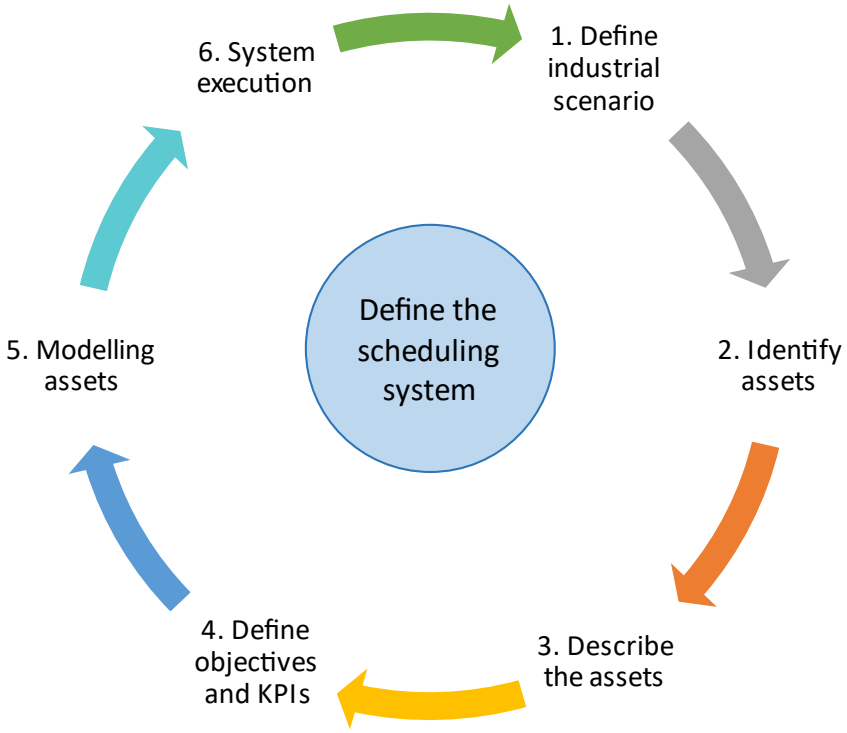

Figure 4.2 - Manufacturing scheduling methodology steps

According to the proposed methodology to implement a production scheduling system it is necessary to create digital representations of the physical production assets, such as machinery and equipment, and integrate them into the software platform. The AAS would provide secure access to the data and functions of the production scheduling software model, enabling remote monitoring and control of the production process.

As stated before, I4.0 defines an asset as anything that requires a connection for the I4.0 solution, which includes machines, components, products, or contracts and orders. All these various assets must be identifiable, which means that their status, operational and technical data, and other information must be understandable by other systems within the I4.0. To accomplish this, RAMI 4.0 introduced AAS. The AAS plays a significant role in facilitating various industrial operations, including dynamic scheduling, since it will aid to improve asset utilization, enhance efficiency and productivity and reduce maintenance costs, by including contextual information about the assets and allowing real-time data integration, interoperability between different systems, and more flexibility and scalability.

4.2 Assets Identification

In the context of production scheduling for SM, very little has been done to develop a common way to design scheduling solutions. Although scheduling solutions are usually case-oriented, a part of the process is common for most cases, and it can be unified and standardized with RAMI 4.0, shortening the development phase of the scheduling (Alemao et al., 2022). To do so, it is crucial to identify the assets that are present in any scheduling problem and then model them as generic as possible, so little to no changes need to be made when adopting this approach.

Before identifying the assets, it is necessary to consider three domains when developing manufacturing scheduling solutions, which are aligned with the main industry paradigms developed recently, namely CPPS and RAMI4.0. Those are the physical layer, the integration layer, and the cyber (or application layer), as exemplified in Figure 4.3. The physical layer comprises the shop floor and the physical assets it contains, such as sensors, actuators, products, and machinery. The cyber layer includes the software necessary to process and manage data and develop tools and algorithms to improve manufacturing systems, business processes, and services communication to external systems. The integration layer is responsible for linking the other two, providing the interoperability capabilities necessary for a smooth interaction between cyber and physical domains, such as services and interfaces.

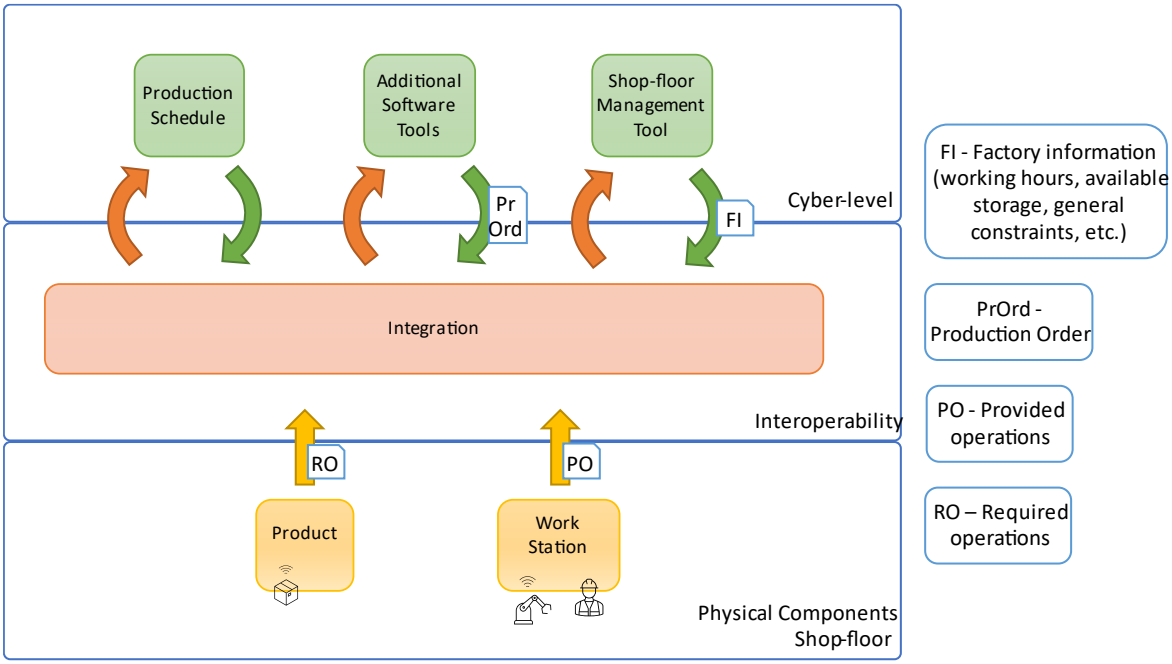

Figure 4.3 – Three domains to consider when developing manufacturing scheduling solutions.

The production scheduling framework could include the following broader modules in order to successfully implement a manufacturing scheduling solution:

1. Production scheduling module: This module would enable the creation of efficient and effective production schedules based on customer demand, available resources, and production capacity, ensuring that production orders are executed on time while specific objectives are optimized. It plays a crucial role in the methodology as it assists to optimize the production process. This module should interact closely with a MES for better coordination of the system and, consequently, obtain reliable information on the products to be manufactured, required quantities, and production timeline listed in the production plan. This module would be responsible for gathering all the important data from products, resources, facility constraints, production orders, requirements, and so on, and run its inner algorithms to provide the best schedule solution possible for the problem in hands. The system will benefit in several ways from the scheduling module. It will allow to allocate resources efficiently by considering the availability and capabilities of different resources, to generate an optimal schedule that optimizes their utilization. It considers customers' orders, delivery dates and priorities to ensure the production is following the demand and helps to avoid delays and missing deadlines. Furthermore, it may help to minimize production costs by reducing idle time, changeovers, and minimizing waste. The scheduling module should manage customer orders (including quantities, deadlines, priorities, and requirements or constraints), consider the available station's capacity, contain scheduling algorithms to generate optimized solutions, and integrate real-time data with the resource and inventory management modules. Additionally, it should provide a visual representation of the production schedule, allowing an easy understanding of the schedule.

2. Resource management module: This module would provide details on the availability and utilization of production resources, such as machinery, equipment, and personnel, as well as its capabilities. It would also include additional information on maintenance schedules and downtime, for example, that may directly interfere with the production schedule. As so, it will ensure not only an optimal utilization of resources in the production environment, considering their availability, capacity, capabilities, and requirements, but also resource tracking from the integration with sensors and stations monitoring systems to gather data about status and availability in real-time. The real-time interaction with a resource management module will allow to highly improve the output of the schedule and help to define better alternatives when unexpected events arise in the production line. It should also comprise other elements such as a database of all resources available in the facility and their associated attributes; estimate future resource requirements by analyzing historical data, production forecasts, and order demand; maintenance and downtime activities management to help ensure that these activities are scheduled at optimal times to minimize disruptions.

3. Inventory management module: This module would provide information on the availability and quantities of raw materials, work-in-progress, and finished goods. The scheduling system may benefit from good inventory management in different ways, such as an ERP. First of all, the production scheduling module must know which type and quantity of raw materials are available for production, so it can allocate the tasks effectively. Otherwise, if raw materials are missing the schedule needs to be delayed. Second, if there is work-in-progress when a new schedule generation starts, it is important to know the current status of the production, namely what is being produced and how long the stations will be occupied. Lastly, the schedule may be adjusted according to the finished goods stored in the warehouse, and so, attribute different priorities to different production tasks when there are enough goods stored. Thus, it ensures that production orders are scheduled based on the availability of inventory, preventing stock ruptures or excess inventory. Efficient inventory management will improve the cost optimization associated with excess and lack of inventory.

4. Analytics module: This module would provide real-time data on production performance, resource utilization, inventory level trends, and quality metrics. It would include customizable dashboards and reports for management and stakeholders. The scheduling system should work perfectly without an analytics module, however, it would benefit from it in several different ways. Analytics tools can provide not only real-time information but also alerts, predictions, and trends that may be detected in the production system. Thus, the schedule module may transform this data into useful knowledge and provide improved solutions. This will allow to handle changes and unforeseen events, provide the ability to quickly reschedule production orders, and adapt efficiently in real-time in response to dynamic conditions.

Although it is bold to indicate which entities to model within this ecosystem, from the author's own experience (Alemão et al., 2021; Alemão, Rocha, et al., 2019) and the analysis of the literature, such as (K. Gao et al., 2019; Klement et al., 2021; Mohammadi et al., 2020; Nikolakis et al., 2018; Zeng et al., 2018), and focusing on the relevant data for the production scheduling module, the following elements were identified as vital for any scheduling solution development. Many others can be defined as optional, depending on the context, and may also be important for the development of scheduling solutions, but it is still possible to implement generic solutions without considering them.

In the context of this work, the following entities are considered mandatory to develop a robust scheduling system:

- Production schedule: It encompasses the information contained in the production schedules, such as the planned start and end times for each production operation, the stations where they are being performed, as well as any dependencies between operations, and any additional information necessary to help to understand the schedule output
- Product: It serves to define all the items that are manufactured in the facility and includes product specifications, such as size, weight, and composition, which affect the manufacturing process. It may be defined in a Bill of Materials (listing all the components and raw materials required to produce the parts). A product may not be a finalized good, but a part of a final product.
- Resource: Represents all the machinery, human workers, equipment, and tools required to manufacture a product. It includes, for instance, capability, capacity, availability, maintenance schedules, and available functionalities, such as screw, drill, paint, or pick and place.
- Production constraints and requirements (Shop-floor related data): include limitations on the amount of time, labor, and materials available for manufacturing, the number of shifts or working hours and any maintenance or downtime requirements, raw materials, work in progress, finished goods inventory levels, and transportation units in the shop floor. Also, it needs to consider business rules, such as production quotas for specific products, as well as safety restrictions or energy consumption requirements.
- Scheduling software: represents the production scheduling software as a whole, with information describing its capabilities, such as creating and managing production orders, scheduling resources, and monitoring production status. It could also provide service interfaces to interact with other components

Optional entities:

- Task: This entity represents the specific activities required to manufacture a product, such as a specific production technique, assembly, testing, inspection, and packaging. To manufacture a product, it is necessary to have one or more operations. In the cases where there is more than one operation, the system will have different approaches depending on whether they have dependencies amongst them or not, as it will influence the order of the operations in the schedule. The operations may have different setup and execution times depending on which stations they are allocated to. Considering the different times of several products helps in achieving more accurate schedule solutions and avoiding delays.
- Production process: The steps involved in the manufacturing process, including the sequence of operations to manufacture each product, and their respective durations.
- Work center: A physical location where a specific operation takes place. It may be able to produce different parts according to its configurations, which also leads to different execution times. Work centers can be shared by multiple resources. Sometimes, for instance, if the work center is only composed of one resource, the work center and the resource may be considered the same, as it will not interfere with the schedule solution.
- Production Order: Represents the data for individual production orders, such as the quantity of the product to be manufactured, the desired delivery date, where the products should be delivered once they have been completed, material requirements, and any special instructions or constraints.
- Demand Forecast: Software benefits from having access to demand forecast, which can be based on historical sales data, customer orders, or market trends. This information helps the software to optimize production schedules based on expected demand.
- Additional Software tools (MES, ERP…): Other tools that may be important to support the scheduling process, such as MES, ERP, monitoring tools, and so on, for instance, to obtain the production orders.
- User interface: a graphical user interface available to the users that facilitates the implementation of the schedule on the shop floor. If there is already software providing a user interface it is not required within the scheduling tool.

Based on this information, the production scheduling software will generate a production schedule that optimizes multi-objectives, such as production optimization regarding completion times, completed jobs or cost of workers, optimization of energy efficiency or reduction

of material waste, and meets customer demand. The software will highly benefit from real-time monitoring and control of the production process, allowing for quick adjustments to be made as needed. This can help to improve efficiency, reduce costs, and increase the overall effectiveness of the manufacturing plant.

In Figure 4.4, is represented a component diagram composed of the main assets to be considered when using the proposed methodology. The Scheduler component handles the scheduling process. It interacts with other components to get the necessary data and applies the algorithms to generate new schedules. The scheduler may benefit from a GUI to display the generating schedules, although it may provide an output that is not visual. The Resource component deals with the various resources (e.g., machinery, labor) needed for production. It includes details about resource availability, capabilities, and current status. The Product component manages product details, including product specifications and requirements. If the product includes more than one task for processing, then a Task component is necessary to handle the data corresponding to each individual task. The Shop-floor Management component includes relevant information about the shop-floor, such as resources present in the system, working hours or any other constraints. Likely, it will include the production orders to be sent to the scheduler, although it. In yellow are represented the mandatory components of the system.

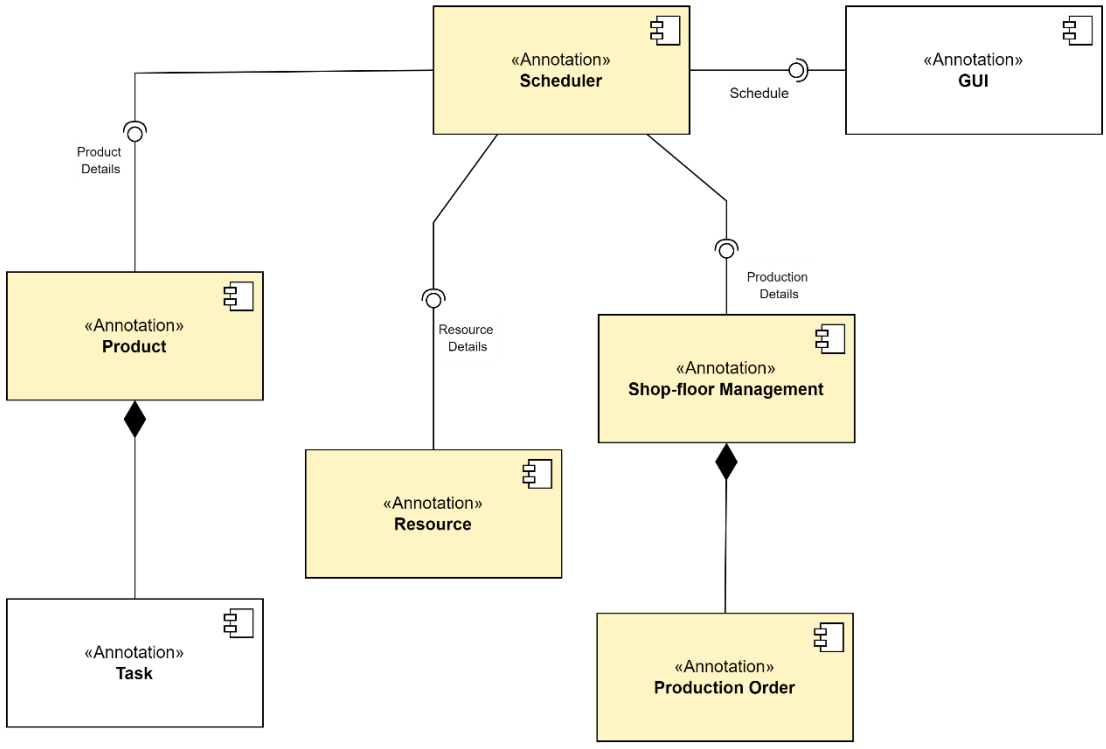

Figure 4.4 - Scheduling system components diagram

To implement a solution based on this methodology, it is essential to follow the steps outlined below. The software development process would likely involve creating and integrating several key components, including:

1. A data model: This would define the types of data and metadata that the production scheduling software needs to store and manage, such as production orders, resource capabilities, material requirements, and production schedules.
2. A service interface: This would provide standardized APIs for other AAS-enabled systems and devices to interact with the production scheduling software, such as to query the status of production orders or to update the production schedule.
3. A data storage and management system: This would be responsible for storing and managing the production scheduling data and providing efficient access to that data for the service interface and other components of the production scheduling software.
4. A user interface: This would provide a graphical interface for production managers and operators to interact with the production scheduling software, such as to create and modify production orders, view production schedules, and monitor the status of production operations.

In addition to these core components, production scheduling software might also include other features and capabilities, such as:

- Integration with other manufacturing systems and devices, such as MES, ERP, and SCADA (Supervisory Control and Data Acquisition) systems.
- Real-time monitoring and optimization of production schedules, based on changing conditions such as machine availability, material availability, and order priorities.
- Advanced analytics and reporting capabilities, such as identifying production bottlenecks, optimizing production schedules, and tracking performance metrics over time.
- Integration with AI and machine learning algorithms to help automate and optimize production scheduling decisions.

Figure 4.5 illustrates the RAMI 4.0 layers in which each of the four most relevant assets for manufacturing scheduling is represented. To successfully implement a scheduling system, it is vital to have a thorough description of the Factory and the Scheduling Tool, even in terms of business processes, while in the Product case, the business necessities and the functions it provides (if any) are not relevant for the scheduling execution, as the important is to know how to communicate with the product and access its information. In the case of Stations, it is necessary to know their functionalities so the products can be correctly allocated.

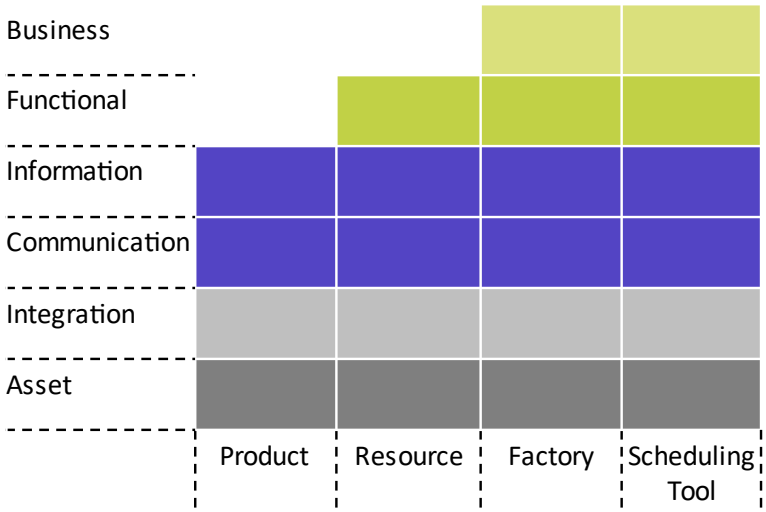

Figure 4.5 – Asset's representation in the RAMI 4.0 layers.

Although this work provides a methodology for implementing manufacturing scheduling solutions, it is important to notice that not all properties will be relevant for all application scenarios. The transfer of properties and data between different participants in a manufacturing or software development process, specifically focusing on how properties and functions are managed within an AAS is not linear and may include the addition or removing of some properties. Figure 4.6 represents the transferring of properties between the different recipients, mainly the asset manufacturer, which develops the software, the software builder, which may improve or integrate the software in their ecosystem and, finally, the plant operator, which will operate the software. As can be observed, not all the properties implemented in the first stage need to be adopted later. Some predefined properties will remain, which are usually the most generic ones, while user-specific properties (in green) may not be transferred for subsequent users or may not be adopted by them, as it may represent content that is unnecessary.

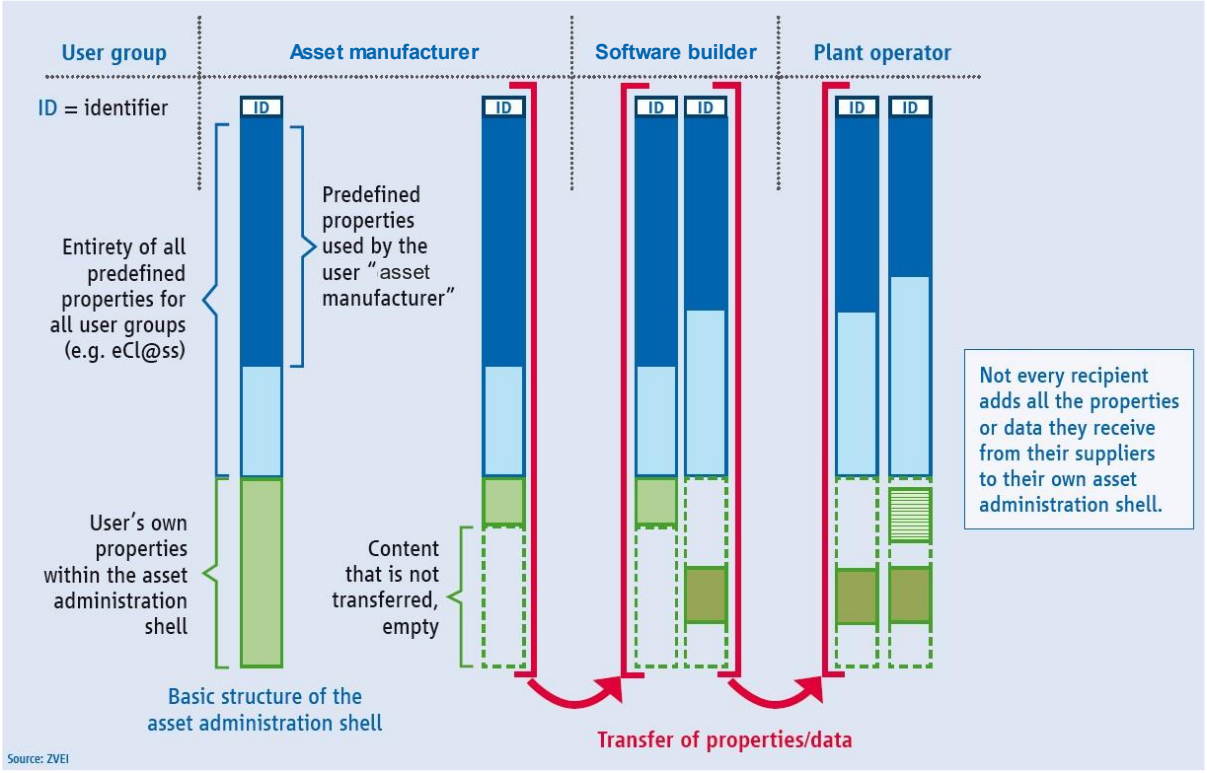

Figure 4.6 - Transfer of Properties/Data from the Asset Administration Shell (Adapted from (ZVEI, 2018))

Additionally, a complete scheduling solution should contain several functionalities represented as submodels in the AAS, as depicted below, however not all may be necessary for each specific case:

- Generate production optimization schedule, by allocating production and/or maintenance operations to the available resources, respecting the constraints and requirements identified for the use case.
- Generate energy optimization schedule, by allocating production and/or maintenance operations to the available resources, respecting the constraints and requirements identified for the use case.
- Objective function definition to be applied when generating a new schedule.
- Take advantage of the implementation of different algorithms to generate schedules and evaluate which one performs better.
- Provide a Graphical User Interface, so it is easy for end-users to understand the output of the schedule and possible warnings.
- Perform data analysis on the data collected from the shop floor, namely from resources and products, so it can identify relevant deviations to the current schedule and reallocate operations when possible or provide a new schedule (rescheduling) in real-time.
- Execute simulation activities, to analyze how the current schedule will behave when faced with different scenarios.
- Evaluate and provide algorithm-related KPIs, such as execution time, used parameters, better solution obtained, mean solution, invalid solutions, and so on.

- Evaluate production-related KPIs, for instance, cycle time, on-time delivery, throughput rate, resources utilization, and production cost, along with others, and compare them between different algorithms implemented.
- Documentation regarding configuration or runtime data.
- Communication identifying the different possible ways to communicate with the asset and how its services are provided.
- Lifecycle status of the tool.
- Integration with smart and legacy systems.

In the next sub-section are represented the AAS submodels of the main assets, as well as their properties, that may be "mandatory" to adopt, "preferred" if they are relevant but not essential, or "optional" if they are not relevant but may represent an advantage. In dark grey are represented the submodels and in light grey the submodels properties.

4.2.1 Scheduler

The scheduler asset is the most relevant in this work, and so the one with more information. Table 3 presents the AAS submodels and their properties for the Scheduler asset. It will gather data from resources, products and requirements and constraints from the shop-floor to build the scheduling system and the desired output.

Table 3 - Scheduler AAS submodels and properties

Name	Explanation	Required	Example
Identification			
Id	Global unique ID of an asset, which can be read by both human and machine	Mandatory	asset0123
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	"software tool responsible for generating production schedules"
Version	Number to distinguish the version of an asset	Preferable	001
Further Info	Additional info necessary to implement the solution	Optional	
Information			

ProductsList	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	[ProdId1, ProdId2, …]
TasksList	A collection of tasks to be allocated in the production schedule.	Mandatory	[TaskId1, TaskId2, …]
Type	Indicates the type of schedule to be performed. E.g. production-oriented, energy consumption-oriented, etc.	Optional	"Full throughput"
SchedulerConfigurations			
MaxTasksToAllocate	Maximum number of tasks allowed to be allocated by the algorithm. May be limited by algorithm restrictions.	Optional	200
PriorityStrategy	Defines the priority to give to each task. E.g. FIFO, importance, execution time duration, etc.	Preferable	FIFO
ListOfSchedulingAlgorithms	List of scheduling algorithms included in the scheduling tool. The user (or the tool itself) may choose the algorithm according to the output expectations.	Preferable	[Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, Mathematical Model]
ListOfObjectivesPossibleToOptimize	Collection of objectives that can be chosen to be optimized	Preferable	[Makespan, Load Balance, Maximize delivery orders, tardiness]
ListOfObjectiveFunctions	Collection of objective functions to be used, depending on the	Mandatory	

	objectives chosen to be optimized. At least one objective function needs to be defined.		$Fitness = \frac{1}{makespan} \times penalty$
ConfigureScheduler	Function to set scheduler configurations for the next schedule generation. It takes the previous five parameters as inputs.	Preferable	
DefineNewObjectiveFunction	Function to create and insert new objective function into the list of objective functions	Optional	
DeleteObjectiveFunction	Function to delete an objective function from the list	Optional	
DefineNewAlgorithm	Function to create and insert a new algorithm into the algorithms list	Optional	
DeleteAlgorithm	Function to delete an algorithm from the list	Optional	
SchedulerOperations			
GenerateNewSchedule	Function call to generate a new schedule based on the saved configurations	Mandatory	
AddTask	Function to add a new task to the next schedule	Optional	
RemoveTask	Function to remove a specific task from the next schedule	Optional	
AddTasksList	Function to add a list of tasks to be allocated in the next schedule	Mandatory	
Reschedule	Function to reschedule a previously generated schedule automatically, if some perturbances occur during the production	Optional	

CurrentSchedule			
Id	Unique ID of current schedule	Mandatory	"schedule_01062023"
Name	Name consisting of one or more words to be easily identified by a human	Optional	"Energy optimization schedule for production order XYZ"
GenarationDate	Date of generation	Preferable	01-06-2023 10:00:00
Description	Information that uniquely describes the meaning of the schedule	Optional	"Energy consumption optimization schedule for production order XYZ starting on June 10th, 2023"
AssociatedProductionOrder	Refers to the ID of the corresponding production order	Mandatory	ProductionOrder1234
TasksList	List of tasks allocated in this schedule, with all the information for each one, including start and end times of production.	Mandatory	
ResourcesList	List of resources used for this schedule, including start and end times, and which tasks are allocated in each moment	Mandatory	
StartTime	Refers to the schedule start time	Mandatory	10-06-2023 08:00:00
EndTime	Refers to the schedule end time	Mandatory	14-06-2023 14:32:26
QuantityProduced	Number of manufactured products	Optional	250
AlgorithmAdopted	Refers to which algorithm was adopted for this schedule instance	Optional	"Genetic Algorithm"
OptimizedObjectives	Refers to the objectives that were optimized during the process	Optional	[Makespan, tardiness]

GanttChart	Visual description of the generated schedule	Preferable	It can be a PDF document, an image, a dynamic web-based chart, etc.
Communication	Description of the different ways to communication with this AAS	Mandatory	REST Services datasheet explaining how to reach the AAS (e.g. by providing IP address and port)
GraphicalUserInterface	A graphical user interface that eases the interaction with the tool, both on shop-floor level and on administrative level	Preferable	Java interface to allow to generate schedules and easily define the required configurations
Integration	Description of how to integrate the tool with other systems	Optional	Integration with smart or legacy systems

4.2.2 Resource

The submodels and properties for the Resource asset are represented in Table 4. It contains all the relevant information to make it possible for the scheduler to allocate the different task effectively to the available resources.

Table 4 - Resource AAS submodels and properties

Name	Explanation	Required	Example
Identification			
Id	Global unique ID of an asset, which can be read by both human and machine	Mandatory	Resource0123
Name	Name consisting of one or more words to be easily identified by a human	Preferable	"Cutting machine"
Description	Information that uniquely describes the meaning of the asset	Preferable	"machine responsible for cutting metal parts"

Version	Number to distinguish the version of an asset	Optional	001
Further Info	Additional info necessary to implement the solution	Optional	
Documentation	Documents related to the resource	Mandatory	manuals for installation and operation
Communication	External interfaces provided	Mandatory	OPC-UA, REST, MQTT
Configuration			
Function	Describes the function or set of functions of the machine	Mandatory	OpMode1
SetupTime	Defines the time to setup the machine	Mandatory	[sec]
OperationModes	Specifies the operation mode being used (which may affect, for example, the execution velocity or energy consumption)	Mandatory	["OpMode1", "OpMode2"]
Capacity	Buffer capacity of the resource	Optional	5
ProductsAbleToManufactured	Identifies the type of products the resource is able to produce	Preferable	"ProdA", "ProdC"
KPIs	Describes some KPIs that may be relevant	Optional	
Maintenance	Defines information related to maintenance in the resource (more important if maintenance tasks should be considered)	Optional	
MES Connection	Show a simple connection between the asset and an MES system.	Optional	
Production status	Determines if the asset is able to execute a production task at the time being	Preferable	[Idle, Running, Down Time, Failure]

Operating hours	Determines how long the asset has been operating	Preferable	[sec]
Additional sub-models			Energy Efficiency; Drilling; Painting; Cutting

4.2.3 Product

Table 5 outlines the AAS submodels and properties for the Product asset. Most important here, the product contains the list of tasks necessary to execute during the production.

Table 5 - Product AAS submodels and properties

Name	Explanation	Required	Example
Identification			
Id	Global unique ID of an asset, which can be read by both human and machine	Mandatory	Product012
Name	Name consisting of one or more words to be easily identified by a human	Preferable	"PieceA"
Description	Information that uniquely describes the asset's meaning	Preferable	"metal part"
Version	Number to distinguish the version of an asset	Optional	001
Further Info	Additional info necessary to implement the solution	Optional	
Documentation	Documents related to the resource	Mandatory	manuals for installation and operation
Communication	External interfaces provided	Mandatory (for smart products)	OPC-UA, REST, MQTT
Configuration			
ListOfTasks	List of tasks to be executed in this product. At least one task is necessary	Mandatory	[Task1, Task2]

Associat-edResources	Indication of the resources where the tasks will be executed, if already defined	Preferable	[Res01, Res05]
RawMaterials	Raw materials necessary to manufacture the product	Optional	[screws, paint]

4.2.4 Shop-Floor Management

Table 6 reflects the submodels and properties related to the shop-floor that are most relevant for the scheduling execution, as information about working hours, available resource in the facility or production orders.

Table 6 - Shop-floor AAS submodels and properties

Name	Explanation	Required	Example
Identification			
Id	Global unique ID of an asset, which can be read by both human and machine	Mandatory	Factory01SF02
Name	Name consisting of one or more words to be easily identified by a human	Preferable	"Car Factory Shop floor 2"
Description	Information that uniquely describes the meaning of the asset	Preferable	"Shop floor where are produced cars of model X"
Version	Number to distinguish the version of an asset	Optional	001
Further Info	Additional info necessary to implement the solution	Optional	
Parameters			
WorkingHours	Description of the shop floor working hours	Mandatory	06h00m to 22h00m
WorkingShifts	Description of the shop floor working shifts	Optional	[06h00m-14h00m, 14h00m-22h00m]

Maintenance-Hours	Specific hours to perform maintenance tasks (if different from working hours	Optional	[08h00m-16h00m]
ResourcesList	List of resources available in the shop floor	Preferable	[Res01, Res02, Res06]
ProductsA-bleToBeManufac-tured	List of products that can be manufactured in this shop-floor	Preferable	[Prod012, Prod021, Prod025]
RawMaterialAvail-able	Raw material available in the fa-cility	Optional	
ProductionCur-rentStatus	Current status of the produc-tion, showing work in progress, finished products, resources in-formation, etc.	Preferable	
Transportation-Options	Options available to carry prod-ucts between stations in the shop-floor.	Preferable	[Human worker, con-veyorA, conveyor B, forklift]
Storage	Storage availability within the shop floor	Optional	
Constraints	Constraints regarding the shop floor, eg. start working hours	Preferable	"start working hours: 08h00m"
ProductionOrder	List of products to be manufac-tured	Mandatory	10 Prod012, 55 Prod021
Id	Global unique ID of the produc-tion order, which can be read by both human and machine	Mandatory	Order122024
Type	Identify if it is a production or maintenance order	Optional	Production
Quantity	Identify the quantity of prod-ucts to be manufactured in that order	Preferable	50
ListOfProducts	List of the products to be man-ufactured	Mandatory	[Prod012, Prod 013, Prod 101, Prod, 113]

Deadline	Identifies tge deadline to finish that order	Preferable	30-06-2024
Priority	Identify the priority given to that order	Optional	Define order of priorities

4.3 Data model

The data model of the full manufacturing scheduling system is represented in Figure 4.7, with the classes considered during the next section highlighted. It is represented by the main classes in the manufacturing system that may be adopted when developing scheduling solutions. The data model proposed for the scheduling solution shall follow the RAMI4.0 implementation standards. It is considered that the shop-floor management class manages the entire system, as it interacts with the physical assets as well as to the software tools that run in the higher level, and has access to all data regarding the physical system. The shop-floor management component includes relevant information about working hours, maintenance shifts, transportation options within the factory, the factory's topology, available raw materials and storage, the products being manufactured in that factory, available working stations (including machines and human workers), available production processes to follow, and production orders reaching to the factory. On its turn, the scheduling tool will gather all the information it needs, such as available stations, processing times, and working hours to provide a schedule solution for each production order, based on the optimization processes included in the tool.

In the process of designing the scheduling system, a complex UML diagram was initially designed to capture the full breadth of the system's potential features and interactions. This comprehensive version serves as a detailed blueprint that explores all possible components and their relationships, ensuring that every aspect of the scheduling system is thoroughly understood.

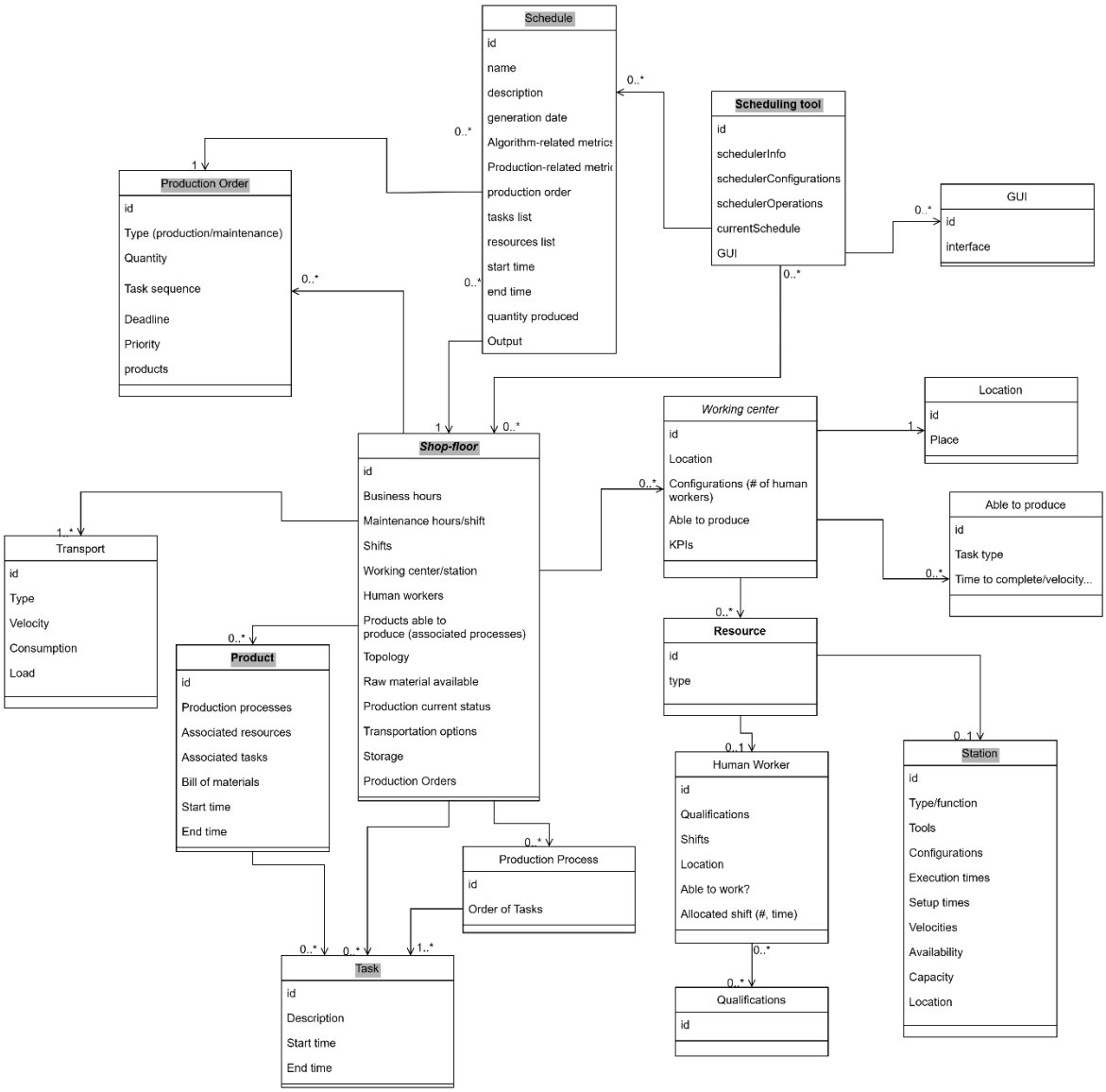

Figure 4.7 - Complete manufacturing scheduling system UML diagram

The *Shop-floor Manager* class serves as the central repository and controller for all information related to the shop floor. It manages real-time status updates of the production floor, ensuring that all operations run smoothly. The Shop-floor Manager collects data from various resources and products present in the factory, providing this crucial information to the scheduling tool. It also communicates with resources to monitor their status and availability, ensuring that production orders are executed efficiently. By coordinating with different resources and products, the Shop-floor Manager plays a pivotal role in maintaining the workflow and addressing any issues that arise on the shop floor. It encompasses data such as an *id*, which serves as a unique identifier for the shop-floor, the *business hours* attribute that defines the operating times of the shop-floor, the *resources* attribute which encompasses all the

machines and labor available for production, the *products able to be produced* attribute that lists the products that can be manufactured on the shop-floor. Furthermore, the *topology* attribute describes the layout and organization of the shop floor, so it is possible to know the paths and distances between resources, the *raw material available* attribute tracks the inventory of raw materials necessary for production, the *storage* attribute manages the space and conditions for storing raw materials and finished products, while the *transportation system* attribute details the logistics for moving products within the facility. Finally, the *production current status* attribute provides real-time updates on the progress and status of ongoing production activities. These attributes collectively enable the shop-floor manager to effectively oversee and optimize the manufacturing process.

The *Product* class encapsulates all relevant information about the products being manufactured. This includes an *id*, which uniquely identifies each product. The *associated tasks* attribute lists the specific tasks that need to be performed to manufacture the product. The *associated resources* attribute identifies the resources required to complete these tasks, which are assigned after the schedule being generated. The *production process* attribute outlines the sequence of tasks involved in creating the product, ensuring that the production follows a structured plan. Finally, the *raw material* attribute details the raw materials needed for producing the product. These attributes collectively provide a comprehensive view of the product's manufacturing requirements and ensures that all product-related data is organized and readily available for the shop-floor manager and scheduling tool, facilitating efficient production planning and execution.

The *Station* class represents the different resources within the factory and includes attributes that provide detailed information about each one. The *id* serves as a unique identifier for the resource, while the type identifies the type of work that station can perform. The *execution time* attribute specifies the time required for the resource to complete its tasks. *Setup times* denote the time needed to prepare the resource for production. The *velocities* attribute details the operational speed of the resource. *Configurations* include the different settings and adjustments that the resource can operate under. The *maintenance* attribute tracks the maintenance schedules and history of the resource. The *products able to produce* attribute lists the products that the resource can manufacture in a certain moment. *Operations able to execute* describes the various tasks the resource is capable of performing. The *availability* attribute provides real-time information on whether the resource is available, in use, or under maintenance. These attributes ensure that the resource’s capabilities and status are comprehensively managed and monitored.

The *Scheduling Tool* class is vital for generating production schedules and includes attributes that support this function. The *id* uniquely identifies the scheduling tool. *SchedulerInfo* contains general information about the scheduler, such as its purpose and scope, or the products that are being processed. *SchedulerConfigurations* include settings and parameters that define how the scheduler operates, such as the algorithms being used, the maximum tasks it is able to process, or add and remove optimization techniques. *SchedulerOperations* describe the various operations the scheduler can perform, as generate new schedules, reschedule a previous solution, or add or remove tasks to be included or excluded from schedule. The *current schedule* attribute shows the active production schedule generated by the tool. The *GUI* attribute provides the graphical user interface through which users interact with the scheduling tool. These attributes collectively facilitate the creation and management of optimized production schedules. The tool handles various scheduling constraints and dependencies, adapting to changes in resource availability and production progress.

The *Production Order* class defines what needs to be produced, including detailed information on production requirements, deadlines, and quantities. Each order is identified by a unique *id*. The *type* attribute specifies whether the order is for production or maintenance. *Quantity* indicates the number of units to be produced, while the *task sequence* outlines the sequence of tasks required to complete the order, when relevant. The *deadline* specifies the required completion date for the order, and the *priority* indicates the importance level of the order relative to others that may be under consideration. The *expected optimization type* describes the type of optimization expected, such as minimizing time or maximizing resource utilization. These attributes ensure that each production order is clearly defined and prioritized, facilitating an efficient execution. The production order includes specific instructions or constraints that must be followed during manufacturing. This class provides a clear and structured directive for the production process, ensuring that all necessary details are communicated to the shop-floor manager and scheduling tool.

The *Schedule* class is responsible for generating a human-readible or a visual representation of the production schedule and includes several detailed attributes. The *id* identifies the schedule output, whilst the *name* provides a descriptive name for the schedule. The *Description* offers additional details about the schedule, and the *Generation date* indicates when the schedule was created. On the other hand, *Algorithm-related metrics* track the performance of the scheduling algorithm used, which make it possible to do a comparison between algorithms, while *Production-related metrics* measure various aspects of production performance. The *associated production order* attribute links the schedule to the production order that originated

it. The *tasks list* details the tasks included in the schedule, at the same time as *Resources list* lists the resources allocated in the schedule. *Start time* and *end time* specify the timeline for the schedule, since the first product started being produced until the last one is finished. *Quantity produced* indicates the number of units expected to be produced. The *output* attribute provides a Gantt chart or other representation of the schedule that is easy for human-workers understanding. These attributes ensure that the schedule output is comprehensive and informative, supporting effective production management.

Following this data model, it is possible to create a cohesive and integrated manufacturing scheduling system. The Shop-floor Manager gathers and manages data from the Product, Statopm, and Scheduling Tool classes, ensuring all necessary information is available for efficient scheduling and production. The Scheduling Tool utilizes this data, along with specific instructions from the Production Order, to generate optimized production schedules. The Schedule Output then provides a visual and detailed representation of these schedules, helping workers to monitor and manage the production process. This integrated approach optimizes resource utilization and enhances overall production efficiency.

4.4 Design Principles aligned to the methodology: Summary

This section demonstrates how the identified DPs relate to the proposed methodology and the respective RAMI4.0 layers. In Table 7 each principle is systematically categorized according to specific steps in the methodology, linked with corresponding layers of the RAMI 4.0, which provides a structured framework for digitalizing industrial environments. The table also provides examples, when relevant, and justifications for clarifying the practical applications and benefits of each principle, emphasizing their importance in creating efficient, integrated, and responsive manufacturing scheduling systems that are crucial for SM adoption.

Table 7 - Summary of the Design Principles and their relation to the methodology

Design Principle	Methodology Step	RAMI 4.0 Layer	RAMI 4.0 Lifecycle	Justification
1.1.1 - Digital factory replica	Identify assets	Integration	Development	Creating a digital replica of the factory is the foundational step in identifying and cataloging all assets within the manufacturing environment. This replica serves as a comprehensive digital model that provides detailed information about the physical layout, equipment, and workflows. It enables accurate simulations and analyses, facilitating better planning and optimization of the manufacturing process.
1.1.2 - Communication protocols	Describe assets	Communication	Development	Describing assets using standardized communication protocols ensures interoperability and effective data exchange across different systems and devices. This step is crucial for integrating disparate tools and technologies, allowing seamless communication without compatibility issues, thus supporting a connected and efficient manufacturing ecosystem.

1.2.1 - Common interfaces	Describe assets	Communication	Development	The adoption of common interfaces across different tools and systems prevents the need for reprogramming whenever a new tool is introduced. This standardization enables tools to interact smoothly, significantly reducing integration time and costs, and enhancing the scalability of the manufacturing system.
1.2.2 - Data model	Describe assets	Information	Development	The adoption of data models allows the data structure and semantics to be harmonized. This consistency is crucial for ensuring that data can be accurately and efficiently shared, understood, and processed across various components, facilitating improved data analytics and decision-making.
1.2.3 - Uniform data among components	Describe assets	Information	Development	Maintaining uniform data standards among components promotes collaboration between tools from different manufacturers. This uniformity is vital for integrating heterogeneous systems, ensuring that data is consistently interpreted and utilized, which enhances overall system performance and reliability.
1.3.1 - Harmonized KPIs description	Define objectives and KPIs	Business	Development	Establishing uniform rules for KPI creation is essential for consistent performance measurement and evaluation. Harmonized KPIs enable a standardized approach to monitoring and improving various aspects of the manufacturing process, ensuring that all stakeholders are aligned on key performance criteria.
1.3.2 - Save production objectives	System execution	Information	Production	Storing production objectives allows for real-time tracking and updating of KPIs, ensuring that the system remains aligned with the current production goals. This dynamic updating capability is crucial for adapting to changes and maintaining optimal performance throughout the manufacturing process.

1.3.3 - Editable KPIs priorities	Define objectives and KPIs	Functional	Development	Allowing the prioritization of KPIs during the process enables the system to focus on the most critical metrics based on the current production context. This flexibility ensures that resources are allocated efficiently, and production strategies can be adjusted in response to emerging priorities.
1.3.4 - Considering different KPIs for schedule generation	Define objectives and KPIs	Functional	Development	Providing the option to select specific KPIs for schedule generation empowers managers to tailor the scheduling process according to the most relevant performance indicators. This customization enhances the accuracy and relevance of the scheduling outcomes.
1.4.1 - Define boundaries	Define industrial scenario	Information	Development	Defining operational boundaries, guides the allocation of production tasks within acceptable limits. This ensures that production stays within safe and efficient operational parameters, aligning with sustainability and efficiency goals.
1.4.2 - Evaluate production process evolution	System execution	Functional	Production	Real-time data acquisition from the shop floor enables continuous evaluation of the production process. This monitoring capability is crucial for identifying deviations, assessing performance, and making timely adjustments to maintain optimal process flow.
1.4.3 - Analyze new orders	System execution	Functional	Production	Evaluating new orders in the context of the current schedule helps in determining the feasibility of integrating them without disrupting existing plans. This step is essential for maintaining schedule integrity and ensuring that the production system can handle additional workloads efficiently.
1.4.4 - Automatic rescheduling	Modelling assets	Functional	Development	Implementing automatic rescheduling in response to significant deviations or disruptions ensures that the production plan remains aligned with real-time

				conditions. This adaptability is critical for minimizing downtime and optimizing resource utilization.
1.5.1 - Schedule validation	System execution	Functional	Production	he GUI should provide a comprehensive view of the system, allowing production managers to validate and approve the proposed schedule. This transparency ensures that the schedule is robust and meets all operational requirements.
1.5.2 - New schedules request	System execution	Functional	Production	The system should support the generation of new schedules not only in response to deviations but also based on managerial requests. This flexibility allows for proactive adjustments to the schedule, improving responsiveness to changing conditions.
1.5.3 - Choose KPIs	Modelling assets	Functional	Development	The GUI must include options for selecting which KPIs are considered in the next scheduling generation. This feature allows for fine-tuning the scheduling process based on the most relevant performance indicators, ensuring that the outcomes are closely aligned with strategic objectives.
1.5.4 - Human worker inputs	Modelling assets	Functional	Development	Incorporating human feedback, enables the system to capture insights and learn from past experiences. This feedback loop is essential for continuous improvement and refinement of the scheduling process.
1.6.1 - Implementation of different optimization techniques	Modelling assets	Functional	Development	Using various optimization techniques allows for exploring different approaches to enhance system performance. Comparing these techniques helps identify the most effective strategies for optimizing scheduling and resource allocation.

1.6.2 - Communication with external systems	Modelling assets	Communication	Development	Establishing communication with external systems is crucial for gathering the necessary data for scheduling. This integration ensures that the scheduling process is updated by comprehensive and accurate information, improving decision-making and planning accuracy.
1.6.3 - Start schedule not before the current state	Modelling assets	Communication	Development	Ensuring that the starting point for any new scheduling generation is the current state of the system, or never before the current state, prevents inconsistencies and misalignments. This principle maintains the continuity and relevance of the scheduling process, ensuring that it accurately reflects the current operational conditions.

Table 7 positions the proposed DPs in the different steps of the methodology for implementing scheduling systems in manufacturing, mapped against RAMI 4.0 Layers axis and Lifecycle axis, and covering Hierarchy Levels from Product to Enterprise level. It covers essential DPs such as creating a digital factory replica, defining communication protocols, and adopting common interfaces, all aimed at integrating and harmonizing manufacturing scheduling processes. The methodology steps are aligned with various RAMI 4.0 layers such as Integration, Communication, and Information, focusing primarily on the Development phase, as it aids mainly to aid in the design and development phases of the scheduling process. The table emphasizes the importance of standardized data models, uniform data across components, and harmonized KPIs for ensuring consistent, reliable, and efficient operations. Additionally, it addresses the need for adaptive production scheduling, including automated rescheduling, schedule validation, and the incorporation of human inputs, to maintain operational efficiency and respond dynamically to production changes. Overall, the table provides a comprehensive approach to implementing manufacturing scheduling systems by leveraging the capabilities of Industry 4.0 technologies.

5

PERFoRM Project Revisited: Methodological Adaptation

In this section is demonstrated how an already existing scheduling system may be adapted in order to be compliant with the methodology proposed, and consequently with RAMI 4.0. Namely, it was taken the solution developed in PERFoRM project, where the main objective was to schedule both production and maintenance tasks to several stations, while trying to optimize the total production execution time. Although it is a methodology implementation case, it was divided from section 6, as it was done to validate that the proposed methodology can be adopted in already existing approaches. On section 6, the methodology was implemented in new application scenarios.

5.1 PERFoRM Revisited

PERFoRM (Production harmonizEd Reconfiguration of Flexible Robots and Machinery) was an EU HORIZON2020 project targeting the need for increasing flexibility and reconfigurability in the manufacturing sector. The increasing demand, in the last two decades, for more customized, cheaper and high-quality products from the customer led to a necessity for the manufacturers to produce with less delays and breakdowns to reduce production costs. In that way, in PERFoRM, was developed a smarter and pluggable system, where resources are able to communicate with each other more efficiently, in a distributed system. The PERFoRM architecture, illustrated in Figure 5.1, is based on a network of hardware devices and software applications, connected by an industrial middleware. Namely, it contains a scheduling software tool responsible for allocating production and maintenance tasks from different products to specific stations on the shop-floor. It also adopts standard interfaces as the main drivers for

pluggability and interoperability, allowing the connection between hardware devices, such as robots and controllers, and software applications, such as databases and management tools.

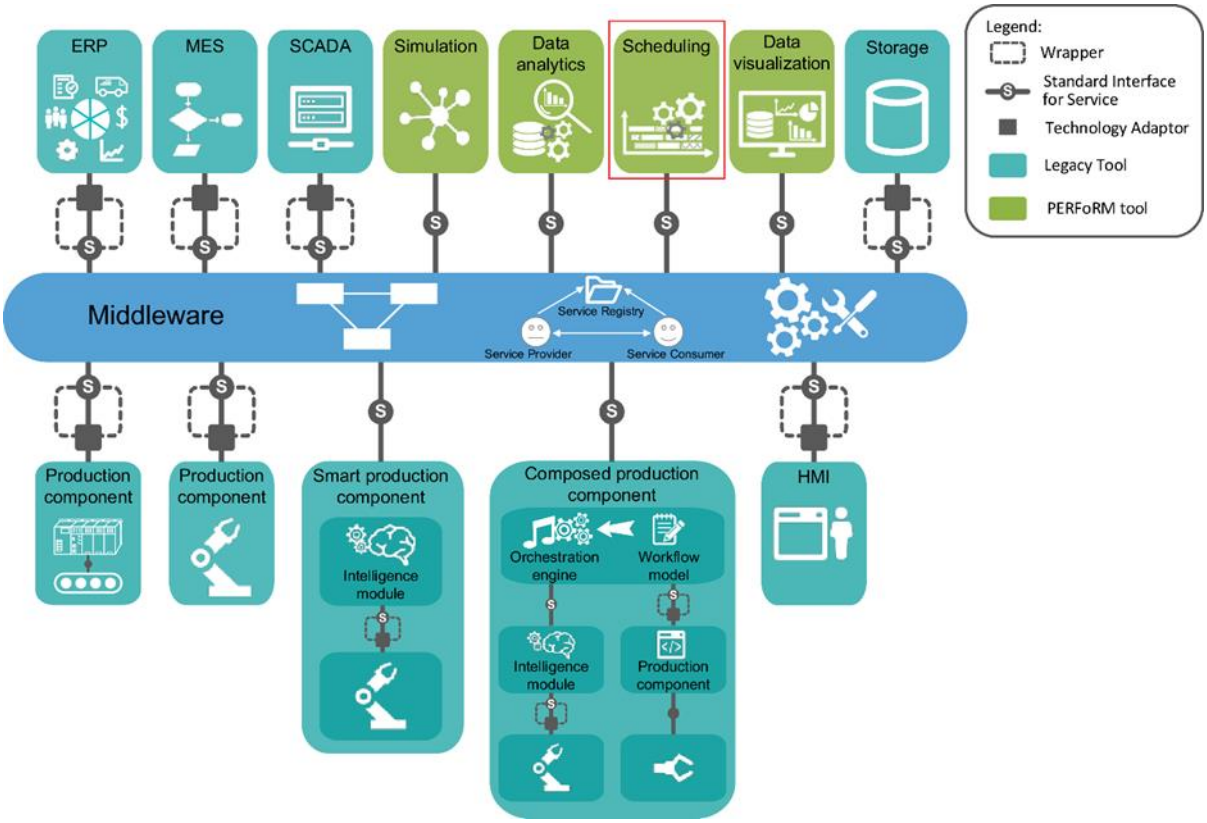

Figure 5.1 - Overview of the PERFoRM system architecture (PERFoRM, 2016)

Despite being outside of the scope of the developed work, it is relevant to provide a brief explanation of the system's data model (Figure 5.2), to understand which and how the classes of the system are relevant to the proposed scheduling methodology, and how they can be mapped.

This data model was developed to allow the PERFoRM-compliant elements to share the same data exchange format. Consequently, it is possible to interact between the scheduling tool and the middleware, allowing to access classes such as *PMLEntity, PMLSkill, PMLConfiguration, PMLProduct, PMLOperation and PMLSchedule*, which will be briefly described for more context in this scenario.

The PMLEntity class is the generic representation of shop-floor entities, which encapsulates all the necessary information associated to both components and subsystems. PMLComponent and PMLSubsystem classes extend the characteristics of a PMLEntity. This class allows the existence of generic collections of elements that can be either components or subsystems. The PMLComponent is basically a single entity that can perform skills and present relevant data regarding its state and operation. On the other hand, the PMLSubsystem is a recursive element

once it can contain other PMLEntities within. This generic design allows that for example a robot could be the lowest entity of a system, being treated as a simple component, and the same robot can be seen as a subsystem inside the system, involving different sensors as its components, using the same data model. PMLEntity class is related to the *Resource* class proposed earlier, while the PMLComponent is related to the *Station* class.

The PMLSkill class represents the tasks that a PMLComponent or PMLSubsystem can perform. A certain skill is characterized by a unique identifier, as well as by series of associated configurations and parameters. Like PMLEntity, PMLSkill is not directly used, enabling instead the creation of generic collections of PMLAtomicSkill and PMLComplexSkill. The PMLAtomicSkill represents the simplest form of a skill, a single action performed by a given entity. On the other hand, the PMLComplexSkill specifies a skill which consists in the combination of multiple skills, which can be atomic or complex.

The PMLOperation class is characterized by a unique identifier and contains all the information related to a task. Namely, the product to which that task is associated, the start and end times, the associated skill used to perform the task and the operation type, indicating if it is a production or maintenance task. In this context, the PMLOperation is related to the *Task* class proposed in this methodology.

The PMLConfiguration class provides a high-level description of a possible configuration to execute a given skill, according to a set of specified parameters. Is this class that provides the duration time of a task.

The PMLProduct class provides an abstraction of a given product variant, along with its core-defining characteristics to enable a process-oriented description of the product. The product is defined by a unique identifier and contains a collection of skills, representing the skills by which it can be produced. It can contain a link to an external description of its geometric characteristics, using for this purpose the COLLAborative Design Activity (COLLADA) interchange format, an opened standard XML schema for the exchange of 3D assets among software applications. In this case, it is directly related to the *Product* class proposed before.

The PMLSchedule class represents a unique schedule, associated with a certain station. It contains the start and end times and a collection of the corresponding associated PMLOperations, and so, it is associated with the *Schedule* class.

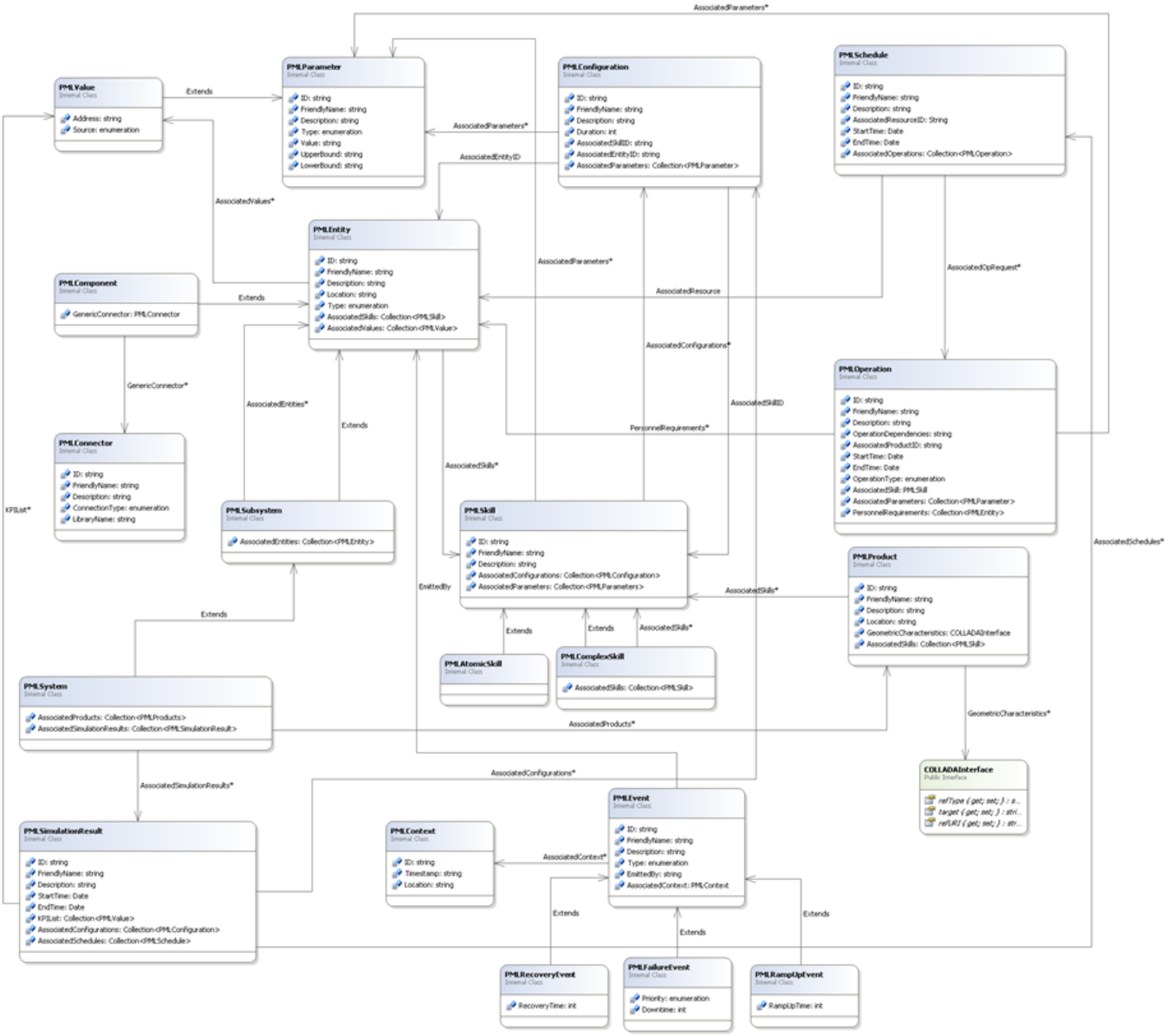

Figure 5.2 - PERFoRM class diagram at machinery level

5.2 Methodological Adaptation

In (Alemão, Parreira-Rocha, et al., 2019) was presented an architecture to solve a task allocation problem, namely a Job-Shop Scheduling Problem, considering production and maintenance tasks, aiming to minimize the total execution time in the shop-floor. As a final goal, these approaches should decrease delays and unexpected number of failures during the production process. This solution allows to reach more reliable schedules by considering the maintenance operations to be performed on the production stations. To solve this problem, a Genetic Algorithm solution was implemented where the goal is to optimize the task allocation based on the

information provided about tasks and working stations. In this section, this work will be analyzed to verify in each way it follows the guidelines of the methodology, and the generic architecture presented in this document and how it can be improved.

Figure 5.3 demonstrates the proposed architecture of the solution. Several tools interact with each other in order to gather the necessary data to generate the scheduling output solutions. This architecture can easily be divided into two parts that match with the high and low layers in the architecture presented in this document.

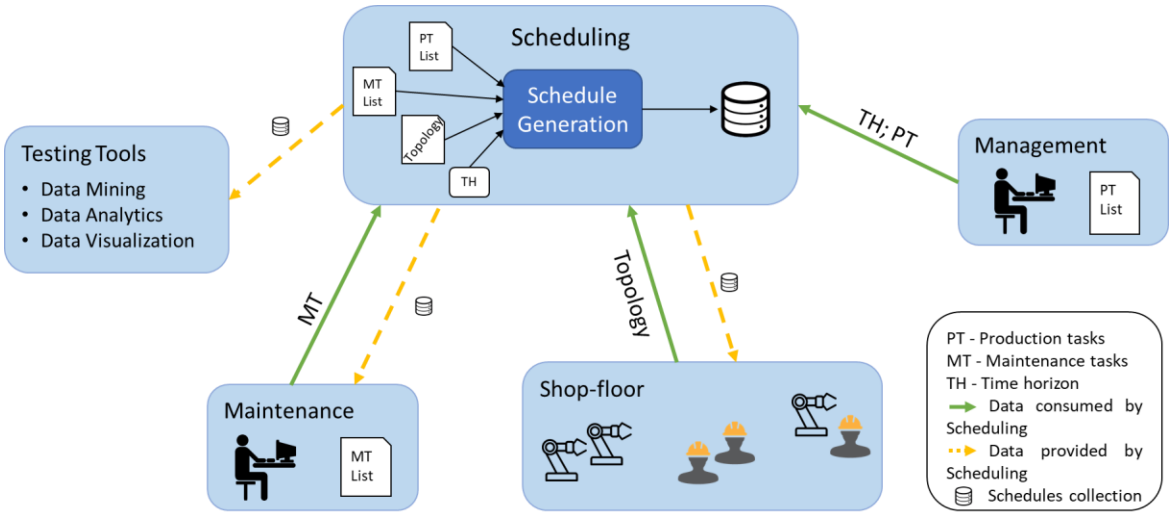

Figure 5.3 - System interactions overview (Alemão, Parreira-Rocha, et al., 2019)

One regarding the lower-level, i.e., the shop-floor, where the hardware assets are present. This includes available working stations, human workers, which will be decisive to allocate both production and maintenance tasks. A Graphical User Interface (GUI) is used to trigger new schedule orders by a human operator, which can also be done automatically by the system at predefined times. This is completely in accordance with the lower-level layer suggested in the architectures proposed in this work.

The second part is the higher-level, intended to host the software tools and applications. In this layer there is a management module, which can be an ERP, where are kept the orders from the customers to be processed. There is, also, a maintenance module, where the maintenance interventions are analyzed and maintenance tasks may be triggered, and the shifts for maintenance teams may be defined. Furthermore, there is the scheduling module, corresponding to the main module in that study. The scheduling tool collects the necessary data, such as production and maintenance tasks to be executed, the factory topology (which gives information about the working stations, and the time horizon to execute the schedule and, through the implementation of a Genetic Algorithm, generates a collection of three types of schedules

based on the gathered information. The adopted optimization technique allows to optimize the function cost, which in that case is to minimize the total production time of the entire shop-floor. The three types of schedules are a) allocate the tasks as soon as possible (which allows to free the machines sooner), b) allocate the tasks as late as possible (favoring a just-in-time production), and c) a hybrid approach which tries to maximize the distribution of the tasks and minimize the overall production time. To complement, there may be other software tools to complement the scheduling optimization. These tools may be used to perform data mining to explore the data coming from the shop-floor, data analysis searching for patterns or deviations that help to recalibrate the scheduling parameters, data visualization to present advanced output tools to help operators and decision-makers, and so on.

The approach in (Alemão, Parreira-Rocha, et al., 2019) is partially aligned with the methodology presented in this work. The scheduling optimization module interacts with other modules to collect important data to be processed, namely a management module (equated to an ERP system) to receive production and maintenance orders, and external tools to gather additional data from data analysis, or to provide visualization outputs. Furthermore, there is a connection with the shop-floor to acquire data related to the working stations or human workers. All the information is stored in a database. However, the components do not use a common middleware to communicate among them, instead they have point-to-point interactions, where each one interacts directly with the other one. This fact means that more communication points are needed, which brings more complexity in terms of design and implementation, since the components do not use the same protocols to communicate.

To implement the scheduling tool, it was developed an algorithm which processes all the information received from the PERFoRM middleware, generates a collection of schedules and sends it back to the middleware. The class diagram developed is presented in Figure 5.4, and specifies the interactions between the created classes and the main methods used in the algorithm.

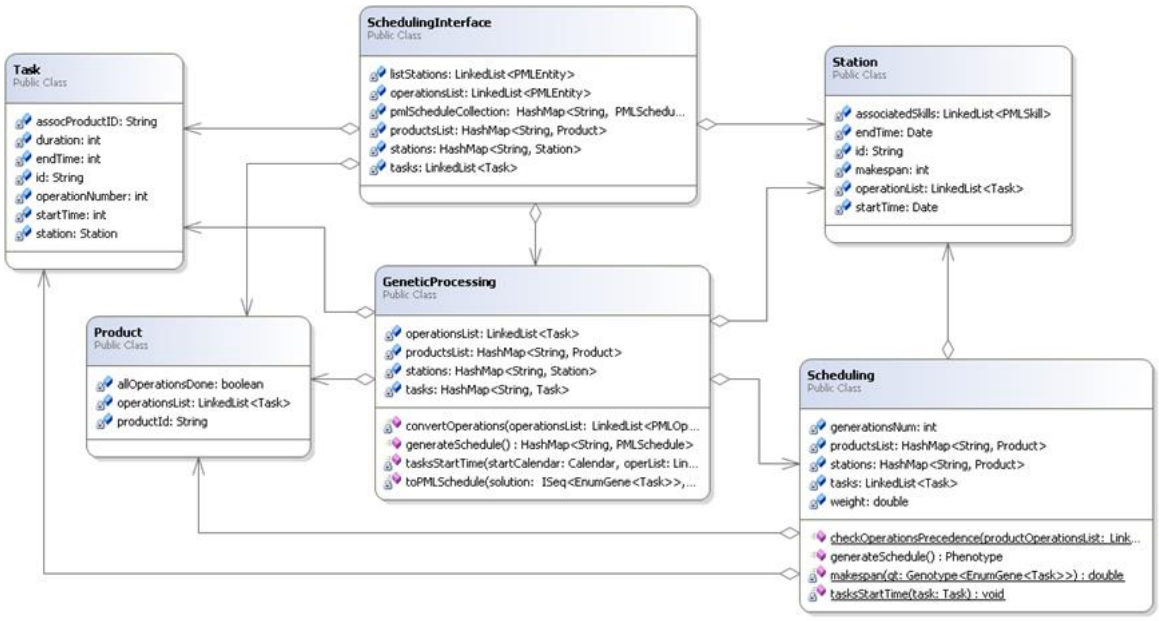

Figure 5.4 - Scheduling tool class diagram

The station class is used to instantiate each station available, thus, it is possible to access any of them. Each instance is characterized by a unique identifier, a collection of the skills which the station can perform, a start and an end time, a collection of operations to be performed by that station and the total time the station is planned to be operating. The Station class provides an abstraction of the working stations available on the shop-floor. It contains all the necessary information about the stations to perform the scheduling, including a collection of the associated skills it can execute, the total working time since the first operation starts till the last one ends, a collection containing all the operations which will be performed in the station and the start and end times of each station.

The *Station* class is closely related to the *Resource* asset proposed in Figure 4.7, with each serving the same level of abstraction and functionality. The *Resource* class provides detailed information about individual machines, including their capabilities, setup times, operational speeds, maintenance schedules, and real-time availability. This granular data is essential for managing each station's tasks and status. On the other hand, the *Station* class operates as one of the possible types of resources that may be grouped into work centers or production lines. In this way, it is simple to convert and slightly adapt a *Station* class into an AAS.

The Task class is the generic representation of the operations to be executed on the shop-floor. Converting each PMLOperation into a Task object makes it more generic, once it can be adapted to a different situation, and easier to handle the GA once the order of the operation (*operationNumber*) and the station associated are available in this class, but not in

the PMLOperation class. Otherwise, would be harder to access them each time a new task needs to be allocated. Similarly, the duration of the operation is not available in PMLOperation class, it is needed to access the *associatedSkill* parameter and then the configurations collection to access the D*uration* field. The start and end times of the operation are represented as an *int* which makes it easier to process the times once the duration is an *int* too. Each task contains an identifier and the associated product identifier that lets it know to which product it belongs to.

On the other hand, the Product class is the representation of the existing products on the shop-floor ready to be performed. Each one has an identifier and contains a collection of the operations needed to be executed. There is also a *Boolean* field to indicate if all the operations are done or not and, consequently, know if a product can already be scheduled, since some products depend on other to be executed. However, the precedence of products was not implemented in this work.

The Product class suggested in Figure 4.7 expands on the *Product* class from PERFoRM by providing a more comprehensive and structured approach to managing product-related information, as it aggregates products and tasks. It encapsulates all relevant details about the products being manufactured, including a unique identifier, a list of specific tasks required for production, and the stations needed to complete these tasks, which are assigned after scheduling. It also outlines the production process, detailing the sequence of tasks involved in manufacturing the product, and specifies the raw materials required. This level of detail ensures that all product-related data is organized and readily accessible, facilitating efficient production planning and execution. The PERFoRM *Product* class lacks detailing task sequencing and station assignment, as it is done at *Task* level, but by merging both of them or considering them independently, it is straightforward to implement an AAS for tasks and products.

On the other hand, the schedule-related information is divided into *SchedulingInterface, GeneticProcessing,* and *Scheduling* classes. The GeneticProcessing class basically treats the data before and after performing the schedule. It converts the collection of PMLOperations into a *HashMap* of Tasks, this way it is easy to access each one of them to alter the start and end times. The *generateSchedule* method instantiates the Scheduling class to obtain a scheduling solution and returns a collection of PMLSchedules. The *toPMLSchedule* method converts the solution obtained in the Scheduling class to a collection of PMLSchedule, assigning the times to each PMLOperation, ready to be sent to the middleware. Finally, the *tasksStartTime* method translates the times of each operation coming from the Scheduling class into Date type values. This class resembles with the *Scheduling Tool* component proposed in Figure 4.7

as it is the component responsible by managing the scheduling and perform operations related to it. One of its submodels may be the equivalent to the *Scheduling* class, which implements the Genetic Algorithm. *Scheduling* class stores the collections of operations, products and stations needed to execute the algorithm. The *generateSchedule* method sets the parameters of the GA and starts the algorithm. The *eval* method is used as the fitness function and evaluates the algorithm. For that, *tasksStartTime* method assigns the start and end times to the operations, according to the sequence of operations received and the *checkOperationPrecedence* method verifies if the precedence between operations is respected. There is a variable, *weight*, which is used to increase the value of the fitness function if some task is in a wrong order. Thus, it may be considered a specific parameter for the *ScheduleConfiguration* submodel which consists of applying a Genetic Algorithm to generate new schedules.

The *SchedulingInterface* class implements the interface where it is possible to generate new schedules and add maintenance tasks manually to the list of tasks to allocate. When initialized it gets the information of the stations available on the shop-floor and then the operations to be performed by those stations. This information is obtained from files stored in the middleware and are, posteriorly, converted to the respective classes. The stations are converted to a Station object and stored in a Hashmap collection identified by its ID. On the other hand, the operations are stored in a list of PMLOperation and each time an operation of a non-existing product arrives that product is created and stored in an Hashmap to, later, be possible to identify the precedence of operations in each product. This class matches the *GraphicalUserInterface* submodel proposed in the scheduling methodology, as it consists mainly in the interaction to execute commands as generating new schedules or adding new tasks.

The data acquisition is done through the PERFoRM middleware and the GUI, as demonstrated in Figure 5.5. Each time a new schedule is requested, the algorithm gets the necessary data to perform the task allocation, namely production operations, maintenance operations and stations available on the shop-floor.

Both production and maintenance operations are loaded from the middleware and stored in a common *LinkedList* of operations. Each new entry of the list is stored on the form of a *PMLOperation*. Also, the maintenance tasks could be inserted from the GUI interface. In this case, they are stored in a *LinkedList* with the same format, where they can be removed, until a new schedule is requested. At this point, the maintenance tasks are added to the common list of operations.

Each operation present in the file is converted to a PMLOperation object and contains necessarily an operation ID, an associated skill, with the respective ID and configuration list and

the operation type, production or maintenance. Each configuration should have the associated entity ID and the duration of that operation. In the production operations case, it should, additionally, contain the operation number and the associated product ID.

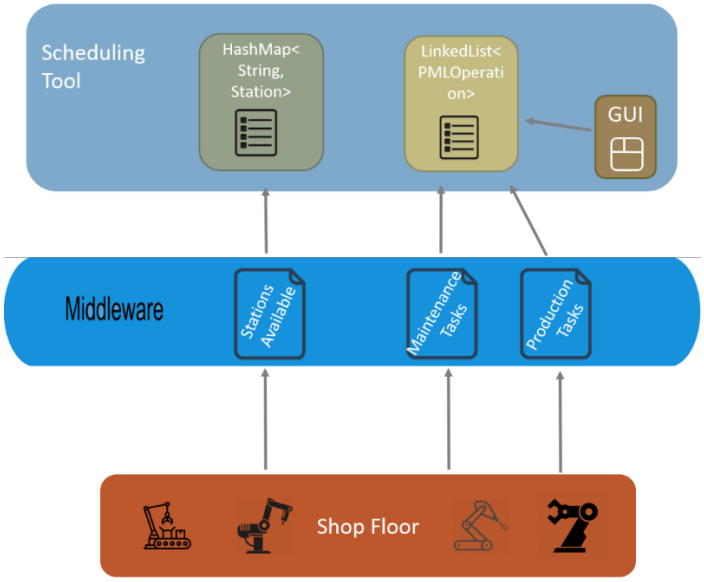

Figure 5.5 - Data acquisition from PERFoRM

Moreover, it is necessary to get information about all the stations on the shop-floor available to execute operations. That information is stored in a *HashMap*. The key to each entry in the *HashMap* is the station ID and each station is stored in the *HashMap* over the format of a *Station* object. So, the information is loaded from the middleware and for each different station present on it a new *Station* object is instantiated, containing the ID and the skills of each station. Each skill is a PMLSkill instance and contains an ID, the corresponding station ID, task's duration, and a PMLConfiguration collection, since each skill could have different configurations, depending on which station it is executed.

Additionally, was implemented an *HashMap* to store the different products whose operations were loaded from the middleware. The key to each entry is the product ID and each product is stored in the format of a *Product* object. It contains an ID and the number of operations to be executed in that product. In this way, is possible to check the precedence between products.

By defining the middleware as an AAS, the integration with the scheduling tool is facilitated and the previous data may be harmonized and well-structured within the scheduler AAS as properties required to executing the schedule, making easy to communicate between the components of the system to exchange information.

Production order and *Shop-floor management* were very simplified in PERFoRM approach. *Production order* was done by simply sending files through middleware, where the scheduling tool could access them. Those files already had the right data structure to make it easy for scheduling tool to treat them. With an AAS implementation the data in those files can be established to structure the way orders should arrive to the system, and from there any IT tool will know how to access it and treat the data. It may also be designed within an ERP AAS, making it more complex but more robust.

Little constraints to the shop-floor operation were considered in PERFoRM, only for working hours and different shifts. *Shop-floor management* AAS should be created in order to take into consideration the requirements and constraints of the shop-floor, related to working hours, available shifts, available stations or transportation options. In this way, access to this information will be facilitated and harmonized, as all data will be very well defined.

Briefly, it is possible to take each PERFoRM component, as showed in Figure 5.6 (left) and transform it in an AAS, a submodel of an AAS or even a property (right). Thus, it provides a standardized digital representation of the assets, enabling seamless communication and interoperability across different components and systems. This facilitates easier integration of assets into digital ecosystems, enhancing the efficiency of data exchange and process automation. Furthermore, it ensures scalability, allowing manufacturers to expand their operations without significant reconfiguration, and it enhances transparency and traceability by providing comprehensive, up-to-date information about assets throughout their lifecycle, which aids in better decision-making.

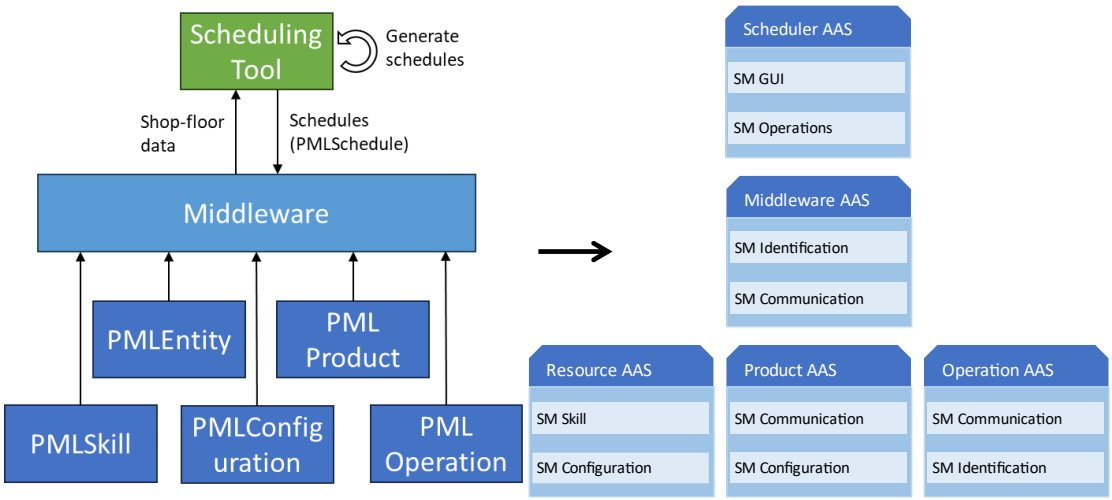

Figure 5.6 - PERFoRM approach adapted to AAS (right)

Another possible approach, simpler than the previous one, is to consider each PML component as an asset or a submodel, as part of a broader Middleware AAS (Figure 5.7). The PERFoRM Middleware can be considered an AAS that manages all the entities, the same way it already does, thus forming a nest of assets. In this way, it is necessary to define the properties and submodels for the PERFoRM Middleware AAS. Then, the scheduling tool is modelled independently and interacts directly with the Middleware AAS, which acts as a black box.

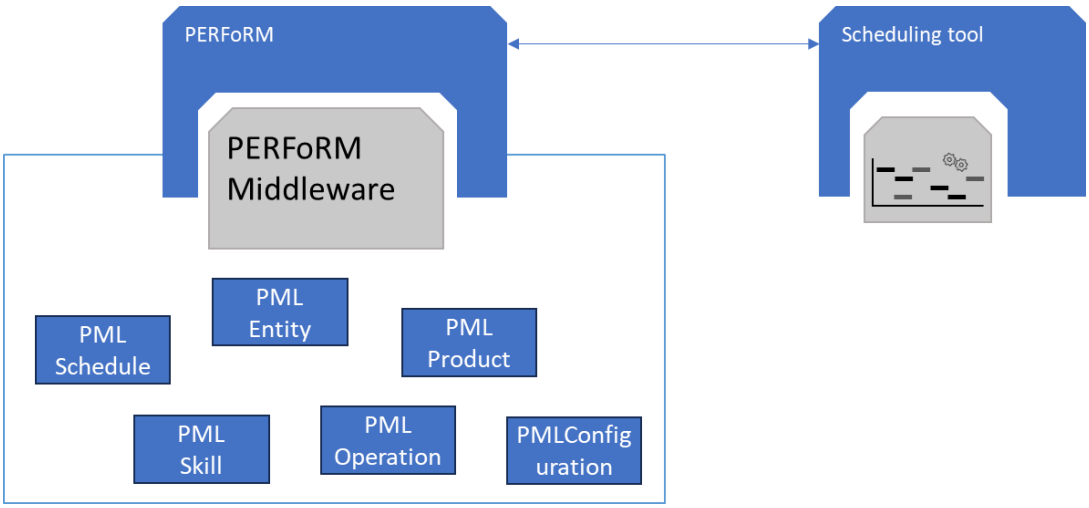

Figure 5.7 - PERFoRM Middleware and Scheduling tool adapted to AAS.

5.3 Relation to DPs

In this sub-section is presented the relation between the DPs for developing the scheduling system, and the PERFoRM project.

Design Principle	Covered	How is it related?
1.1.1 - Digital factory replica	✓	Assets stored in PERFoRM's AutomationML server
1.1.2 - Communication protocols	✓	Standard interfaces between middleware and other components
1.2.1 - Common interfaces	✓	Common interfaces provided through existing APIs
1.2.2 - Data model	✓	Data model defined in PERFoRM project and at the scheduling tool level
1.2.3 - Uniform data among components	✓	The data format is harmonized between the different components of the system
1.3.1 - Harmonized KPIs description	✓	KPIs defined on PERFoRM system
1.3.2 - Save production objectives	✓	Minimize makespan and improve maintenance tasks allocation

1.3.3 - Editable KPIs priorities		Not considered
1.3.4 - Considering different KPIs for schedule generation	✓	Different KPIs are considered, allocate tasks as soon as possible, later, or with better resource balance
1.4.1 - Define boundaries	✓	Boundaries defined on PERFoRM system
1.4.2 - Evaluate production process evolution	✓	Evaluation of the production process executed by monitoring tools
1.4.3 - Analyze new orders	✓	New orders lead to a reschedule of the system
1.4.4 - Automatic rescheduling		Not considered
1.5.1 - Schedule validation	✓	The schedule output is provided for validation
1.5.2 - New schedules request	✓	The user can order for new schedules at anytime
1.5.3 - Choose KPIs	✓	The user can choose which KPIs to adopt
1.5.4 - Human worker inputs		Not considered
1.6.1 - Implementation of different optimization techniques		Not considered
1.6.2 - Communication with external systems	✓	Communication established between the scheduling tool and PERFoRM ecosystem
1.6.3 - Start schedule not before the current state	✓	It is guaranteed that the schedule is always started in a later time then the current one

In the PERFoRM project, the DPs are closely linked to the development of the scheduling system. Communication protocols, common interfaces, and a unified data model are integrated across components. The KPIs defined in PERFoRM focus on improving makespan and maintenance task allocation, with flexibility to adjust KPI priorities and optimize scheduling based on different resource considerations. The system evaluates production processes and allows for automatic rescheduling when new orders arrive. Additionally, users can validate schedules, request new ones, and choose which KPIs to adopt. Communication between the scheduling tool and the PERFoRM ecosystem ensures seamless operations.

6

APPLICATION SCENARIOS

In this section are presented two different applications scenarios. The first one developed in a NOVA University of Lisbon's laboratory, using a demonstration kit, while the second one was developed in collaboration with MUVU Technologies and a plastic injection molding factory.

6.1 Laboratory scenario

This section aims to implement the proposed methodology in a laboratory scenario, following the defined DPs. For this purpose, it was used a demonstration kit simulating a production line, where the scheduling output was applied to.

6.1.1 Scenario Definition

The demonstration kit composed of four stations two transportation methods, as demonstrated in Figure 6.1. The stations simulate different functions: station "1" applies glue; "2" has the functionality of screwing; "3" functions as a welding machine; and station "4" is for painting. The transportation method is a conveyor belt that serves stations "1" and "2", while the transport between other stations is done by a human operator.

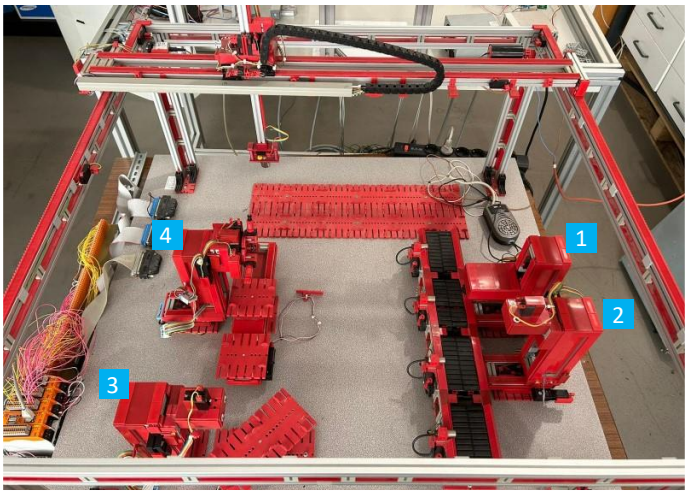

Figure 6.1 - Production line demonstration kit

The modelling of the physical assets composing the production line was done using AASX Package Explorer provided by Platform Industrie 4.0 (Plattform Industrie 4.0), which was chosen over other possibilities (BaSyX, NOVAAS, or RACAS Wizard) since it was developed by the mentors of AAS, it is simple to use, and it has been updated regularly. It includes features like an internal REST and OPC UA server for interacting with AAS packages. The tool facilitates easy interaction with the data related to each asset, and enhancing interoperability between various components of the system. It also facilitated the organization of asset data into structured, standardized formats, ensuring that asset's information could be effectively integrated into the production scheduling workflow.

The AAS models were structured to represent the physical assets as stations, products, and the shop-floor, but also the scheduling tool. Thus, it was created four AAS for the stations, four for the products that may be produced in this system, one for the shop-floor and one for the scheduler.

This production line demo is represented as a shop-floor AAS in Figure 6.2. All the relevant information for the scheduling implementation is described in detail.

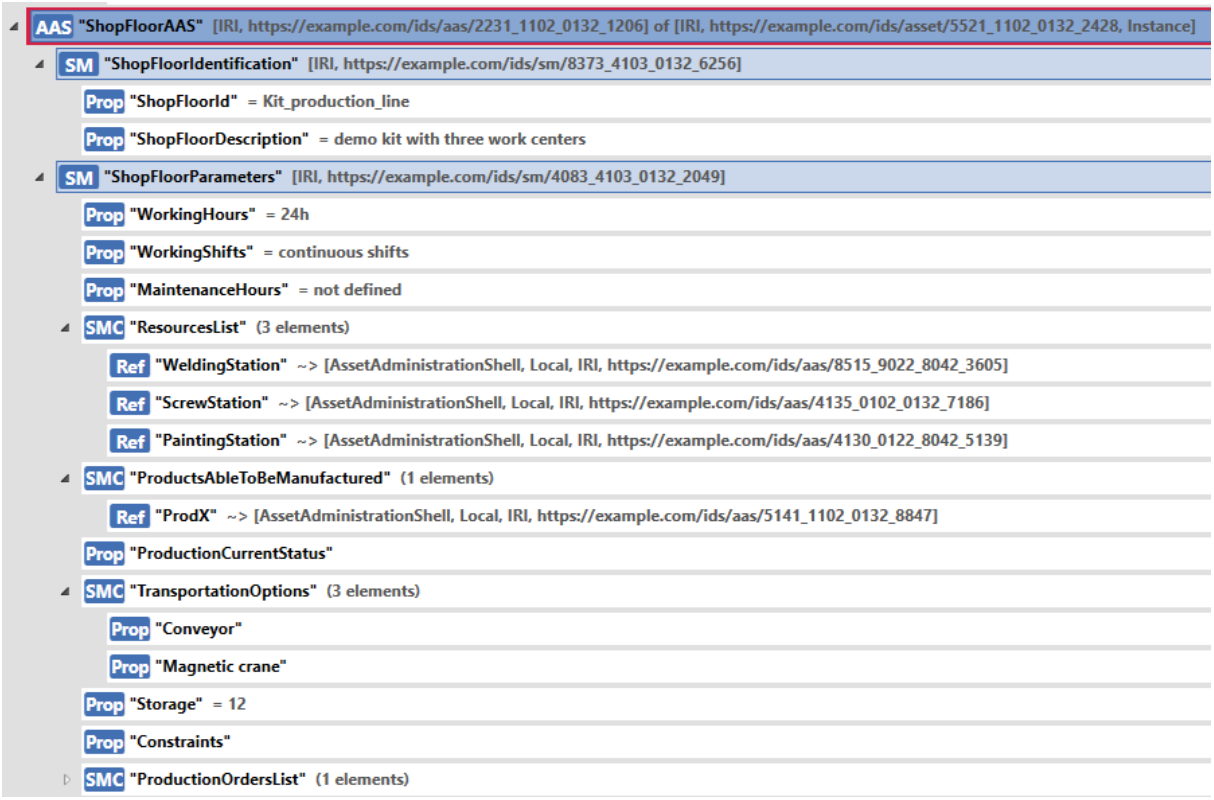

Figure 6.2 - Demonstration kit AAS instantiated in AASX Package Explorer tool.

It contains a submodel for the identification and another submodel for parameters related to the shop-floor. As it is possible to observe, it already contains references to stations

(the same named previously) and products that are available in this demo. The shop-floor contains a *ProductionOrdersList* parameter, which lists the products to be allocated. Each product type in that production order has its own AAS.

The reference for a specific station allows to access all data about that station, as represented in Figure 6.3. The figure shows the representation of the screwing station AAS, composed by its identification, including the ID and the location within the shop-floor, the communication interfaces, and specific configurations of the station.

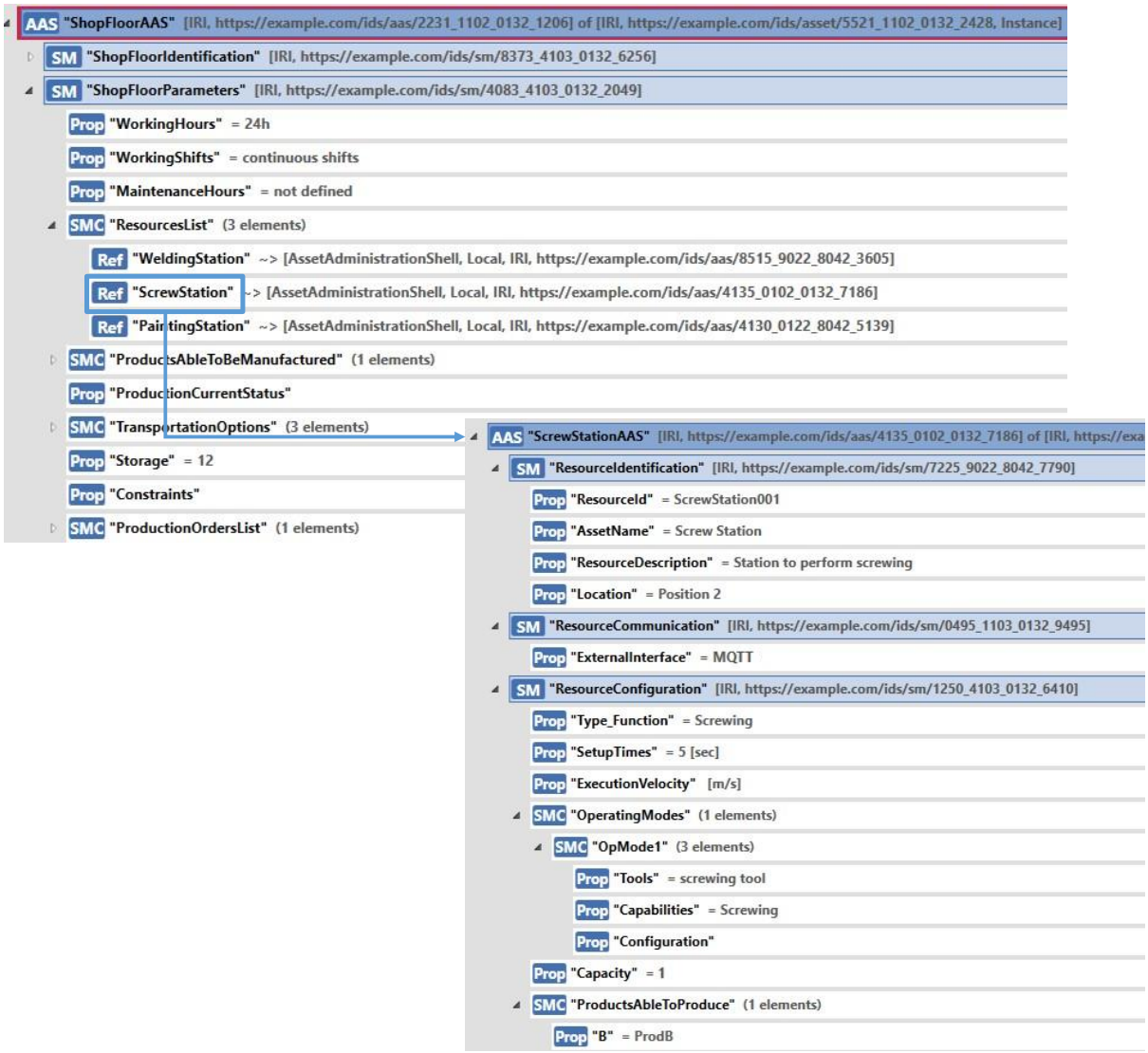

Figure 6.3 - Connection between the shop-floor and screwing station AAS

The scheduling tool AAS, detailed in Figure 6.4, is composed of different submodels that allow to access asset's identification and information, such as the products allocated in the

schedule, to configure a new schedule, to execute new schedules, or access the information of the most recent schedule.

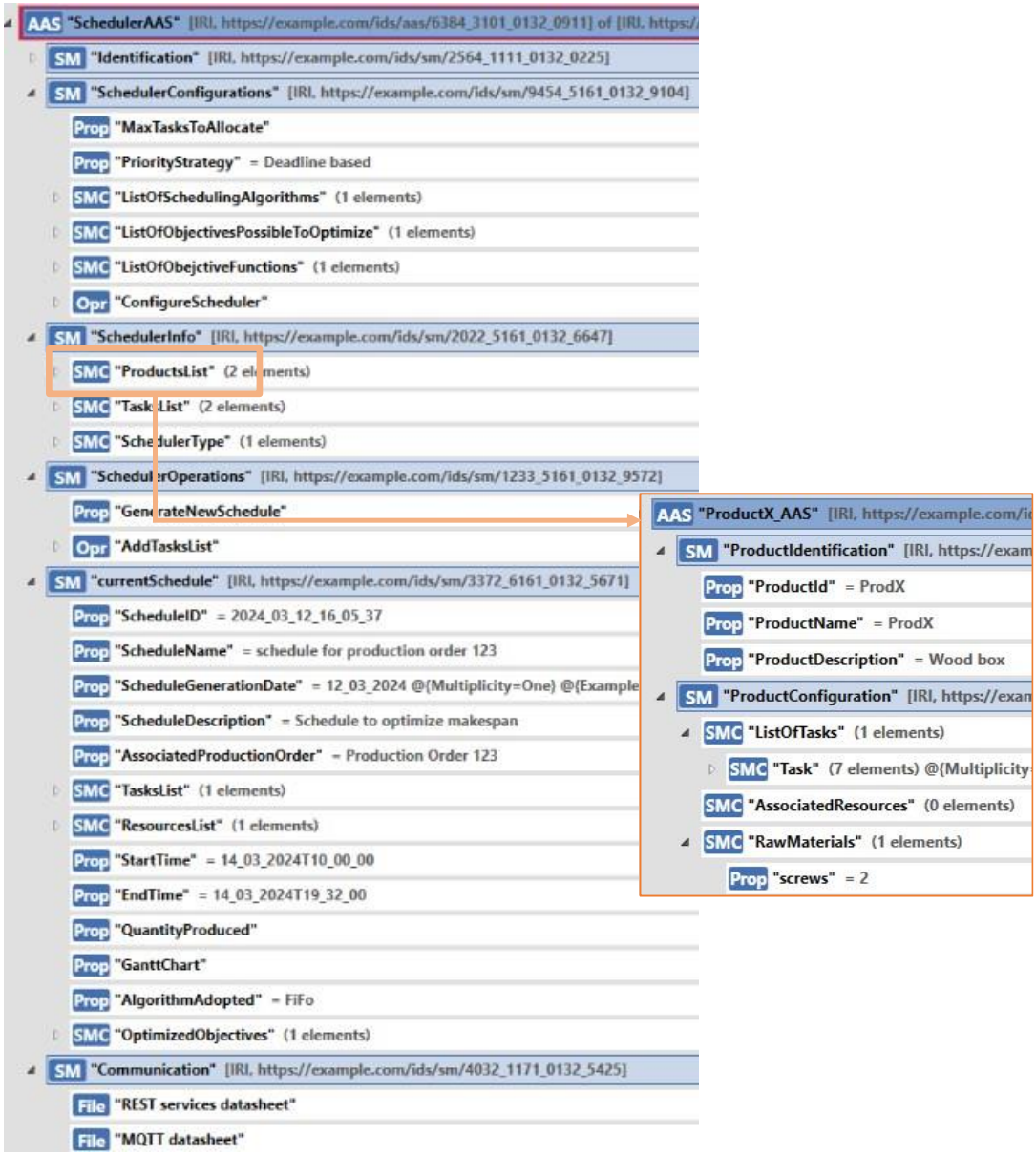

Figure 6.4 - Scheduler and Product AAS

This definition of the use case and the description of its assets serves as the foundation for its implementation, as detailed in the next sub-section.

6.1.2 Implementation

The AAS-based architecture adopted for the implementation of this scenario comprises all the previously defined assets, the adopted communication protocols, and technologies used to implement the manufacturing scheduling system, as illustrated in Figure 6.5. It covers three

levels, namely the physical world where the physical machines and products are, the cyber world encompassing the scheduler system and the AASs server, and the middleware which does the connection between the first two, gathering data from the shop-floor and sending execution plans.

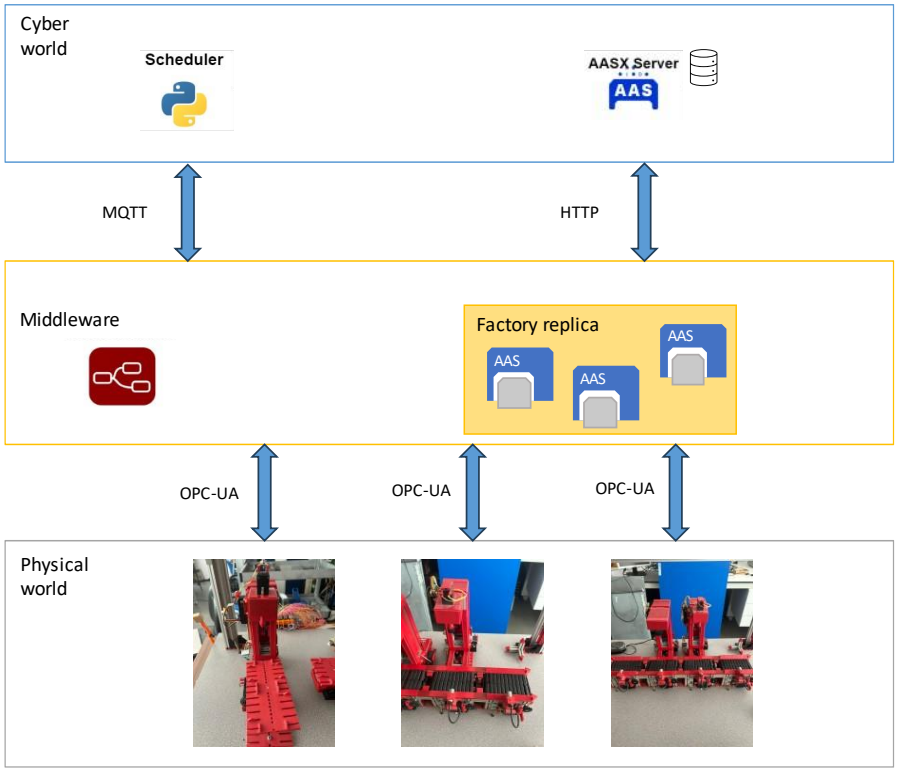

Figure 6.5 - The AAS-based architecture for manufacturing scheduling

Here is outlined the production scheduling system process, 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 6.6.

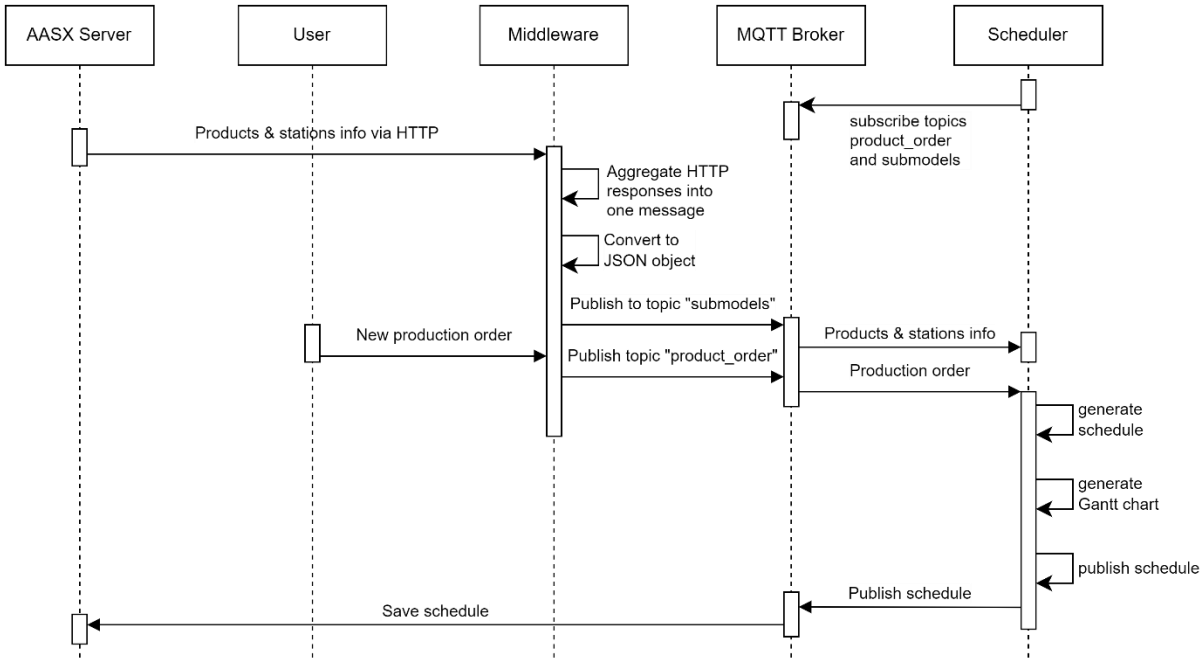

Figure 6.6 - Schedule process overview

The system starts by retrieving data from the AASX server using the Node-RED middleware. Through HTTP requests, information about stations and products is fetched from the server and wrapped in a structured payload. This ensures the production scheduler has up-to-date information on the capabilities and status of all relevant assets.

Once the data payload is received, Node-RED processes it in a function node, where URLs are constructed using encoded values. These URLs facilitate subsequent HTTP requests to obtain additional data from the server, such as operational parameters for stations.

The responses from these HTTP requests are aggregated into a unified message payload using a join node in Node-RED. This step consolidates all relevant information about stations and products into a single dataset. The aggregated data is then validated to ensure it conforms to JSON standards, with any invalid entries discarded, resulting in an array of JSON objects that accurately represent the current state of stations and products.

Once validated, the refined JSON data is published to the *submodels* MQTT topic. This topic acts as a communication channel, enabling the production scheduling system to receive up-to-date information on station functions, product specifications, and other critical data. By subscribing to this topic, the scheduler ensures it consistently has the latest data to guide its decision-making process.

At the same time, the scheduler system is listening to MQTT topics, including *product_orders*, where new product orders are received. When a user sets a new order, the scheduler

decodes the message payload to extract the order details. These orders are then queued for processing via the *process_pending_orders* function, ensuring timely handling and scheduling of production tasks. The *process_pending_orders* function iterates through the queued orders, extracting product requirements for each one. It then assigns suitable stations based on their availability and capabilities. This assignment process ensures that product specifications are matched with station functionalities for optimal task allocation.

After assigning stations, the Gantt chart is updated using the *matplotlib* library to generate Gantt charts, allowing to visualize the production schedule by plotting tasks' start and end times on specific stations. This visual representation helps to identify potential bottlenecks and aids in optimizing the scheduling process for improved efficiency and productivity.

Additionally, the production times are saved to a JSON file to ensure the persistence of data. This file serves as a historical record of the production schedule, facilitating traceability and analysis of past operations. The scheduling data is published to a predefined MQTT topic, *Production_schedule*, enabling other systems to access and stay updated on the current production plan.

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

The implementation of each of these components will be detailed in the next sub-sections.

6.1.2.1 AASX Server

The AAS database is managed using an AASX server, deployed through Docker, illustrated in Figure 6.7. Docker is a platform that enables developers to package applications and their dependencies into standardized units called containers. These containers are lightweight, portable, and ensure consistent performance across different computing environments. By using Docker, it is possible to encapsulate the AASX server along with its configuration and dependencies, guaranteeing reliable operation regardless of the deployment environment. This setup provides a robust and scalable solution for hosting and managing AAS data, ensuring that it is readily accessible for various applications, including the production scheduler.

The AASX server is launched using a *Docker Compose* configuration, which defines how to set up and run Docker containers as part of a single application, and is configured to restart automatically unless stopped manually, ensuring high availability. The AASX server overview is

demonstrated in Figure 6.7, and the server was accessed via HTTP at "http://localhost:8000". The server configuration directs the server to use an external Blazor application hosted at the same URL.

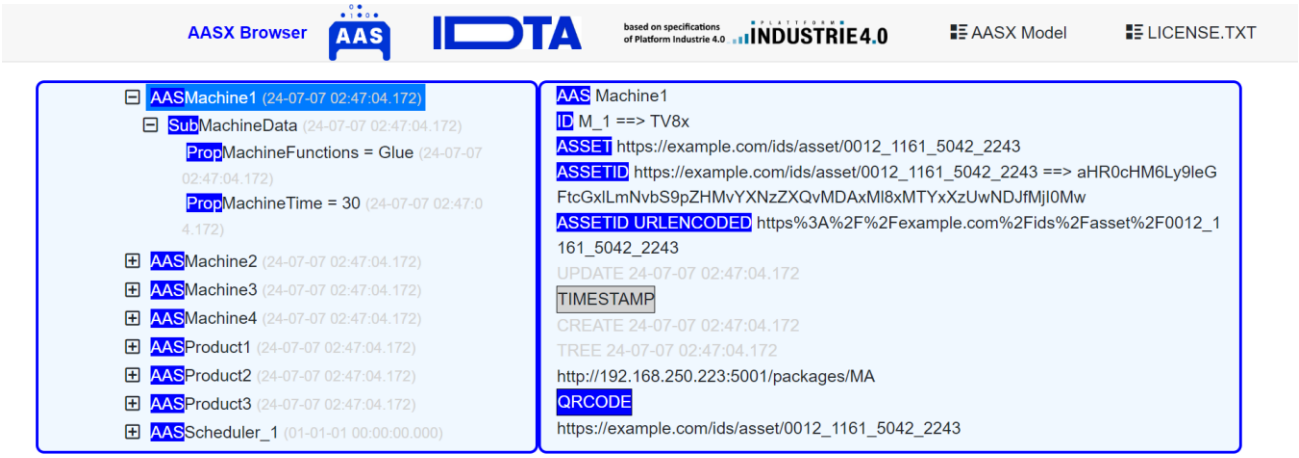

Figure 6.7 - AASX Server overview

The *volumes* configuration in Docker allows the server to access and manage the AASX files stored on the host machine, ensuring that any changes made to the AAS data are persistent, even after the container restarts or updates.

In addition to hosting the AAS data, the AASX server includes a Swagger API, offering a standardized interface for interaction with the AAS data. The API requires that data be base64 encoded, ensuring safe transmission without corruption. This encoding is handled in the middleware, ensuring that all data exchanged with the API meets this requirement.

Using Docker for the AASX server provides several key advantages, such as portability, consistency across different environments, and simplified deployment. The addition of a Swagger API enhances the server's capabilities by enabling seamless access to the AAS data. This setup not only supports the efficient management of AAS data but also allows smooth integration with the production scheduling system.

6.1.2.2 Scheduler

The scheduler consists of several components, including the MQTT broker, the scheduler algorithm, and the Gantt chart generator. The MQTT broker acts as the communication core, enabling message exchange between the scheduler and the middleware. The scheduler, implemented in Python, subscribes to specific MQTT topics to receive data on stations, products, and orders, and maintains records of product orders, station availability, and usage times. The Gantt chart generator, utilizing the *matplotlib* library, allows the visualization of production

schedules. Additionally, the scheduler is wrapped within an AAS, for efficient management and interaction with other assets.

The flowchart showing the steps within the scheduling system is represented in Figure 6.8, and will be described with more detail in this section.

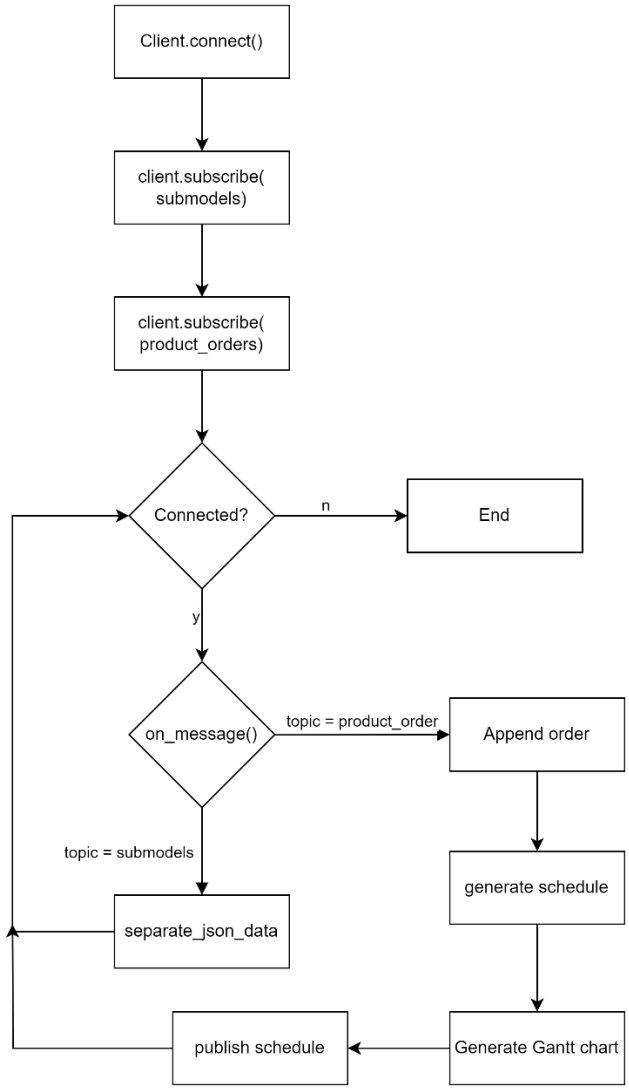

Figure 6.8 - Flowchart detailing the scheduling system.

After the scheduler successfully connects to the MQTT broker, it subscribes to two main MQTT topics: *submodels* and *product_orders*. The first topic provides data on stations and products, while the second one provides the incoming product orders. While the scheduler is connected to the broker, it keeps listening to these topics.

When messages are received on the *submodels* topic, containing stations and products data, they are separated and stored in different JSON files for stations and products, to facilitate the access and comprehension of both lists.

On the other hand, when a message is received on the *product_order* topic, it extracts the product order information and adds it to the *product_orders* queue, namely products' IDs. If there is already a pending order, it appends the second order to the existing one. It then calls *generate_schedule* function to handle the new order immediately. The main functionality of the *generate_schedule* function is to assign products to stations, which is essential for efficiently managing the allocation of stations to process product orders. It evaluates the product's requirements and determines the most suitable stations based on their availability. Besides the processing times, it also considers a wait time between the completion of one task and the start of another to allow the products transition between stations. The function starts by evaluating the product requirements, allowing it to process each requirement step-by-step across all products. For each step, it iterates through the requirements, assessing available stations that meet the criteria, and allocates them to the next available time. If multiple stations can execute the task, the one with the earliest available time is selected, to ensure tasks are assigned as soon as possible. The selected station’s availability is then updated to reflect the end task's time, and the start and end times for each task are determined.

These times are determined by considering the station's current availability, the processing time required for the task, and a defined wait time interval, for product transport. The processing time is retrieved from the station JSON data, which specifies how long each station takes to complete a task. The start time for a task is set to the later of the station'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 keeps track of start and end times for each station assignment, ensuring accurate updates during the scheduling process. If no suitable station is available for a given requirement, the scheduler informs the system. The pseudo-code for this function is demonstrated in Figure 6.9.

```
Find suitable stations for current requirement:
  - Call find_available_machines_for_requirement()
  If suitable stations found:
     Iterate through suitable machines:
          - Select station available sooner
     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 station choices, start times, and end times
     Update station availability
     Update product time
  Else:
     Print message indicating no available  stations found
```

Figure 6.9 - Pseudo-code for assigning stations

The next step is to provide a clear visual representation of the production schedule. The visualization process focuses on displaying the results in a Gantt chart. For this, the start and end times of tasks and stations are used to generate a Gantt chart, which provides a visual timeline of stations allocation. For different product instances, it assigns different colours, ensuring that repeated products are distinguished. Each product is given a unique identifier, and a colour map is created to facilitate visual differentiation. Using the *matplotlib* library, the function plots horizontal bars representing each station's tasks on the Gantt chart. The Gantt chart is further configured to include labels for stations and a legend for products, making it easy to interpret the schedule. Overall, the Gantt chart provides an intuitive and visually appealing representation of the production schedule.

The processed scheduling data is stored in a JSON file, keeping the details of station assignments, including start times, end times, and processing times for each task and maintaining a record of the production schedule.

Finally, the function publishes this scheduling data to a predefined MQTT topic, *Production_schedule*, enabling other systems to access the generated schedule, which allows other systems to access the latest schedule. This approach facilitates ongoing monitoring and integration with other production management tools, enhancing the overall efficiency and transparency of the scheduling process.

6.1.2.3 Middleware

The middleware was developed using Node-RED, a flow-based development tool. Node-RED enables seamless integration and management of data flows between different systems and components. This middleware was designed to get data from a database and process it, encode necessary information, generate specific URLs, and then publish the processed data to an MQTT topic.

Figure 6.10 provides a detailed illustration of the Node-RED, showing the sequence of nodes used for tasks such as data retrieval, encoding, URL construction, HTTP requests, data aggregation, JSON processing, and MQTT publication. This flow diagram visually represents the step-by-step process described below.

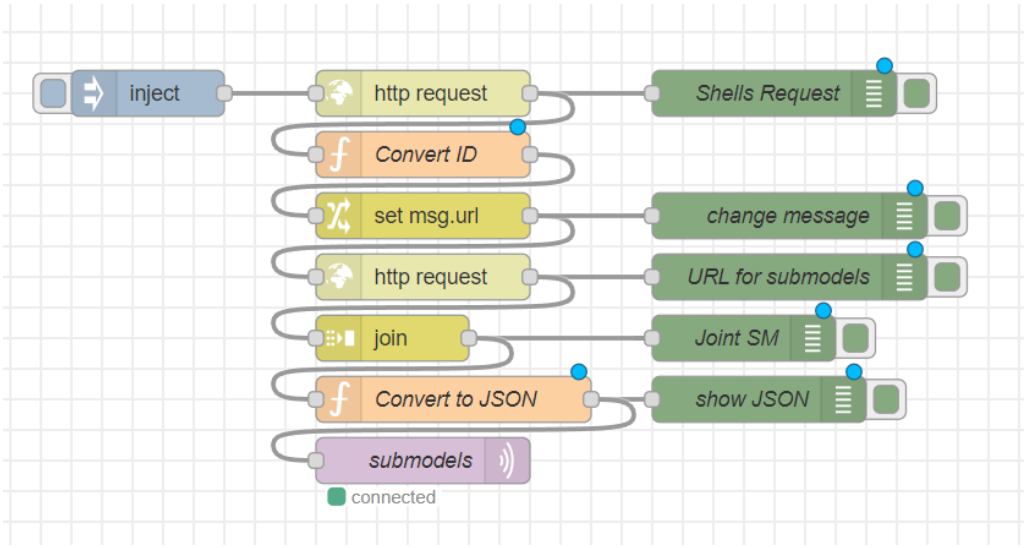

Figure 6.10 - Flow for AAS Data retrieving and processing

The process starts with an HTTP request, which fetches all the AASs (like the one present in Figure 6.11) from the database. This request is configured to obtain information on stations and products, encapsulated within a structured payload. Once the data is fetched, it is forwarded 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 6.11 - Example of a AAS gathered by the HTTP request

The function node then processes the message payload by iterating through each item in the array. For each item, it extracts the *id* ("M_1" in Figure 6.11) and the first submodel value ("SM_M1". If either value is missing, a default value is assigned, and a warning is logged. These values are then used to construct a URL, with the encoded *id* and submodel values included as query parameters. The URL is set as the payload of a new message, which is sent out.

After encoding and construction the URLs, the middleware uses a "set message URL" node to assign the constructed URLs as the payload. Another HTTP request node is then used to send these URLs and retrieve the corresponding data from the server. The responses from these HTTP requests are serialized, and can be interpreted as JSON strings, containing the information required for the scheduler, as Figure 6.12 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 6.12 - Example of a serialization taken by the HTTP request

The aggregated data, now an array of JSON strings, is forwarded to another function node. This function checks whether the payload is a valid array and whether each string in the array is a valid JSON. It then converts each valid JSON string into a JSON object, disregarding invalid entries. The resulting array of JSON objects is then set as the new message payload.

Finally, the processed data, now in the form of a JSON object array, is published to the *submodels* MQTT topic. This ensures that updated station and product information are made available to the production scheduling system, allowing it to retrieve the most recent

information. The seamless flow from data retrieval to processing and publishing ensures that the scheduler receives accurate and timely updates, facilitating efficient production management.

In addition to handling station and product details, the middleware includes a separate flow to manage product orders as shown in Figure 6.13.

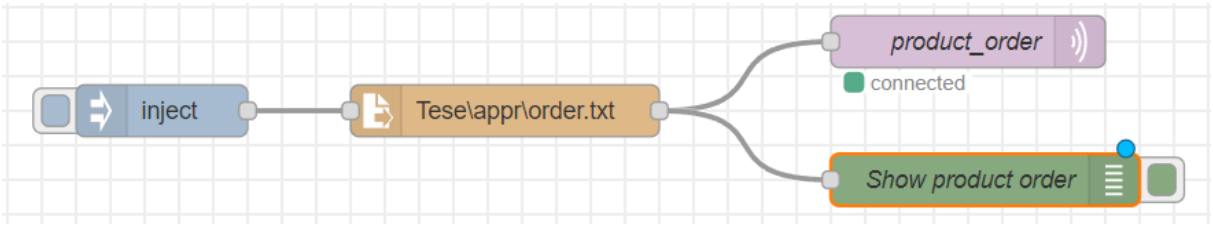

Figure 6.13 - Flow for product order retrieval

This flow begins by reading the product order data from a specified file. The content of the file, representing new orders, is then passed to an MQTT node, which publishes the product order data to the *product_orders* topic. This ensures that the production scheduler is updated with new orders arriving, enabling the system to process incoming orders efficiently and maintain an up-to-date production schedule.

Figure 6.14 illustrates the process of updating the product list, showing a flow that receives product orders via MQTT. In this flow, a function node sets the message headers to accept all content types and specifies the content type as JSON. The message payload is then converted into a JSON string with a key named *Products_List* that contains the received product order. Next, a method updates the *products_list* property within the information submodel of the scheduler with the latest product order. Finally, an HTTP request node completes the update process, by sending the update to the AAS.

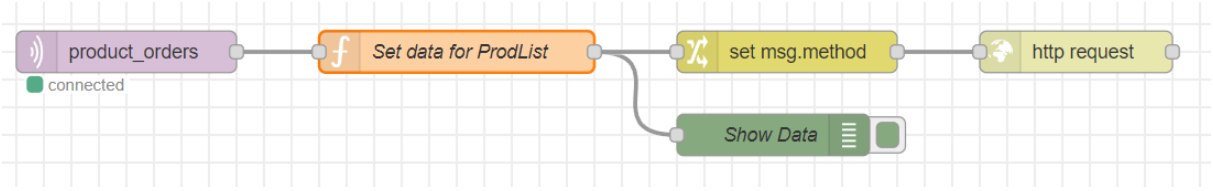

Figure 6.14 - Flow responsible for Updating the Product List in the Scheduler AAS

6.1.3 Results

This sub-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 assess the system's performance, several simulated test scenarios were developed, each involving different numbers of products and stations. The main objective was to evaluate how the system adapts to changes in workload and station availability, providing insights into its capability to manage diverse production demands. Additionally, the test on the laboratory kit was performed.

For some of these tests, the stations in the laboratory scenario were duplicated, having eight stations, where each pair of stations was set to perform the same function to simulate increased production capacity like follows:

- M_1 and M_5 perform the glue function.
- M_2 and M_6 perform the screw function.
- M_3 and M_7 perform the weld function.
- M_4 and M_8 perform the painting function.

The selected configuration allows to evaluate the scalability of the system when additional stations are available. By doubling the number of stations assigned to each function, the tests evaluated the system's ability to manage more tasks simultaneously, demonstrating its effectiveness in reducing idle times and enhancing overall production efficiency.

The tests were carried out across various scenarios to evaluate 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 five products in the laboratory kit.

In test 1, the system was dealing with scheduling five products across four stations. The Gantt chart, presented in Figure 6.15, demonstrates the schedule with effective task allocation, resulting in a compacted schedule. The computation time was, on average, 0.34 seconds, from the arrival of a new production order until generation the Gantt chart. This test proved that the system can handle simple scheduling scenarios with a manageable number of stations and products, laying a solid foundation for more complex scenarios.

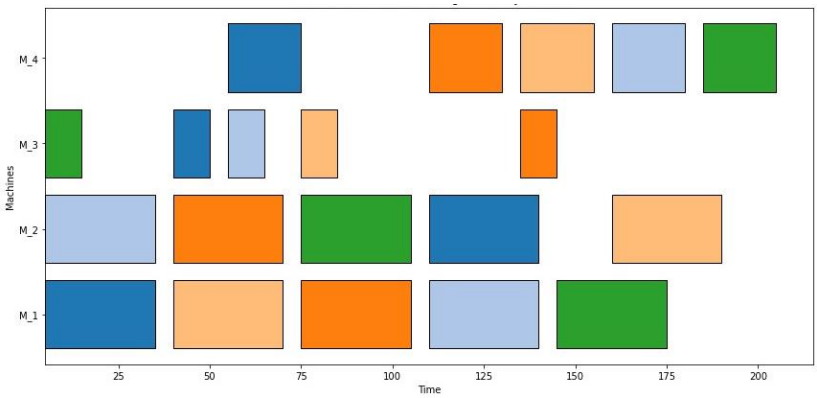

Figure 6.15 - Gantt Chart for five products and four machines

Increasing the station number to eight while keeping the product count at five provided a chance to assess the benefits of additional stations. The Gantt chart for this scenario, shown in Figure 6.16, illustrates a more distributed schedule, shorter but with more idle times.

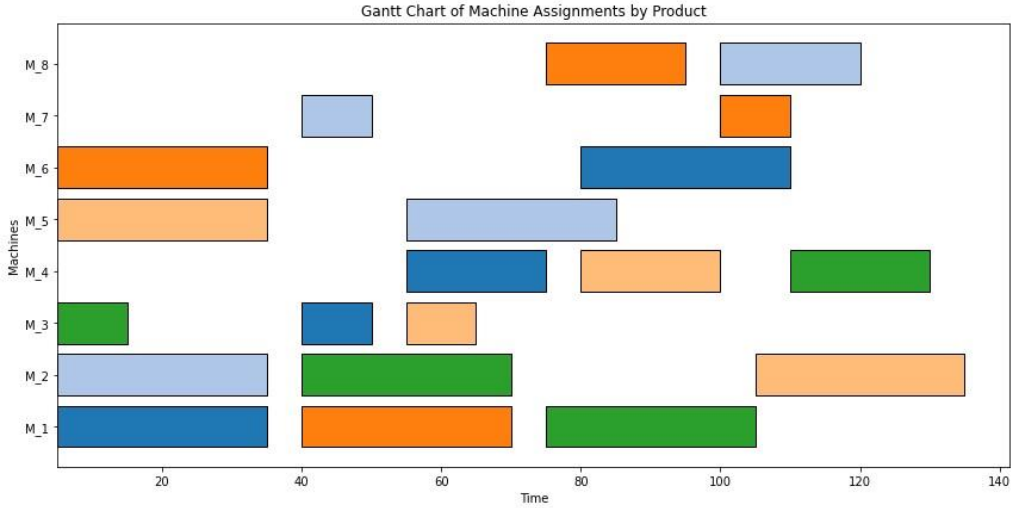

Figure 6.16 - Gantt Chart for five products and eight machines

The scheduling generation time slightly increased to 0.39 seconds. Despite this small increase, there was a decrease in the makespan compared to Test 1, as would be expected. This test underlined the advantage of having more stations available to reduce the scheduling time.

In the next test, the system was supposed to schedule sixty products across four stations. The Gantt chart, in Figure 6.17, reveals a complex schedule with increased idle times, as the processing times in M_3 are shorter than the others, however all tasks were successfully allocated. The scheduling generation took, on average, 1.21 seconds.

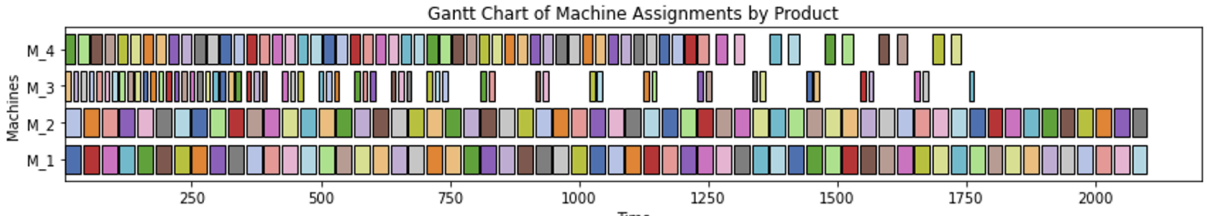

Figure 6.17 - Gantt Chart for sixty products and four machines

The limited number of stations led to longer processing times as it was not possible to allocate all tasks earlier. Having only four stations for sixty products leads to this scenario. Nevertheless, all tasks were successfully assigned and all the restrictions, mainly the order of the tasks within each product, were respected. Naturally, the number of available stations on the system will define the effectiveness of the scheduling process.

The scenario of scheduling sixty products with eight available stations led to a reduction of the schedule makespan, as observed in Figure 6.18. The Gantt chart for this test illustrates that there are mode idle times, namely on M_3 and M_7, as they are similar and have shorter execution times than the others. In this situation, the scheduling time slightly increased to 1.24 seconds.

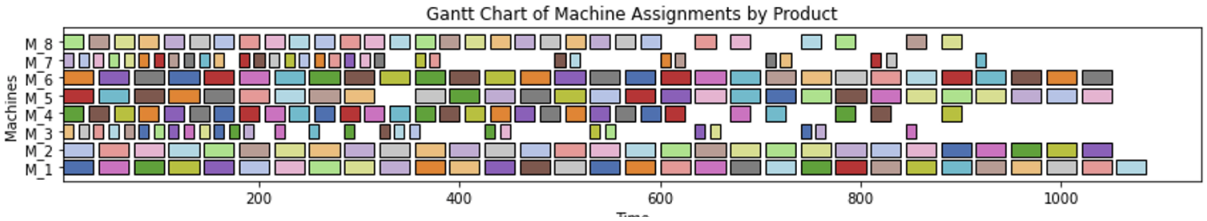

Figure 6.18 - Gantt Chart for sixty products and eight machines

The increased number of stations allowed for better parallel processing and task distribution. This test highlighted the system's ability to handle large product orders effectively when more stations are available.

The challenge in the next test was scheduling one hundred and twenty products using only four stations. The Gantt chart, shown in Figure 6.19, reflects a highly congested schedule with extended production times. However, it proves the system's ability to scale and operate even with large production orders. The computation time was 2.40 seconds.

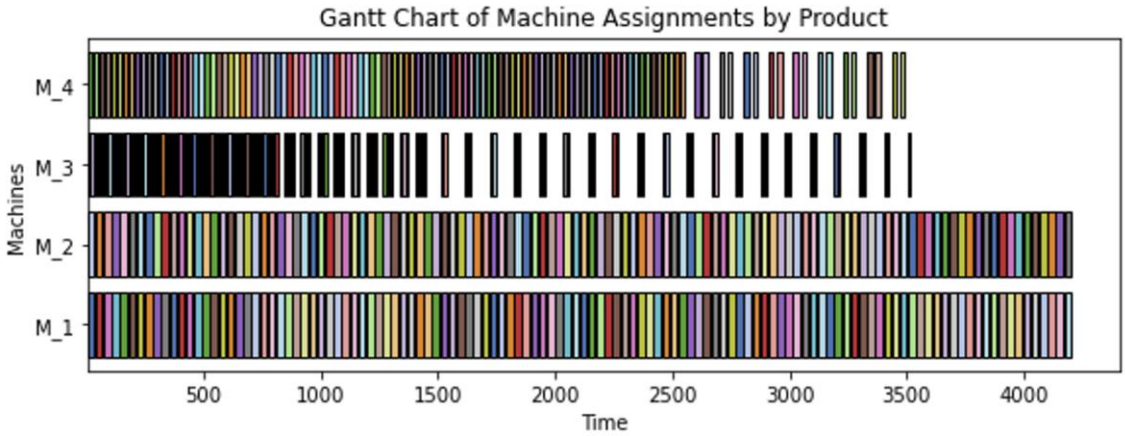

Figure 6.19 - Gantt Chart for one hundred and twenty products and four machines

The limited number of stations led to a very congested schedule, however all tasks were successfully assigned and all constraints were respected.

When using the same number of products with eight stations available, the scheduler system managed to deal with it, reducing the makespan, and distributing the tasks efficiently by the stations. The Gantt chart, in Figure 6.20, also shows that, as expected from the observation of previous scenarios, the idle time in some stations increased. In this scenario, the scheduling process took, on average, 2.45 seconds

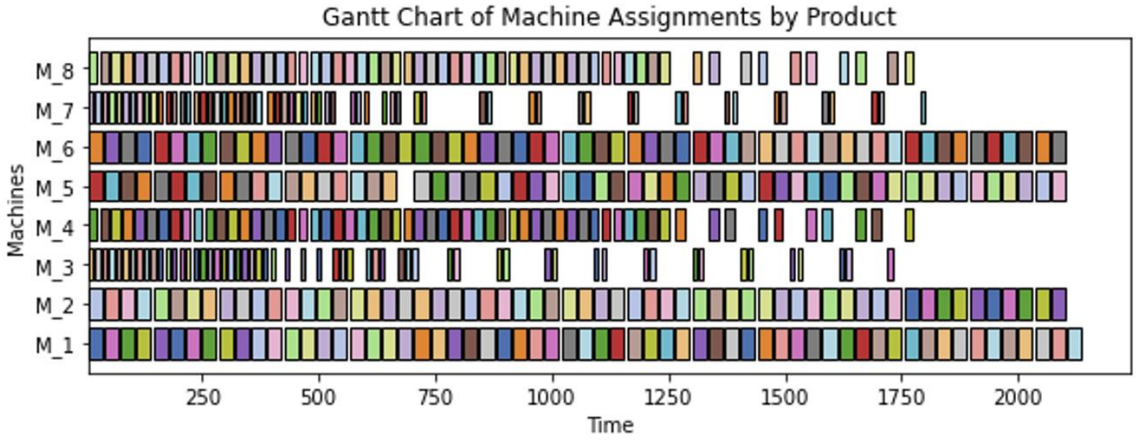

Figure 6.20 - Gantt Chart for one hundred and twenty products and eight machines

The increased number of stations allowed for effective task distribution, shortening the overall production duration. The well-organized schedule highlights the system's capability to scale effectively with both product volume and number of stations, providing an efficient production process.

Finally, a test was conducted using a production simulation kit in a NOVA University of Lisbon laboratory. This scenario, defined in 6.1.1, involves four stations handling a production order of four products, simulating an environment with software and hardware interactions in

both ways. The main goal was to test the capabilities of the scheduler system to interact with a real scenario, while monitoring the production progress.

To manage the execution of the production scheduling in this scenario, a monitoring function was employed to iterate through the JSON object, represented in Figure 6.21, published on the *schedule_info* MQTT topic, to continuously check the station IDs and their corresponding start times. When the scheduled start time for a station is reached, the function sends a value of 1 to an MQTT topic designated for that station. It signals the middleware to initiate the station's operation.


```
{
   "product": "P_3_1",
   " station ": "M_2",
   "start_time": 5,
   "process_time": 30,
   "end_time": 35
},
{
   "product": "P_2_2",
   "station": "M_2",
   "start_time": 40,
   "process_time": 30,
   "end_time": 70
}
```


Figure 6.21 - Schedule JSON object

Once a station completes its assigned task, it sends a "done" message to the middleware. This acknowledgment allows the system to track the progress of each station and ensure that subsequent tasks can begin promptly. This dynamic interaction between the scheduling system and the physical station ensures real-time synchronization, enabling smooth production flow and minimizing delays. If a task takes more time than expected and this value goes above a predefined threshold, an automatic rescheduling with the remaining tasks is performed, while that station is dismissed, as it is considered unable. This feature ensured that the scheduling system could adapt to unexpected disruptions and continue operations with minimal impact on overall production flow.

This test demonstrated a reliable integration between the software and hardware components, providing two-way communication while executing the scheduling, as well as the ability to manage real-world unexpected events and dynamic interactions between the system.

The results across the various tests provide several key insights:

- **Product Volume Management:** The system demonstrates its ability to handle both small and large product orders. The overall time performance is significantly enhanced with more stations, as expected. However, larger orders with more stations lead to increased idle times.
- **Scalability:** The tests validate the system's scalability, showing that it can effectively manage different volumes of products and number of stations. The time efficiency improvements observed in scenarios with more stations emphasize the system's robustness and adaptability in diverse manufacturing scenarios. The improvement in scheduling efficiency with more products and stations, while more and less keeping the execution time, 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 ensures that the system functions cohesively and effectively across different conditions, meeting the expected objectives of this work.

Overall, the evaluation confirms that the production scheduling system performs effectively across different conditions, optimizing manufacturing processes, while the system's components are seamlessly integrated.

6.1.4 Relation to DPs

In this sub-section is presented the relation between the DPs and the implementation of the laboratory use case.

Design Principle	Covered	How is it related?
1.1.1 - Digital factory replica	✓	Assets stored in the AAS server
1.1.2 - Communication protocols	✓	Communication protocols defined for each asset (HTTP, MQTT)
1.2.1 - Common interfaces	✓	Interfaces defined through the developed APIs
1.2.2 - Data model	✓	Data model defined within the AAS, e.g. for stations and products

1.2.3 - Uniform data among components	✓	The data format is harmonized between the different components of the system
1.3.1 - Harmonized KPIs description	✓	The KPI considered was the makespan, measured in seconds
1.3.2 - Save production objectives	✓	Minimize makespan and produce in the shortest time possible
1.3.3 - Editable KPIs priorities		Not considered
1.3.4 - Considering different KPIs for schedule generation		Not considered
1.4.1 - Define boundaries	✓	The specification of the shop-floor and stations defines the boundaries of the system.
1.4.2 - Evaluate production process evolution	✓	Shop-floor monitoring is performed to know if the processes are being executed on time
1.4.3 - Analyze new orders	✓	New orders lead to a reschedule of the system
1.4.4 - Automatic rescheduling	✓	If delays in processing are identified, is performed a rescheduling with the remaining tasks
1.5.1 - Schedule validation	✓	The schedule output is provided for validation
1.5.2 - New schedules request	✓	The user can order for new schedules at anytime
1.5.3 - Choose KPIs		Not considered
1.5.4 - Human worker inputs		Not considered
1.6.1 - Implementation of different optimization techniques	✓	Different optimization algorithms may be adopted as far as they follow the provided data model and defined API, to ensure it receives the right inputs and provides the right outputs
1.6.2 - Communication with external systems		Not considered
1.6.3 - Start schedule not before the current state	✓	It is guaranteed that the schedule is always started in a later time then the current one

The table presents the relationship between the DPs and their implementation in the laboratory scenario. Key aspects like digital replica of the factory, communication protocols, and data models are covered, ensuring that the system components communicate and operate efficiently. Other principles such as defining boundaries, analyzing new orders, and implementing automatic rescheduling are incorporated to improve scheduling and process monitoring. Also, the implementation of diverse optimization techniques and schedule validation is

supported. Some elements like editable KPI priorities, different KPI considerations, and communication with external systems were not considered in the current implementation due to implementation complexity.

6.2 Industrial Scenario

Multi-objective planning, including strict adherence to deadlines, energy efficiency and minimizing setup time, represents a significant challenge for the manufacturing industry in general but in particular for the plastic injection molding sector. The urgency of this challenge is amplified in the plastics industry because of its key role in global sustainability. The growing environmental impact of plastics production and disposal has caught the attention of legislative bodies, especially in the European Union. In response, policies have been developed to decrease waste, promote resource reuse, and improve the sustainability of production cycles in an attempt to establish a resilient circular economy. The KITT4SME platform is a cloud architecture created to connect AI components from a marketplace into the shop floor and enable them to store and exchange data in an interoperable, secure, and scalable manner.

6.2.1 Scenario Definition

The KITT4SME platform is made up of connected RESTful services that operate in a cluster environment and rely on a specific cluster software infrastructure. The KITT4SME architecture includes data collection from various factory-deployed devices, such as wearable sensors, CPS, and environmental sensors, supplying both raw and pre-processed data to the system's knowledge base. It is supported by a middle layer composed by a FIWARE Orion Context Broker, which enables the handling of information in a decentralized and large-scale way. The higher-level layer comprises big data, AI-based and optimization tools, which may analyze historical data to add value for process optimization or improvements. The FIWARE services use JSON for data exchange and it can be accessed by HTTP requests, while KITT4SME platform services should be exposed as RESTful services (Angel et al., 2021).

RAILES is a real-time industrial production line management application, developed by MUVU Technologies, to help manufacturers make better decisions based on information collected from the shop-floor. In this scenario, it will be responsible for providing information from the shop-floor to the AI-based solutions. In Figure 6.22 is represented an adaptation of the KITT4SME platform, where the scheduling tool will be operating within the AI-based solutions module, more precisely in the data processing engine module. It will interact with Railes

platform through the Fiware Context broker, so it is possible to gather all the necessary data to execute the scheduling.

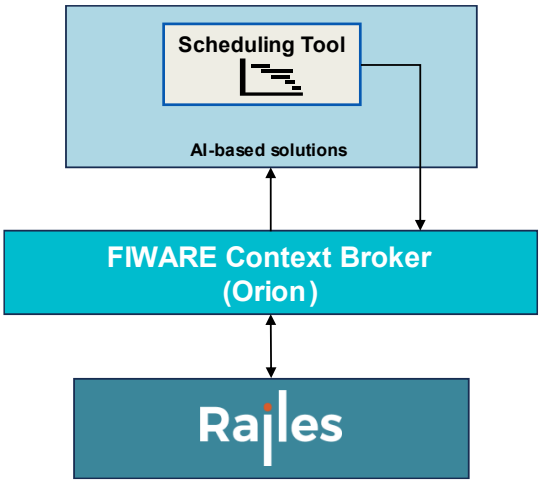

Figure 6.22 - *Interaction between the scheduling tool and Railes platform in the context of KITT4SME Platform (Adapted from* (Angel et al., 2021))

The proposed work aims to develop a solution capable of optimizing production scheduling with multiple objectives, such as optimizing deliveries, energy consumption, and reducing waste. As mentioned, the use case will rely on the plastic sector, which has had several issues related to environmental sustainability in the last years. It will allow to test and validate the interoperability of the scheduling system in the plastic injection industry. For this, the developed solution was deployed and validated within a Portuguese SME that specializes in plastic packaging production due to the substantial impact and importance of plastic production in this sector.

The validation procedure entailed close cooperation with factory floor operatives, as their hands-on experience was vital to testing the solution in a practical setting. This process was carried out in coordination with various key individuals. The production manager, quality operators, machine engineers/operators were included due to their comprehensive understanding of the production process and hands-on interaction with the machines. The initial KPIs were measured before and after the implementation of the developed solution.

Tests were conducted during normal operation on the factory floor, supervised by the factory workers, with the RAILES platform used to monitor the KPIs. Having been tested in a 'laboratory' environment, the tool was put to work on the factory floor, and the differences were monitored. In this particular scenario, the procedure is rather straightforward. It simply requires setting up the production according to the priorities suggested by the algorithm and monitoring the KPIs before and after the changes.

As mentioned, some tests were performed during the validation phase with various productions and jobs scheduled, but in detail, the one that allowed obtaining the KPIs presented in the results chapter, took place over three weeks of production, on three different machines, and included four scheduling jobs. This is due to the fact that in each week a job was performed to allocate production to the machines and an extra one to test the introduction of new orders, observing the correct allocation of machines.

In this scenario, ten different productions were made in three different machines. Four of them on Machine 1, another four on Machine 2, and two on Machine 3, because the cycle time and the number of mold cavities of the produced products varies among them. On Machine 1, a total of 361394 pieces were produced in the four productions, on Machine 2, 335391 pieces were produced, and on Machine 3, 339298 pieces were produced.

6.2.2 Assets Relation

The implementation and integration of an AI module into an existing platform demands a systematic and structured approach. This subchapter outlines the methodology used to acquire data from a primary source, process it, and use it to construct Next Generation Service Interfaces (NGSI)-conforming entities compatible with the Orion Context Broker. Although it is not the focus of this work, it will be briefly described here for better context.

The developed module was initially configured to acquire data from at least one primary data source in its unprocessed state and funnel it into the Orion Context Broker using NGSI-conforming payloads. The *StructuredValueAttr* class was leveraged to expedite the data input process for the AI module. Data were sourced in a JSON format via a direct query to the RAILES database. The FiPy python library was used to form an entity from this data. The *StructuredValueAttr* class was selected as it permits the storage and handling of structured data, such as the information retrieved from the database containing shop-floor specifics. The data encapsulates a series of production instances, each bearing unique attributes including production ID, product designation, quantity, delivery due date, the machine involved, and cycle durations. The *StructuredValueAttr* class allowed to effectively store and retrieve this data within the AI module, facilitating precise analysis and forecast. Utilizing the *StructuredValueAttr* class facilitated the formulation of an organized entity that accurately represents data in a comprehensible and manageable manner. This methodology enabled the creation of an NGSI entity, which can be conveniently utilized by other applications and services, highlighting its high degree of interoperability. The *StructuredValueAttr* class also offers flexibility in data

representation, as it supports the creation of dynamic structures that can be effortlessly expanded in the future to incorporate new data fields.

The FiPy library simplified the process of linking the AI component with the FIWARE Orion data broker, facilitating data exchange via NGSI entities. By connecting the component to the Orion Context Broker, the AI component is now prepared for integration with other FIWARE-based services and applications, thus offering a highly interoperable solution capable of effortless expansion and scalability.

The connection between the scheduling tool and RAILES is done through the context broker, which means that all data is stored as a black box, that is unknown for the scheduling tool, but where the data is provided correctly as an output of the broker, as a RESTFul service, Figure 6.23. Thus, the information regarding working stations, products, and shop-floor management may be developed as submodels of a broader AAS, the KITT AAS.

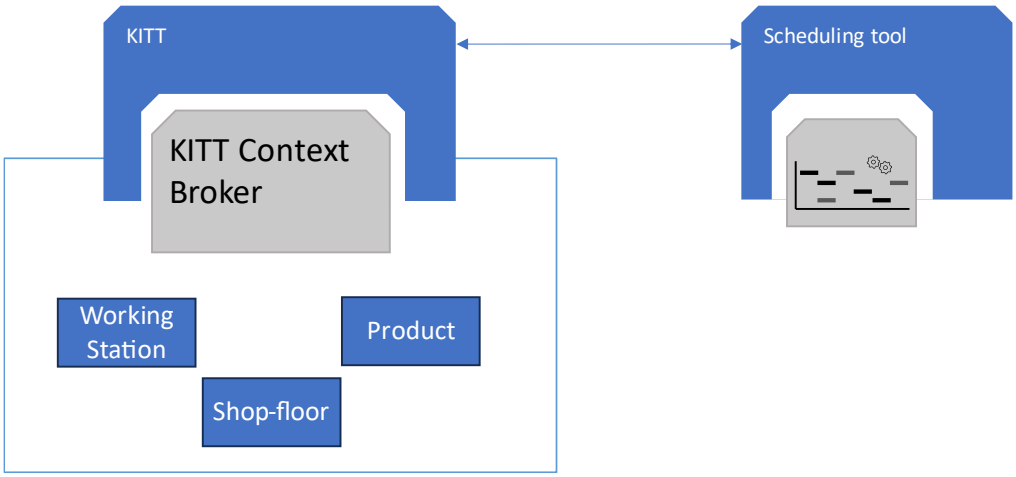

Figure 6.23 - AAS relation between KITT and the scheduling tool.

The solution data model for this scenario is represented in Figure 6.24, and depicts the data related to the physical assets (Product and Working Stations), relevant information and constraints about the Shop-floor, and the software assets (Scheduling Tool and Schedule Output).

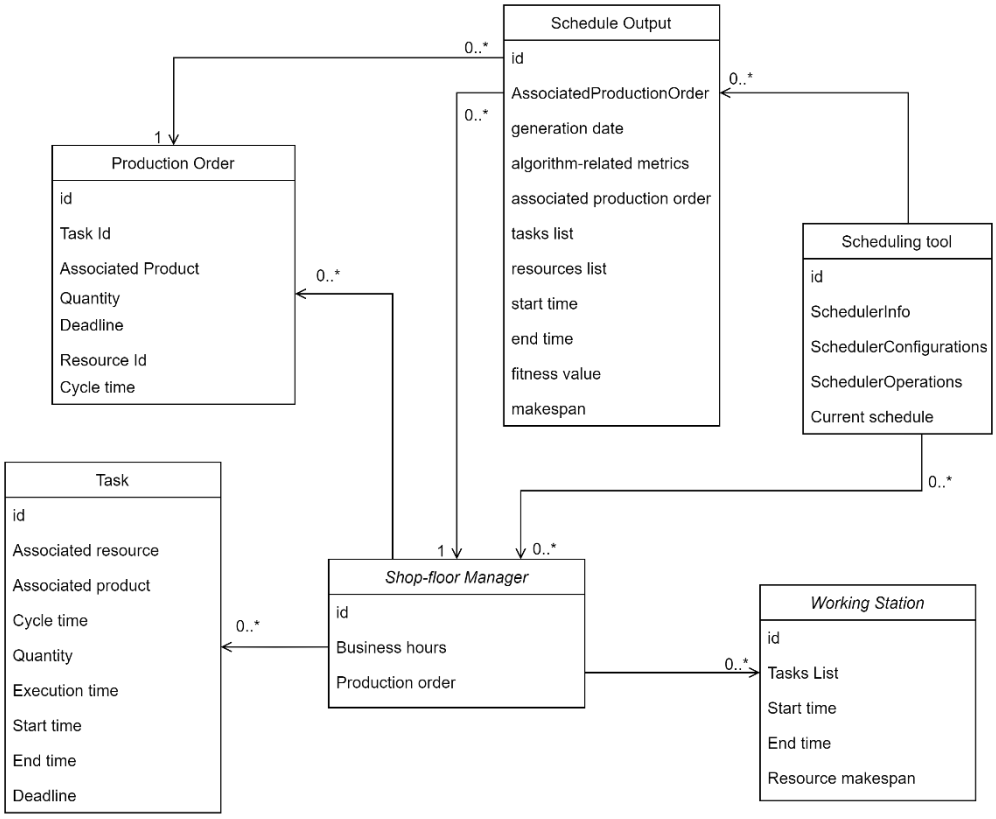

Figure 6.24 - Production scheduling solution data model

For a deeper understanding of the AAS structure, the scheduling AAS was modeled as below, illustrated in Figure 6.25.

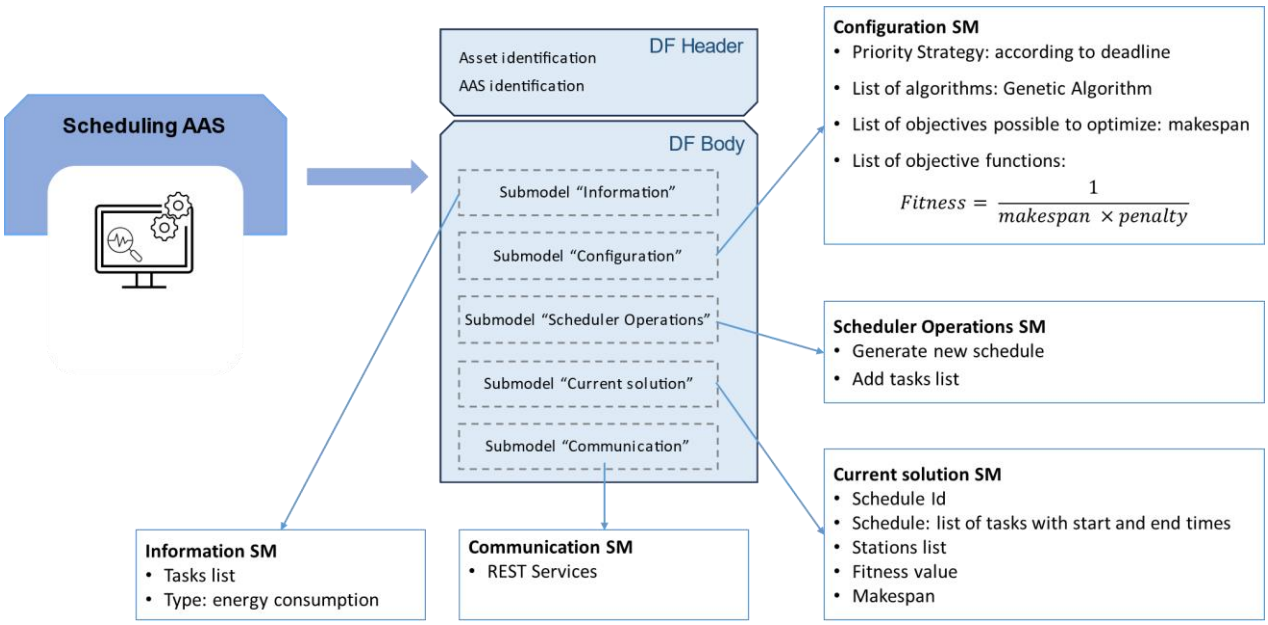

Figure 6.25 - Scheduling AAS configuration

Besides its identification, it is composed by five submodels, namely, the Information submodel with information about the list of tasks and the type of schedule (related to energy efficiency), the Configuration submodel composed by the priority strategy, the list of available algorithms, the list of objectives possible to optimize, and the list of objective functions, the Scheduler Operations submodel that comprises the operations to generate new schedules and add new list of tasks to allocate, the Current Solution submodel containing information about the most recent generated schedule, and the Communication submodel, which specifies how the AAS communicates with other systems (in this case, the communication is done through REST services.

6.2.3 Scheduling algorithm implementation

Genetic algorithms (GAs) are a class of optimization algorithms that are inspired by the process of natural selection in biological organisms. Complex optimization problems can be solved with GAs when more conventional approaches may be ineffective.

A population of potential solutions (known as "individuals" or "chromosomes") is generated randomly, and each individual represents a potential solution to the optimization problem. A fitness function, which measures the quality of the solution, is used to assess each individual of the population.

The fundamental idea underlying GAs is to mimic natural selection by repeatedly choosing the fittest members of the population and utilizing them to produce the subsequent generation of individuals. This process involves several steps, including selection, crossover, and mutation.

To create the next generation, selection entails picking the population's fittest individuals. An individual's likelihood of being chosen for the upcoming generation increases with their level of fitness. This step is comparable to the process of natural selection, in which organisms with beneficial features have a higher chance of surviving and reproducing.

Crossover consists of generating a new individual by combining two individuals' genetic characteristics. This process is comparable to reproduction in biology, where two parents' genetic material is combined to produce a new offspring.

Mutation requires introducing a little of random variations into an individual's information to create a new, slightly different individual. This step helps to introduce genetic diversity into the population and can prevent the algorithm from converging on a suboptimal solution.

Selection, crossover, and mutation steps can be repeated iteratively until a stopping condition is satisfied (e.g. a predetermined number of generations or a satisfactory fitness threshold). The population tends to get better over time, as the algorithm looks for the fittest individual or individuals which represent optimal or near-optimal solutions.

GAs have been applied to solve a variety of optimization problems. They have been demonstrated to be especially useful in situations where traditional optimization techniques may find it challenging to locate a solution due to the size and complexity of the search space. However, GAs can be computationally expensive, especially for large population sizes or complex fitness functions. Also, the choice of fitness function and other algorithmic parameters has a significant impact on the quality of the solution produced by a GA.

The fitness function is a crucial component of a genetic algorithm implementation. It serves as the evaluation method for individuals, determining their quality in the population with respect to the problem being solved. The fitness function guides the evolutionary process by assigning a numerical value, the fitness score, to each individual based on how well it meets the objectives or constraints of the problem.

For the specific scheduling problem being solved, the main steps are describing next:

1. Define chromosome(individual): a chromosome is composed of genes, where each gene represents an operation to be performed on the shop-floor. Thus, each chromosome represents a different scheduling solution.
2. Create a new population of random individuals.
3. Calculate the fitness score of each individual:
   a. Define execution times each time a new individual is generated: the algorithm goes through the chromosome to set the execution times (start and end times) of each operation. It will allocate the current operation to the next empty time of the corresponding station. At the end, it will update the completion time (makespan) of that station (where several operations will be performed).
   b. The total makespan of each schedule (individual) is represented by the maximum makespan from all stations, i.e., if one station has a makespan of 1 day and another one has a makespan of 3 days, the total makespan of the schedule is 3 days.
   c. The schedule fitness is given by the inverse of the schedule makespan multiplied by a penalty: $Fitness = \frac{1}{makespan \times penalty}$

      The penalty is compounded each time an operation is allocated beyond the deadline. The higher the fitness score the better.
4. Stop if stop criteria are met. Which may be maximum generations, number of generations without changes in fitness value, or fitness score threshold.
5. Increment generation number.
6. Elitism: chooses a percentage of the best individuals to automatically pass on to the next generation.

7. Selection process: selection of two of the best individuals (parents) for reproduction based on tournament selection mechanism
8. Crossover: the two individuals generate a new one (offspring), containing parts of each one solution
9. Mutation: appliance of random changes in the chromosome
10. Go to number 3.

The objective here is to find the shortest solution to complete all the production tasks, while avoiding those tasks going beyond the deadline. In Table 8 is the data format of each of the assets.

Table 8 - Assets data format

Production order	**Schedule**

{
"TaskID":"74104345-4eab-415f-a630-6bc52996534a",
"AssociatedProduct":"garrafa 800 ml cristal tritan",
"Quantity":10000,
"Deadline":"2023-01-26 16:35:47+00:00",
"ResourceID":"Nissei ASB 12M 3",
"CycleTime":"20" [sec]
}
	{
	"ScheduleID":"schedule_2024_03_11_1051",
	"schedule": list of tasks,
	"resourcesList": list of resources,
	"fitness":"3.944*10^(-6)",
	"makespan": "253500" [sec],
	"algorithm-related metrics": execution time
	}

Resources	**Task**

{
"ResourceID":"Nissei ASB 12M 3",
"tasksList": Collection of tasks allocated to this resource,
"startTime":"2023-01-24 00:00:00+00:00",
"endTime": "2023-01-26 22:25:00+00:00",
"makespan": "253500" [sec]
}
	{
	"TaskID":"74104345-4eab-415f-a630-6bc52996534a",
	"associatedResource": Nissei ASB 12M 3,
	"associatedProduct": "garrafa 800 ml cristal tritan",
	"cycleTime":"32" [sec],
	"quantity":"6250",
	"executionTime":"200000" [sec],
	"startTime":"2023-01-24 00:00:00+00:00",
	"endTime": "2023-01-26 07:33:20+00:00",
	"deadline": "2023-01-26 16:35:47+00:00"
	}

6.2.4 Results

In this chapter a comprehensive analysis of the key performance indicators that testify the efficiency of the developed system is presented. The results were not compared with the state of the art or benchmarks, but with the performance indicators analyzed in the company before the integration of this solution. As a starting point for the analysis, Table 9 presents a comprehensive overview of the KPIs' status.

Table 9 – KPI status before and after the scheduling algorithm is implemented.

Nº	KPI name and description	Start Value	Target value	Actual Value
1	**Performance**	95,5%	96% (↑ ~0,5%)	96,1% (↑ 0,6%)
2	**Quality**	93,7%	94% (↑ ~0,3%)	95,2 % (↑ 1,5%)
3	**Availability**	50,7%	56% (↑ ~5,3%)	57,1% (↑ 6,4%)
4	**OEE**	45,4%	50,5% (↑ ~5,1%)	52,2% (↑6,8%)
5	**Waste**	6,3%	5.7% (↓~0,6%)	4.8% (↓1,5%)

Prior to the implementation of the solution, performance indicator was already evident, indicating that scheduling might not significantly influence this metric. As expected by the process experts and observable in the table, a modest increase of approximately 0.5% was predicted. Interestingly, implementation led to a 0.6% increase, validating the initial projection and attesting to the effectiveness of the developed system.

Significant enhancement was observed in the quality metric, surpassing initial expectations. This can be primarily attributed to improved setup procedures facilitated by schedule optimization. Specifically, optimized production planning allows productions of identical products to be sequenced together, thus avoiding setup time for activities such as mold changes. It should be noted that in industries such as this one, the majority of quality-related issues typically occur during the setup phase and early stages of production. One notable advantage of employing genetic algorithms lies in their flexibility: the fitness function can be adjusted to accommodate specific targets, whether it involves minimizing or maximizing a particular factor. Consequently, the algorithm can be tailored to meet the unique objectives of various industry contexts. This adaptability is crucial in situations where the primary sources of waste differ. For example, in industries where waste is not predominantly tied to setup time, the algorithm can be recalibrated to assign less weight to setup time restrictions, thereby more accurately aligning with the industry's specific requirements.

Machine availability experienced one of the most substantial improvements, surpassing all expectations. Improved planning and machine allocation management led to a more efficient utilization of the company's productive capacity. As a result, production was achieved in less time, which in turn increased machine availability. This KPI saw a 6.4% increase, compared to the initially expected 5,3%. This KPI is straightforward, as the developed solution optimizes production planning based on consumption, cycle times, and quantities to be produced.

Furthermore, it holds potential for application across various productive industries, as long as the intention for improvement is clearly defined.

Finally, it was calculated the new OEE, which is 52.2%, representing a 6,8% increase regarding the initial target. This highlights the validity and feasibility of this application in a shop floor setting. In the case of injection molding processes, quality, a component of the OEE calculation, is measured by the proportion of correctly molded parts to the total number of parts produced. This quality component takes into account the good parts produced during a production run that meet quality standards. In this case, since it was known the produced quantity and the quantity that was actually accepted and delivered to the customer, it was possible to quantify this KPI as indicated in the table.

Contrary to quality, waste is the value representing the percentage of parts that were produced and were not in compliance with the quality standards. As it is a directly related calculation, if the quality value is 95.2%, the waste value is 4.8%, which represents a decrease in this value, since the initial value measured and indicated in the action plan report was 6.3%, and a decrease of 0.6 was expected, a decrease of 1.5% was observed.

Lastly, the Energy Consumption indicator merits separate discussion. The primary focus was to measure the energy consumption relative to the ratio energy/quantity produced, not the total energy consumed. In the observations, it was noticed a reduction in energy consumption of about 1.12%. The decrease (although small) in energy consumption might be due to the limited scope of the analysis. For industries like this, an accurate evaluation should cover at least a year to account for variations in demand and production. Nevertheless, the metrics and KPIs gathered during the months of validation already show the effectiveness of the solution.

In conclusion, the observed decrease in energy consumption, coupled with an increase in machine availability, points to the broader applicability of the solution to any industry seeking to enhance their production planning. This analysis holds value for any manufacturing industry requiring multi-objective production planning, making this solution sufficiently flexible to expand into various markets.

6.2.5 Relation to DPs

In this sub-section is presented the relation between the DPs and the implementation of the scheduling system for KITT4SME use case.

Design Principle	Covered	How is it related?
1.1.1 - Digital factory replica	✓	Assets stored in Railes
1.1.2 - Communication protocols	✓	Communication protocols defined for each component
1.2.1 - Common interfaces	✓	Common interfaces provided through existing (communication with KITT4SME broker) and developed (scheduling tool) APIs
1.2.2 - Data model	✓	Data model defined in KITT4SME project and at the scheduling tool level
1.2.3 - Uniform data among components	✓	The data format is harmonized between the different components of the system
1.3.1 - Harmonized KPIs description	✓	KPIs defined on Railes
1.3.2 - Save production objectives	✓	Minimize makespan and improve energy efficiency
1.3.3 - Editable KPIs priorities		Not considered
1.3.4 - Considering different KPIs for schedule generation	✓	Different KPIs were considered for scheduling evaluation
1.4.1 - Define boundaries	✓	Boundaries defined on Railes.
1.4.2 - Evaluate production process evolution	✓	Railes performs the evaluation of the production process
1.4.3 - Analyze new orders	✓	New orders lead to a reschedule of the system
1.4.4 - Automatic rescheduling		Not considered
1.5.1 - Schedule validation	✓	The schedule output is provided for validation
1.5.2 - New schedules request	✓	The user can order for new schedules at anytime
1.5.3 - Choose KPIs		Not considered
1.5.4 - Human worker inputs	✓	Human workers can provide feedback on Railes
1.6.1 - Implementation of different optimization techniques	✓	Different optimization algorithms may be adopted as far as they follow the provided data model and defined API, to ensure it receives the right inputs and provides the right outputs
1.6.2 - Communication with external systems	✓	Communication established between the scheduling tool and Railes
1.6.3 - Start schedule not before the current state	✓	It is guaranteed that the schedule is always started in a later time then the current one

In the KITT4SME use case, the relation between DPs and system implementation is clearly outlined. Communication protocols, common interfaces, and the data model are well-supported, with assets and data harmonized across components using the Railes system. KPIs are defined and used to minimize makespan and improve energy efficiency, with Railes also handling boundary definition, process evaluation, and automatic rescheduling when new orders are analyzed. Schedule validation and request features allow for flexibility, while human workers can provide feedback. Optimization techniques and external communication are integrated, ensuring a robust and efficient scheduling system that respects both system boundaries and operational constraints.

7

CONCLUSION AND FUTURE WORK

To summarize the findings of this work and highlight potential avenues for further exploration, this section presents the conclusions drawn from the work and outlines key directions for future work that can expand upon the contributions made.

7.1 Conclusion

The research presented in this thesis has addressed the key challenges associated with the design and implementation of scheduling systems in the context of smart manufacturing, particularly aligned with the RAMI 4.0 architecture, focusing on addressing key requirements, adhering to DPs, and employing a structured methodology to implement manufacturing scheduling systems. The rapid evolution of I4.0 technologies has led to the emergence of highly interconnected, autonomous systems, which demand more sophisticated scheduling mechanisms to maintain production efficiency, meet KPIs, and adapt to unforeseen disturbances. Traditional scheduling systems often fall short of the flexibility and scalability needed in modern industrial environments. Thus, this research has sought to bridge the gap by proposing a more generic, adaptive, and comprehensive approach to manufacturing scheduling within smart factories.

A thorough literature review was conducted to identify the functional and non-functional requirements for modern scheduling systems. These requirements were distilled into key DPs that serve as the foundation for the proposed methodology, which aims to develop robust and scalable scheduling solutions tailored for RAMI 4.0-compliant environments. This work has also explored the intricacies of integrating production assets into the scheduling process,

specifically through the use of AASs, which play a crucial role in representing both the physical and digital components of the manufacturing process.

By structuring the scheduling system around RAMI 4.0 and incorporating AAS for managing asset data, the proposed approach enhances the transparency, adaptability, and efficiency of scheduling systems. The use of AASs ensures that the components within the smart manufacturing ecosystem can communicate seamlessly, enabling real-time updates and automatic rescheduling when required. This alignment with I4.0 standards also facilitates greater interoperability between different resources and systems, thus reducing downtime, optimizing resource utilization, improving sustainability-related metrics, and enhancing overall production flow.

The methodology developed in this thesis has been validated through its application to three distinct use cases: adaptation to PERFoRM project, implementation in a laboratory scenario, and implementation in an industrial scenario (KITT4SME project). These use cases were selected to represent varying complexities and production environments, demonstrating the versatility and scalability of the proposed scheduling solution. In each scenario, the system was tested against different configurations of stations and product orders, allowing for a detailed evaluation of its performance under different conditions. The results obtained from these tests highlight the system's capacity to dynamically adjust to changing conditions, balance workloads, and minimize idle time, even when faced with unexpected disruptions.

The results from the testing phases confirmed that the proposed methodology based on the defined DPs, can effectively manage production tasks and optimize scheduling processes. The research findings suggest that the integration of AAS within the RAMI4.0 framework not only enhances the adaptability of scheduling systems but also contributes to the overall improvement of manufacturing processes. The combination of a well-structured methodology, real-time communication through AASs, and the use of a standardized reference model such as RAMI4.0 represents a significant step forward in the development of scheduling systems for smart manufacturing. This research also underscores the importance of interoperability, flexibility, and automation in achieving efficient production management in highly dynamic industrial settings.

7.2 Future Work

Future research and development should address several key areas to enhance the system's capabilities. One avenue is to apply the proposed methodology across a broader range

of scenarios and with greater detail, incorporating additional assets, features, and constraints. This includes integrating factors such as human workers, production shifts, storage options, raw materials, bill of materials for each product, or energy consumption of transportation options.

Additionally, exploring scalability solutions will be crucial for managing larger production environments. Optimizing algorithms and processing mechanisms to handle increased production loads efficiently will be essential. Developing more sophisticated error detection and recovery mechanisms will further enhance the system's resilience, ensuring continuous operation despite unexpected disruptions.

Expanding the system’s integration with other industrial systems, including supply chain management and ERP systems, could provide a more holistic approach to production management. Also, incorporating machine learning for predictive analytics could significantly improve the system’s ability to forecast production needs and potential issues, allowing for proactive scheduling adjustments.

Furthermore, investigating the potential of integrating emerging I4.0 technologies, such as edge computing and advanced IoT sensors, could provide real-time insights into machine and process statuses. This would enable the system to make more informed decisions based on live data streams and pave the way for fully autonomous scheduling systems capable of optimizing production processes without human intervention.

These future directions aim to build on the system’s current strengths, address emerging challenges, and contribute to a more robust and adaptable solution for production scheduling.

BIBLIOGRAPHY

Abreu, L. R., Cunha, J. O., Prata, B. A., & Framinan, J. M. (2020). Computers and Operations Research A genetic algorithm for scheduling open shops with sequence-dependent setup times. *Computers and Operations Research*, *113*, 104793. https://doi.org/10.1016/j.cor.2019.104793

Afzalirad, M., & Shafipour, M. (2015). Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. *Journal of Intelligent Manufacturing*, *29*(2), 423–437. https://doi.org/10.1007/s10845-015-1117-6

Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers and Operations Research*, *73*, 56–66. https://doi.org/10.1016/j.cor.2016.03.009

Alam, K. M., & El Saddik, A. (2017). C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems. *IEEE Access*, *5*, 2050–2062. https://doi.org/10.1109/ACCESS.2017.2657006

Alemão, D., Araújo, S., Rocha, A., Peres, R., Tripa, J., Barata, J., Camarinha-Matos, L., Nikghadam-Hojjati, S., Jafarali-Jassbi, J., & Okatakyie, K. (2020). *CESME - Deliverable 4.1*.

Alemão, D., Parreira-Rocha, M., & Barata, J. (2019). Production and Maintenance Scheduling Supported by Genetic Algorithms. In *IFIP Advances in Information and Communication Technology* (Vol. 530, pp. 49–59). https://doi.org/10.1007/978-3-030-05931-6_5

Alemão, D., Rocha, A. D., & Barata, J. (2021). Smart manufacturing scheduling approaches—systematic review and future directions. *Applied Sciences (Switzerland)*, *11*(5), 1–20. https://doi.org/10.3390/app11052186

Alemão, D., Rocha, A. D., & Barata, J. (2019). Production Scheduling Requirements to Smart Manufacturing. In L. M. Camarinha-Matos (Ed.), *10th IFIP WG 5.5/SOCOLNET Advanced Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS 2019* (pp. 227–237). Cham Springer. https://doi.org/10.1007/978-3-030-17771-3_19

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

Alotaibi, A., Lohse, N., & Vu, T. M. (2016). Dynamic Agent-based Bi-objective Robustness for Tardiness and Energy in a Dynamic Flexible Job Shop. *Procedia CIRP*, *57*, 728–733. https://doi.org/10.1016/j.procir.2016.11.126

Angel, M., Aguilar, A., Albertario, S., Alonso, R., Falconi, A., Kolehmainen, K., & Kolyvas, T. (2021). *KITT4SME D2.1 - KITT4SME Reference Architecture*. *952119*, 1–63.

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

Asadzadeh, L. (2015). A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Computers & Industrial Engineering*, *85*, 376–383. https://doi.org/10.1016/j.cie.2015.04.006

Asghar, A., Hosseinabadi, R., Vahidi, J., Saemi, B., Kumar, A., & Elhoseny, M. (2019). Extended Genetic Algorithm for solving open-shop scheduling problem. *Soft Computing*, *23*(13), 5099–5116. https://doi.org/10.1007/s00500-018-3177-y

Azami, A., Demirli, K., & Bhuiyan, N. (2018). Scheduling in aerospace composite manufacturing systems: a two-stage hybrid flow shop problem. *International Journal of Advanced Manufacturing Technology*, *95*(9–12), 3259–3274. https://doi.org/10.1007/s00170-017-1429-0

Bader, S., Barnstedt, E., Bedenbender, H., Billman, M., Boss, B., & Braunmandl, A. (2020). Details of the Asset Administration Shell Part 1 - The exchange of information between partners in the value chain of Industrie 4.0. In *Plattform Industrie 4.0* (3.0; Vol. 0). https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html

Balin, S. (2011). Non-identical parallel machine scheduling using genetic algorithm. *Expert Systems with Applications*, *38*(6), 6814–6821. https://doi.org/10.1109/TM

Bandi, C., & Gupta, D. (2020). Operating Room Staffing and Scheduling. *Manufacturing & Service Operations Management*, *22*(5), 958–974. https://doi.org/10.1287/msom.2019.0781

Barbosa, J., Leitão, P., Adam, E., & Trentesaux, D. (2015). Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. *Computers in Industry*, *66*, 99–111. https://doi.org/10.1016/j.compind.2014.10.011

Benešl, T., Kaczmarczyk, V., Sýkora, T., Arm, J., Dvorský, P., Husák, M., Marcon, P., & Bradác, Z. (2022). Asset Administration Shell - Manufacturing processes energy optimization. *IFAC-PapersOnLine*, *55*(4), 334–339. https://doi.org/10.1016/j.ifacol.2022.06.055

Branke, J., Nguyen, S., Pickardt, C. W., & Zhang, M. (2016). Automated Design of Production Scheduling Heuristics: A Review. *IEEE Transactions on Evolutionary Computation*, *20*(1), 110–124. https://doi.org/10.1109/TEVC.2015.2429314

Bürgy, R., & Bülbül, K. (2018). The job shop scheduling problem with convex costs. *European Journal of Operational Research*, *268*(1), 82–100. https://doi.org/10.1016/j.ejor.2018.01.027

Çaliş, B., & Bulkan, S. (2015). A research survey: review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, *26*(5), 961–973. https://doi.org/10.1007/s10845-013-0837-8

Carosi, S., Frangioni, A., Galli, L., Girardi, L., & Vallese, G. (2019). *A Tool for Practical Integrated Time-Table Design and Vehicle Scheduling in Public Transport Systems* (pp. 207–217). https://doi.org/10.1007/978-3-030-25842-9_16

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. (2012). *Smart Grid Reference Architecture* (Issue November).

ftp://ftp.cen.eu/EN/EuropeanStandardization/HotTopics/SmartGrids/Security.pdf
Chan, F. T. S., Choy, K. L., & Bibhushan. (2011). A genetic algorithm-based scheduler for multiproduct parallel machine sheet metal job shop. *Expert Systems with Applications*, *38*(7), 8703–8715. https://doi.org/10.1016/j.eswa.2011.01.078
Chang, H.-C., & Liu, T.-K. (2017). Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. *Journal of Intelligent Manufacturing*, *28*(8), 1973–1986. https://doi.org/10.1007/s10845-015-1084-y
Chaudhry, I. A., & Khan, A. A. (2016). A research survey: Review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, *23*(3), 551–591. https://doi.org/10.1111/itor.12199
Chou, Y. C., Cao, H., & Cheng, H. H. (2013). A bio-inspired mobile agent-based integrated system for flexible autonomic job shop scheduling. *Journal of Manufacturing Systems*, *32*(4), 752–763. https://doi.org/10.1016/j.jmsy.2013.01.005
Chung, B. Do, & Kim, B. S. (2016). A hybrid genetic algorithm with two-stage dispatching heuristic for a machine scheduling problem with step-deteriorating jobs and rate-modifying activities. *Computers and Industrial Engineering*, *98*, 113–124. https://doi.org/10.1016/j.cie.2016.05.028
Costa, A., Cappadonna, F. A., & Fichera, S. (2016). Minimizing the total completion time on a parallel machine system with tool changes. *Computers & Industrial Engineering*, *91*, 290–301. https://doi.org/http://dx.doi.org.ezproxy.javeriana.edu.co:2048/10.1016/j.cie.2015.11.015
De Giovanni, L., & Pezzella, F. (2010). An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. *European Journal of Operational Research*, *200*(2), 395–408. https://doi.org/10.1016/j.ejor.2009.01.008
Delorme, X., Fleury, G., Lacomme, P., & Lamy, D. (2023). Modelling and solving approaches for scheduling problems in reconfigurable manufacturing systems. *International Journal of Production Research*, 1–22. https://doi.org/10.1080/00207543.2023.2224446
Ding, B., Ferràs Hernández, X., & Agell Jané, N. (2021). Combining lean and agile manufacturing competitive advantages through Industry 4.0 technologies: an integrative approach. *Production Planning & Control*, *34*(5), 442–458. https://doi.org/https://doi.org/10.1080/09537287.2021.1934587
Dr. Heinz, B. (VDI), Meik, B. (ZVEI – Z. E. E., & Birgit, B. (Robert B. (2019). Which criteria do Industrie 4 . 0 products need to fulfil ? *Platform Industrie 4.0*, 32. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/criteria-industrie-40-products.html
Framinan, J. M., Leisten, R., & Ruiz García, R. (2014). Manufacturing Scheduling Systems. In *Manufacturing Scheduling Systems*. Springer London. https://doi.org/10.1007/978-1-4471-6272-8
Framinan, J. M., & Ruiz, R. (2010). Architecture of manufacturing scheduling systems: Literature review and an integrated proposal. *European Journal of Operational Research*, *205*(2), 237–246. https://doi.org/10.1016/j.ejor.2009.09.026
Freitag, M., & Hildebrandt, T. (2016). Automatic design of scheduling rules for complex manufacturing systems by multi-objective simulation-based optimization. *CIRP Annals - Manufacturing Technology*, *65*(1), 433–436. https://doi.org/10.1016/j.cirp.2016.04.066
Freitas, N., Araújo, S. O., Alemão, D., Ramos, J., Guedes, M., Gonçalves, J., Peres, R. S., Rocha, A. D., & Barata, J. (2023). Cloud-Based Machine Learning Application for Predicting Energy

Consumption in Automotive Spot Welding. *Processes*, *11*(1), 284. https://doi.org/10.3390/pr11010284

Fu, Y., Ding, J., Wang, H., & Wang, J. (2018). Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Applied Soft Computing*, *68*, 847–855. https://doi.org/10.1016/j.asoc.2017.12.009

Gahm, C., Denz, F., Dirr, M., & Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research*, *248*(3), 744–757. https://doi.org/10.1016/j.ejor.2015.07.017

Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. N. (2019). Flexible Job-Shop Rescheduling for New Job Insertion by Using Discrete Jaya Algorithm. *IEEE Transactions on Cybernetics*, *49*(5), 1944–1955. https://doi.org/10.1109/TCYB.2018.2817240

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. S. (2016). Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. *Journal of Intelligent Manufacturing*, *27*(2), 363–374. https://doi.org/10.1007/s10845-014-0869-8

Gao, Kai Zhou, Suganthan, P. N., Pan, Q. K., Chua, T. J., Chong, C. S., & Cai, T. X. (2016). An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. *Expert Systems with Applications*, *65*, 52–67. https://doi.org/10.1016/j.eswa.2016.07.046

Gao, Kai Zhou, Suganthan, P. N., Pan, Q. K., Tasgetiren, M. F., & Sadollah, A. (2016). Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-Based Systems*, *109*, 1–16. https://doi.org/10.1016/j.knosys.2016.06.014

González-Neira, E. M., Montoya-Torres, J. R., & Barrera, D. (2017). Flow-shop scheduling problem under uncertainties: Review and trends. *International Journal of Industrial Engineering Computations*, 399–426. https://doi.org/10.5267/j.ijiec.2017.2.001

Han, L., Xing, K., Chen, X., & Xiong, F. (2015). A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems. *Journal of Intelligent Manufacturing*, *29*(5), 1083–1096. https://doi.org/10.1007/s10845-015-1161-2

He, L., Cao, Y., Li, W., Cao, J., & Zhong, L. (2022). Optimization of energy-efficient open shop scheduling with an adaptive multi-objective differential evolution algorithm. *Applied Soft Computing*, *118*, 108459. https://doi.org/10.1016/j.asoc.2022.108459

Helo, P., Phuong, D., & Hao, Y. (2018). Cloud manufacturing – Scheduling as a service for sheet metal manufacturing. *Computers & Operations Research*, *0*, 1–12. https://doi.org/10.1016/j.cor.2018.06.002

Helu, M., Libes, D., Lubell, J., Lyons, K., & Morris, K. C. (2016). Enabling Smart Manufacturing Technologies for Decision-Making Support. *Volume 1B: 36th Computers and Information in Engineering Conference*, *March 2017*, V01BT02A035. https://doi.org/10.1115/DETC2016-59721

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0 scenarios. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2016-March*, 3928–3937. https://doi.org/10.1109/HICSS.2016.488

Industrial Digital Twin Organization. (2023). *Part 3a: Data Specification – IEC 61360*. *April*. https://industrialdigitaltwin.org/en/content-hub/aasspecifications/specification-of-the-asset-administration-shell-part-3a-data-specification-iec-61360-idta-number-01003-a-

3-0
Ivanov, D., Dolgui, A., Sokolov, B., Werner, F., & Ivanova, M. (2016). A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0. *International Journal of Production Research*, *54*(2), 386–402. https://doi.org/10.1080/00207543.2014.999958
Iwamura, K., & Sugimura, N. (2010). A study on real-time scheduling for autonomous distributed manufacturing systems. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 1352–1357. https://doi.org/10.1109/ICSMC.2010.5642451
Jamili, A. (2016). Robust job shop scheduling problem: Mathematical models, exact and heuristic algorithms. *Expert Systems with Applications*, *55*, 341–350. https://doi.org/10.1016/j.eswa.2016.01.054
Jamrus, T., Chien, C.-F., Gen, M., & Sethanan, K. (2018). Hybrid Particle Swarm Optimization Combined With Genetic Operators for Flexible Job-Shop Scheduling Under Uncertain Processing Time for Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, *31*(1), 32–41. https://doi.org/10.1109/TSM.2017.2758380
Jia, H. Z., Fuh, J. Y. H., Nee, A. Y. C., & Zhang, Y. F. (2007). Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems. *Computers & Industrial Engineering*, *53*(2), 313–320. https://doi.org/10.1016/j.cie.2007.06.024
Jiang, J. R. (2018). An improved cyber-physical systems architecture for Industry 4.0 smart factories. *Advances in Mechanical Engineering*, *10*(6), 1–15. https://doi.org/10.1177/1687814018784192
Jung, S., Woo, Y. Bin, & Kim, B. S. (2017). Two-stage assembly scheduling problem for processing products with dynamic component-sizes and a setup time. *Computers and Industrial Engineering*, *104*, 98–113. https://doi.org/10.1016/j.cie.2016.12.030
Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: securing the future of German manufacturing industry; Final report of the Industrie 4.0 working group (Technical Report)*. *April*.
Kaplanoğlu, V. (2014). Multi-agent based approach for single machine scheduling with sequence-dependent setup times and machine maintenance. *Applied Soft Computing Journal*, *23*, 165–179. https://doi.org/10.1016/j.asoc.2014.06.020
Karimi, S., Ardalan, Z., Naderi, B., & Mohammadi, M. (2016). 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
Klement, N., Abdeljaouad, M. A., Porto, L., & Silva, C. (2021). Lot-Sizing and Scheduling for the Plastic Injection Molding Industry—A Hybrid Optimization Approach. *Applied Sciences*, *11*(3), 1202. https://doi.org/10.3390/app11031202
Kosanke, K. (1995). *CIMOSA - Overview and status*. *27*, 101–109.
Kuhpfahl, J., & Bierwirth, C. (2016). A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers and Operations Research*, *66*, 44–57. https://doi.org/10.1016/j.cor.2015.07.011
Kumar Khaitan Siddhartha, & McCalley, D. J. (2015). Design Techniques and Applications of Cyber Physical Systems: A Survey. *IEEE Systems Journal*, 1–15.

https://www.academia.edu/23178627/Design_Techniques_and_Applications_of_Cyber_Physical_Systems_A_Survey

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

Kusiak, A. (2018). Smart manufacturing. *International Journal of Production Research*, *56*(1–2), 508–517. https://doi.org/10.1080/00207543.2017.1351644

Ladj, A., Varnier, C., & Tayeb, F. B. S. (2016). IPro-GA: an integrated prognostic based GA for scheduling jobs and predictive maintenance in a single multifunctional machine. *IFAC-PapersOnLine*, *49*(12), 1821–1826. https://doi.org/10.1016/j.ifacol.2016.07.847

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

Lee, W. C., Wang, J. Y., & Lin, M. C. (2016). A branch-and-bound algorithm for minimizing the total weighted completion time on parallel identical machines with two competing agents. *Knowledge-Based Systems*, *105*, 68–82. https://doi.org/10.1016/j.knosys.2016.05.012

Lei, D., Gao, L., & Zheng, Y. (2018). A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop. *IEEE Transactions on Engineering Management*, *65*(2), 330–340. https://doi.org/10.1109/TEM.2017.2774281

Lei, H., Xing, K., Han, L., & Gao, Z. (2017). Hybrid heuristic search approach for deadlock-free scheduling of flexible manufacturing systems using Petri nets. *Applied Soft Computing*, *18*(2), 240–245. https://doi.org/10.1016/j.asoc.2017.01.045

Leitao, P., Barbosa, J., Papadopoulou, M.-E. C., & Venieris, I. S. (2015). Standardization in cyber-physical systems: The ARUM case. *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2988–2993. https://doi.org/10.1109/ICIT.2015.7125539

Leitão, P., Barbosa, J., Pereira, A., Barata, J., & Colombo, A. W. (2016). Specification of the PERFoRM architecture for the seamless production system reconfiguration. *IECON Proceedings (Industrial Electronics Conference)*, 5729–5734. https://doi.org/10.1109/IECON.2016.7793007

Leitao, P., Cristalli, C., Paone, N., Chiariotti, P., Utz, W., Stojanovic, N., Barata, J., & Woitsch, R. (2022). Integrating Standardization in Research and Innovation Projects: the GO0DMAN Experience. *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, 1147–1152. https://doi.org/10.1109/ISIE51582.2022.9831486

Li, Q., Tang, Q., Chan, I., Wei, H., Pu, Y., Jiang, H., Li, J., & Zhou, J. (2018). Smart manufacturing standardization: Architectures, reference models and standards framework. *Computers in Industry*, *101*(July), 91–106. https://doi.org/10.1016/j.compind.2018.06.005

Li, R., Gong, W., & Lu, C. (2022). Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time. *Computers and Industrial Engineering*, *168*(February), 108099. https://doi.org/10.1016/j.cie.2022.108099

Li, X., & Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, *174*, 93–110. https://doi.org/10.1016/j.ijpe.2016.01.016

Lin, L., Hao, X. C., Gen, M., & Jo, J. B. (2012). Network modeling and evolutionary optimization for scheduling in manufacturing. *Journal of Intelligent Manufacturing*, *23*(6), 2237–2253. https://doi.org/10.1007/s10845-011-0569-6

Liu, Q., Dong, M., & Chen, F. F. (2018). Single-machine-based joint optimization of predictive maintenance planning and production scheduling. *Robotics and Computer-Integrated Manufacturing*, *51*(January), 238–247. https://doi.org/10.1016/j.rcim.2018.01.002

Liu, Y., Wang, L., Wang, X. V., Xu, X., & Zhang, L. (2018). Scheduling in cloud manufacturing: state-of-the-art and research challenges. *International Journal of Production Research*, *7543*, 1–26. https://doi.org/10.1080/00207543.2018.1449978

Liu, Y., Wang, L., Wang, Y., Wang, X. V., & Zhang, L. (2018). Multi-agent-based scheduling in cloud manufacturing with dynamic task arrivals. *Procedia CIRP*, *72*, 953–960. https://doi.org/10.1016/j.procir.2018.03.138

Lu, C., Gao, L., Li, X., Pan, Q., & Wang, Q. (2017). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *Journal of Cleaner Production*. https://doi.org/10.1016/j.jclepro.2017.01.011

Lu, P.-H., Wu, M.-C., Tan, H., Peng, Y.-H., & Chen, C.-F. (2018). A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. *Journal of Intelligent Manufacturing*, *29*(1), 19–34. https://doi.org/10.1007/s10845-015-1083-z

Madureira, A., Pereira, I., & Abraham, A. (2013). Towards Scheduling Optimization through Artificial Bee Colony Approach. *2013 World Congress on Nature and Biologically Inspired Computing*, *August*, 253–258. https://doi.org/10.1109/NaBIC.2013.6617872

Majdzik, P. (2022). A Feasible Schedule for Parallel Assembly Tasks in Flexible Manufacturing Systems. *International Journal of Applied Mathematics and Computer Science*, *32*(1), 51–63. https://doi.org/10.34768/amcs-2022-0005

Marzouki, B., Belkahla Driss, O., & Ghédira, K. (2017). Multi Agent model based on Chemical Reaction Optimization with Greedy algorithm for Flexible Job shop Scheduling Problem. *Procedia Computer Science*, *112*, 81–90. https://doi.org/10.1016/j.procs.2017.08.174

Mattfeld, D. C. (1996). Evolutionary search and the job shop : investigations on genetic algorithms for production scheduling. *Production and Logistics*, *November*, ix, 152 p. https://doi.org/10.1007/978-3-662-11712-5

Mohammadi, S., Al-e-Hashem, S. M. J. M., & Rekik, Y. (2020). An integrated production scheduling and delivery route planning with multi-purpose machines: A case study from a furniture manufacturing company. *International Journal of Production Economics*, *219*(May 2019), 347–359. https://doi.org/10.1016/j.ijpe.2019.05.017

Mokhtari, H., & Hasani, A. (2017). An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers and Chemical Engineering*, *104*, 339–352. https://doi.org/10.1016/j.compchemeng.2017.05.004

Mokhtari, H., & Noroozi, A. (2018). An efficient chaotic based PSO for earliness/tardiness optimization in a batch processing flow shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(5), 1063–1081. https://doi.org/10.1007/s10845-015-1158-x

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

Moosavi, A., Ozturk, O., & Patrick, J. (2022). Staff scheduling for residential care under pandemic conditions: The case of COVID-19. *Omega (United Kingdom)*, *112*(April), 102671. https://doi.org/10.1016/j.omega.2022.102671

Mou, J., Li, X., Gao, L., & Yi, W. (2018). An effective L-MONG algorithm for solving multi-

objective flow-shop inverse scheduling problems. *Journal of Intelligent Manufacturing*, *29*(4), 789–807. https://doi.org/10.1007/s10845-015-1129-2

Mourtzis, D., & Vlachou, E. (2018). A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance. *Journal of Manufacturing Systems*, *47*, 179–198. https://doi.org/10.1016/j.jmsy.2018.05.008

Nahavandi, S. (2019). Industry 5.0—A Human-Centric Solution. *Sustainability*, *11*(16), 4371. https://doi.org/10.3390/su11164371

Nguyen, S., Mei, Y., & Zhang, M. (2017). Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, *3*(1), 41–66. https://doi.org/10.1007/s40747-017-0036-x

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2014). Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation*, *18*(2), 193–208. https://doi.org/10.1109/TEVC.2013.2248159

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

Nikolakis, N., Kousi, N., Michalos, G., & Makris, S. (2018). Dynamic scheduling of shared human-robot manufacturing operations. *Procedia CIRP*, *72*, 9–14. https://doi.org/10.1016/j.procir.2018.04.007

Parsa, N. R., Karimi, B., & Husseini, S. M. M. (2017). Exact and heuristic algorithms for the just-in-time scheduling problem in a batch processing system. *Computers and Operations Research*, *80*, 173–183. https://doi.org/10.1016/j.cor.2016.12.001

PERFoRM, P. (2016). *Deliverable D2.2 - Definition of the System Architecture*. 1–82. http://www.horizon2020-perform.eu/files/documents/D2_2_public.pdf

Perlo, P., Rocha, A., Alemao, D., Freitas, N., & Oliveira, F. (2022). *ICT infrastructure supporting seamless integration of certification procedures in microfactories*. *November*, 0–5.

Petrović, M., Vuković, N., Mitić, M., & Miljković, Z. (2016). Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. *Expert Systems with Applications*, *64*, 569–588. https://doi.org/10.1016/j.eswa.2016.08.019

Pinedo, M. L. (2016). *Scheduling Theory, Algorithms, and Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-319-26580-3

Platenius-Mohr, M., Malakuti, S., Grüner, S., Schmitt, J., & Goldschmidt, T. (2020). File- and API-based interoperability of digital twins by model transformation: An IIoT case study using asset administration shell. *Future Generation Computer Systems*, *113*, 94–105. https://doi.org/10.1016/j.future.2020.07.004

Platform Industrie 4.0. (2018). *The Structure of the Administration Shell: Trilateral Perspective from France, Italy and Germany*. 64. www.entreprises.gouv.fr

Platform Industrie 4.0. (2020). Details of the Asset Administration Shell Part 2 – Interoperability at Runtime – Exchanging Information via Application. *Plattform Industrie 4.0*, 1–90. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html

Platform Industrie 4.0. (2022). *Details of the Asset Administration Shell Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC02)*.

https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html

Plattform Industrie 4.0. (n.d.). *AASX Packge Explorer*. https://github.com/eclipse-aaspe/package-explorer

Plattform Industrie 4.0. (2016). *Structure of the Administration Shell Continuation of the Development of the Reference Model for the Industrie 4.0 Component*. www.bmwi.de

Pourabbas, E., Parretti, C., Rolli, F., & Pecoraro, F. (2021). Entropy-Based Assessment of Nonfunctional Requirements in Axiomatic Design. *IEEE Access*, *9*, 156831–156845. https://doi.org/10.1109/ACCESS.2021.3128686

Qu, S., Wang, J., Govil, S., & Leckie, J. O. (2016). Optimized Adaptive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach. *Procedia CIRP*, *57*, 55–60. https://doi.org/10.1016/j.procir.2016.11.011

Quadrini, W., Cimino, C., Abdel-Aty, T. A., Fumagalli, L., & Rovere, D. (2022). Asset Administration Shell as an interoperable enabler of Industry 4.0 software architectures: a case study. *Procedia Computer Science*, *217*, 1794–1802. https://doi.org/10.1016/j.procs.2022.12.379

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

Ribeiro, L., & Bjorkman, M. (2018). Transitioning From Standard Automation Solutions to Cyber-Physical Production Systems: An Assessment of Critical Conceptual and Technical Challenges. *IEEE Systems Journal*, *12*(4), 3816–3827. https://doi.org/10.1109/JSYST.2017.2771139

Rocha, A. D., Freitas, N., Alemão, D., Guedes, M., Martins, R., & Barata, J. (2021). Event-Driven Interoperable Manufacturing Ecosystem for Energy Consumption Monitoring. *Energies*, *14*(12), 3620. https://doi.org/10.3390/en14123620

Rossit, D. A., Tohmé, F., & Frutos, M. (2019). Industry 4.0: Smart Scheduling. *International Journal of Production Research*, *57*(12), 3802–3813. https://doi.org/10.1080/00207543.2018.1504248

Rossit, D., & Tohmé, F. (2018). Scheduling research contributions to Smart manufacturing. *Manufacturing Letters*, *15*, 111–114. https://doi.org/10.1016/j.mfglet.2017.12.005

Rupp, C. (2020). *REQUIREMENTS-ENGINEERING UND -MANAGEMENT* (7th ed.). Carl Hanser Verlag GmbH & Co. KG.

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., & Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Computers and Industrial Engineering*, *86*, 2–13. https://doi.org/10.1016/j.cie.2015.01.003

Sakurada, L., Leitao, P., & La Prieta, F. De. (2022). Agent-Based Asset Administration Shell Approach for Digitizing Industrial Assets. *IFAC-PapersOnLine*, *55*(2), 193–198. https://doi.org/10.1016/j.ifacol.2022.04.192

Salido, M. A., Escamilla, J., Barber, F., & Giret, A. (2017). Rescheduling in job-shop problems for sustainable manufacturing systems. *Journal of Cleaner Production*, *162*, S121–S132. https://doi.org/10.1016/j.jclepro.2016.11.002

Scholze, S., Stokic, D., Kotte, O., Barata, J., Orio, G. Di, & Candido, G. (2013). Reliable Self-Learning Production Systems Based on Context Aware Services. *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 4872–4877. https://doi.org/10.1109/SMC.2013.829

Schweichhart, K. (2016). *RAMI 4.0 reference architectural model for Industrie 4.0*. https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf

Serrano-Ruiz, J. C., Mula, J., & Poler, R. (2022). Development of a multidimensional conceptual model for job shop smart manufacturing scheduling from the Industry 4.0 perspective. *Journal of Manufacturing Systems*, *63*(January), 185–202. https://doi.org/10.1016/j.jmsy.2022.03.011

Shahrabi, J., Adibi, M. A., & Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers and Industrial Engineering*, *110*, 75–82. https://doi.org/10.1016/j.cie.2017.05.026

Shen, L., Dauzère-Pérès, S., & Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, *265*(2), 503–516. https://doi.org/10.1016/j.ejor.2017.08.021

Shi-Wan, L., Bradford, M., Jacques, D., Graham, B., Chigani, A., Martin, R., Murphy, B., & Crawford, M. (2017). The Industrial Internet of Things Volume G1 : Reference Architecture. *Industrial Internet Consortium White Paper*, *Version 1.*, 58 Seiten.

Shiue, Y.-R., Lee, K.-C., & Su, C.-T. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering*, *101*, 0–1. https://doi.org/10.1016/j.cie.2018.03.039

Silva, C., Klement, N., & Gibaru, O. (2018). *A Generic Decision Support Tool for Lot-Sizing and Scheduling Problems with Setup and Due Dates* (Issue Ijc 2016, pp. 131–138). https://doi.org/10.1007/978-3-319-58409-6_15

Singh, M. R., & Mahapatra, S. S. (2015). A Quantum Behaved Particle Swarm Optimization for Flexible Job Shop Scheduling. *Computers & Industrial Engineering*, *93*, 36–44. https://doi.org/10.1016/j.cie.2015.12.004

Sobeyko, O., & Mönch, L. (2016). Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Computers and Operations Research*, *68*, 97–109. https://doi.org/10.1016/j.cor.2015.11.004

Souza, R. L. C., Ghasemi, A., Saif, A., & Gharaei, A. (2022). Robust job-shop scheduling under deterministic and stochastic unavailability constraints due to preventive and corrective maintenance. *Computers and Industrial Engineering*, *168*(October 2021), 108130. https://doi.org/10.1016/j.cie.2022.108130

Stock, T., & Seliger, G. (2016). Opportunities of Sustainable Manufacturing in Industry 4.0. *Procedia CIRP*, *40*(Icc), 536–541. https://doi.org/10.1016/j.procir.2016.01.129

Tang, D., Dai, M., Salido, M. A., & Giret, A. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry*, *81*, 82–95. https://doi.org/10.1016/j.compind.2015.10.001

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *International Journal of Advanced Manufacturing Technology*, *94*(9–12), 3563–3576. https://doi.org/10.1007/s00170-017-0233-1

Tao, F., Qi, Q., Liu, A., & Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems*, *48*, 157–169. https://doi.org/10.1016/j.jmsy.2018.01.006

Tao, F., Qi, Q., Wang, L., & Nee, A. Y. C. (2019). Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. *Engineering*, *5*(4), 653–661. https://doi.org/10.1016/j.eng.2019.01.014

Tran, L. V., Huynh, B. H., & Akhtar, H. (2019). Ant Colony Optimization Algorithm for Maintenance, Repair and Overhaul Scheduling Optimization in the Context of Industrie 4.0. *Applied Sciences*, *9*(22), 4815. https://doi.org/10.3390/app9224815

Uhlmann, I. R., Zanella, R. M., & Frazzon, E. M. (2022). Hybrid flow shop rescheduling for contract manufacturing services. *International Journal of Production Research*, *60*(3), 1069–1085. https://doi.org/10.1080/00207543.2020.1851422

Voulgaridis, K., Lagkas, T., & Sarigiannidis, P. (2022). Towards Industry 5.0 and Digital Circular Economy: Current Research and Application Trends. *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 153–158. https://doi.org/10.1109/DCOSS54816.2022.00037

Wang, G. G., Gao, D., & Pedrycz, W. (2022). Solving Multiobjective Fuzzy Job-Shop Scheduling Problem by a Hybrid Adaptive Differential Evolution Algorithm. *IEEE Transactions on Industrial Informatics*, *18*(12), 8519–8528. https://doi.org/10.1109/TII.2022.3165636

Wang, J., Tang, J., Xue, G., & Yang, D. (2016). Towards Energy-efficient Task Scheduling on Smartphones in Mobile Crowd Sensing Systems. *Computer Networks*. https://doi.org/10.1016/j.comnet.2016.11.020

Wang, L., Cai, J., Li, M., & Liu, Z. (2017). Flexible Job Shop Scheduling Problem Using an Improved Ant Colony Optimization. *Scientific Programming*, *2017*. https://doi.org/10.1155/2017/9016303

Wang, X., Zhang, L., Liu, Y., Li, F., Chen, Z., Zhao, C., & Bai, T. (2022). Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning. *Journal of Manufacturing Systems*, *65*(July), 130–145. https://doi.org/10.1016/j.jmsy.2022.08.004

Wuest, T., Weimer, D., Irgens, C., & Thoben, K.-D. (2016). Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, *4*(1), 23–45. https://doi.org/10.1080/21693277.2016.1192517

Xiang, W., Member, S., Yu, K., Han, F., Member, S., & Fang, L. (2023). Advanced Manufacturing in Industry 5 . 0 : A Survey of Key Enabling Technologies and Future Trends. *IEEE Transactions on Industrial Informatics*, 1–15. https://doi.org/10.1109/TII.2023.3274224

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

Xiong, H., Shi, S., Ren, D., & Hu, J. (2022). A survey of job shop scheduling problem : The types and models. *Computers and Operations Research*, *142*(May 2021), 105731. https://doi.org/https://doi.org/10.1016/j.cor.2022.105731

Xiong, W., & Fu, D. (2015). A new immune multi-agent system for the flexible job shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(4), 857–873. https://doi.org/10.1007/s10845-015-1137-2

Xu, X., Lu, Y., Vogel-Heuser, B., & Wang, L. (2021). Industry 4.0 and Industry 5.0—Inception, conception and perception. *Journal of Manufacturing Systems*, *61*, 530–535.

https://doi.org/10.1016/j.jmsy.2021.10.006
Yamada, T., & Nakano, R. (1997). Job-shop scheduling. In *Genetic algorithms in engineering systems* (pp. 134–160).
Yazdani, M., Aleti, A., Khalili, S. M., & Jolai, F. (2017). Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem. *Computers and Industrial Engineering*, *107*, 12–24. https://doi.org/10.1016/j.cie.2017.02.019
Ye, X., & Hong, S. H. (2019). Toward Industry 4.0 Components: Insights into and Implementation of Asset Administration Shells. *IEEE Industrial Electronics Magazine*, *13*(1), 13–25. https://doi.org/10.1109/MIE.2019.2893397
Yoo, J., & Lee, I. S. (2016). Parallel machine scheduling with maintenance activities. *Computers & Industrial Engineering*, *101*, 361–371. https://doi.org/10.1016/j.cie.2016.09.020
Zarook, Y., & Abedi, M. (2014). JIT-scheduling in unrelated parallel-machine environment with aging effect and multi-maintenance activities. *International Journal of Services and Operations Management*, *18*(1), 99. https://doi.org/10.1504/IJSOM.2014.060455
Zeng, Z., Hong, M., Man, Y., Li, J., Zhang, Y., & Liu, H. (2018). Multi-object optimization of flexible flow shop scheduling with batch process — Consideration total electricity consumption and material wastage. *Journal of Cleaner Production*, *183*, 925–939. https://doi.org/10.1016/j.jclepro.2018.02.224
Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing*, *30*(4), 1809–1830. https://doi.org/10.1007/s10845-017-1350-2
Zhang, L., Gao, L., & Li, X. (2013). A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem. *International Journal of Production Research*, *51*(12), 3516–3531. https://doi.org/10.1080/00207543.2012.751509
Zhang, M., Tao, F., & Nee, A. Y. C. (2021). Digital Twin Enhanced Dynamic Job-Shop Scheduling. *Journal of Manufacturing Systems*, *58*(PB), 146–156. https://doi.org/10.1016/j.jmsy.2020.04.008
Zhang, Yi, Zhu, H., Tang, D., Zhou, T., & Gui, Y. (2022). Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, *78*(July), 102412. https://doi.org/10.1016/j.rcim.2022.102412
Zhang, Yingfeng, Liu, S., Liu, Y., Yang, H., Li, M., Huisingh, D., & Wang, L. (2018). The 'Internet of Things' enabled real-time scheduling for remanufacturing of automobile engines. *Journal of Cleaner Production*, *185*, 562–575. https://doi.org/10.1016/j.jclepro.2018.02.061
Zhang, Z., Wu, F., Qian, B., Hu, R., Wang, L., & Jin, H. (2023). *A Q -learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation*. *234*(March).
Zhao, F., Tang, J., Wang, J., & Jonrinaldi. (2014). An improved particle swarm optimization with decline disturbance index (DDPSO) for multi-objective job-shop scheduling problem. *Computers and Operations Research*, *45*, 38–50. https://doi.org/10.1016/j.cor.2013.11.019
Zhong, R. Y., Dai, Q. Y., Qu, T., Hu, G. J., & Huang, G. Q. (2013). RFID-enabled real-time manufacturing execution system for mass-customization production. *Robotics and Computer-Integrated Manufacturing*, *29*(2), 283–292. https://doi.org/10.1016/j.rcim.2012.08.001

Zhuang, C., Liu, J., & Xiong, H. (2018). Digital twin-based smart production management and control framework for the complex product assembly shop-floor. *International Journal of Advanced Manufacturing Technology*, *96*(1–4), 1149–1163. https://doi.org/10.1007/s00170-018-1617-6

ZVEI. (2011). Manufacturing Execution Systems (MES): Industry specific Requirements and Solutions. In *Berthold Druck GmbH* (Vol. 49, Issue July). www.zvei.org

ZVEI. (2016). *Implementation Strategy Industrie 4.0*. 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

ZVEI. (2018). *Drive 4.0 – Vision Becomes Reality*.

ZVEI. (2019). *Communication in the Context of Industrie 4.0*. https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2019/Maerz/Communication_in_the_Context_of_Industrie_4.0/ZVEI_WP_Kommunikation_Industrie-4.0-Umfeld_ENGLISCH.pdf

2024

Duarte José Marques Alemão

Modelling Smart Manufacturing Assets Targeting Scheduling Optimisation