University of Nottingham
UK | CHINA | MALAYSIA

A holistic methodology for manufacturing systems configuration

Thesis submitted to the University of Nottingham for the degree of **Doctor of Philosophy**

Agajan Torayev

Faculty of Engineering

University of Nottingham

Signature:

January 30, 2024

Abstract

The manufacturing sector is undergoing rapid transformations driven by global economic factors, technological advancements, and fluctuating market demands. These dynamics necessitate continuous innovation and adaptability in manufacturing systems to maintain competitiveness.

Therefore, this research addresses the manufacturing systems configuration (MSC) problem, aiming to develop a methodology to make manufacturing systems resilient and adaptable that balances resource management, production efficiency, and costs.

Despite the critical nature of the MSC problem, existing solutions are fragmented and suffer from significant limitations, including underutilisation of data models, software integration issues, and the inadequacy of traditional optimisation and decision-making methods in dealing with uncertainties and multiple objectives.

Therefore, this research formulates the problem as "developing a holistic solution for addressing the MSC problem for adapting manufacturing systems to rapidly changing manufacturing requirements" to address these gaps. In this context, a holistic solution synergistically combines data modelling, software integration, adaptive optimisation and decision-making algorithms.

The research objectives include the development of adaptable data models that encapsulate the complexities of manufacturing systems, plug-and-produce manu-

facturing software solutions that address system scalability and adaptability, and adaptive optimisation algorithms capable of navigating complex solution spaces.

The research employs a multi-staged validation approach, initially testing the proposed methodologies in two distinct manufacturing processes with unique challenges: sorting cylinders and bin-picking parts of industrial pipe couplers. These processes serve as a comprehensive testing ground for the proposed solutions. Three research hypotheses were sequentially assessed, focusing on the adaptability of object-oriented data models, the effectiveness of manufacturing apps in achieving interoperability, and the efficiency of optimisation and decision-making algorithms in managing multiple objectives and uncertainties. Each hypothesis was successfully validated, confirming the research contributions.

Subsequently, empirical validation was extended to real-world industrial settings, focusing on aerospace and custom product manufacturing sectors. In the aerospace sector, the task was to find optimal manufacturing system configurations for changing and multiple conflicting manufacturing costs for assembling a generic hinged product. In the custom product manufacturing sector, the task involved planning a machining process that required balancing multiple manufacturing costs. These validations substantiate the research hypotheses and demonstrate the proposed methodology's generalisability and adaptability.

By developing a holistic approach, this research contributes significantly to the field. It addresses the limitations of existing fragmented solutions and provides a robust, adaptable, and holistic framework for manufacturing systems. The research has practical implications for manufacturing entities aiming to be agile and responsive to market changes, fulfilling the main aim of developing a holistic solution to the MSC problem.

Acknowledgements

Firstly, I want to express my sincere thanks to Professor Svetan Ratchev. His constant help, advice, and support were essential throughout my PhD journey. He was much more than a supervisor to me, guiding me in many aspects of life and helping shape my worldview. His practical advice that a problem should be identified before its solution has played a huge role in my development as a researcher and my general mindset. I will never forget his research and life-related advice in our supervision meetings and casual chats.

My gratitude extends to Dr Giovanna Martínez-Arellano for her supervision and support. Our fruitful discussions on machine learning, artificial intelligence, and other technical topics were crucial to the development of the technical parts of my thesis. She encouraged me to explore and brainstorm ideas, some of which seemed far into the future, but proved incredibly valuable.

I am also grateful to Dr Jack C Chaplin for his steady supervision and guidance. Whenever I faced a problem, he was there with an answer or some wisdom to help. His practical support with my secondments, the DiManD project, and in proofreading my publications was invaluable. He not only highlighted my mistakes but provided clear explanations, which helped me to grow and improve as a researcher.

I owe immense gratitude to Dr David Sanderson for his unwavering supervision and support throughout my PhD. His aid in merging my backgrounds in Computer Science and Manufacturing Engineering was invaluable. Dave's critical reviews of my work significantly enhanced my research quality. His help in integrating my solutions into industrial robots, along with guidance on addressing reviewers' comments succinctly, deeply enriched my research.

I also want to thank Professor Atanas Popov for his supervision and advice. His straightforward suggestions for my annual reports and PhD thesis were of great help. He always provided motivational conversations and positive advice, which I found essential during my PhD.

My thanks also go to Helena Arrand, Kryssa Roycroft, and Nancy Martin for all their administrative help. They took care of organising my supervision meetings, secondments, purchase orders, and documents, which allowed me to focus more on my research.

I cannot forget to thank my lovely wife, Bilbil. She provided me with emotional support throughout my PhD journey, especially during the lockdown period. She took care of our two children, enabling me to focus on my work. Her motivation and faith in me, even when I made less than ideal choices, were truly helpful.

I want to express my gratitude to my parents, Akjagul and Bazargeldi. They supported me both emotionally and materially from the start, and I wouldn't be where I am today without them. I will always remember my father's wise words, that the best investment is in knowledge.

My brother, Dovletmyrat, deserves my sincere thanks. Despite being younger than me, he often acted like an older brother and supported me throughout my PhD journey. His wise advice sometimes made me reconsider my decisions.

Lastly, a message to my children, Selbi and Selim: If you ever read this PhD thesis, remember the importance of staying curious and exploring the unknown. Only by doing this can you progress towards a brighter future. I hope you will also choose to pursue a PhD, just like your father.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 814078

This research's scope was rooted in the digital manufacturing and design (DiManD) innovative training network (ITN), a programme funded by the European Union's Horizon 2020 under the Marie Skłodowska-Curie grant agreement No 814078. The project's core objective was to enhance Europe's industrial competitiveness by designing and implementing an integrated programme focused on intelligent informatics-driven manufacturing. This programme aimed to become a benchmark for training future practitioners in Industry 4.0. It also aspired to develop a high-quality multidisciplinary, multi-professional, and cross-sectorial research and training framework that formed the backbone of Europe's industrial future.

The programme's vision was to facilitate the design and development of integrated systems to effectively overcome barriers to Industry 4.0 adoption. The project sought to foster experts equipped with a multidisciplinary skill set, enabling them to generate a new generation of solutions congruent with the Industry 4.0 ethos, which included adapting current solutions to this novel approach to producing complex, high-value products.

Contents

List of Tables

List of Figures

Abbreviations

API application programming interface.

ASF achievement scalarisation function.

CAD computer-aided design.

CIDENA changes identification algorithm.

CNC computer numerical control.

CPS cyber-physical system.

CSV comma-separated values.

DBMS database management system.

DFO derivative-free optimisation.

DiManD digital manufacturing and design.

DMS dedicated manufacturing systems.

EA evolutionary algorithms.

EAS evolvable assembly systems.

E-R entity-relationship.

ERP enterprise resource planning.

EU European Union.

FMS flexible manufacturing systems.

GA genetic algorithms.

IGES initial graphics exchange specifications.

IoT internet of things.

ITN innovative training network.

KPI key performance indicator.

LiDAR light detection and ranging.

MAPPDK manufacturing apps development kit.

MCDM multiple-criteria decision-making.

MES manufacturing execution system.

MFGRL manufacturing reinforcement learning environment.

MILP mixed-integer linear programming.

MPM manufacturing process management.

MSC manufacturing systems configuration.

NSGA-II non-dominated sorting genetic algorithm 2.

O3PARAMS online optimisation of operational parameters.

OEM original equipment manufacturer.

OODM object-oriented data modelling.

PDM product management.

PW pseudo-weights.

RL reinforcement learning.

RMS reconfigurable manufacturing systems.

SCADA supervisory control and data acquisition.

SDK software development kit.

STEP standard for the exchange of product data.

TOPSIS technique for order of preference by similarity to ideal solution.

UML unified modelling language.

WP work package.

Chapter 1

Introduction

1.1 Background and motivation

Global economic factors, technological progress, and market trends influence modern manufacturing systems by constantly changing the operational landscape [1]. Changes in product design, production capacity, and operational efficiency standards are some examples of factors driving these changes. Given this dynamic environment, manufacturing systems must constantly innovate and improve their practices. They must fine-tune their operations to meet changing manufacturing needs, maintain high quality, and stay competitive [2].

This persistent need for adaptability often manifests in notable disruptions within the sector. For instance, if not efficiently addressed, a surge in product demand could trigger a production bottleneck, adversely impacting potential sales and overall consumer satisfaction levels. Severe instances of this cause-effect scenario could push an organisation out of the market competition. Furthermore, matters escalate when inefficient optimisation of critical operational parameters, like energy consumption, escalates production costs and breaches environmental standards.

Addressing these disruptions requires systematic configuration, re-configuration, and enhancement of manufacturing systems through strategic planning, detailed data analytics, comprehensive system modelling, focused algorithmic optimisation and decision-making. This problem is referred as manufacturing systems configuration (MSC) problem in this thesis, which aims to nurture a resilient system capable of swiftly adapting to pervasive manufacturing changes whilst maintaining a balance of resource management, production efficiency, and expense overheads.

However, the current approaches to tackle the MSC problem are fragmented and characterised by significant limitations, most of which originate from the existing knowledge gaps identified in the literature:

1. **Underutilisation of data models.** Data models play a crucial role in representing the complexities of manufacturing systems and facilitating optimisation efforts. However, in the context of the MSC problem, there is a concerning shortfall in harnessing the full potential of suitable data models. Manufacturing systems are intricate ecosystems with multifaceted interactions between various components such as machines, processes, materials, and human operators. These interactions lead to dynamic behaviours, uncertainties, and variations that shape the operational landscape [3].

   Despite the critical role of data models in system representation, existing approaches often overlook their application in addressing the MSC problem. This oversight hampers optimisation efforts by hindering the accurate capture of the intricacies and variability of manufacturing systems. For instance, intricate interdependencies between production processes and resource utilisation are challenging to model accurately without appropriate data models [4–6].

2. **Software integration and interoperability.** In an era of digital transformation, manufacturing systems increasingly depend on digital platforms for efficient operations. However, the issue of software integration and interoperability poses a significant challenge. Manufacturing systems often rely on a mix of software applications, such as enterprise resource planning (ERP), manufacturing execution system (MES), and supervisory control and data acquisition (SCADA) systems, to manage various aspects of operations [7, 8].

   Despite this reliance on digital platforms, seamless software integration solutions remain elusive. Incompatibility between different software systems results in data silos, limited communication, and challenges in data exchange. This lack of integration hampers system efficiency and adaptability in a rapidly changing manufacturing environment. For instance, disruptions caused by sudden market changes may require rapid adjustments in production schedules, which become difficult when software systems cannot

communicate seamlessly [9, 10].

3. **Limitations of traditional optimisation and decision-making methods.** Traditional optimisation methods have been extensively applied in various manufacturing contexts. However, these methods exhibit significant limitations when addressing the complexities of the MSC problem. Manufacturing systems are subject to uncertainties in demand, evolving operational objectives, and the need for sequential decision-making [11, 12]. Traditional optimisation techniques, such as linear programming or heuristic algorithms, struggle to handle these complexities effectively.

   For instance, evolving market demands require manufacturing systems to adapt rapidly to production volume and product mix changes. Traditional methods may not be equipped to handle dynamic objectives or real-time adjustments in resource allocation. Furthermore, uncertainties in supply chain disruptions or raw material availability can significantly impact production schedules and resource utilisation, necessitating adaptive optimisation strategies [13, 14].

The outlined limitations in the current solutions for the MSC problem paint a clear picture of a fragmented landscape [1, 2]. While efforts are being made in various areas, these endeavours are not cohesive and lack the interconnectedness necessary to address the problem holistically. For instance, improving data models alone will not suffice if the software systems they operate lack integration or the optimisation methods employed are not dynamic enough to adjust to evolving manufacturing scenarios [3, 7]. Current solutions resemble pieces of a jigsaw puzzle that, while essential, do not quite fit together perfectly to reveal the full picture [15].

Such fragmentation impedes fostering resilient and adaptable manufacturing systems [2]. For example, an optimally designed data model, which captures all nuances of the manufacturing process, loses its efficacy if integrated into a software

system that fails to communicate seamlessly with other mission-critical systems [7, 8]. The consequential data silos undermine the model's potential and introduce inefficiencies into the system, defeating the primary objective. Similarly, even if we achieve software interoperability but employ outdated optimisation methods, the result will be sub-optimal decision-making that may not account for rapidly changing demand patterns or supply chain disruptions [11, 12].

By its very nature, manufacturing is a system of interconnected components [1]. The same interdependence observed in the manufacturing process must be mirrored in the solutions crafted to address its challenges. Though they might offer improvements in isolated areas, fragmented solutions can introduce unforeseen complications when implemented in tandem [15]. For instance, enhancing one area without considering the full system dynamics might inadvertently stress another, leading to unforeseen bottlenecks or inefficiencies [3].

Moreover, in the modern manufacturing landscape, marked by Industry 4.0 and the rise of the internet of things (IoT), the need for holistic solutions becomes even more pressing [16]. As manufacturing entities increasingly rely on interconnected devices and real-time data flows, the margin for error shrinks [9]. A disjointed approach in such an environment can lead to inefficiencies and potential vulnerabilities, as a minor change in one area can cascade across the system [10]. Thus, only a unified approach that considers the manufacturing system's entirety and addresses its challenges cohesively can ensure robustness, adaptability, and longevity in the face of evolving global demands and technological advances.

For manufacturing systems to be agile and responsive to market changes, every piece of the solution – from data models and software systems to optimisation methodologies – must work in harmony, emphasising the dire need for a more holistic approach.

1.2 Aims and objectives

Considering the current state of fragmented solutions to the MSC problem, ***the primary aim of this research is to develop a holistic methodology that combines data models, software, and optimisation algorithms for addressing the MSC problem***.

The aim is to be achieved by addressing the below objectives:

1. **Development of comprehensive and adaptable data models:** The initial objective involves the development of data models that encapsulate the complexities inherent in manufacturing systems. These models should serve as a fundament for data that facilitates the analysis and interpretation. Importantly, the models should be adaptable, allowing for modifications to accommodate shifts in demand, design specifications, or changes in overarching objectives. This adaptability ensures that the data models remain relevant and useful throughout the lifecycle of the manufacturing system, thereby providing a solid foundation for subsequent research activities.

2. **Development of plug-and-produce manufacturing software solutions:** The second objective focuses on creating software solutions that utilise plug-and-produce technology. This specific technology is crucial for rapidly integrating various manufacturing equipment into a unified system. The software aims to streamline both the initial setup and any subsequent re-configurations, effectively addressing system scalability and adaptability challenges. By facilitating easier integration and disconnection of various components, the software aims to enhance its overall flexibility, allowing it to adapt to changing operational requirements without requiring extensive modifications.

3. **Development of adaptive optimisation and decision-making algo-**

rithms: The final objective concerns formulating algorithms capable of navigating the complex solution space defined by the data models. These algorithms should be designed to identify optimal configurations for the manufacturing system under various conditions and constraints. The optimisation process should be adaptive and capable of handling multiple, often conflicting, objectives. By building upon the comprehensive data models, these algorithms should aim to provide actionable insights that can be directly implemented in the manufacturing system.

A rigorous validation process will be conducted to ensure the robustness and applicability of the proposed holistic solution. The validation process will sequentially assess the adaptability and robustness of the developed data models, the effectiveness of the plug-and-produce manufacturing software, and the efficiency of the adaptive optimisation and decision-making algorithms. Two distinct manufacturing processes with unique challenges and mechanisms will be the experimental backdrop for these validations. This approach offers a comprehensive testing ground for the proposed methodologies.

Following the initial validation, empirical substantiation will be done in real-world industrial settings. The aim is to extend the validation to complex manufacturing environments known for stringent quality requirements and high variability. These case studies will corroborate the initial findings and provide actionable insights into how the proposed solutions can be tailored to meet the unique challenges of different manufacturing environments. This multi-layered validation approach will significantly enhance the credibility and practical relevance of the research contributions, thereby fulfilling the main aim of developing a holistic solution to the manufacturing systems configuration (MSC) problem.

1.3 Thesis outline

The rest of the thesis focused on addressing the MSC problem holistically and organised as in the below structure:

Chapter 1 - Introduction: This chapter sets the stage by offering a comprehensive background and motivation for the research. It aims to address the MSC problem holistically, contrasting with the fragmented approaches in existing literature. The chapter outlines a holistic approach combining data models, manufacturing software, optimisation, and machine learning algorithms to adapt to changing manufacturing requirements.

Chapter 2 - Literature review: This chapter thoroughly reviews existing literature on the MSC problem, covering manufacturing paradigms, reconfigurability, and enabling technologies. It identifies three critical knowledge gaps and sets the foundation for the research by discussing these gaps and potential strategies to address them.

Chapter 3 - Research methodology: This chapter defines the research problem and its associated challenges, introducing relevant terminology and mathematical notations for precise problem formulation. It outlines the research questions and hypotheses and proposes a validation methodology that combines qualitative and quantitative analysis. The chapter aims to develop a research methodology for developing a holistic approach that enhances the responsiveness and resilience of manufacturing systems.

Chapter 4 - Data model for MSC problem: This chapter introduces an object-oriented data model tailored for the MSC problem. It defines key classes and associations that capture the complexities of modern manufacturing systems. The proposed object-oriented data model is a modular and flexible foundation for addressing the MSC problem.

Chapter 5 - Manufacturing apps for MSC problem: This chapter tackles the issue of software interoperability in manufacturing by introducing the concept of "manufacturing apps." It establishes a theoretical and practical foundation for developing modular software solutions and proposes a methodology for developing a manufacturing app development kit. The chapter contributes to creating an adaptable and flexible manufacturing environment which enables plug-and-produce functionality.

Chapter 6 - Optimisation and decision-making algorithms for MSC problem: This chapter focuses on optimisation and decision-making modules building on top of the proposed data model. It introduces three modules: changes identification, optimisation, and decision-making modules. These modules employ various algorithms and methods to address uncertainty and multiple objectives in selecting optimal manufacturing configurations.

Chapter 7 - Validation of the research contributions: This chapter validates the research contributions through experiments in two distinct manufacturing processes. It sequentially assesses the data model's adaptability, the manufacturing apps' effectiveness, and the ability of optimisation and decision-making algorithms to deal with given requirements. The chapter confirms the research hypotheses and presents a holistic solution to modern manufacturing challenges.

Chapter 8 - Industrial validations: This chapter extends the validation to real-world industrial settings, focusing on aerospace and custom product manufacturing. It demonstrates the generalisability and adaptability of the proposed methodology, confirming its practical relevance and credibility.

Chapter 9 - Conclusions: This chapter synthesises the research contributions and highlights the remaining knowledge gaps. It critically evaluates what was achieved and what is remaining. It outlines future work that can further advance the field, thereby wrapping up the thesis in a comprehensive manner.

Chapter 2

Literature review

2.1 Introduction

The manufacturing industry is undergoing rapid transformations due to changing market dynamics, technological advancements, and evolving customer demands. In this dynamic environment, manufacturing systems must adapt quickly to new products, production requirements, and supply chain configurations. This literature review chapter reviews academic works on making manufacturing systems adaptable to rapidly changing requirements.

The chapter begins by exploring different manufacturing systems paradigms and their characteristics, comparing dedicated, flexible and reconfigurable manufacturing systems.

It then dives into the concept of reconfigurability and existing research on the MSC problem. Enabling technologies like data models, interoperability approaches, and optimisation algorithms are also reviewed in detail.

This chapter identifies critical knowledge gaps in current research by thoroughly analysing the literature. Primarily, the limitations stem from insufficient representation of system complexity, lack of software integration and interoperability, and underutilisation of advanced algorithms like reinforcement learning based methods. These gaps motivate the need for a holistic approach to the MSC problem.

The literature review lays a strong foundation for the overall research by highlighting important concepts, analysing existing methodologies, and revealing avenues for improvement.

The identified knowledge gaps shape the direction for developing a robust framework to address the MSC problem holistically.

2.2 Manufacturing systems paradigms

It is important to understand the types of manufacturing systems and their suitability for adapting to rapid changes. Hence, this section explores different manufacturing systems paradigms and their characteristics.

Manufacturing systems have evolved significantly due to changing market demands and technological advancements. Historically, these systems transitioned from dedicated to flexible and, most recently, to reconfigurable models, each shift representing the manufacturing industry's response to specific challenges and opportunities.

Though numerous manufacturing systems paradigms exist, such as holonic [17–19], evolvable [20–22], fractal [23–25], bionic [26–28], cellular [29–31], and matrix-structured [32–34] manufacturing systems, among others, the three most widely recognised and foundational paradigms are dedicated, flexible, and reconfigurable manufacturing systems.

2.2.1 Dedicated manufacturing systems

Dedicated manufacturing systems (DMS), the traditional approach to production, are designed for high volume manufacturing with low product variety [35, 36]. These systems are particularly suitable for mass production, where there is a need to minimise unit cost and the product complexity is low. DMS's importance becomes evident in industries like automobile and electronic component manufacturing, where they provide significant efficiency in an economy of scale due to their highly specialised design and operation [35].

The rapid transformation of manufacturing has identified limitations of DMS, with the increasing focus on mass customisation, product variety, and frequent design changes [36]. While DMS are efficient in industries where the product life cycle

is long and stable, and demand is predictable, the inflexibility of these systems can lead to costly and time-consuming processes when there is a need to redesign or retool for new products and requirements. Their main drawback lies in their rigidity and limited adaptability to changes in product design or market demand [37].

These limitations of DMS, coupled with the changing manufacturing landscape, necessitated a shift towards more flexible systems. Therefore, the next section discusses the emergence of flexible manufacturing systems and their added adaptability in manufacturing processes.

2.2.2 Flexible manufacturing systems

Flexible manufacturing systems (FMS) symbolise an industry adaptation to the increasing demand for product variety and customisation, enabling the production of different types of products without significant downtime for changeovers. These systems retain the efficiency of DMS while providing enhanced flexibility to cater to design changes and variety. The infrastructure typically comprises several automated machine tools connected by an automatic material handling system, overseen by a central computer control system [38, 39].

According to Koren [36], the inherent versatility of FMS offers a balanced approach to productivity and flexibility, which suits industries with medium-volume production and frequent design changes. The reduced turnover time from product design to production marks a leap in manufacturing agility, offering a swift response to market changes [39, 40].

However, as noted by Zhang et al. [41], while FMS offer considerable advantages, their adoption comes with challenges. For instance, the initial cost of implementation can be high. Despite the increased flexibility over dedicated systems, they are still constrained by the hardware specifications of the installed machines, po-

tentially inhibiting rapid adaptation to extensive design changes.

Therefore, the next section explores reconfigurable manufacturing systems building upon the flexibility of FMS. They are considered the next generation of manufacturing systems that enhance flexibility with rapid reconfigurability.

2.2.3 Reconfigurable manufacturing systems

Reconfigurable manufacturing systems (RMS) stand at the forefront of modern manufacturing paradigms, characterised by high flexibility, speed, scalability, and rapid reconfiguration. These systems can adapt to shifts in market dynamics, new product introductions, and fluctuating product requirements [42–44].

RMS distinguish themselves from FMS by their inherent capacity to swiftly adjust production capability and capacity. This unique attribute amplifies the competitiveness of manufacturing firms and significantly reduces their reaction time to market changes [45, 46].

Stemming from these unique capabilities, RMS are fundamentally designed to facilitate modifications to their structure, hardware, and software components. This property of RMS allows them to maintain increased responsiveness to unforeseen shifts in product demand, thereby aligning the system with the evolving market conditions and customer requirements [47–50]. In essence, RMS harness the strengths of both DMS and FMS, designed to adjust production capability and capacity precisely when needed [47, 51, 52].

As detailed by Koren et al. [42], RMS are underpinned by the following core characteristics:

- **Scalability (design for capacity changes)**: The ability to modify production capacity by adding or removing resources or altering system com-

ponents.

- **Convertibility (design for functionality changes)**: The potential to transform the functionality of existing systems and machines to meet new production requirements.
- **Diagnosability (design for easy diagnostics)**: The capability for real-time monitoring of product quality and rapid root-cause analysis of product defects.
- **Customisation (flexibility limited to part family)**: The system or machine flexibility designed around a part family, yielding customised flexibility within the part family.
- **Modularity (modular components)**: The division of operational functions into independent units or modules that can be rearranged between alternative production schemes.
- **Integrability (interfaces for rapid integration)**: The capability for swift and precise integration of modules via hardware and software interfaces.

The above core characteristics make RMS well equipped to adapt to rapidly changing manufacturing requirements. The concept of reconfigurability, at its core, enables a customised flexibility on-demand within a condensed time frame, empowering companies to adapt to a constantly evolving marketplace [35].

However, despite RMS being theoretically suited to adapting to changing manufacturing requirements, the practical implementation of the core characteristics is still challenging. Therefore, the next section transitions to the literature review on selecting the optimal manufacturing configurations based on the characteristics of RMS, emphasising the concept, measurement, and implementation challenges.

2.3 Manufacturing systems configuration (MSC) problem

The manufacturing systems configuration (MSC) problem requires determining the best manufacturing configuration and resource allocation to optimise performance indicators such as production capacity and capability, system availability, and operational efficiency while minimising costs. Effectively solving this problem is important for customised flexibility and responsiveness to market changes. However, to tackle this problem, it is necessary first to comprehend the concept of reconfigurability and understand how to measure its degree and effectiveness.

2.3.1 Concept of reconfigurability

The concept of reconfigurability in manufacturing has been widely studied in the literature, and RMS and RMS-based manufacturing systems have been recognised as the future of manufacturing for adapting to rapidly changing manufacturing requirements [53, 54]. Therefore, understanding the concept of reconfigurability is important for addressing the MSC problem.

Reconfigurability refers to the ability of the system to change its capability and capacity cost-effectively and rapidly to meet changing market conditions [48]. The ability to reconfigure the system elements over time allows for producing a diverse set of individualised products in small quantities and with short delivery lead times [55]. Furthermore, two main aspects of reconfigurability are discussed in the literature: physical and logical.

Physical reconfigurability involves changing the manufacturing system's physical components, such as the layout, machines, and material handling devices [56]. It allows for modifying the system's physical structure to adapt to new manufac-

turing requirements [57]. Examples of physical reconfiguration activities include adding or removing modules and machines from the production process to achieve new functionalities [57], changing the layout of the system [56], and modifying material handling devices [56]. Physical reconfigurability enables the system to be adjusted to accommodate different product families and production volumes [58].

On the other hand, logical reconfigurability involves changing the system's software, control and information flow [59]. It allows modifying the system's logical structure to adapt to new manufacturing requirements [60]. Examples of logical reconfiguration activities include reprogramming machines to perform different tasks [59], re-planning the production schedule [59], and re-routing the flow of materials [59]. Logical reconfigurability enables the system to adapt to changes in product mix, production volume, and production sequence.

Both physical and logical reconfigurability are essential for achieving the flexibility and adaptability required in modern manufacturing systems. By combining both aspects of reconfigurability, manufacturing systems can effectively respond to changing market conditions, achieve cost-effective production, and provide customised flexibility on demand.

After understanding the concept of reconfigurability, the next step in solving the MSC problem is measuring reconfigurability, as it provides a quantifiable and objective basis for making decisions and implementing optimisation algorithms.

Quantifying reconfigurability helps assess a manufacturing system's capability to adapt to changes and compare different manufacturing systems for their adaptability. This quantification forms the basis for deploying optimisation algorithms to find the best manufacturing configurations under given conditions. Therefore, the next section delves into measuring reconfigurability, a necessary step towards selecting optimal manufacturing configurations for adapting to new manufacturing requirements.

2.3.2 Measuring reconfigurability

Research on the MSC problem has stimulated the development of multiple configuration and reconfiguration selection methodologies using different reconfigurability metrics. These methodologies aim to define the costs and objectives required for assessing reconfigurability.

Youssef and ElMaraghy [61] presented a metric for evaluating reconfiguration smoothness across market, system, and machine levels. However, this method confronts challenges related to the association of costs and specific data requirements. On the other hand, Goyal et al. [62] developed metrics to assess machine reconfigurability and operational capability but faced similar limitations.

Benderbal et al. [63] offered a novel approach using the non-dominated sorting genetic algorithm 2 (NSGA-II)-guided design methodology and robustness index to improve adaptability of manufacturing systems. However, this methodology focuses too narrowly on single-machine unavailability, neglecting other essential aspects of the system.

Efforts towards measuring reconfigurability were also pursued by Huang et al. [64] and Prasad and Jayswal [65], who suggested a similarity coefficient and a layout based on RMS principles, respectively. Despite their contributions, these methodologies have limitations as they predominantly rely on heuristics, hindering the development of fully optimised solutions.

Ameer and Dahane [66] brought forth a process-level metric called reconfiguration index for measuring the reconfigurability of manufacturing systems. Although innovative, this approach did not fully address the diverse types of effort required in the reconfiguration process.

This review of reconfigurability metrics illustrates the multifaceted nature of defining a cost-optimal manufacturing configuration selection problem. The choice of

metrics significantly influences the performance and suitability of optimisation algorithms. Therefore, understanding these nuances is essential for selecting appropriate metrics and algorithms.

After reviewing the metrics for measuring and quantifying reconfigurability, the next section focuses on the current literature addressing the MSC problem, analysing the strengths and weaknesses of the existing approaches. The ultimate goal is to understand the factors that limit addressing the MSC problem and implementing the solutions practically.

2.3.3 Current approaches to the MSC problem

The MSC is a complex problem in making manufacturing systems adaptable to rapidly changing manufacturing requirements. Hence, many researchers looked at this problem from different angles.

Huang et al. [67] presents a design framework for optimising supply chain systems by incorporating the generic bills of materials concept to model supply chain structure. The framework applies genetic algorithms (GA) to solve the proposed mathematical model. Limitations include the single focus on total supply chain cost, the assumption of unlimited capacity at each stage and the lack of consideration for uncertainties or non-stationary demand.

Youssef and ElMaraghy [53] introduces optimal configurations selection approach, combining constraint satisfaction procedures with meta-heuristics to optimise capital cost and system availability. However, the methodology does not consider material handling systems and layout design, consults only two performance criteria and does not account for unexpected future demand changes that could affect configuration requirements.

Dou et al. [68] proposes a GA-based approach for optimising multi-part flow-line configurations. However, it focuses only on minimising capital cost, neglecting other key performance indicators, and unrealistically assumes 100% machine availability, allowing unrestricted operations assignment to any stage.

Goyal et al. [69] presents an approach that measures a reconfigurable machine tool's reconfigurability and operational capability, considering these measures and costs for the optimal machine assignment. It assigns fixed weights to the performance measures, neglects the impact of machine interactions, and assumes all machines are fully reliable and available.

Bensmaine et al. [70] addresses optimal manufacturing configurations selection problem by selecting machines optimised for product specifications and reconfigurable machine capabilities using an NSGA-II algorithm. However, the methodology only considers a single product and does not consider machine reliability or additional costs associated with system reconfiguration.

Hasan et al. [71] discusses an optimal configuration sequence of multiple part families in the RMS considering the total benefit. The limitation of this work includes no consideration for sequence-dependent setup times, machine reliability, or key RMS performance indicators beyond economic factors.

Mittal et al. [72] proposes optimal configuration selection based on minimum loss for various part families. However, the model fails to consider the stochastic nature of orders arrival, assumes reconfiguration cost is directly proportional to reconfiguration effort, and assumes a fixed sequence of part families.

Koren et al. [51] formulates design and operational principles for RMS, highlighting the challenges in maintaining product precision in RMS. Nevertheless, it lacks a systematic design framework, does not specify the RMS type considered and neglects the influence of recent intelligent manufacturing technologies and environmental impact considerations.

Moghaddam et al. [73] proposes a two-phased method for configurations design and reconfigurations adapting to demand rate changes. The work, however, falls short as it considers a single-part family for production in the RMS, overlooks operating and maintenance costs and does not consider system design constraints such as maximum space or cost.

Ashraf and Hasan [74] introduces a simulation-based multi-objective optimisation approach for configuring multi-part flow lines. The study does not, however, consider cost, quality, sustainability, and other crucial factors, neglects the dynamics in ramp-up and ramp-down of production volumes, and provides limited model validation.

Diaz et al. [75] introduces a simulation-based multi-objective optimisation approach for optimal configurations selection. It, however, only considers throughput and total buffer capacity as optimisation objectives, neglects the dynamics in ramp-up and ramp-down, and lacks comprehensive model validation and inclusion of more sources of uncertainty.

Although the existing body of research on the MSC provides considerable insights into the problem, it also presents numerous limitations that must be addressed in future works for them to be implementable in a holistic and integrated way.

A significant limitation lies in the insufficient representation of the complexity and variability in manufacturing systems, leading to oversimplification of the assumptions about manufacturing systems. This problem can be addressed using data models for modelling manufacturing systems. Without data models, it is challenging to accurately represent the interaction of different system elements and their impact on system performance.

Practical implementation of the proposed methodologies is also limited due to issues related to software integration and interoperability. This lack of integration may prevent the effective operation and control of various system components and

hamper the ability to respond to system or operating environment changes.

Finally, current works predominantly use traditional optimisation methods and do not address optimisation and decisions over time. Although traditional algorithms such as genetic algorithms (GA) and mixed-integer linear programming (MILP) are the baseline choice for multi-objective optimisation in manufacturing, they cannot efficiently handle uncertainties and struggle when sequential optimisation and decision-making are necessary. Advanced methods such as machine learning or reinforcement learning (RL), which can model and predict system behaviour under uncertainty and sequentially, are largely absent in the existing works on the MSC problem. Therefore, there is a need for developing robust algorithms in combination with classical optimisation and machine learning algorithms.

The limitations in the existing literature motivate to explore the key enabling technologies such as data modelling techniques, software and interoperability, optimisation and machine learning algorithms. Therefore, the next section reviews these key enabling technologies focusing on manufacturing.

2.4 Review of enabling technologies

The existing literature on the MSC problem showed that the key enabling technologies such as data models, software and interoperability approaches, and advanced machine learning algorithms are absent in the current solution considerations. However, these technologies have been extensively used in other parts of manufacturing. Therefore, the goal of this section is to explore the usage of these key technologies in manufacturing in general and to understand how these technologies can be applied in addressing the MSC problem. The integration of these key enabling technologies is important to develop a framework that addresses the problem in a holistic way.

2.4.1 Data modelling in manufacturing

Data modelling plays an important role in the design and implementation of manufacturing systems. It aids in visualising data flow, reducing redundancy, and improving data consistency, thereby making the data management process more efficient.

Data modelling acts as a medium for understanding business needs, and through this understanding, improvements in manufacturing processes can be made, including streamlining production, improving accuracy and reducing costs. It also provides a structural representation of data that assists in defining business workflows, ensuring seamless communication among different functional units and setting the stage for continuous process improvement.

Data modelling techniques can be divided into relational, entity-relationship, hierarchical, network, object-oriented, and semantic data modelling techniques.

2.4.1.1 Relational data modelling

Relational data modelling is a method where data is organised in a series of tables or relations. These tables are interlinked based on shared attributes or keys, allowing for high data manipulation and retrieval flexibility. It provides a clear and intuitive structure for representing complex data relationships. It is particularly useful in manufacturing contexts where multiple entities must be cross-referenced, such as inventory management, supply chain management, and production scheduling.

Relational data modelling techniques have effectively harmonised manufacturing data from various sources, aiding production decisions and concurrent data access across devices.

Drstvensek et al. [76] employed a database management system (DBMS), cou-

pled with genetic algorithms (GA), to store and analyse historical data to inform production decisions.

Corallo et al. [77] leveraged PostgreSQL to develop a data integration system that aggregates data from disparate platforms, enabling efficient data handling and concurrent data access across manufacturing equipment.

2.4.1.2 Entity-relationship data modelling

Entity-relationship (E-R) data modelling is a technique for system design that represents the entities involved in a system and the relationships between them. This approach is instrumental in identifying and structuring the various components of a manufacturing system, such as resources, operations, and dependencies. It provides a strong foundation for designing databases and helps ensure data is accurately represented and properly linked.

The E-R data modelling technique has proved effective in providing unique representations of manufacturing flexibilities and creating adaptable and modular manufacturing execution system (MES) software.

Zhou et al. [6] applied the extended E-R model to develop a generic and adaptable data model for MES, leveraging the model's ability to represent complex relationships among entities, which served as a fundamental framework for creating adaptable and modular MES software.

Chowdary et al. [78] utilised the E-R model to capture manufacturing flexibilities, enabling the unique representation and evaluation of flexibility options, with an algorithm to derive flexibility paths from these models, which demonstrated effectiveness in dealing with decision context changes and operational risks.

2.4.1.3 Hierarchical data modelling

Hierarchical data modelling represents data in a tree-like structure, with each piece having a single parent and zero or more children. This top-down approach benefits manufacturing contexts with clear hierarchical relationships, such as assembly line organisation, material ordering, or production sequence planning. It allows for efficient data retrieval along predefined paths.

The broad utility of hierarchical data modelling in manufacturing is seen in its applications for mapping energy and material flows, assessing the geometric accuracy of parts, and structuring manufacturing processes to optimise machine utilisation.

Alvandi et al. [79] used this technique to construct a simulation-based model that maps energy and material flows within manufacturing systems. This hierarchical model was key in identifying efficiency hotspots and evaluating the effectiveness of retrofits through simulated scenarios.

Yang et al. [80] developed a hybrid hierarchical modelling approach to assess the geometric accuracy of parts in additive manufacturing. The hierarchical model allowed for capturing variability at the part-to-part and feature levels, facilitating manufacturing scalability.

Gaiardelli et al. [81] used hierarchical modelling to structure manufacturing processes into tasks and sequences of actions, enabling a runtime scheduling algorithm to optimise machine utilisation and minimise makespan.

2.4.1.4 Network data modelling

Network data modelling is a flexible approach that allows for many-to-many relationships between entities. Unlike the hierarchical model, each entity in a network model can have multiple parent and child entities. Network data modelling is ben-

eficial in complex manufacturing systems with multiple interdependencies, such as in multi-stage production processes or when handling intertwined supply chains.

Network data modelling, given its flexibility, has been employed to tackle intricate scheduling problems and to study the correlation between system structure and performance in large-scale manufacturing.

Lin et al. [82] used this modelling technique to represent intricate scheduling problems in manufacturing, combining it with evolutionary algorithms (EA) for auto-scheduling. It was deemed efficient in capturing the complexity and interdependencies inherent in scheduling.

Becker et al. [83] employed a network model backed by complex network theory to study the link between system structure and performance in large-scale manufacturing. The model provided a universal, low-effort approach to assess systems, revealing unique, non-random network characteristics and a non-linear correlation between topological and performance measures.

2.4.1.5 Object-oriented data modelling

Object-oriented data modelling treats data and the operations that can be performed on it as a single unit called an "object". Each object in the system has a type, and different types can inherit characteristics from other types. Object-oriented data modelling is helpful in representing complex relationships in manufacturing systems, where the same types of operations may apply to different parts of the system, such as various types of machinery or different stages of a production process.

The object-oriented data modelling technique has found extensive use in enhancing plan generation, improving software development efficiency, capturing complex equipment interaction, and increasing interoperability in manufacturing systems.

Sormaz and Khoshnevis [4] used it to represent knowledge in computer aided process planning, reducing search space and enhancing plan generation. Zhang et al. [84] improved computer aided process planning's adaptability by using object-oriented manufacturing resources modelling, which encapsulates system knowledge.

Chengying et al. [85] used the object-oriented method for developing a generalised manufacturing resource model to facilitate information sharing throughout the product development process. Law and Woo [86] applied an object-oriented approach to manufacturing information systems, increasing data abstraction and reusability. Ehm et al. [87] proposed an object-oriented model to capture complex supply chains in high-tech manufacturing.

Pullan et al. [88] and Hedman et al. [89] used it for collaborative design and manufacturing and human resource modelling to increase interoperability and provide comprehensive representation.

Young et al. [90] created a model for manufacturing control systems using unified modelling language (UML) to adapt to changes swiftly. Similarly, Anglani et al. [91] used UML for creating flexible manufacturing system simulation models, improving software development efficiency.

2.4.1.6 Semantic data modelling

Semantic data modelling focuses on the meaning of data within a specific context. This approach is particularly useful in manufacturing systems that need to capture intricate business rules or complicated relationships between entities. It offers a high level of abstraction and is advantageous when a clear understanding and representation of business concepts and their relationships are required, like in regulatory compliance or quality assurance.

Semantic data modelling techniques are effectively used to automate decision-making processes, facilitate knowledge integration, promote information exchange and interoperability, and optimise unused industrial capacity within the manufacturing ecosystem.

Garcia-Crespo et al. [92] leveraged ontology-based implementation to automate industrial manufacturing decision-making. Similarly, Negri et al. [93] examined semantic languages' application in manufacturing systems control through the semantic annotation of web service-based architectures.

Zhang et al. [94] employed a semantic-predictive model supported by ontology representations to facilitate knowledge integration and reuse. Hildebrandt et al. [95] developed semantic metamodels to promote information exchange and interoperability in Industrie 4.0. Wan et al. [96] introduced an ontology-based resource reconfiguration method to manage resources in complex manufacturing systems efficiently.

Landolfi et al. [97] designed a semantic data model to optimise unused industrial capacity by mapping complex relationships and flows within the manufacturing ecosystem. Köcher et al. [98] detailed a formal ontology-based model to represent machine capabilities, enabling adaptability and integration in response to environmental pressures.

The literature review on data models shows that they play a vital role in manufacturing. Moreover, they act as a backbone for developing interoperable and plug-and-produce solutions and are necessary for software solutions. Therefore, the next section is about how interoperable manufacturing solutions are developed on top of data models.

2.4.2 Interoperable manufacturing solutions

The current body of the literature on the MSC problem lacks research on integrating developed MSC algorithms with manufacturing systems. This problem is mainly due to the focus of the existing literature solely on the optimisation part and the omission of the integration part. However, integration has challenges, such as dealing with different manufacturing equipment, which necessitates interoperable manufacturing solutions. Therefore, this section aims to understand how interoperability, plug-and-produce functionality, and software development methodologies are addressed in manufacturing in general.

2.4.2.1 Interoperability

Interoperability in manufacturing refers to the ability of different systems, processes, and stakeholders to exchange and use information, data, and resources seamlessly. It plays a crucial role in ensuring a smooth flow of information and coordination between various stages of the manufacturing process, from design to production.

One of the key challenges in achieving interoperability in manufacturing is the lack of integration between different information systems and equipment from original equipment manufacturers (OEMs).

Elheni-Daldoul et al. [99] highlight the lack of interoperability between product management (PDM) systems, manufacturing process management (MPM), and enterprise resource planning (ERP) systems as a major obstacle to ensuring a continuous and bidirectional flow of information from design to manufacturing and assembly.

Ahmed and Han [100] emphasize efficient communication and exchange of product and manufacturing information between computer-aided design (CAD) and

computer-aided manufacturing and analysis systems to avoid information loss and errors in design and fabrication.

Another challenge is the heterogeneity of data formats and standards used in different systems.

Sanchez-Londono et al. [101] point out that the lack of common standards for device interoperability is a major challenge in smart manufacturing. This argument is supported by Nassehi et al. [102], who state that interoperability is achieved when manufacturing process plans can be executed by various resources with minimal human involvement. The use of neutral formats such as standard for the exchange of product data (STEP) and initial graphics exchange specifications (IGES) are proposed as a solution to facilitate integration between different systems by Ahmed and Han [100].

Researchers have proposed various approaches and solutions to address the interoperability challenges in manufacturing. One approach is using data models and ontologies for knowledge representation and management.

Wicaksono et al. [103] discuss the application of ontologies in energy management systems in manufacturing, which helps address interoperability issues among different stakeholders and entities. The ontologies and data models provide a formal knowledge representation that enables a common understanding between stakeholders with different background knowledge.

Another approach is integrating different software tools and systems through interoperable interfaces. Gürdür and Gradin [104] discuss integrating software tools in cyber-physical system (CPS) manufacturing, emphasizing the importance of including sustainability aspects in the interoperability strategies. They propose an approach that integrates sustainability metrics into interoperable toolchains to improve sustainability in CPS manufacturing.

The use of standards and reference architectures is also highlighted as a solution to achieve interoperability in manufacturing.

Georgios et al. [105] mention the importance of open standards, open source software, and multilateral solutions in ensuring interoperability in the context of Industry 4.0. They emphasize the need for common standards to ensure high accuracy and efficiency in manufacturing processes.

Similarly, Da Rocha et al. [106] discuss the syntactic interoperability between the IEEE 1451 and IEC 61499 standards in an industrial environment, which enables information exchange between smart transducers and supports Industry 4.0 requirements.

2.4.2.2 Plug-and-produce concept

Another aspect currently overlooked in the current research related to optimal manufacturing configurations selection is the "plug-and-produce" concept. The plug-and-produce concept, originally put forth by Arai et al. [107], offers a promising avenue for enhancing the adaptability and scalability of manufacturing systems.

Plug-and-produce is a key element in manufacturing systems to integrate new devices and configure machines with minimum reconfiguration effort [108]. It aims to enhance the interoperability and reusability of modules, reducing integration times and enabling rapid system configuration and reconfiguration [108].

The plug-and-produce paradigm represents the idea of a quick and seamless connection of production equipment with minimal or no setup needed [109]. It is based on the idea of simply integrating new devices into the manufacturing ecosystem, including simplified data exchange mechanisms [110]. The concept involves three main stages: physical attachment of a device to the ecosystem, communication

establishment to the device or the representative entity, and device integration on a logical level [110].

The plug-and-produce concept is closely related to other paradigms and technologies in manufacturing. It is often associated with modularity, agent technology, and evolvable assembly systems (EAS) [111]. Evolvable assembly systems (EAS) propose an agile solution by considering a plug-and-produce environment, where assembly capabilities are allocated to process requirements [111].

The concept of plug-and-produce is also aligned with the key features found in RMS, such as modularity, integrability, customisation, convertibility, and diagonalisability [112]. The agility introduced in the adaptability and reconfigurability of the software and hardware layer is crucial to fulfilling the requirements of modern manufacturing systems [112].

Multi-agent technology is often utilised to implement the plug-and-produce concept. Multi-agent systems provide autonomy, openness, and communication features, extending the plug-and-produce concept to deal with workflow changes and automatic task assignment [108]. Using multi-agent technology in plug-and-produce systems allows for the integration of resources using standardised hardware connectors and automatic inclusion in the manufacturing process [113].

The plug-and-produce concept is also closely related to the broader context of Industry 4.0 and the need for flexible production facilities. In the era of Industry 4.0, manufacturing systems must be equipped with adaptable fabrication facilities and flexible production patterns to acquire plug-and-produce capabilities for future production lines [114]. The concept of plug-and-produce is frequently used in Industry 4.0 to describe the need for flexible production facilities where devices can be added quickly and configured automatically [115].

2.4.2.3 Software development methodologies

In the pursuit of enhanced interoperability and a more seamless plug-and-produce environment, there has been a shift in software development methodologies within the manufacturing sector.

The conventional monolithic architecture, characterised by rigidity and lack of adaptability, has been increasingly scrutinised for its inefficiencies, necessitating a transition towards more flexible systems [116, 117].

Embracing a more modular approach, such as microservices architecture, is poised to bring higher adaptability and efficiency in manufacturing software solutions.

The rise of microservices and containerisation techniques signifies an important step in the manufacturing software landscape, providing crucial enhancements in flexibility, interoperability, independence, and scalability [118, 119].

Microservices architecture, acclaimed for its potential to streamline the complexities often associated with monolithic systems, paves the way to better flexibility and some level of interoperability [118]. A number of studies, including those by Ibarra-Junquera et al. [120], Goldschmidt et al. [121], González-Nalda et al. [9], and Thramboulidis et al. [122], have employed this architecture with promising outcomes. However, these studies primarily address internal system interoperability, focusing less on interoperability across diverse communication protocols. This underscores the need for a more comprehensive and robust approach to integrating various modules within manufacturing systems.

Containerisation, another critical technology in manufacturing software development, offers benefits in terms of compatibility and scalability [123]. Despite its broad adoption, as shown by studies from Senington et al. [124], Nikolakis et al. [125], and Rufino et al. [10], containerisation, like microservices, still faces challenges in managing diverse technologies and ensuring complete interoperability.

This implies that there is still room for improvement in leveraging containerisation technologies to simplify manufacturing processes and control structures.

Recent trends indicate a shift towards an app-based strategy in manufacturing software development, borrowing from successes in the personal device industry. This shift promises benefits such as centralised data storage and simpler user setup [126]. However, works by Chen et al. [127], Singh et al. [128], Gröger et al. [126], and Goerzig et al. [7], while informative, have not sufficiently delved into the development of manufacturing apps that empower users to control manufacturing equipment, thereby limiting their potential active usage.

After reviewing enablers for interoperability for the MSC problem, such as plug-and-produce and interoperable software development methodologies, one final piece that remains to review is the review of traditional optimisation algorithms and machine learning algorithms for the optimal manufacturing configurations selection problem.

2.4.3 Optimisation and machine learning algorithms

Optimisation algorithms for ensuring optimality of the chosen solutions are the main elements in solving the MSC problem. Therefore, researchers have proposed several methods for optimisation. However, the most commonly used optimisation algorithms in the MSC research directions are mixed-integer linear programming (MILP) and genetic algorithms (GA).

Machine learning algorithms that can deal with uncertainties and generate optimal policies, such as RL algorithms, have not been directly utilised for the MSC problem. However, RL approaches have been extensively used in other parts of manufacturing optimisation.

The goal of this section is to explore and understand how the above optimisa-

tion and machine learning algorithms can be combined and utilised for the MSC problem.

2.4.3.1 Mixed-integer linear programming

Mixed-integer linear programming (MILP) is a branch of optimisation that deals with linear equations and inequalities subject to discrete or integer constraints. This approach can model various industrial scenarios in manufacturing with mathematical precision, enabling highly detailed decision-making. For instance, MILP can represent decisions regarding the number of machines, their configurations, and the assignment of parts to machines [129], allowing it to formulate complex and versatile solutions.

Numerous studies have proposed using MILP to optimise RMS, highlighting the method's capability for handling dynamic and complex scenarios.

Hees et al. [130] developed a novel production planning method using MILP to determine feasible configurations for RMS in dynamic environments. The approach supported capacity scalability and functionality changes, contributing significantly to optimising production planning processes.

Similarly, Liu et al. [131] addressed the multi-module RMS optimisation problem for multi-product manufacturing using MILP model. The research notably aimed at minimising total cost and cycle time, combining an ε-constraint method and a multi-objective simulated annealing algorithm for small and large problem sizes.

More advanced models were proposed by Wikarek et al. [132] and Dou et al. [129], integrating constraints and multiple decision factors for better decision support in the configuration and reconfiguration of manufacturing systems. The flexibility of these models is particularly significant, allowing for the concurrent optimisation of configuration design and scheduling in RMS.

Moghaddam et al. [52] developed two different approaches for configuration design using MILP and integer linear programming formulations, showcasing the utility of this method in the context of scalable RMS.

2.4.3.2 Genetic Algorithms

Genetic algorithms (GA) are a class of adaptive heuristic search algorithms based on natural selection and genetics principles. GA are used in manufacturing due to their inherent capacity to handle multi-objective problems. The non-dominated sorting genetic algorithm and its variants, as used by Goyal et al. [69] and Khettabi et al. [133], are particularly useful for their ability to generate a set of optimal solutions (a Pareto front), providing manufacturers with an assortment of trade-off options to choose from. For instance, Goyal et al. [69] proposed using NSGA-II and a multiple attribute decision-making approach for optimal machine assignment based on module interactions and machine capability.

Subsequent studies expanded upon this approach to consider multiple objectives. Khettabi et al. [133] tackled the problem of environmentally conscious multi-objective process planning in RMS, employing modified versions of the non-dominated sorting genetic algorithm and a decision-making technique called TOPSIS.

Houimli et al. [134] introduced a hybrid approach combining GA and high-level Petri nets to select configurations in RMS optimally, prioritising resource optimisation and preservation of the best properties.

The viability of GA was further demonstrated by Chaube et al. [135] and Bensmaine et al. [136], who used adapted versions of NSGA-II for dynamic process planning and process plan generation, respectively, aiming to reduce manufacturing cost and time in a reconfigurable manufacturing context.

2.4.3.3 Reinforcement Learning (RL)

Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by taking actions in an environment to achieve a goal. Deep RL, an extension of RL, utilises deep learning architectures for decision-making. RL can handle real-time dynamic situations in manufacturing, making it particularly useful for dynamically reconfigurable flow shops and smart manufacturing systems [137, 138].

Recent studies have begun exploring the application of reinforcement learning for optimisation problems in manufacturing systems. Tang and Salonitis [139] proposed the use of deep RL to make autonomous decisions in RMS to complete assigned order lists while minimising reconfiguration actions.

Epureanu et al. [138] and Yang et al. [137] furthered this trend, investigating how deep RL and its variants, like the deep Q-network (DQN) and its improved version expected DQN (EDQN), can facilitate decision-making in smart manufacturing systems, and optimise scheduling and reconfiguration in real-time respectively.

Viharos and Jakab [140] introduced RL for statistical process control in production, utilising the Q-table method and several novel RL extensions to optimise production cost while maintaining high-quality outputs.

Zimmerling et al. [141] and Li et al. [142] explored the use of RL for estimating optimal process parameters in variable component geometries and for assembly line design, respectively, demonstrating the technique's adaptability across different application domains.

The optimisation of manufacturing configurations continues to be an active field of research. While considerable advances have been made using MILP, GA, and RL, the dynamic nature of reconfigurable manufacturing systems and the increasing complexity of production environments underline the need for ongoing exploration

of these and potentially other computational techniques.

In the context of the MSC problem, it is important to realise that a single algorithm is not a 'one size fits all' solution due to the multifaceted nature of manufacturing environments. Every manufacturing situation has different constraints, multiple objectives, varying degrees of complexity, and uncertainties.

MILP has proven capable of formulating complex solutions and can precisely handle discrete or integer constraints but might not optimally manage dynamic, non-linear scenarios or uncertainties. GA, though adept at multi-objective problems, may suffer from premature convergence and might not always guarantee the global optimum solution. RL, despite its potential to handle real-time dynamic situations, is subject to substantial computational costs and can sometimes struggle with stability and convergence issues.

Given this diversity, the most effective approach to the MSC problem may involve not relying on a single algorithm but combining and coordinating these algorithms depending on the specific requirements of each scenario. One could consider adopting a hybrid approach, where the strength of one algorithm compensates for the weaknesses of another, or a multi-algorithm ensemble approach, where different algorithms are applied in parallel and their outputs are aggregated.

For instance, GA can be used to explore the solution space broadly and quickly, followed by MILP to fine-tune solutions within the promising regions identified by GA precisely. RL can handle dynamic changes and uncertainties in the production environment. Ultimately, the combination should be adaptive, dynamic, and flexible to suit the ever-evolving needs of the manufacturing configurations problem. This integrated approach could potentially lead to robust, efficient, and optimal decision-making in diverse manufacturing environments.

2.5 Knowledge gaps

A thorough exploration of existing literature on the MSC problem reveals several key knowledge gaps:

1. **The first knowledge gap** lies in the underutilisation of data models capable of capturing the complexity and variability in manufacturing systems. While data models have been used extensively for different problems in manufacturing, their application in the context of the MSC problem remains largely under-explored. This gap is important to address because data models can accurately represent system dynamics essential for making decisions that optimise system performance. Specifically for the MSC problem, incorporating these data models can help determine the best structural configuration and resource allocation, optimising key performance indicators.

2. **The second knowledge gap** concerns software integration and interoperability. Despite some suggested solutions in the literature, a modular and plug-and-produce solution has yet to be found. This gap becomes increasingly important as manufacturing systems depend more heavily on digital platforms. Achieving seamless software integration is important for the MSC problem as it can significantly improve system efficiency and responsiveness, both key aspects for rapidly adjusting to changes in the manufacturing environment and market.

3. **The third knowledge gap** is associated with the optimisation algorithms and their capability to handle evolving, uncertain, and multiple objectives. The current body of research primarily relies on traditional optimisation methods. While these methods can be effective in certain scenarios, they often fail when addressing uncertainties, sequential optimisation, and dynamically changing objectives. By its nature, the MSC problem requires handling uncertainties and sequential decision-making while simultaneously

adapting to changing, and at times conflicting, objectives. Therefore, it is important to develop methodologies that combine traditional optimisation methods with more robust techniques, such as machine learning and reinforcement learning.

These individual gaps, taken collectively, point to the main knowledge gap — **a lack of a holistic approach to tackle the MSC problem**. Each gap presents its challenge, but when combined, they highlight the importance of bringing together robust data modelling, interoperable software integration, and applying adaptable algorithms to adapt to changing manufacturing requirements.

2.6 Chapter summary

This chapter extensively explored existing literature surrounding the MSC problem. It started by identifying the most relevant manufacturing systems paradigms, providing an essential foundation for the literature review.

The review then transitioned to an in-depth discussion on reconfigurability and analysing existing works focusing on optimal configuration selections. It highlighted both the value and limitations of the existing works, offering valuable insights into the current state of research on the MSC problem.

This understanding paved the way to investigate enabling technologies that could address the limitations identified in the existing works. A significant portion of the chapter was dedicated to an intensive review of these technologies. Specifically, the discussion centred around data models, the interoperability and plug-and-produce concept, software development methodologies, and optimisation algorithms.

By scrutinising the literature in this manner, three critical knowledge gaps were identified in relation to solving the MSC problem, which led to an elaborate dis-

cussion on these gaps, examining the nature and implications of these gaps and the possible strategies to address them.

This chapter, thus, serves as a basis for the entire research work. The concepts and ideas discussed in this chapter are explored in detail in the technical chapters of this dissertation. The next chapter presents a research methodology for the MSC problem building upon the literature review presented in this chapter.

Chapter 3

Research methodology

3.1 Introduction

Three main knowledge gaps for holistically approaching the MSC problem were identified in Chapter 2. These three gaps led to the main knowledge gap: the fragmented approach to dealing with the MSC problem. Therefore, this chapter aims to develop a research methodology to address the main knowledge gap for the MSC problem and have a holistic approach instead. To this end, the comprehensive research and validation methodology is presented in this chapter.

The chapter starts with research problem formulation. It is done by providing terminology, definitions, and mathematical notations. Then, the cross-cutting research challenges are discussed concerning the above three knowledge gaps. The research question and hypotheses are formulated after identifying challenges with each research gap. Formulating research questions and the corresponding research hypotheses helps tackle the problem in a structured way.

After the research problem formulation, a methodology is established to validate the hypotheses. The validation methodology is established in two stages.

Initially, the research contributions provided in the technical chapters are validated in a controlled robotics environment where the manufacturing processes and use cases are designed from scratch. In particular, two representative manufacturing processes are proposed as use cases: sorting industrial products and bin-picking industrial items.

After validating the research contributions in the lab environment, the proposed methodologies are applied to real industrial use cases. In particular, one of the industrial validations is carried out for the aerospace manufacturing sector, and the other industrial validation is carried out for the custom product machining sector.

3.2 Research problem formulation

Exploring existing literature on the MSC problem has unveiled critical knowledge gaps, as detailed in Chapter 2. These gaps are the underutilisation of intricate data models, software integration and interoperability challenges, and the limitations of current optimisation algorithms in handling evolving and uncertain objectives. Moreover, these challenges must be addressed holistically because of the interconnected nature of the manufacturing systems. Consequently, this chapter defines the research problem as follows:

"Developing a holistic methodology for addressing the manufacturing systems configuration (MSC) problem for adapting manufacturing systems to rapidly changing manufacturing requirements".

In the context of the above-defined research problem, a "holistic solution" means an approach that combines data modelling, software integration, and adaptive optimisation and decision-making algorithms.

Rather than addressing each knowledge gap in isolation, a holistic solution synergistically merges these domains, ensuring that data models are compatible with the software solutions and that the optimisation and decision-making methods can effectively utilise both. This cohesive approach aims to offer a holistic solution, ensuring efficiency and adaptability, by leveraging the strengths of each component to address the challenges of rapidly changing manufacturing requirements.

To tackle the above research problem, defining the key terms and notations is necessary. Also, as the MSC problem involves optimisation and relies on the utilisation of optimisation and decision-making algorithms, it is important to model the problem mathematically. Therefore, the next section lays down and formalises the above research problem by defining the critical terms and mathematical notations.

3.2.1 Definitions and mathematical notations

The terms defined in this section are referenced and used in technical chapters to develop a data model, interoperable software solutions, optimisation and decision-making algorithms. Although the definitions of terms and mathematical notations are abstract, they provide the foundational framework for the holistic solution. These definitions ensure clarity, consistency, and precision in communication and interpretation across various manufacturing system components.

3.2.1.1 Definitions

The key definitions of the terms used throughout the thesis are defined below:

- **Manufacturing asset**: A tangible or intangible valuable resource that plays a significant role in manufacturing, contributing directly to producing goods or services (e.g., manufacturing equipment, manufacturing software, staff).
- **Manufacturing configuration**: A manufacturing configuration is a constituent of a manufacturing system representing a specific manufacturing process setup. It systematically combines assets to form capabilities with given capacities and costs to facilitate adaptive optimisation in response to changing manufacturing requirements (e.g., a setup involving a CNC machine, an industrial robot, control software, and a human operator).
- **Manufacturing capability**: A specific skill or ability of a manufacturing asset or manufacturing configuration resulting from combination of manufacturing assets enabling the production of a specific product or component (e.g., drilling, assembling, welding capabilities).
- **Manufacturing capacity**: The maximum level of value-added activity a manufacturing asset or manufacturing configuration can achieve in a desig-

nated period under standard operating conditions for a specific capability (e.g., the capacity to perform ten drilling operations per hour).

- **Operational parameter**: This term denotes any run-time parameter of a manufacturing asset or manufacturing configuration that can affect its effectiveness measured by key performance indicator (e.g., parameters of the manufacturing equipment that affect the energy consumption).

- **Changes in manufacturing requirements**: This term defines the dynamic adjustments or alterations in manufacturing capabilities, capacities, and operational parameters to accommodate rapidly changing requirements (e.g., a surge in production volume, alterations in product design, or a new requirement to operate in a carbon-neutral manner).

3.2.1.2 Mathematical notations

The mathematical notations and functions provided below enable the mathematical formulation of the MSC problem for utilisation of optimisation algorithms:

- The set of available assets:

$$\mathcal{A} = \{A_1, \ldots, A_{N_\mathcal{A}}\} \tag{3.1}$$

where A_i is i-th manufacturing asset, and $N_\mathcal{A}$ is the total number of assets.

For example:
$\mathcal{A} = \{\text{Industrial robot}, \ldots, \text{Gripping end-effector}\}$

- The set of manufacturing configurations:

$$\mathcal{F} = \{F_1, \ldots, F_{N_\mathcal{F}}\} \tag{3.2}$$

where $F_i = \{A_{i,1}, \ldots, A_{i,N_{F_i}}\}$ is i-th manufacturing configuration, $A_{i,j}$ is the

j-th asset of i-th manufacturing configuration, and $N_{\mathcal{F}}$ is the total number of manufacturing configurations.

- A configuration-to-assets matching function that matches a manufacturing configuration to a set of assets:

$$g : \mathcal{F} \rightarrow 2^{\mathcal{A}} \tag{3.3}$$

where $2^{\mathcal{A}}$ is a power set of assets (that is, the set of all subsets of $\mathcal{A}$). For example: $g(F_1) = \{A_2, A_4, A_7\}$

- The set of available capabilities:

$$\mathcal{B} = \{B_1, \ldots, B_{N_{\mathcal{B}}}\}. \tag{3.4}$$

where $N_{\mathcal{B}}$ is the total number of capabilities.

For example: $\mathcal{B} = \{\text{drilling}, \ldots, \text{welding}\}$

- A capability-to-configurations matching function that matches a capability to a set of manufacturing configurations:

$$f : \mathcal{B} \rightarrow 2^{\mathcal{F}} \tag{3.5}$$

For example: $f(\text{drilling}) = \{F_2, F_5\}$ returns all manufacturing configurations that have a drilling capability.

- A capacity function that matches a manufacturing configuration and a capability to a capacity per unit time:

$$h : \mathcal{B} \times \mathcal{F} \rightarrow \mathbb{Z}^+ \tag{3.6}$$

For example: $h(\text{drilling}, F_3) = 50$ operations per unit time, that is, the manufacturing configuration F_3 has a drilling capability with a capacity of

50 operations per unit time.

- A manufacturing requirement in the requirement period t with the sequence of required capability and capacity pairs:

$$D_t = [(B_{t,1}, P_{t,1}), \ldots, (B_{t,N_t}, P_{t,N_t})] \tag{3.7}$$

where $P_{t,i} \in \mathbb{Z}^+$ is the required capacity for a capability $B_{t,i} \in \mathcal{B}$ in the demand period t, and N_t is the total number of required capabilities in the demand period t.

- Requirement periods with a sequence of requirements and requirement period lengths:

$$\mathcal{R} = [(D_1, L_1), \ldots, (D_T, L_T)] \tag{3.8}$$

where $L_t \in \mathbb{Z}^+$ is the duration of the requirement period D_t, and T is the total number of requirement periods.

3.2.1.3 Costs

Once definitions and notations are established, defining clear costs for optimisation problems is crucial. In this case, three types of costs have been defined as functions on the number of manufacturing configurations $x_t \in \mathbb{Z}^+$ used at given requirement period t. Each of these costs can be broken down into individual components:

- *Investment cost for requirement period t:* $I(x_t)$. Investment costs are incurred only once to set up manufacturing configurations. Investment costs can be decomposed into the cost of purchasing new assets, the required space, and other one-time costs.

- *Recurring cost for requirement period t:* $R(x_t)$. Recurring costs reoccur for a unit time of the utilisation of manufacturing configurations. Recurring costs

can be decomposed into the cost of energy consumption, staff salaries, and other costs that recur during the duration of the demand period.

- *Transition cost from requirement period t to period $t+1$:* $Q(x_t, x_{t+1})$. Transition costs are costs for transitioning from one requirement period to another by changing existing manufacturing configurations. Transition costs depend on previous and current manufacturing configurations. The cost is zero if there is no transition to another requirement period.

3.2.1.4 Manufacturing requirements satisfaction problem

After defining the necessary components, the MSC problem for changing manufacturing requirements is mathematically formulated as below:

Given requirement periods:

$$\mathcal{R} = [(D_1, L_1), \ldots, (D_T, L_T)] \tag{3.9}$$

Solve:

$$\min_x \left(\sum_{t=1}^{T} I(x_t), \sum_{t=1}^{T} L_t R(x_t), \sum_{t=1}^{T-1} Q(x_t, x_{t+1}) \right) \tag{3.10}$$

such that

$$\sum_{j=1}^{N_{\mathcal{F}}} h(B_{t,i}, F_j) \cdot x_{t,i,j} \geq P_{t,i} \tag{3.11}$$

and

$$x_{t,i,j} \in \mathbb{Z}^+ \tag{3.12}$$

$\forall t \in [1, \ldots, T]$, $\forall i \in [1, \ldots, N_t]$ and $\forall j \in [1, \ldots, N_{\mathcal{F}}]$.

Equation 3.11 is a capacity-satisfaction constraint, and Equation 3.12 is an integrality constraint.

3.2.2 Cross-cutting research challenges

Upon defining the research problem, terminology, and notations, three primary challenges emerge in developing a holistic solution for the MSC problem:

- **Challenge 1: Data modelling in complex manufacturing systems.** Manufacturing systems comprise intricate components and processes. The challenge lies in effectively capturing these complexities to address the MSC problem. The solution demands data models that integrate system components and account for dynamic operational conditions and potential fluctuations.

- **Challenge 2: Software integration in diverse equipment.** Manufacturing systems often comprise heterogeneous equipment with varied capabilities, interfaces, and communication protocols. This diversity complicates unified software solutions. The key is to develop modular software that can integrate different equipment types, enhancing overall system efficiency and adaptability.

- **Challenge 3: Optimisation and decision-making involving multiple and uncertain objectives.** Manufacturing objectives and costs constantly shift, and traditional optimisation methods struggle with these dynamics. Addressing this challenge requires adaptive optimisation techniques that merge traditional methods with advanced strategies like machine learning and reinforcement learning. Integrating adaptive optimisation and machine learning methods will ensure adaptability to evolving manufacturing objectives.

3.2.3 Research questions and hypotheses

The research problem and the challenges in developing the holistic solution for the MSC problem are divided into separate research questions. Each research question is addressed using a research hypothesis that builds upon the literature review on enabling technologies.

3.2.3.1 Research question 1

The first research question is formulated as follows to tackle the first research challenge:

What type of data models can accurately represent the complexities and dynamic nature of manufacturing systems while also suitable for software application integration?

In pursuit of an answer to this research question, a comprehensive scrutiny of existing data modelling techniques, their relevance to manufacturing systems, and their compatibility with software integration is essential.

Given manufacturing systems' intricate and dynamic nature, the ideal data models must encompass various system components, such as machinery, processes, workflows, and human involvement. They should also account for the interrelationships and dependencies among these factors and exhibit flexibility towards operational or system changes.

The sought data models need to accurately represent the extensive network of operations within the manufacturing system while maintaining adaptability for future modifications.

3.2.3.2 Research hypothesis 1

In response to the first research question, the first research hypothesis is postulated as follows:

Object-oriented data modelling can effectively capture the complexities and dynamism integral to manufacturing systems while being suitable for integrating with software applications.

This first research hypothesis is based on the concept that the inherent qualities of object-oriented data models - encapsulation, inheritance, and polymorphism - make them particularly suited for illustrating the varied components of a manufacturing system, their interconnections, and compatibility with software applications.

3.2.3.3 Research question 2

The second research question to address the second research challenge states as follows:

What software development approaches can reduce the interoperability challenges posed by utilising diverse equipment in manufacturing systems while enhancing plug-and-produce capabilities?

This research question delves into the software development approaches that can mitigate the interoperability challenges associated with incorporating various manufacturing equipment types within manufacturing systems. These software development strategies should be built on top of data models and ease the integration of diverse equipment, leading to enhanced plug-and-play capabilities. Here, plug-and-produce indicates the ease with which new machinery or system components can be incorporated into manufacturing with minimal configuration efforts.

3.2.3.4 Research hypothesis 2

In response to the second research question, the second research hypothesis is defined as follows:

Developing manufacturing software as modular manufacturing apps, incorporating a modular approach to integrate various equipment and communication protocols, can enhance a manufacturing system's interoperability and plug-and-produce capabilities.

The modular app methodology breaks down a complex system into smaller, manageable units or apps, each designed to handle a specific function or process within the manufacturing system. This approach enhances the system's scalability, adaptability, and maintainability.

Such an approach could offer a scalable and flexible solution to enhance interoperability. Different equipment types and communication protocols can be more efficiently managed by designing software modules or apps that employ a unified integration strategy.

3.2.3.5 Research question 3

The third research question to address the third research challenge states as follows:

What types of algorithms can effectively manage uncertainties and facilitate sequential decision-making in the dynamic and evolving multi-objective environment of manufacturing processes?

This research question investigates the algorithms that could effectively handle uncertainties and support sequential decision-making in manufacturing optimisation and decision-making processes.

Given manufacturing systems' dynamic and multi-objective nature, the research explores algorithmic methodologies capable of handling uncertainties and managing sequential decision-making while adapting to evolving manufacturing objectives. The key factors under consideration include the algorithms' capacity to adapt to changes in manufacturing requirements, reconcile conflicting objectives, mitigate uncertainties, and manage sequential decision-making to enhance the overall performance of the manufacturing system.

3.2.3.6 Research hypothesis 3

In response to the third research question, the third research hypothesis is defined as follows:

An optimisation and decision-making framework that integrates traditional optimisation and machine learning algorithms can effectively manage uncertainties and facilitate sequential decision-making in dynamic, multi-objective manufacturing environments.

Integrating traditional optimisation algorithms with machine learning algorithms could provide an optimal balance between performance and adaptability while effectively managing uncertainties and facilitating efficient sequential decision-making. Traditional optimisation methodologies offer a reliable base, while machine learning algorithms, such as Bayesian optimisation and reinforcement learning, can continually refine the solutions based on real-time feedback. Consequently, this approach could potentially handle the challenges associated with fluctuating manufacturing objectives and conditions, leading to a more resilient and efficient manufacturing system.

3.3 Validation methodology

The validation methodology of the proposed research hypotheses consists of theoretical and practical works. Initially, theoretical works are established. After establishing theoretical works, they are experimentally validated in a controlled lab environment and extended into real industrial manufacturing use cases.

3.3.1 Theoretical knowledge contributions

Initially, for each of the research hypotheses, theoretical knowledge contributions are provided. These theoretical works form the basis for practical validations. Consequently, the next three chapters provide theoretical works addressing the research hypotheses.

Chapter 4 provides theoretical work on the object-oriented data model for representing manufacturing systems' complexities and dynamic nature. The object-oriented data model is developed by dissecting the manufacturing system into individual components.

Chapter 5 provides theoretical work on developing modular manufacturing software, introduces the concept of manufacturing apps and describes the components necessary to develop a manufacturing app. This chapter also presents insights into developing a modular architecture for deploying manufacturing apps.

Chapter 6 provides theoretical work on optimisation and decision-making algorithms for the MSC problem. New algorithms and decision-making environments are proposed to tackle manufacturing processes' multiple objectives and uncertainties.

3.3.2 Validation tasks and objectives

After developing theoretical contributions for each of the hypotheses, the practical validation of each of them is done using the below validation process in the following steps:

1. Selection of two manufacturing processes that exhibit similarities but necessitate distinct manufacturing assets and capabilities.
2. Identification of the required manufacturing assets for the aforementioned processes.
3. Introduction of new manufacturing requirements to evaluate the adaptability and resilience of the proposed solutions:
    - Capability change requirement.
    - Capacity change requirement.
    - Operational parameters change requirements.
4. Assessment of the success rate, both qualitatively and quantitatively.

For each research hypothesis, the following objectives and success metrics are defined:

1. **Data model validation**
    - **Objective:** Develop an object-oriented data model for the manufacturing processes to test the hypothesis on object-oriented data modelling using the theoretical contributions in Chapter 4.
    - **Success metric:**
        - The object-oriented data model can adapt to changes in requirements without extensive modifications.

  - The model can represent both the static and dynamic aspects of the manufacturing process.

2. **Manufacturing apps validation**

   - **Objective:** Develop modular manufacturing apps, using theoretical contributions provided in Chapter 5, that incorporate a modular approach to integrating various equipment and communication protocols, ensuring seamless interoperability across diverse manufacturing equipment.
   - **Success metric:**
     - The apps can be easily integrated with different manufacturing equipment without extensive customisation.
     - Enhanced plug-and-produce capabilities are evident, allowing for quick setup and deployment in diverse manufacturing environments.

3. **Optimisation and decision-making validation**

   - **Objective:** Integrate and develop apps for optimisation and decision-making algorithms developed in Chapter 6 that combine traditional optimisation techniques with machine learning algorithms tailored for dynamic, multi-objective manufacturing environments.
   - **Success metric:**
     - The algorithms can effectively manage uncertainties in the manufacturing process, adapting to changes in real-time.
     - Optimisation and decision-making algorithms can be integrated with the developed data models and apps.

Measure of success for the holistic approach: The proposed holistic solution will be deemed successful if each outlined criteria is effectively validated, building on the previous step.

3.3.3 Validation scenarios

The validation of research contributions is critical in ensuring the applicability and effectiveness of proposed methodologies. In this research, the experimental validations are conducted in two distinct stages, each offering unique insights into the adaptability and efficiency of the proposed solutions in different environments.

3.3.3.1 First stage: controlled robotics environment

The experimental validations in the first stage are performed in a controlled robotics environment using two distinct manufacturing processes: sorting cylinders based on attributes like diameter, height, and colour and bin-picking of industrial pipe coupler parts. The former is a static classification task, focusing on the accurate differentiation of cylinders with closely related specifications. The cylinders are typically presented in a consistent orientation for sorting. Conversely, bin picking is dynamic, where parts are randomly oriented within a container. The bin-picking process necessitates advanced spatial recognition algorithms to identify the correct part, discern its orientation, and plan a collision-free path for the robotic arm, making it a more intricate task for robotic manipulation.

Several new manufacturing requirements are introduced in the validation process, as depicted in Figure 3.1. These requirements include transitioning from sorting to a bin-picking process (capability change), increasing the number of bin-picking operations per time unit (capacity change), and reducing energy consumption without compromising cycle time (operational parameters change). Introducing these requirements during the validation steps offers a comprehensive insight into the adaptability of the proposed solutions to real-world manufacturing constraints and variations.

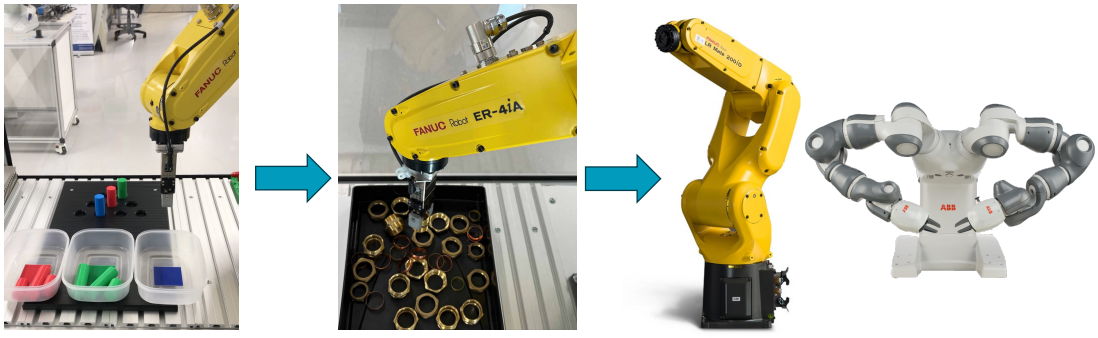

Figure 3.1: The first validation stage.

3.3.3.2 Second stage: industrial validations

In the next stage, the experiments extend the validation of the proposed research hypotheses by examining them in real-world industrial settings, as illustrated in Figure 3.2 and Figure 3.3. Each of these projects presents distinct manufacturing environments with complexities, requirements, and challenges.

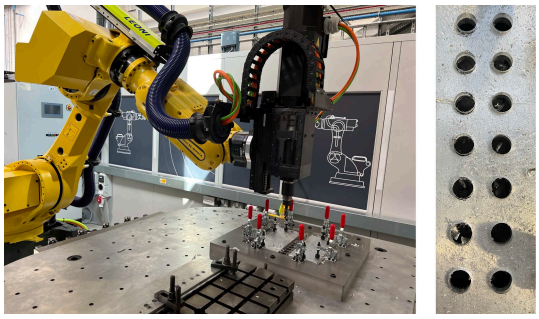

Figure 3.2: Validation process at the Omnifactory facility, University of Nottingham, UK. This focuses on optimal manufacturing configurations for aerospace manufacturing use cases.

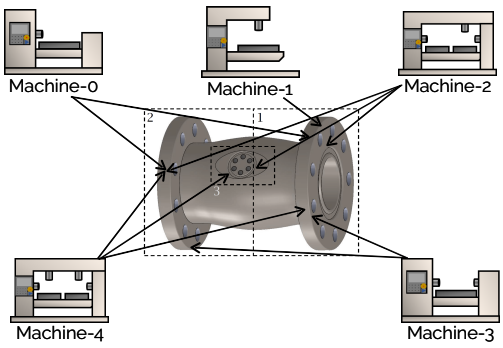

Figure 3.3: Validation process at the University of Mondragon, Basque Country, Spain. This emphasizes optimal manufacturing configurations for machining process planning use cases.

The efficacy of the proposed solutions is assessed in these varied contexts using a mix of quantitative and qualitative performance indicators. These encompass efficiency metrics, resource allocation, adaptability to alterations, and stakeholder satisfaction in multi-criteria decision-making.

3.4 Chapter summary

This chapter laid the groundwork for the research by clearly defining the problem statement and associated challenges. It introduced relevant terminology and mathematical notations to enable precise problem formulation.

The research problem was broken down into three key challenges, leading to the development of corresponding research questions. Each research question was paired with a research hypothesis grounded in technical literature.

A rigorous validation methodology was proposed, combining qualitative and quantitative analysis. This approach will comprehensively assess the research hypotheses through theoretical analysis and real-world case studies.

The aim is to develop a holistic solution to the defined research problem. This solution encompasses novel data modelling techniques, modular software applications, adaptive optimisation and decision-making algorithms. The goal of synergistically integrating these elements is to enhance manufacturing systems' responsiveness and resilience to evolving requirements.

With the problem context established, subsequent chapters will provide an in-depth investigation into each research hypothesis. The next chapter, Chapter 4, explores object-oriented data modelling methodologies for manufacturing systems.

Chapter 5 introduces the concept of modular manufacturing apps to interoperable and plug-and-produce software integration.

Finally, Chapter 6 delves into adaptive optimisation algorithms tailored for dynamic manufacturing environments.

The experimental validations are done in Chapter 7 and in Chapter 8.

Chapter 4

Data model for MSC problem

4.1 Introduction

One of the key challenges in holistically addressing the MSC problem lies in the complexity of components and processes in modern manufacturing systems. This thesis proposes a data model specifically designed to capture the intricate details of these systems to address this challenge. This unified modelling methodology enables effective management of various manufacturing equipment, software, costs, and objectives.

Chapter 2 reviewed the literature to identify research gaps in creating a data model tailored to the MSC problem. Although numerous data models exist, none cater specifically to the MSC problem. This shortcoming hinders the development of interoperable software solutions and the integration of optimisation and machine learning algorithms. Therefore, this chapter focuses on developing a data model suited to the challenges posed by the MSC problem.

Following the research methodology detailed in Chapter 3, the aim is to address the following research question and hypothesis:

- **Research question 1:** What type of data models can accurately represent the complexities and dynamic nature of manufacturing systems while also suitable for software application integration?
- **Research hypothesis 1:** Object-oriented data modelling can effectively capture the complexities and dynamism integral to manufacturing systems while being suitable for integrating with software applications.

The central research contribution of this chapter is an object-oriented data model designed specifically for the MSC problem. An object-oriented data modelling technique is used to create the entire model, representing each element as a class, depicted using UML class diagrams. This approach provides an inclusive view of

the structure and relationships within the data model.

It is important to note that manufacturing systems vary considerably based on industry, scale, and other factors. While the proposed data model captures many critical elements of manufacturing systems, it may not cover all components. Nevertheless, it offers valuable insights for decision-makers aiming to optimise their manufacturing operations to meet changing requirements.

The contributions provided in this chapter are peer-reviewed and disseminated in the following venues:

- Torayev et al. [143] "Optimal Selection of Manufacturing Configurations using Object-Oriented and Mathematical Data Models", 10th International Conference on Industrial Engineering and Applications. 4-6 April 2023, Phuket, Thailand.
- Torayev et al. [144] "Multi-Criteria Decision-Making for Optimal Manufacturing Configuration Selection Using an Object-oriented Data Model and Mathematical Formalization", ASME International Manufacturing Science and Engineering Conference (MSEC 2023), 12-16 June 2023, New Brunswick, New Jersey, USA.

The next section of this chapter starts by explaining the key components and principles of an object-oriented data modelling technique. It introduces UML concepts for the visual presentation of the proposed data model. The further sections detail modelling different manufacturing components for the MSC problem.

4.2 Object-oriented data modelling (OODM) in manufacturing

Object-oriented data modelling (OODM) is a data modelling technique that incorporates the principles of object-oriented programming based on the concept of "object". It facilitates the definition of data structures in a way that represents real-world entities, enabling efficient manipulation and interpretation of data.

This thesis proposes using OODM for modelling manufacturing systems in addressing the data modelling challenges in the MSC problem. This modelling approach can represent manufacturing components, their attributes, methods, and interactions as objects within classes. Consequently, manufacturers can explore various configurations and choose the most efficient one that meets the specified criteria. Furthermore, the OODM enables interoperable software integration.

It is helpful to consider a simple manufacturing process involving a pick-and-place operation using an industrial robotic arm to provide a clearer understanding of the concepts and principles of OODM. This process also incorporates a vision camera for positioning and a part that needs to be picked up and placed at a specified location.

4.2.1 Components of OODM

OODM comprises four primary components: objects, classes, attributes, and methods:

- **Objects** are fundamental entities in an object-oriented model. They represent real-world elements and encapsulate data and methods associated with these elements. For instance, in the pick-and-place manufacturing example, an object can be a "FanucRobot" with attributes like "speed" or "reacha-

bility" and methods like "pick" or "place".

- **Classes** act as a blueprint or template for creating objects. It defines the common characteristics shared by all instances of this class. In the manufacturing scenario, one could define a class "IndustrialRobot", and all specific industrial robots in the manufacturing system will be objects of this class.

- **Attributes** refer to the properties or characteristics that define an object. They store information about the state of an object. For example, a "FanucRobot" object may have attributes such as "payload", "number of axes", or "precision" in the above example.

- **Methods** signify a behaviour or operation that an object can perform. Methods define what an object can do and what can be done to it by changing its attributes. For instance, the "FanucRobot" object can have a method "moveTo", which instructs the robotic arm to move to a certain location.

4.2.2 Principles of OODM

The four fundamental principles that govern OODM are abstraction, encapsulation, inheritance, and polymorphism:

- **Abstraction** is the process of simplifying complex systems by modelling classes appropriate to the problem and working at the most suitable level of inheritance for a given aspect of the problem. For instance, the "IndustrialRobot" abstraction could hide complex details like motor movements and present simple methods such as "pick" and "place" instead.

- **Encapsulation** refers to the bundling of related properties and behaviours into individual objects, with access to these properties restricted to the same object's methods. In the case of the "FanucRobot", details like kinematic

parameters could be encapsulated within the object, restricting direct manipulation of these attributes.

- **Inheritance** is a mechanism where a new class can be derived from an existing class, inheriting all its attributes and methods. For instance, an "IndustrialRobot" class can be a subclass inheriting from the "ManufacturingEquipment" superclass, possessing additional specific attributes like "width", "height", or "weight".

- **Polymorphism** allows objects of different types to be treated as objects of a parent type, enabling different behaviours to be executed depending on the specific object type. For instance, the "pick" method of the "FanucRobot" may behave differently when picking different industrial parts based on their attributes.

4.2.3 Unified modelling language (UML)

Transitioning from OODM's principles and components, this subsection introduces a valuable tool for OODM – the unified modelling language (UML). UML serves as a standardised language offering a platform-independent set of diagrams to visually express the structure and behaviour of systems, thereby enhancing their understandability and maintainability. It embodies a rich collection of symbols and notations, making it a powerful tool for describing various aspects of an object-oriented system.

With a broad range of diagrams under its umbrella, this thesis primarily utilises class and object diagrams of UML due to their high relevance in expressing the static structure of a system and the instances of classes at a specific point in time.

4.2.3.1 UML class diagram

Class diagrams, one of the diagrams of UML, present a visual depiction of the static structure of an object-oriented system. They illustrate the system's classes, their attributes and methods, and, crucially, the relationships among these classes. As a result, they serve as a tool for visualising a system's blueprint.

The relationships between classes in class diagrams are primarily expressed through three types:

- **Association**: This relationship signifies a link between two classes indicating that they are connected or interact with each other. In the context of the pick-and-place example, an association might exist between the "FanucRobot" class and the "IndustrialPart" class, suggesting that a robot interacts with industrial parts during the manufacturing process.
- **Aggregation**: This term describes a "whole-part" relationship where a class (the whole) comprises other classes (parts) but does not exclusively own them. For instance, the "ManufacturingProcess" class could aggregate the "IndustrialRobot" and "IndustrialPart" classes, as these components constitute part of the manufacturing process.
- **Composition**: This relationship is a stronger form of aggregation where the "whole" class owns the "part" class, and the parts cannot exist without the whole. In the above example, a "FanucRobot" object might comprise several "Motor" objects. If the "FanucRobot" ceases to exist, the associated "Motor" objects would also cease to exist, as they are integral parts of the robot.

4.2.3.2 UML object diagram

Object diagrams, in contrast to class diagrams, depict a snapshot of the instances within a system and their interrelations at a specific moment. Essentially, they materialise a class diagram's structure at a given time, detailing the relationships between instances and their present state.

For instance, an object diagram might portray several instances of the "IndustrilabRobot" class interacting with various instances of the "IndustrialPart" class, revealing the intricacies of their interaction and the state of each object during a certain phase of the manufacturing process.

4.2.3.3 OODM for the MSC problem

Employing OODM and UML concepts, Figure 4.1 provides a comprehensive UML class diagram for the proposed object-oriented data model for the MSC problem, illustrating class relationships.

The UML class diagram in Figure 4.1 consists of 12 key classes. The relationships between these classes are shown with specified arrows. An association line is a solid line connecting two classes, sometimes with an arrowhead to indicate direction if the association is not bidirectional. The aggregation relationship is depicted as a hollow diamond on the side of the "whole" class pointing towards the "part" class. The composition relationship is depicted as a filled diamond on the side of the "whole" class, pointing towards the "part" class. The inheritance relationship is depicted by a hollow arrow pointing from the child class (the more specific class) to the parent class (the more general class). A dashed line often represents a dependency relationship, known as a "using" relationship which occurs when one class uses the methods of another class but does not necessarily need to maintain a reference to it.

The next sections of this chapter detail relationships, attributes and methods of each class shown in Figure 4.1.

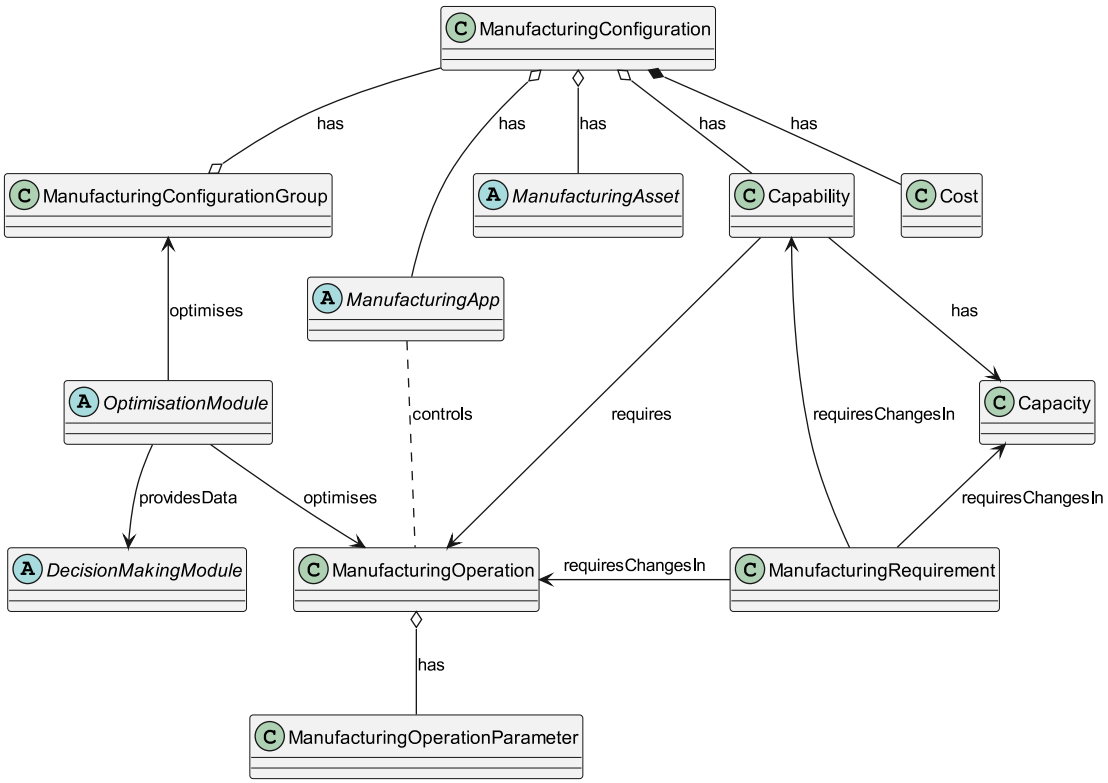

Figure 4.1: UML class diagram for object-oriented data model for the MSC problem for changing manufacturing requirements. Details about attributes and methods are omitted for a detailed discussion in the following sections.

4.3 Modelling manufacturing requirements

Manufacturing operations often face disruptions due to various factors, such as an increase in product volume, changes in product design, or a need to reduce energy consumption. These disruptions, rooted in the dynamic nature of manufacturing requirements, often result in inefficiencies and challenges in the ongoing production system.

As a solution to these challenges, this section aims to classify these various manufacturing requirements into broad categories, thus formalising them into a structured data model.

4.3.1 Changes in manufacturing requirements

Generally, manufacturing requirements in the MSC problem can be classified into three primary categories:

- **Capability change requirements**: This category of manufacturing requirement represents the need to modify manufacturing capabilities that encompass specific skills, technologies, or processes that enable a manufacturing system to produce a particular product or component. Factors such as product design updates, new product introductions, or technological advancements may require changes in capability requirements.
- **Capacity change requirements**: This category of manufacturing requirement signifies the need for adjustments in manufacturing capacity, referring to the maximum production output that a manufacturing system can achieve within a specific time. Capacity changes may be necessary to accommodate fluctuations in production volumes, seasonal variations, or market trends.
- **Operational parameters change requirements**: This category of manufacturing requirement refers to the need to adjust manufacturing operation parameters, such as the run-time operational parameters of manufacturing equipment. These changes are often necessary to optimise manufacturing performance, maintain product quality, reduce costs, or address environmental concerns.

4.3.2 Representation of manufacturing requirements

Building upon the understanding of the types of manufacturing requirements in the MSC problem, it is essential to represent this concept in a structured and coherent way. This facilitates a comprehensive view of manufacturing requirements and their impact on manufacturing systems.

Figure 4.2 shows the UML class diagram for the *ManufacturingRequirement* class, illustrating its relationships with the *Capability*, *Capacity*, and *ManufacturingOperation* classes within the context of the MSC problem.

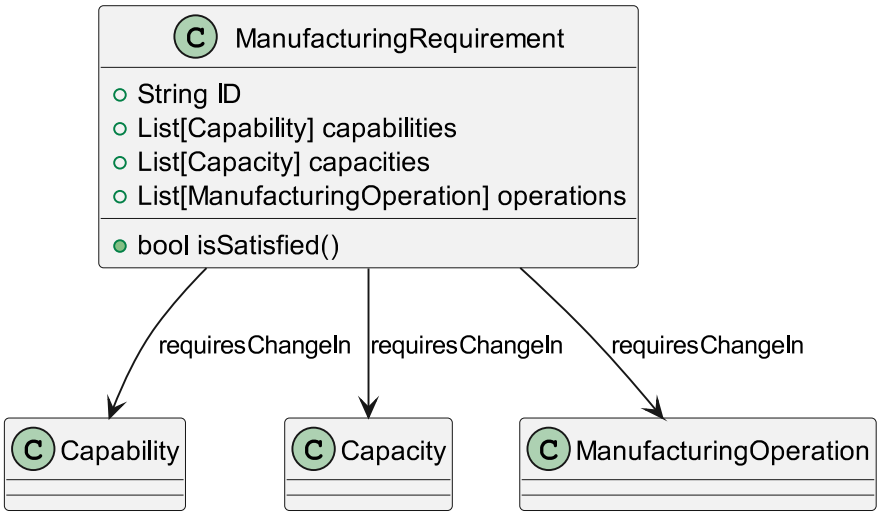

Figure 4.2: UML class diagram of *ManufacturingRequirement* class.

The *ManufacturingRequirement* class encapsulates the requirements, including capabilities, capacities, and operational parameters, needed to fulfill production needs. The class has the following attributes and methods:

- *ID*: A unique string identifier assigned to each *ManufacturingRequirement* instance.
- *capabilities*: A list of *Capability* objects representing the manufacturing capabilities required for this specific requirement.
- *capacities*: A list of *Capacity* objects signifying the necessary manufacturing capacities to accommodate this requirement.
- *operations*: A list of *ManufacturingOperation* objects representing the required changes in operational parameters for this requirement.
- *isSatisfied()*: A method that returns a Boolean value indicating whether the current requirement has been met based on the available capabilities, capacities, and operational parameters within the manufacturing system.

The *ManufacturingRequirement* class interacts with the *Capability*, *Capacity*, and *ManufacturingOperation* classes through *"requiresChangeIn"* associations, which represent the dependencies between these classes:

- *"requiresChangeIn"* relationship with *Capability*: This association indicates that a *ManufacturingRequirement* instance depends on one or more *Capability* instances to satisfy its specific manufacturing requirements. Each *Capability* object embodies a particular skill, technology, or process essential for producing a product or component.

- *"requiresChangeIn"* relationship with *Capacity*: This association denotes that a *ManufacturingRequirement* instance relies on one or more *Capacity* instances to fulfil its production needs. Each *Capacity* object corresponds to the maximum production output attainable.

- *"requiresChangeIn"* relationship with *ManufacturingOperation*: This association signifies that a *ManufacturingRequirement* instance is dependent on one or more *ManufacturingOperation* instances to address the necessary adjustments in operational parameters. Each *ManufacturingOperation* object represents a specific aspect of the manufacturing process.

The proposed way of defining and modelling manufacturing requirements in the MSC problem allows a detailed, structured, and comprehensive representation of the evolving needs in manufacturing systems. By categorising manufacturing requirements into capability changes, capacity changes, and operational parameter changes, it offers an explicit understanding of the different types of adjustments that may be required. This classification is beneficial for precisely capturing the adaptation needs of the manufacturing system and efficiently planning and implementing suitable responses.

Maintaining *Capability*, *Capacity*, and *ManufacturingOperation* as separate classes

provides modularity by encapsulating distinct aspects of manufacturing requirements in their entities. This separation ensures the flexibility to modify or extend one aspect independently of others, enhancing system maintainability and adaptability to changing requirements.

4.4 Modelling manufacturing components

Following the modelling of manufacturing requirements, the next stage in addressing the MSC problem involves developing a representative data model for various manufacturing system components, presented in Chapter 3.2.1 when describing mathematical notations and definitions.

The top-down design strategy is employed for modelling the components of manufacturing systems. It starts with defining broad concepts and gradually refining these into detailed, specific components. This approach is beneficial in the given context, facilitating a comprehensive understanding of the manufacturing configuration and its key elements before exploring detailed configuration selections. This strategy ensures the alignment of all components with the manufacturing requirements.

4.4.1 Manufacturing configuration

Figure 4.3 shows a UML class diagram for the *ManufacturingConfiguration* class.

The *ManufacturingConfiguration* class has associations with several other classes, as described below:

- Aggregation association with the *ManufacturingAsset* class: This indicates that instances of the *ManufacturingAsset* class are combined within a single *ManufacturingConfiguration* instance. *ManufacturingAsset* instances can

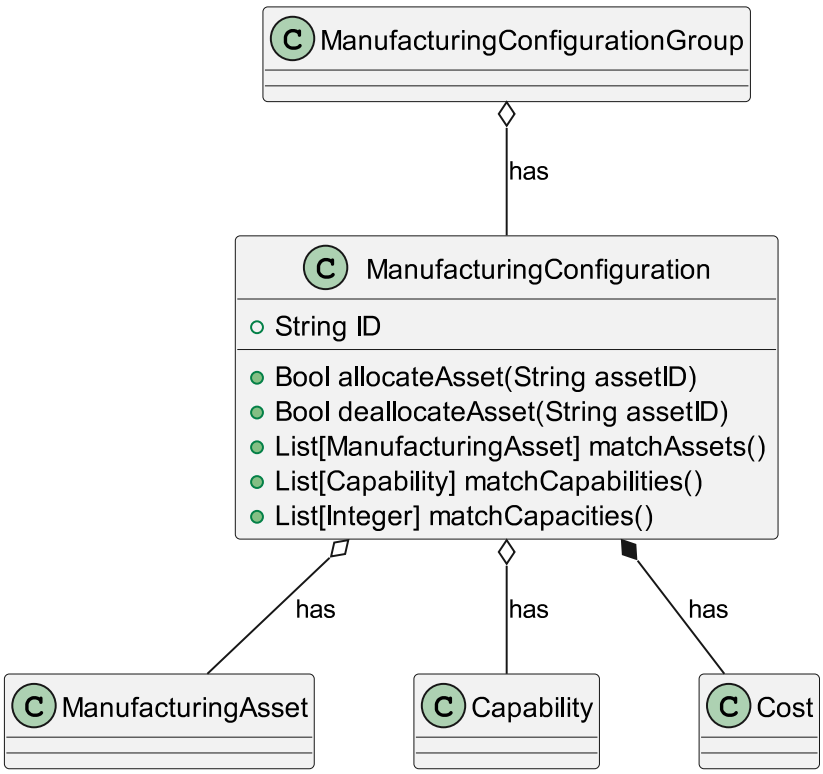

Figure 4.3: UML class of *ManufacturingConfiguration* class. The diagram shows the associations between *ManufacturingConfiguration* class and other classes necessary for the MSC problem.

function independently of the *ManufacturingConfiguration* class but are grouped to execute a manufacturing process.

- Aggregation association with the *Capability* class: This association signifies that *Capability* class instances are independent from the *ManufacturingConfiguration* class. Nevertheless, they are grouped based on the *ManufacturingAsset* class assigned to the *ManufacturingConfiguration* instance.

- Aggregation association with the *ManufacturingConfigurationGroup* class: This relationship facilitates the grouping of related manufacturing configurations, which helps to organise and manage the manufacturing process. The *ManufacturingConfigurationGroup* class groups *ManufacturingConfiguration* instances when a single instance cannot meet the requirements. The primary purpose of the *ManufacturingConfigurationGroup* class is to enable efficient organisation of optimisation algorithms.

- Composition association with the *Cost* class: This connection represents a relationship where each *Cost* class instance depends on the corresponding *ManufacturingConfiguration* class instance. As a result, *Cost* class instances are terminated when the related *ManufacturingConfiguration* class instance is destroyed.

The proposed UML class, *ManufacturingConfiguration*, encompasses the below methods:

- The *allocateAsset()* method accepts a string parameter called *assetID* and returns a boolean value. It assigns a manufacturing asset, identified by the given ID, to the manufacturing configuration instance. The returned Boolean value signifies the success or failure of the allocation process.
- The *deallocateAsset()* method, taking a string parameter named *assetID*, also returns a boolean value. It removes a manufacturing asset, identified by the specified ID, from the manufacturing configuration instance. The returned Boolean value indicates the success or failure of the deallocation process.
- The *matchAssets()* method produces a list of *ManufacturingAsset* objects. This method identifies a set of manufacturing assets compatible with the given manufacturing configuration by searching for assets that correspond to the defined capabilities and costs. The *ManufacturingAsset* objects returned by this method represent suitable assets for manufacturing.
- The *matchCapabilities()* method implements a function that matches a given capability to manufacturing configurations. It accepts a *Capability* object as a parameter and returns a set of *ManufacturingConfiguration* instances. This method aligns a given capability to all manufacturing configurations that possess this capability.

- The *matchCapacities()* method represents a capacity function. It accepts a *Capability* object and a *ManufacturingConfiguration* instance as parameters and produces an integer, representing the capacity per unit of time. The function matches a manufacturing configuration and a capability to a capacity, showing how many operations involving the capability can be performed per unit of time in the given manufacturing configuration.

These methods allow access by any object that has an instance of the *ManufacturingConfiguration* class.

The *ManufacturingConfiguration* class effectively embodies the elements and relationships necessary for the MSC problem. Its class diagram and associations with *ManufacturingAsset*, *Capability*, *ManufacturingConfigurationGroup*, and manufacturing *Cost* classes enable a thorough comprehension of the manufacturing process and its components.

The aggregation and composition associations guarantee the configuration's modular and interdependent nature. Simultaneously, the *allocateAsset()*, *deallocateAsset()*, and *matchAssets()* methods offer vital functionalities for managing and adjusting the manufacturing configuration in response to fluctuating production requirements. These methods ensure adaptability and interaction with other components of the system.

Subsequent subsections delve into the *ManufacturingAsset*, *Capability*, and *Cost* classes in detail. A brief description of the *ManufacturingConfigurationGroup* is provided, as it serves only for the organisation of manufacturing configurations and its use by optimisation algorithms.

4.4.2 Manufacturing asset

Within manufacturing systems, tangible assets include objects such as machinery, tools, and equipment that require physical infrastructure for storage and utilisation. On the contrary, intangible assets constitute software, data, and intellectual property, all of which are in digital formats and accessible through digital devices. An accurate data modelling of these assets is fundamental to the MSC problem. As such, this subsection gives attention to object-oriented modelling of manufacturing assets, providing precise class definitions through the use of UML class diagrams structured comprehensively and logically.

4.4.2.1 Modelling a manufacturing asset

The UML class diagram in Figure 4.4 illustrates the hierarchy of the class of manufacturing assets, including physical and digital assets, and their relationships. Using this diagram makes it possible to see how different manufacturing assets relate to each other and the unique attributes of each asset type.

In Figure 4.4, the enumeration class *AssetType* defines two types of assets within a manufacturing system: physical assets, such as robots and digital assets, such as software. An enumeration class is a data type that consists of a fixed set of named values.

In Figure 4.4, the *ManufacturingAsset* is an abstract class, serving as a blueprint for creating other classes. It is the parent class for all manufacturing assets, providing a common set of attributes and methods that all manufacturing assets inherit from the parent class.

Establishing the *ManufacturingAsset* class as an abstract class offers several advantages in the context of a manufacturing system. First, it ensures that the *ManufacturingAsset* class is a blueprint for creating specialised subclasses that

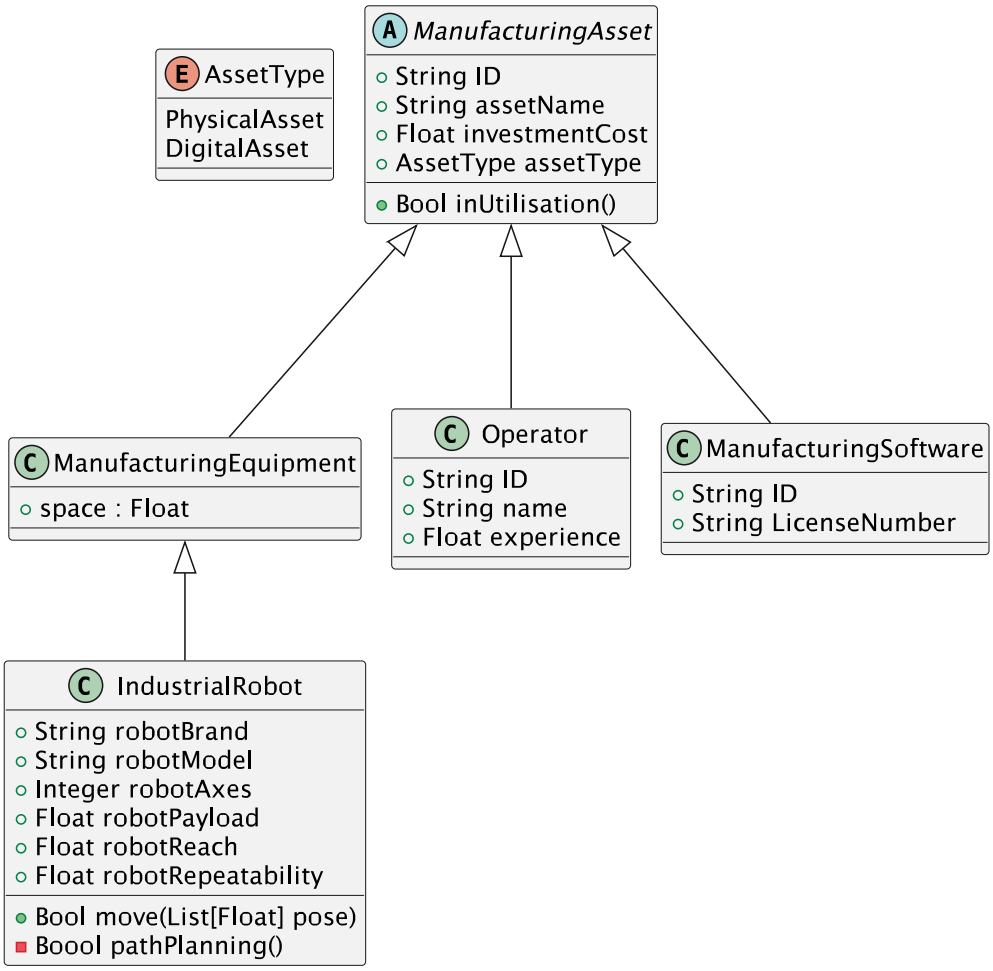

Figure 4.4: UML class of *ManufacturingAsset*. The diagram depicts classes for different types of assets.

represent specific manufacturing assets with distinct characteristics or functionalities. This promotes a consistent structure, enabling modularity and scalability as new manufacturing assets are added to the system.

Secondly, providing a common set of attributes and methods in the abstract *ManufacturingAsset* class enables code reusability and maintainability. All manufacturing assets inherit these attributes and methods from the parent class, reducing code duplication and making it easier to manage changes or updates to the shared properties of manufacturing assets.

Lastly, the abstract nature of the *ManufacturingAsset* class prevents the direct instantiation of generic manufacturing assets, which may lack the specialised attributes and methods required for effective system operation. This enforces the

creation of more specific subclasses that meet the unique requirements of manufacturing assets, ultimately leading to a more robust and efficient manufacturing system.

The attributes and methods, common to all manufacturing assets, defined in *ManufacturingAsset* are:

- *ID* is typically a unique identifier for the asset, differentiating it from other system assets;
- *assetName* is a human-readable name for the asset, often used for display and reference purposes;
- *assetType* specifies the type of asset, as defined in the *AssetType* enumeration class;
- *investmentCost* is used to record the cost of investing in an asset.
- *inUtilisation()* method of *ManufacturingAsset* class shows whether an asset is currently utilised or not.

4.4.2.2 Types of manufacturing assets

Subclasses derived from the parent class *ManufacturingAsset* inherit the attributes and methods and can extend the parent class by defining additional attributes and methods. By defining common attributes and methods in the parent class, this modelling approach facilitates the development process and ensures uniformity among diverse types of manufacturing assets. The UML class diagram in Figure 4.4 includes the following inherited subclasses:

- *ManufacturingEquipment*
- *IndustrialRobot*

- *Operator*
- *ManufacturingSoftware*

ManufacturingEquipment is a subclass that extends *ManufacturingAsset* and has an attribute, *space*, representing the physical space required by the equipment.

IndustrialRobot is another subclass that extends *ManufacturingEquipment*. It has six extra attributes: *robotBrand*, *robotModel*, *robotAxes*, *robotPayload*, *robotReach*, and *robotRepeatability*. The *move()* method is used to control the robot's movement, while *pathPlanning()* is a private method that handles the robot's path planning.

In the given diagram, the *Operator* class also extends the *ManufacturingAsset* class, which means that it inherits the common attributes of *ManufacturingAsset* and can also define its unique attributes and methods. In the context of the *Operator* class, investment cost could refer to the cost associated with the hiring, training, and management of operators within the manufacturing system. The *Operator* class has additional three attributes: *ID*, *name*, and *experience*. The *ID* attribute is inherited from *ManufacturingAsset* and serves as a unique identifier for the operator. The *name* attribute is a human-readable name for the operator, which can be used for display purposes or to identify the operator in reports or other documentation. The *experience* attribute tracks the operator's level of experience or skill, which can be important in determining the appropriate tasks or responsibilities for the operator within the manufacturing system.

ManufacturingSoftware is another subclass that extends *ManufacturingAsset*, and it has two attributes: *ID* and *LicenseNumber*. As a digital asset, manufacturing software can be used to manage and monitor the operation of physical assets within the manufacturing system.

A combination of inheritance, abstract classes, and inherited classes allows easy

extension of the data model with additional classes not presented in Figure 4.4. For example, adding computer numerical control (CNC) machines or end-effectors to the data model can be accomplished by defining the respective classes and inheriting attributes and methods of the *ManufacturingAsset* class.

4.4.3 Manufacturing capability

The assignment of capabilities to a manufacturing configuration rather than an individual manufacturing asset enables for comprehensive functionality. Although individual assets may possess inherent capabilities, they can often be relatively basic or atomic. However, when these assets are combined methodically within a manufacturing configuration, they generate enhanced manufacturing capabilities, facilitating the production of complex goods or services.

Based on the definition of manufacturing capability, it is essential to model this concept in relation to the *ManufacturingConfiguration* class and its connection to other relevant classes. This modelling will provide a clearer understanding of how capabilities interact within manufacturing configurations.

The UML class diagram in Figure 4.5 shows the *CapabilityType* enumeration class and *Capability* class, representing a capability that a class *ManufacturingConfiguration* can possess. The UML class diagram also shows the relationship between *Capability* and *ManufacturingOperation*.

4.4.3.1 Atomic and combined capabilities

The enumeration class *CapabilityType* in Figure 4.5 defines two types of capabilities:

- *Atomic* capability

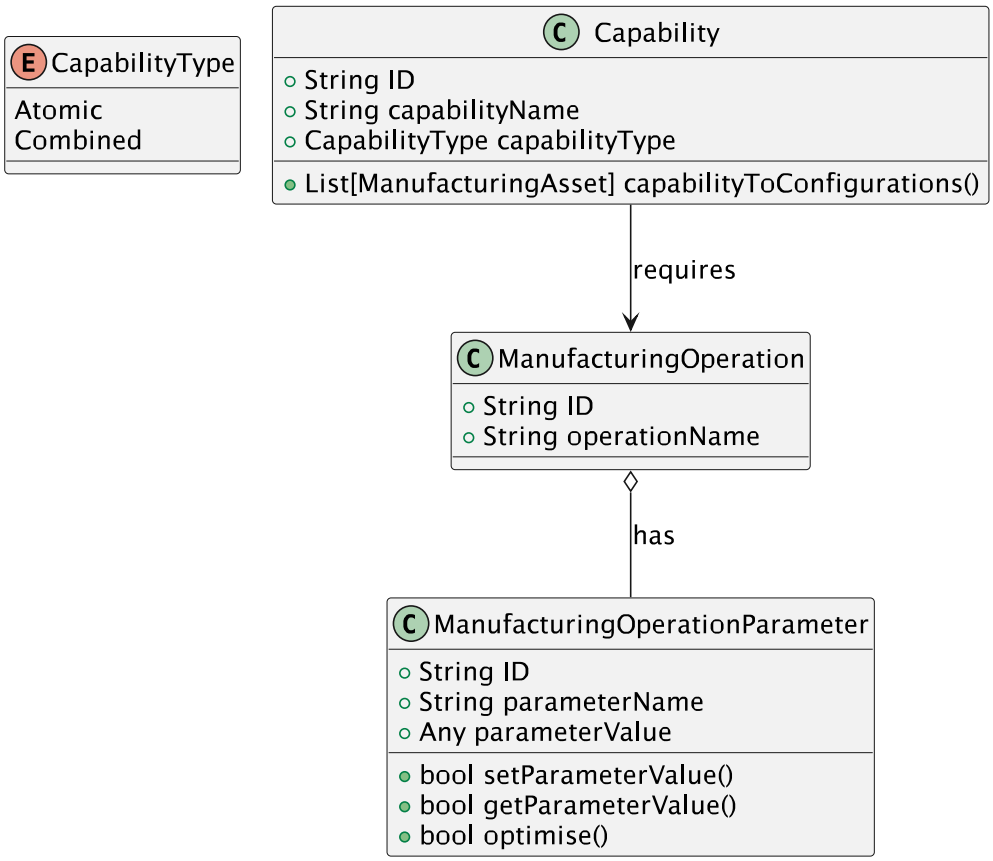

Figure 4.5: UML class diagram representing a manufacturing capability and the related classes.

- *Combined* capability

An *Atomic* capability is a basic capability that cannot be broken down further. Examples of the *Atomic* capabilities include moving robotic arms, drilling and gripping, as shown in Figure 4.6 (left). These capabilities cannot be broken down further and are fundamental to manufacturing processes.

In contrast, a *Combined* capability is a complex capability made up of multiple *Atomic* capabilities. Examples of the *Combined* capabilities include pick and place, sorting, and assembly, as shown in Figure 4.6 (right). These capabilities comprise multiple *Atomic* capabilities and require the coordination of several manufacturing assets combined into a manufacturing configuration to perform the desired manufacturing process.

Based on the types of the capabilities, the *Capability* class has the following prop-

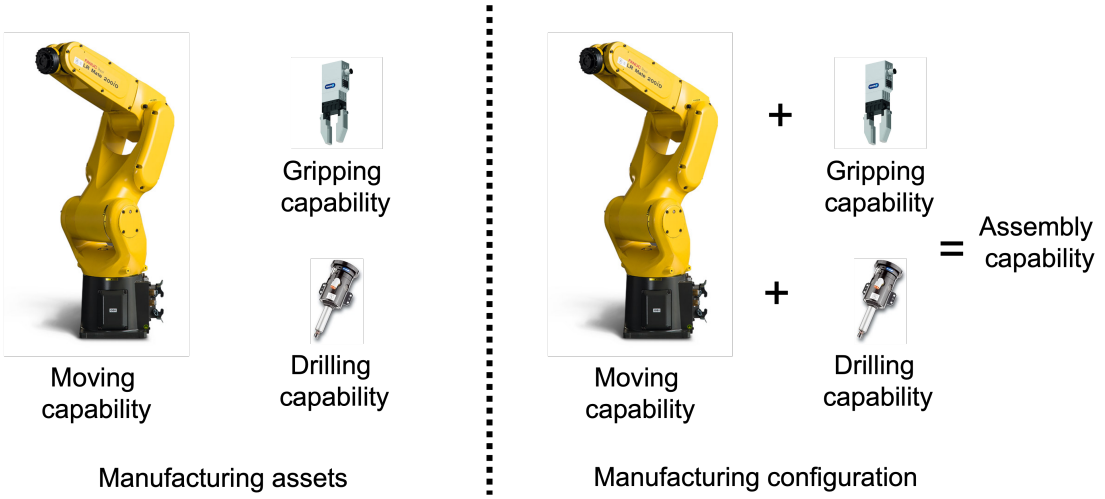

Figure 4.6: The demonstration of combining manufacturing assets and atomic capabilities into manufacturing configuration with a combined capability

erties:

- The *ID* property is of type String and is used to identify a capability uniquely.
- The *capabilityName* property is also of type String and describes the capability in more detail.
- The *capabilityType* property is of type the *CapabilityType*, an enumeration that defines two values: *Atomic* and *Combined*.

The *Capability* class has one method called *capabilityToConfigurations()*, which returns a list of *ManufacturingConfiguration* objects. This method is used to find manufacturing configurations that possess the capability represented by the *Capability* object. For example, if a *Capability* object represents the capability to drill holes, then the *capabilityToConfigurations()* method would return a list of *ManufacturingConfiguration* objects that can drill holes.

4.4.3.2 Manufacturing operation

After building a manufacturing configuration from a group of physical and digital manufacturing assets, the manufacturing configuration possesses one or multiple capabilities. While capability is general, an operation refers to specific actions that a manufacturing configuration performs with a given capability. Thus, manufacturing capability requires manufacturing operations to perform actual production. For example, the atomic capability "move robot arm" must perform operations such as "linear movement" and "circular movements".

Each *ManufacturingOperation*, such as a "linear movement" or a "circular movement" in the provided example, has a unique identity (ID) and *operationName*, indicating the type of operation performed within the manufacturing process. To further refine and optimise these operations, the *ManufacturingOperationParameter* class is defined. These parameters represent the adjustable aspects of a *ManufacturingOperation*, allowing for fine-tuning and optimisation of individual operations.

For example, in the atomic capability "move robot arm", the *ManufacturingOperation* "linear movement" can have parameters such as *ManufacturingOperationParameter* objects such as speed, acceleration, and path curvature. By adjusting these parameters, it is possible to optimise the robot arm's movement to meet specific production requirements or constraints, ultimately improving overall performance and resource utilisation.

4.4.3.3 Capacity

Understanding the concept of capacity is vital when discussing manufacturing capabilities. Each manufacturing configuration has an inherent capacity constraint, which can limit its ability to meet fluctuating production demands. Various fac-

tors, such as equipment availability, workforce skills, or production schedule, can influence these constraints. Accurate capacity assessment is essential for efficient resource allocation, production planning, and overall process optimisation.

To address this limitation, the concept of a manufacturing configuration group has been introduced. This approach assembles various distinct or similar manufacturing configurations to collectively fulfil the requirements. By pooling the capacities of multiple configurations, the manufacturing configuration group can effectively respond to changes in production demands, enhance resource utilisation, and maintain the flexibility needed to adapt to changing manufacturing requirements. Moreover, this grouping strategy enables a more efficient allocation of resources and promotes the continuous improvement of manufacturing processes.

4.4.4 Manufacturing costs

After modelling manufacturing assets, capabilities and operations, it is crucial to model manufacturing costs for optimisation and decision-making purposes. Manufacturing costs provide a quantitative basis for evaluating and comparing manufacturing configurations, assets, capabilities, and operations.

Cost modelling in this context refers to the representation of costs within an object-oriented data model, which offers the potential for nuanced insights and effective cost management strategies. Although the comprehensive modelling of costs, along with all its inherent complexities, surpasses the scope of this thesis, the aim here is to construct a flexible and modular data model that can readily incorporate varying cost data. This approach facilitates the seamless integration of evolving cost data and provides an efficient platform for handling the dynamic nature of manufacturing costs.

Manufacturing costs include all expenses associated with producing a product, including labour, materials, and equipment usage. The provided UML diagram in

Figure 4.7 presents a class called *Cost*, representing various costs associated with the manufacturing configurations.

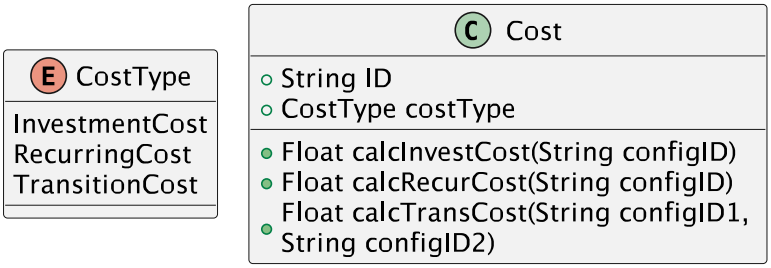

Figure 4.7: UML class diagram for representing manufacturing costs.

The *Cost* class has two properties:

- *ID* is of type String and is used to uniquely identify a cost;
- *costType* is of type *CostType*, an enumeration that defines three cost types: *InvestmentCost*, *RecurringCost*, and *TransitionCost*. The attribute *costType* specifies the cost associated with the *Cost* object.

The *Cost* class also has three methods used to calculate the cost associated with a manufacturing configuration:

- The *calcInvestCost()* method accepts a configuration ID parameter of type String and returns a Float value. This method computes the investment cost associated with a manufacturing configuration identified by the configuration ID. The investment cost represents the expenses of purchasing and establishing the assets required for a manufacturing process.
- The *calcRecurCost()* method accepts a configuration ID parameter of type String and returns a Float value. This method computes the recurring cost associated with a manufacturing configuration identified by the configuration ID. Recurring costs include costs incurred during manufacturing, such as energy consumption and labour costs.

- The *calcTransCost()* method accepts two configuration ID parameters of the type String and returns a Float value. This method computes the transition cost associated with changing from one manufacturing configuration identified by *configID1* to another manufacturing configuration identified by *configID2*. Transition costs include expenses associated with adjusting a manufacturing configuration when requirements change and the current configuration cannot meet demand.

The costs defined in the UML diagram, including investment, recurring, and transition costs, can sometimes conflict. For example, a manufacturing configuration optimised for low recurring costs might require a high investment cost upfront. Conversely, a manufacturing configuration optimised for low investment costs could have higher recurring costs.

Moreover, when manufacturing requirements change, reconfiguring the existing configuration to meet new demands can result in high transition costs. The decision to incur these costs can conflict with the goal of keeping recurring costs low.

4.5 Chapter summary

This chapter presented an object-oriented data model specifically designed for the manufacturing systems configuration (MSC) problem. The model comprises key classes representing manufacturing requirements, assets, capabilities, operations, and costs. Associations between these classes capture the relationships and dependencies in manufacturing systems.

The manufacturing requirement class categorises requirements into changes in capabilities, capacities, and operational parameters. This classification provides a structured representation of evolving production needs. Separate classes for

capability, capacity, and operations enable modularity in the data model.

The manufacturing configuration class and associations integrate assets, capabilities, costs, and configuration groups. Critically, the model incorporates tangible assets like equipment and intangible assets like software, recognising the role of both in modern manufacturing. The manufacturing asset class employs inheritance and abstract classes to define common attributes and methods for various asset types.

Notably, the model associates capabilities with configurations rather than individual assets. This composition of atomic capabilities from assets into combined capabilities for configurations reflects real-world manufacturing more accurately.

The manufacturing operation and cost classes add further granularity. Different cost types provide a quantitative basis for evaluating configurations. However, manufacturing costs involve greater intricacies than those captured in the model.

The object-oriented data model offers a flexible, modular foundation tailored to the MSC problem. It enables the assimilation of diverse manufacturing components and relationships into an interoperable structure. Nevertheless, real-world manufacturing systems have added complexities. The model may benefit from validating classes against specific manufacturing use cases, which is done in Chapter 7.3.

Building on top of the data model presented in this chapter, the next chapter, Chapter 5, addresses the challenges of integrating manufacturing software solutions by introducing the concept of “manufacturing apps”.

Chapter 5

Manufacturing apps for MSC problem

5.1 Introduction

Chapter 4 introduced a data model tailored to address the MSC problem. Despite theoretical contributions, the challenge of rapidly adapting to changing manufacturing requirements persists. Although the proposed data model offers a theoretical methodology, the practical implementation of these solutions still remains a challenge.

One of the primary reasons for this challenge is the lack of interoperability among manufacturing equipment in modern manufacturing systems. Interoperability issues impede rapid adaptation of systems by hindering the development of interoperable manufacturing software solutions.

Therefore, this chapter adheres to the research methodology outlined in Chapter 3, with the aim of addressing the following research question and hypothesis:

- **Research question 2:** What software development approaches can reduce the interoperability challenges posed by utilising diverse equipment in manufacturing systems while enhancing plug-and-produce capabilities?
- **Research hypothesis 2:** Developing manufacturing software as modular manufacturing apps, incorporating a modular approach to integrate various equipment and communication protocols, can enhance a manufacturing system's interoperability and plug-and-produce capabilities.

Interoperability in manufacturing systems is a critical concern as it speaks directly to the ability of various pieces of equipment and software to work cohesively. The reasons for the lack of interoperability are multifaceted, one of which is the diversity of manufacturing equipment used in modern manufacturing environments. Each piece of equipment may come from a different manufacturer, each with its unique set of protocols, standards, and data formats, which makes seamless inte-

gration a daunting task.

In an era where manufacturing requirements change rapidly, the ability to quickly adapt systems can be a significant competitive advantage. However, a lack of interoperability slows this process considerably, as changes must be made on a system-by-system or machine-by-machine basis. This can result in increased downtime, decreased productivity, and ultimately loss of profitability.

Given the challenges mentioned above, this chapter seeks to address the obstacle of interoperability by developing a novel concept: *ManufacturingApp*. Using the principles of modularity and standardisation, the goal is to streamline the software development process and enhance the interoperability of manufacturing systems.

The proposed approach of using *ManufacturingApp* to address the challenges of interoperability in manufacturing systems takes a fundamentally different route from most of the existing literature. While traditional approaches often focus on creating customised solutions for each unique manufacturing system, the *ManufacturingApp* approach aims at designing modular and standardised applications that can interface with a wide range of systems. This modular and generalised approach significantly reduces the time, resources, and complexity typically associated with customised software solutions.

Most of the previous work has focused on attempting to align various existing technologies or redesigning systems to a unified standard. This process can be time-consuming and resource-intensive. In contrast, the *ManufacturingApp* approach is designed to be flexible and adaptable, integrating with different protocols, standards and data formats, thus accommodating the diversity of manufacturing equipment.

In order to realise this concept, the current chapter offers an in-depth exploration of the *ManufacturingApp* class, initially proposed in Chapter 4. The emphasis is on laying the solid theoretical and practical foundation for creating these modular

software applications designed specifically for manufacturing environments. The key contributions of this chapter can be summarised as follows:

- The introduction and clear definition of the "manufacturing apps" concept sets the stage for a standardised approach towards software development in manufacturing systems.
- The development of a manufacturing apps development kit (MAPPDK) serves as a set of tools to design and create these manufacturing apps.
- Lastly, the proposition of a modular reference architecture to deploy these manufacturing apps facilitates the deployment of the apps, enhancing the system's adaptability and interoperability.

The contributions presented in this chapter were subject to peer review and were disseminated through the following venues:

- Torayev et al. [145], "Towards Modular and Plug-and-Produce Manufacturing Apps", 55th CIRP Manufacturing Systems Conference. October 29 - July 1, 2022, Lugano, Switzerland.
- "fanucpy - Python Interface for FANUC robots" software package and library distributed under Apache-2.0 licence and hosted on the GitHub platform[1].

The subsequent sections of this chapter will provide a comprehensive discussion of the contributions mentioned above. Notably, this chapter mostly focuses on the concepts of the manufacturing apps and the modular architecture for the apps, and the actual manufacturing apps are developed in Chapter 7.4.

[1] https://github.com/torayeff/fanucpy

5.2 Manufacturing apps

Breaking down complex software applications into smaller, task-specific modules, known as "appification", has transformed how industries function by enabling them to take advantage of various software applications to improve their operational efficiency and adaptability to emerging trends.

Due to the potential benefits offered by appification in the manufacturing domain, it is important to explore its practical applications and underlying principles. Therefore, this section delves into the appification of manufacturing processes, the definition of a manufacturing app, and the abstract elements that constitute such an app.

5.2.1 Appification of manufacturing processes

In a manufacturing process, such as assembling parts to produce a final product, the whole process can be divided into smaller operations. Each operation in this process involves manufacturing equipment that relies on a software solution for control. With the connectivity of Industry 4.0, the standalone control code is no longer sufficient to control equipment. Instead, the software solution includes control code and access to other data sources to coordinate production. Therefore, in the context of the MSC, the following definition is given to the "appification" of the manufacturing process:

The appification of the manufacturing process refers to separating digital and physical implementations of the manufacturing system to make implementations modular and reusable for similar manufacturing processes.

This definition assumes that each physical and digital component of the manufacturing system can be modularised to enable the development of apps for each system module. In this definition, "digital implementation" refers to the soft-

ware component, while "physical implementation" refers to the hardware in the manufacturing system.

Appification facilitates the reusability of developed solutions. For example, consider a bin-picking manufacturing process that consists of an industrial robot arm, an end-effector, and a vision system, as shown in Figure 5.1.

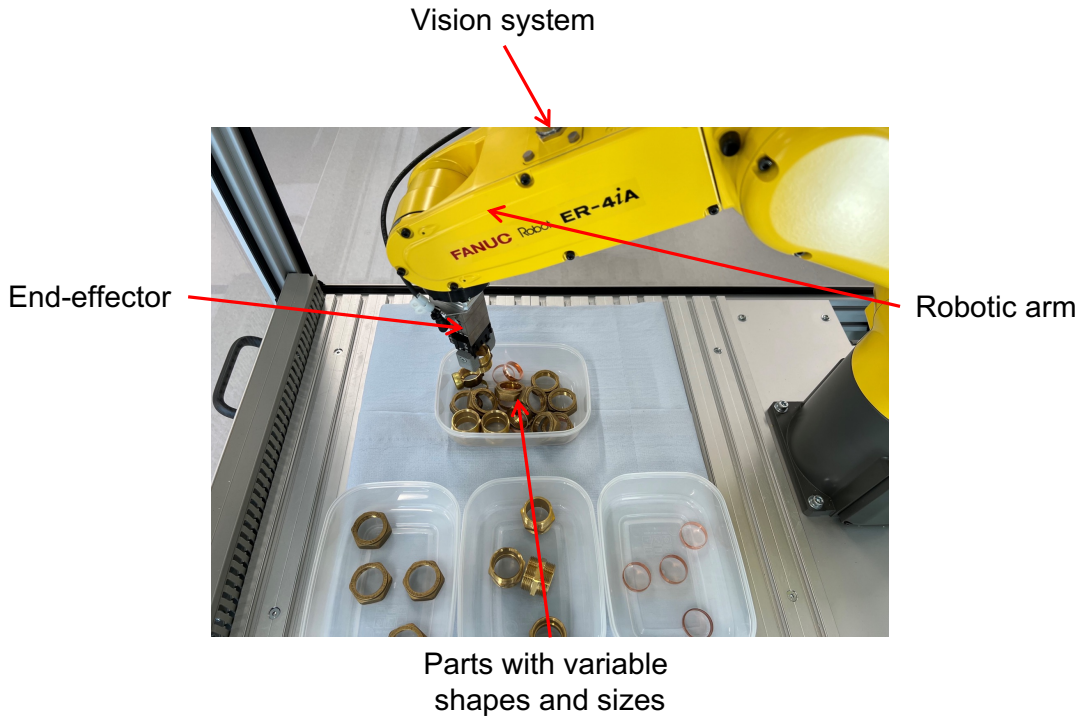

Figure 5.1: FANUC ER-4iA industrial robotic arm performing a bin-picking process using a vision system and a gripping end-effector.

The bin-picking process presents several challenges, as it often involves parts that can change or require a different pose for the robotic arm to pick them effectively. It requires constant adaptations to the vision system, robotic arm, and end-effector to accommodate these variations. Traditionally, these adjustments are made monolithically, which can be time-consuming and inflexible.

Appification provides a more efficient and adaptable solution to these challenges by modularising these three components. If there is a need to replace the vision system, robotic arm, or end-effector, the change can be made without impacting the other two modules. Furthermore, additional capabilities, such as pose esti-

mation, can be integrated through specific apps without disrupting the system. This approach creates a more versatile, time-efficient, and easily maintainable bin-picking process.

Another example is a drilling process that involves a robotic arm, a CNC controller, a drilling end effector, and a workpiece holding fixture, as shown in Figure 5.2.

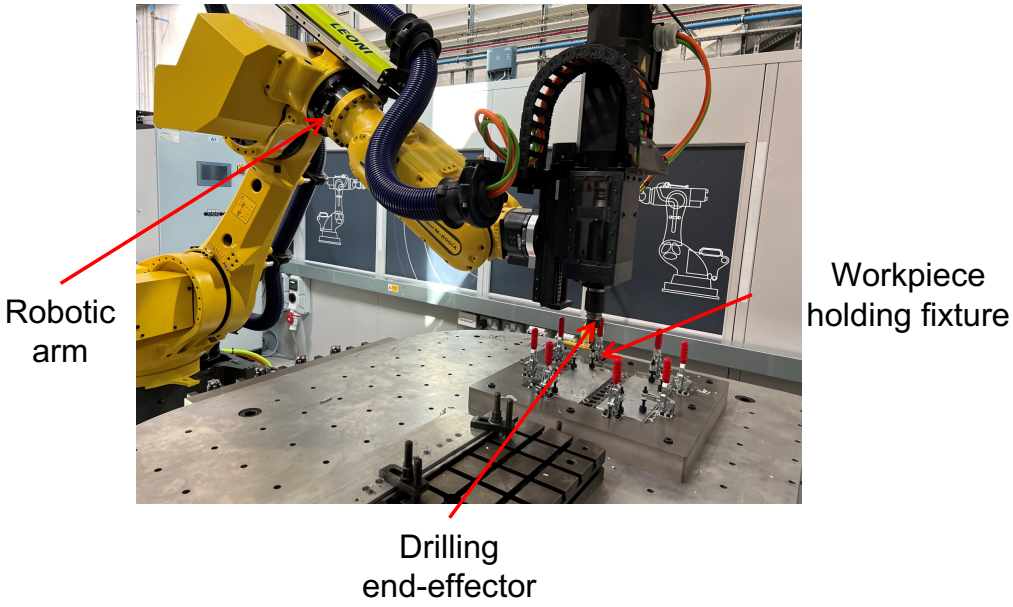

Figure 5.2: FANUC M800iA industrial robotic arm performing a drilling process operation. The robot controller controls the robotic arm and the CNC machine controls the drilling end-effector.

In the drilling process example, each component can be effectively modularised to ensure seamless adaptability and efficiency. When manufacturing apps are employed, this process can be broken down into specific tasks that are easily manageable and can be upgraded or replaced when needed without causing major disruptions to the overall process.

The CNC machine, for instance, can use an app that controls its motion, while the drilling end-effector can have an app that monitors its wear. This modular approach ensures that each component can function independently and effectively, resulting in a more efficient drilling process.

When switching from drilling to milling, the drilling end-effector module can be

replaced with a milling cutter without affecting the other two modules. This simplifies the transition and saves time and resources as the change is limited to a specific module. The appification of the drilling process enhances its overall flexibility and adaptability, making it easier to integrate with other manufacturing processes and to accommodate diverse production requirements.

5.2.2 Definition of a manufacturing app

The appification of manufacturing processes paves the way for a more efficient and adaptable approach to meet changing manufacturing requirements. By breaking down complex tasks into modular, reusable components, it becomes easier to integrate new capabilities and replace existing ones without causing disruptions to the entire system. This leads to the concept of "manufacturing apps". In the context of manufacturing and specifically the MSC problem, a manufacturing app is defined as:

A manufacturing app is a software solution that implements control of the manufacturing process or equipment in a modular and reusable way that allows replacement or addition of capabilities without affecting the other parts of the system.

A manufacturing app has three main properties:

1. **Modularity:** Manufacturing apps are designed to enable modular manufacturing processes. For example, a manufacturing process can be divided into smaller independent processes, and apps should be developed for smaller processes, not for the whole process.

2. **Reusability:** Manufacturing apps are designed to be reusable, which means that they can be used in multiple manufacturing processes and scenarios, reducing development time and cost. This property allows manufacturers to quickly and easily adapt to new production processes and equipment.

3. **Plug-and-produce:** Manufacturing apps are designed to be "plug-and-produce", meaning they can be easily integrated into existing manufacturing systems and processes with minimal configuration and setup required. This property allows for quick and efficient deployment of new equipment and production processes, reducing downtime, and increasing productivity.

The concept of manufacturing apps and the fundamental properties of manufacturing apps—modularity, reusability, and plug-and-produce—are illustrated in the context of manufacturing using industrial robots in Figure 5.3.

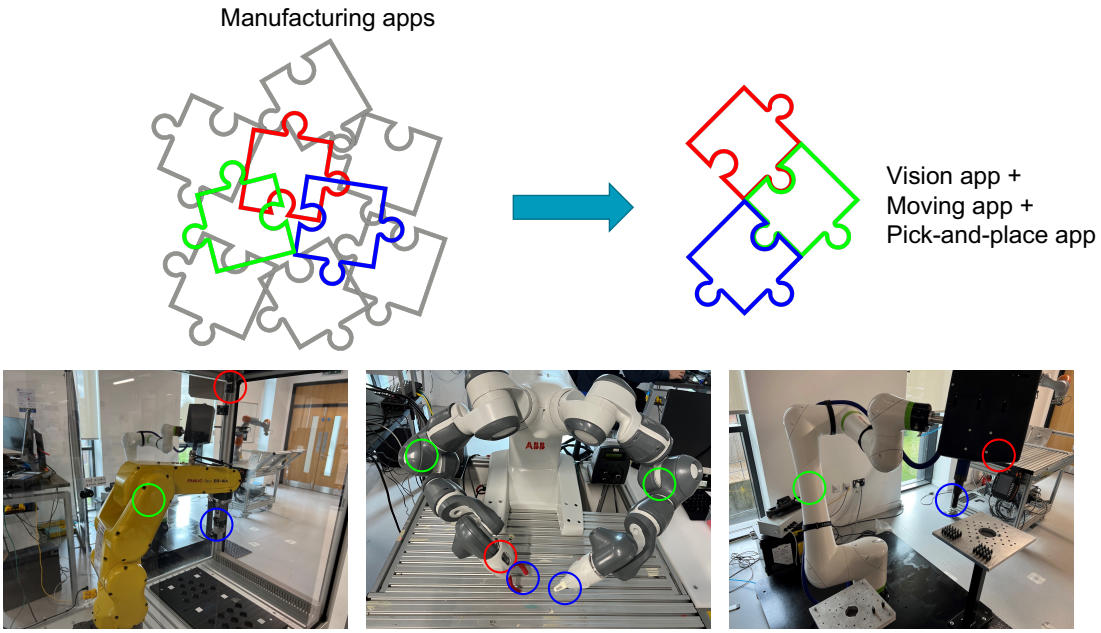

Figure 5.3: Manufacturing process which involves a vision system (red circles) a robotic arm (green circles), and an end-effector (blue circles). Appification facilitates the reusability of developed manufacturing software solutions by developing modular manufacturing apps

The image presents a central hub representing the core manufacturing process and equipment, surrounded by various modules depicted as puzzle pieces, each symbolising a manufacturing app.

Different puzzle pieces highlight the modularity aspect of manufacturing apps, demonstrating that manufacturing processes can be divided into smaller, independent components and manufacturing apps can be reused and recombined in

different ways to enable the modularity of manufacturing processes. The colour-coding or labelling of these puzzle pieces emphasises the reusability of the apps across multiple manufacturing processes and scenarios.

Furthermore, connectors linking the apps to one another signify the standardised interfaces that enable seamless communication between the apps. This element showcases the plug-and-produce property, underscoring the ease with which these apps can be integrated into existing manufacturing systems and processes, requiring minimal configuration and setup.

This representation of the properties of manufacturing apps effectively conveys the potential advantages of adopting appification strategies in the manufacturing industry, highlighting the enhanced flexibility, rapid reconfiguration, and adaptation to new environments and requirements that these apps can provide.

5.2.3 Abstract elements of a manufacturing app

In Chapter 4, a detailed description of a *ManufacturingConfiguration* class was discussed. In contrast, the description of one class that is part of the *ManufacturingConfiguration*, *ManufacturingApp* class has been omitted until this section. Therefore, based on the definitions of "appification" of manufacturing processes and "manufacturing app", this section discusses in detail the elements of a *ManufacturingApp* class.

As a modular and reusable software solution, a manufacturing app comprises several abstract elements that define its structure and functionality within the manufacturing process, as shown in Figure 5.4. These elements facilitate seamless integration and interoperability within various production environments. The following subsections detail the key abstract elements of a manufacturing app.

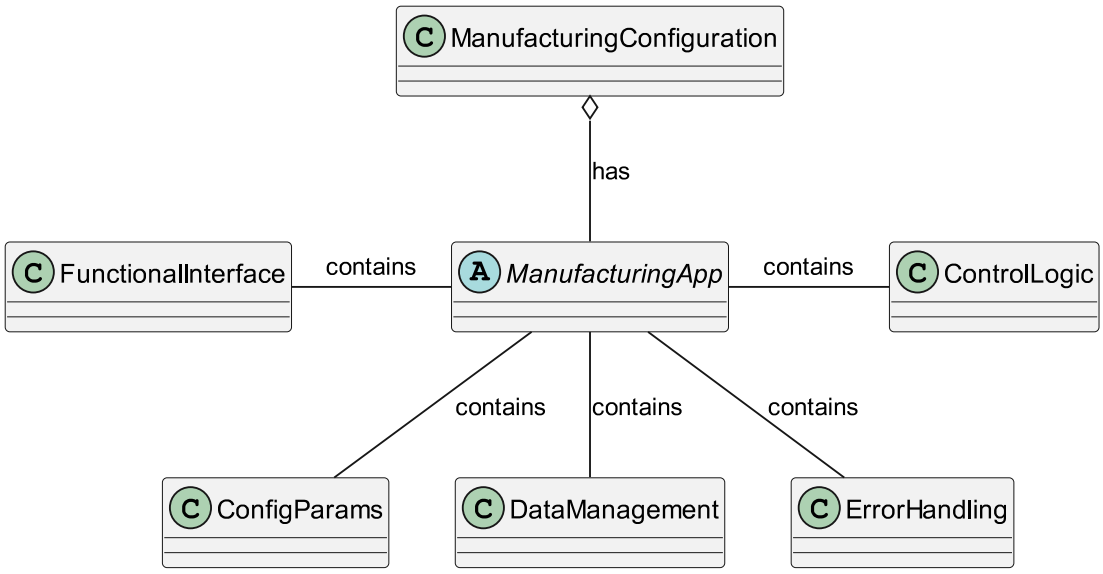

Figure 5.4: ManufacturingApp class and its elements.

5.2.3.1 Functional interface

An important element of a manufacturing app is its functional interface, which defines the inputs, outputs, and operations that it exposes to the rest of the manufacturing system. The functional interface serves as the primary means of communication between the app and other components of the production process. Manufacturing software developers can ensure that different apps interact seamlessly by establishing a standardised functional interface for each manufacturing app, facilitating modularity and reusability.

5.2.3.2 Control logic

Control logic represents the core functionality of a manufacturing app. It includes algorithms, decision-making processes, and control strategies that govern how the app interacts with manufacturing equipment and other apps. The control logic enables the app to perform specific tasks or functions within the manufacturing process, such as controlling a machine's movement or monitoring the status of a production line. Manufacturing software developers can easily update, replace, or modify the logic as needed by encapsulating the control logic within a modular

app without affecting other system parts.

5.2.3.3 Configuration parameters

Configuration parameters allow manufacturers to customise the behaviour and functionality of a manufacturing app according to specific production requirements. These parameters may include settings related to machine speeds, tolerances, or process variables. A manufacturing app can be adapted to various production environments and scenarios by providing a set of configurable parameters, enhancing its reusability and flexibility.

5.2.3.4 Data management

Data management is an essential aspect of a manufacturing app, as it deals with collecting, storing, and processing data generated during manufacturing. These data can include sensor readings, machine status information, or production statistics. A manufacturing app should provide mechanisms for handling data in a structured and efficient manner, enabling real-time decision-making and performance monitoring. Furthermore, effective data management within the app allows for better integration with other software systems, such as manufacturing execution system (MES) or enterprise resource planning (ERP) solutions.

5.2.3.5 Error handling

A manufacturing app must include robust error handling and recovery mechanisms to ensure the smooth operation of the manufacturing process. By anticipating and addressing potential failures, such as equipment malfunctions, software bugs, or communication errors, the app can maintain high reliability and resilience. Error handling and recovery mechanisms can include fault detection algorithms,

fallback strategies, or automated recovery procedures, allowing the app to respond appropriately to unexpected events and minimise the impact on the production process.

5.3 Manufacturing apps development kit

After defining the necessary elements for the development of modular manufacturing applications, the current challenge that hinders the development of manufacturing apps is the lack of a software development kit (SDK) specially designed for this purpose. An SDK is a collection of tools that facilitate the creation of applications for a given environment.

Every original equipment manufacturer (OEM) has an SDK suitable only for that hardware. For example, FANUC has the FANUC PCDK, ABB has the RobotStudio SDK, and KUKA has its software collection for developing robotic applications. It is extremely difficult to make them work together due to different vendor-specific standards. Therefore, this section proposes a methodology for developing a manufacturing apps development kit based on the definitions and class elements of the previous sections.

5.3.1 Requirements for a development kit

A manufacturing apps development kit is a comprehensive set of tools and resources that streamline the design, creation, and deployment of manufacturing applications within various production systems. To ensure successful development and integration of these apps, the development kit must address three key requirements:

1. **Standardised framework:** The development kit should provide a stan-

dardised framework for app creation, ensuring interoperability and seamless communication between different apps. This can be achieved by establishing uniform data structures, communication protocols, and application programming interface (API) that promote compatibility across diverse manufacturing environments. A standardised framework simplifies the development process and fosters consistency in software development practises, making it easier for developers to create and maintain apps on different systems.

2. **Extensive support for manufacturing environments:** The development kit must support various manufacturing environments, equipment, and processes. This involves providing libraries, APIs, and tools compatible with different hardware components, control architectures, and communication protocols. By offering extensive support for a wide range of manufacturing technologies, the development kit ensures that developed apps can be effectively integrated into existing systems and easily adapt to new technologies and processes as they emerge.

3. **Modular architecture:** The development kit should promote a modular architecture, encouraging the development of reusable and adaptable apps. This approach allows developers to create individual app components that can be easily assembled and customised to meet specific manufacturing needs. A modular architecture enhances the flexibility and scalability of developed applications and reduces development time and cost.

By addressing these technical requirements, a manufacturing apps development kit can enable the seamless integration of various hardware and software solutions, significantly improving the overall efficiency and adaptability of the manufacturing industry.

5.3.2 Data model for manufacturing apps development kit

To meet the requirements outlined in section 5.3.1, a comprehensive data model is essential. While a foundational data model has been discussed in Chapter 4, the specific requirements of a manufacturing app development kit require the extension of this base model with new relevant elements.

Figure 5.5 illustrates the data model for a kit for developing manufacturing apps. The diagram effectively demonstrates the use of object-oriented data modelling techniques, highlighting how they can facilitate the logical structure of manufacturing entities and their relationships.

For example, *IndustrialRobotApp* is shown to be associated with the *IndustrialRobot* class, reflecting the role of the app in the interaction with industrial robots. When associated with the *IndustrialRobot* class, it can leverage the properties of the associated manufacturing equipment. Additionally, the *IndustrialRobotApp* class, being a subclass of *ManufacturingApp*, inherits properties such as *FunctionalInterface*, *ControlLogic*, *ConfigurationParameters*, *DataManagement*, and *ErrorHandling*.

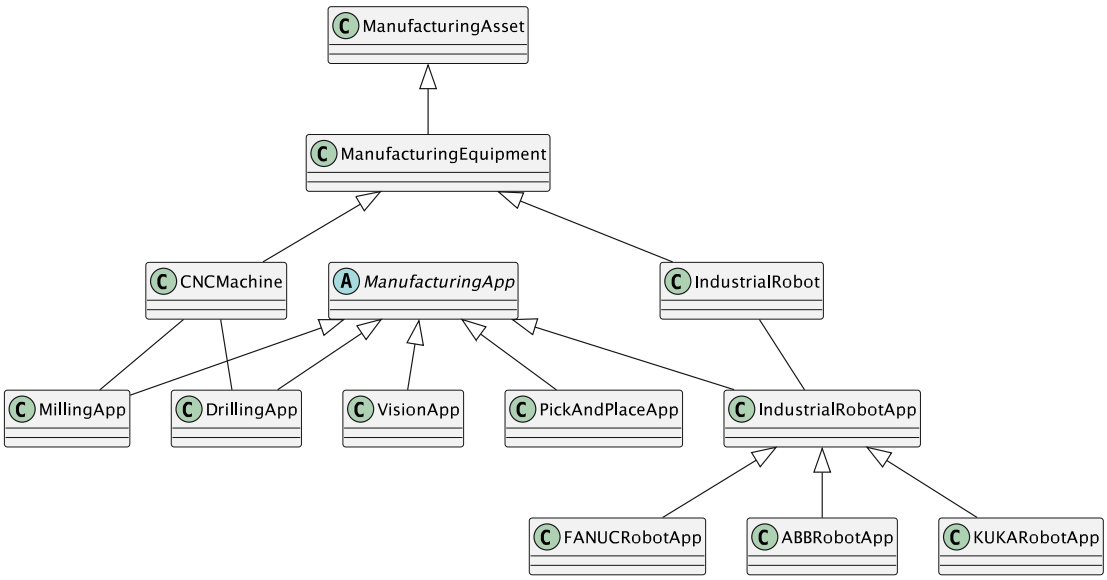

Figure 5.5: Data model for Manufacturing Apps Development Kit

The *IndustrialRobotApp* class is then further specialised into subclasses for specific brands of industrial robots, such as FANUC, ABB and KUKA. This breakdown is critical because each manufacturer uses unique conventions, such as different Euler angle representations and proprietary motion control methods. Segregating each app into its respective vendor-specific apps enhances modularity, allowing for easy adjustments when changes occur in the production setup, as only the relevant app needs to be updated.

Another important aspect of the model is the separate *VisionApp* and *PickAndPlace* apps. These apps are different from equipment-specific apps, reinforcing that the control logic for each piece of equipment is independent of that of others, thus promoting flexibility and modularity.

5.3.3 Industrial robot apps

This section demonstrates how these ideas can be applied to developing a manufacturing apps development kit tailored specifically for industrial robots that draws on the concepts and discussions of previous sections.

5.3.3.1 Robotic app challenges

Industrial robot apps pose unique challenges in manufacturing due to the diversity of robots, the number of axes, different types of robots, mathematical representations and other factors. Addressing these challenges is crucial to seamlessly integrate industrial robots within the manufacturing domain and create robust and adaptable manufacturing apps. These challenges are:

- **Diversity of robots:** Industrial robots come in various sizes, shapes, and designs, each tailored for specific tasks and applications. Robots from different manufacturers often have proprietary control systems and programming

languages, making interoperability a significant challenge. A manufacturing apps development kit for industrial robots must support a wide range of robots from different manufacturers and facilitate seamless integration into existing production systems.

- **Types of robots:** There are numerous types of industrial robots, such as articulated robots, SCARA robots, delta robots, and gantry robots, each with unique kinematics and motion control requirements. A robust manufacturing apps development kit must support various robot types, enabling developers to create customised apps that cater to the specific needs of each robot and its intended application.

- **Mathematical representations:** Different manufacturers may use different mathematical notations or conventions, which can affect understanding, calculation and application. For instance, varying Euler angle representations for orientation in robotics can complicate robot programming and control. The development kit should support various mathematical representations, including different Euler angle conventions, providing a unified platform for developing industrial robot apps.

- **Vendor-specific conventions:** Industrial robot vendors may have their proprietary conventions and methods for motion control, programming, and other aspects of robot operation. The development kit should provide a way to encapsulate and manage these vendor-specific features, allowing for the creation of modular and adaptable apps that can easily accommodate changes in the production setup or the introduction of new robot models and brands.

A manufacturing apps development kit designed for industrial robots can help simplify the integration of various robots into the manufacturing domain by addressing these challenges.

5.3.3.2 Elements of robotic apps

The diagram depicted in Figure 5.6 represents an extension of the manufacturing apps development kit specifically designed for industrial robotics. It demonstrates the various functionalities and components required to develop industrial robot applications. It addresses challenges such as the diversity of robots, robot types, different numbers of axes, Euler angle representations, and vendor-specific conventions.

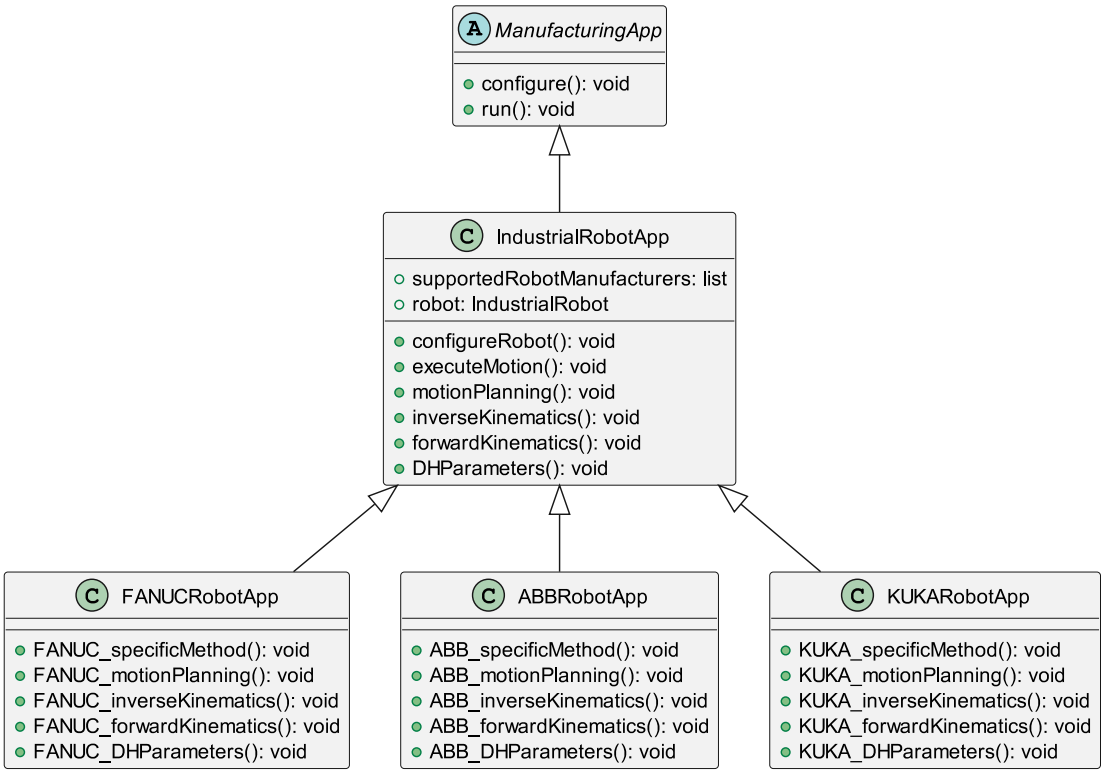

Figure 5.6: Details of *IndustrialRobotApp* class diagram

Here is a detailed description of the classes and their relationships in the diagram:

1. *ManufacturingApp*: This is an abstract class representing the general concept of a manufacturing application. It contains two abstract methods, *configure()* and *run()*, intended to be implemented by subclasses to define application-specific configurations and execution behaviour.

2. *IndustrialRobotApp*: This class is a subclass of *ManufacturingApp*, special-

ising in industrial robot applications. It introduces several new attributes and methods specific to industrial robots:

- *supportedRobotManufacturers*: A list of supported robot manufacturers.
- *robot*: A reference to the *IndustrialRobot* object being controlled by the app.
- *configureRobot()*: A method to configure the robot according to the requirements of the application.
- *executeMotion()*: A method to execute robot motion commands.
- *motionPlanning()*: A method to plan the robot's motion.
- *inverseKinematics()*: A method to compute the inverse kinematics of the robot.
- *forwardKinematics()*: A method to compute the forward kinematics of the robot.
- *DHParameters()*: A method to obtain or set the Denavit-Hartenberg (DH) parameters of the robot.

3. *FANUCRobotApp, ABBRobotApp, and KUKARobotApp:* These subclasses of *IndustrialRobotApp* are tailored for specific robot manufacturers (FANUC, ABB and KUKA). Each class contains a vendor-specific method and methods for motion planning, inverse kinematics, forward kinematics, and DH parameters tailored to the respective manufacturer's conventions and requirements.

By incorporating various robot-specific functions into the *IndustrialRobotApp* class and its subclasses, the manufacturing apps development kit is now better suited for developing apps for industrial robots. This modular design enables developers to create customised apps for different types and manufacturers of robots while maintaining a common interface for configuration and execution.

5.4 Architecture for manufacturing apps

After defining the "appification", "manufacturing apps", "manufacturing apps development kit" and developing a data model for manufacturing apps, the last step is to develop an architecture necessary for deploying manufacturing apps. This architecture is vital to solve the optimal manufacturing configuration selection problem. Therefore, this section starts by defining requirements for such an architecture and presents a conceptual model for building it.

5.4.1 Requirements for a modular architecture

There is a need for an architecture that meets certain requirements to deploy manufacturing apps; in particular, a conceptual architecture must have the following inherent characteristics:

1. An architecture must enable modular physical implementations, i.e., physical components must be independent. This facilitates the easy integration and replacement of manufacturing equipment, sensors, and other hardware elements, while minimising the impact on the overall system.

2. An architecture must enable modular digital implementations, i.e., software components must be independent. This allows for the flexible addition, modification, and removal of software modules, such as manufacturing apps and their associated control algorithms, without disrupting the system's operation.

3. An architecture must enable plug-and-produce properties; that is, physical and digital components must be replaceable and reusable with the minimum necessary configuration. This simplifies the deployment and reconfiguration of manufacturing systems, as components can be quickly swapped in or out depending on production requirements.

4. An architecture must enable the distribution of developed solutions by publishing to the central hub to enable the reusability of developed solutions in similar scenarios. This promotes the sharing and reuse of manufacturing apps, configurations, and best practises between different manufacturing sites, improving overall productivity and reducing development efforts.

The above requirements lay the foundation for a modular architecture that supports the development and deployment of manufacturing apps.

5.4.2 Elements of a conceptual architecture

Several key concepts must be introduced and understood to define a modular architecture that enables plug-and-produce capabilities. These concepts contribute to a flexible and adaptable architecture, allowing manufacturing systems to accommodate various requirements, devices, and processes. The main terms necessary for such an architecture are:

- **Atomic device**. An atomic device is a single physical device designed to perform one specific operation, such as gripping with a gripper, capturing an image with a camera, or measuring force with a force sensor. When every physical device in the manufacturing system is divided into atomic devices with one specific goal, the physical modularity of the architecture is enabled. The concept of an atomic device is consistent with the definition of a manufacturing asset and its subclass, manufacturing equipment, as defined in Chapter 4.

- **Physical node**. A physical node consists of one or more atomic devices that work together to perform a manufacturing process. For example, an industrial robot arm, camera, and gripper may form one physical node to sort products. The definition of a physical node is necessary for managing

atomic devices grouped for a single manufacturing process. A physical node corresponds to the definition of a manufacturing configuration, as defined in Chapter 4.

- **Computational node**. A computational node is a device, such as a Raspberry Pi, that can control the physical nodes or atomic devices within the physical nodes. A computational node must be able to host variable software solutions using appification and containerisation technologies. The definition of computational nodes addresses the interoperability bottleneck in existing manufacturing systems by enabling communication and digital implementations. Computational nodes enable the digital modularity of manufacturing systems.

- **Manufacturing process cluster**. A manufacturing process cluster is a group of physical and computational nodes with one specific goal: sorting, bin-picking, palletising, welding, or drilling. The manufacturing process cluster helps to organise physical and computational nodes for efficient management. The concept of a manufacturing process cluster corresponds to the manufacturing configuration group defined in Chapter 4.

- **Architecture manager**. The architecture manager is responsible for several functions, such as installing, updating, or removing apps and assigning and reassigning computational and physical nodes to manufacturing process clusters. The architecture manager enables the plug-and-produce property of manufacturing systems by providing manufacturing engineers with high-level management tools and an interface for developers to create software solutions without considering management-related issues.

- **Global applications repository**. The global application repository is a central hub for publishing and delivering manufacturing apps. The global applications repository enables the reusability of developed solutions for similar processes.

The terms atomic device, physical node, and manufacturing process cluster correspond to the concepts defined in Chapter 4. However, maintaining separate terminology for the architecture ensures its independence from the underlying data model. If the data model or architecture changes, they will not be affected by each other.

New components, such as computational node, architecture manager, and global applications repository, are defined. These concepts are inherent only to the provided architecture and do not have meaning if no architecture exists.

5.4.2.1 Computational node

A computational node is a combination of hardware and software components engineered to deliver local computing and processing capabilities at the edge of a manufacturing system. Its primary purpose is to execute manufacturing apps on site, ensuring faster processing times and minimising latency. The computational node is designed to facilitate seamless integration into existing manufacturing systems, necessitating minimal configuration and setup efforts. The schematic representation of a computational node is shown in Figure 5.7.

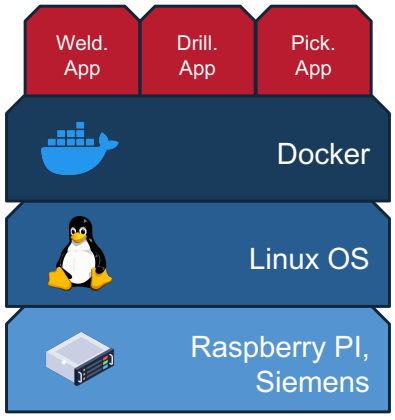

Figure 5.7: Schematic diagram of a computational node. A computational node hosts manufacturing apps in a containerised way.

5.4.2.2 Architecture manager

The architecture manager is a comprehensive software application that serves as a centralised management system for computational nodes within the manufacturing environment. Its primary purpose is to facilitate seamless oversight and administration of the manufacturing apps installed on these computational nodes.

The architecture manager empowers manufacturers to monitor and manage the deployment of manufacturing apps, allowing them to configure and tailor these applications according to their specific production requirements. Additionally, the architecture manager simplifies the maintenance process by enabling app developers to efficiently manage, update and troubleshoot their apps, ensuring continuous and reliable operation.

Additionally, the architecture manager incorporates advanced analytics and reporting tools that help manufacturers optimise their manufacturing processes. Manufacturers can make data-driven decisions to improve overall efficiency and productivity by providing actionable information. The schematic representation of the interaction of the architecture manager with computational nodes can be found in Figure 5.8.

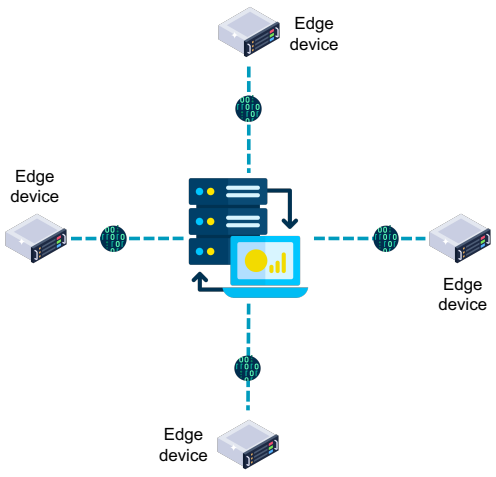

Figure 5.8: Schematic diagram of an architecture manager.

5.4.2.3 Global applications repository

The global applications repository serves as a digital marketplace that enables manufacturers to browse, evaluate, and download manufacturing apps tailored to address various manufacturing challenges. This repository is crucial to connecting app developers with manufacturers, fostering innovation, and driving the adoption of the latest manufacturing technologies.

Manufacturing apps available in the global applications repository are designed with modularity and reusability. This allows manufacturers to seamlessly integrate them into their existing production systems and customise the apps to meet their unique requirements, ultimately enhancing productivity and efficiency.

By providing a platform for app developers to showcase their cutting-edge solutions, the global applications repository broadens its reach, giving them access to a wider audience and the opportunity to contribute to the ongoing advancements in the manufacturing industry.

A schematic representation illustrating the interaction of the global application repository with manufacaturing apps is shown in Figure 5.9.

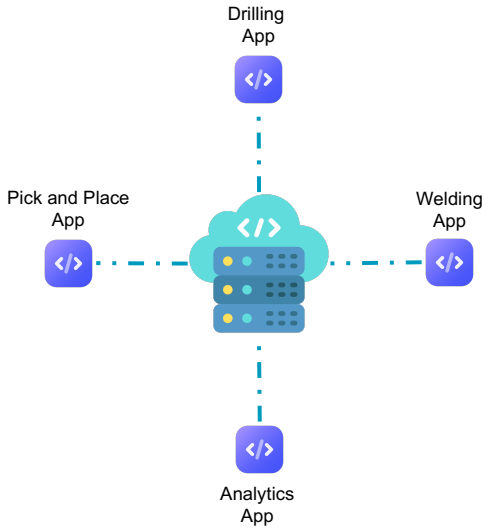

Figure 5.9: Global applications repository.

5.4.3 Conceptual architecture

After defining the requirements and essential components of a conceptual architecture, a schematic representation of the architecture to deploy manufacturing apps is illustrated in Figure 5.10.

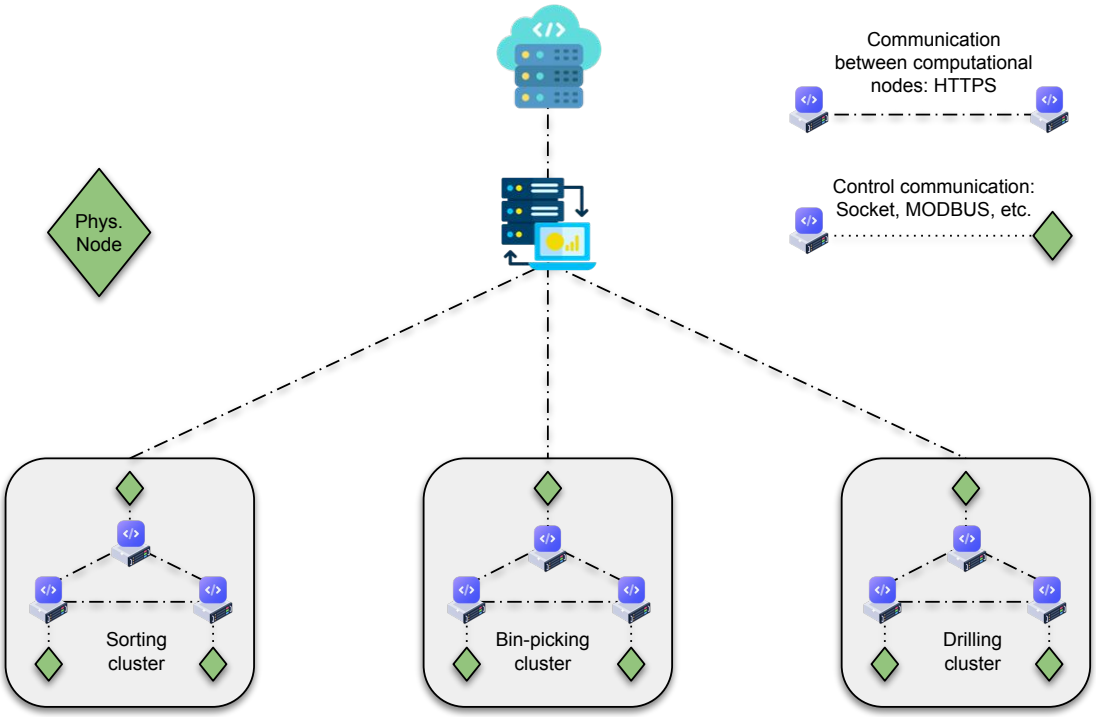

Figure 5.10: Conceptual architecture for deploying manufacturing apps.

Implementing manufacturing capabilities via the proposed architecture and manufacturing apps involves the following steps:

1. The computational node sends a request to the architecture manager for a specific manufacturing app.
2. The architecture manager searches the global applications repository for the requested manufacturing app.
3. The appropriate manufacturing app is retrieved from the repository and delivered to the computational node.
4. The computational node deploys the manufacturing app, enabling the requested capability on the corresponding manufacturing equipment.

For example, consider a bin-picking operation in which an industrial robot identifies and picks parts from a bin. The process using the above architecture would be as follows:

1. The computational node detects a change in the part to be picked and requests a new pose-detection app from the architecture manager.
2. The architecture manager searches the global applications repository for a suitable pose-detection app tailored for the new part.
3. The selected pose detection app is delivered to the computational node.
4. The computational node deploys the new app, allowing the industrial robot to detect and pick up the new part efficiently.

This example demonstrates the flexibility and adaptability of the proposed architecture, allowing for the seamless integration of new apps and updates to optimise manufacturing processes in response to changing requirements.

5.5 Chapter summary

This chapter addressed the practical implementation challenges of the MSC problem. The primary identified issue was the lack of interoperability between manufacturing equipment, which hinders the development of adaptable software solutions. The *ManufacturingApp* class was presented to overcome this challenge, based on the data model discussed in earlier chapters.

The concept of "manufacturing apps" was introduced and defined, and a theoretical and practical foundation was established for creating modular software solutions. The appification of manufacturing processes, the definition of a manufacturing app, and the abstract elements that make up such an app were explored,

enabling a comprehensive understanding of their design, implementation, and integration in various production environments.

A methodology for developing a manufacturing apps development kit was proposed, as current SDKs are vendor-specific and do not facilitate interoperability. The requirements for a development kit, the data model for the manufacturing apps development kit, and the industrial robot apps were outlined.

Lastly, a modular reference architecture was presented for deploying manufacturing apps. The requirements for a modular architecture, the elements of a conceptual architecture, and the conceptual architecture itself were defined.

The proposed object-oriented data model and the concept of "appification" of manufacturing processes have established a solid foundation to address the first two challenges of the MSC problem. An adaptable and flexible environment that can accommodate rapid changes in manufacturing requirements was created by transforming manufacturing operations into modular, manageable applications.

Building on this foundation, the next challenge to overcome is effectively integrating an optimisation and decision-making framework. This is the third and remaining challenge in developing a holistic solution to the MSC problem. The contributions provided in Chapter 4 and this chapter have made strides toward a modular implementation of optimisation and decision-making methods. This challenge will be addressed in depth in the following chapter, rounding out a comprehensive approach to tackling the complexities of the MSC problem.

Chapter 6

Optimisation and decision-making algorithms for MSC problem

6.1 Introduction

This chapter addresses the third challenge of developing a holistic, integrated optimisation and decision-making solution for rapidly changing manufacturing requirements. The remaining challenge is that the manufacturing costs and objectives constantly change with the change in requirements, which makes it difficult to adapt to new requirements cost-effectively. Solving this challenge requires developing robust and modular algorithms that can exploit the underlying data model and respond to changes as they occur.

Therefore, following the research methodology of Chapter 3 and building on top of the research contributions provided in Chapter 4 and Chapter 5, this chapter aims to address the following research question and research hypothesis:

- **Research question 3:** What types of algorithms can effectively manage uncertainties and facilitate sequential decision-making in the dynamic and evolving multi-objective environment of manufacturing processes?
- **Research hypothesis 3:** An optimisation and decision-making framework that integrates traditional optimisation and machine learning algorithms can effectively manage uncertainties and facilitate sequential decision-making in dynamic, multi-objective manufacturing environments.

This chapter proposes three important modules to address the above research question and hypothesis. Consequently, in this chapter three main research contributions are made, which are discussed in detail in the following sections.

The first contribution of this chapter is the development of a module that uses the data model presented in Chapter 4 to identify changes in the manufacturing environment. The changes identification algorithm (CIDENA) is introduced and its functioning is detailed and demonstrated, showcasing its effectiveness in detecting

changes.

The second contribution is the introduction of an optimisation module that employs an optimisation matrix to determine the necessary type of optimisation, methods, and algorithms. The initial step in optimisation is to optimise the manufacturing equipment for a specific objective. Therefore, a novel meta-algorithm called online optimisation of operational parameters (O3PARAMS) is presented to optimise the operational parameters of the manufacturing equipment. Subsequently, the optimisation for capability and capacities is discussed.

Given the stochastic nature of the manufacturing environment and the constant need for adaptation, the third contribution is a reinforcement learning (RL)-based decision-making module that utilises optimised equipment and manufacturing configurations. A novel manufacturing RL-based decision-making environment is presented, designed to effectively meet changing manufacturing demands.

The contributions provided in this chapter are peer-reviewed and disseminated in the following venues:

1. Torayev et al. [146] "Online and Modular Energy Consumption Optimization of Industrial Robots". Journal publication in IEEE Transactions on Industrial Informatics.

2. Torayev et al. [147] "Optimal Manufacturing Configuration Selection: Sequential Decision Making and Optimization using Reinforcement Learning", 56th CIRP Manufacturing Systems Conference. October 24-26, 2023, Cape Town, South Africa.

3. "MFG-RL Manufacturing Reinforcement Learning Environment"[1] software package and library distributed under Apache-2.0 licence and hosted on GitHub platform.

[1] https://github.com/torayeff/mfgrl

The general interaction between the proposed modules and how they build on top of the object-oriented data model is illustrated in Figure 6.1.

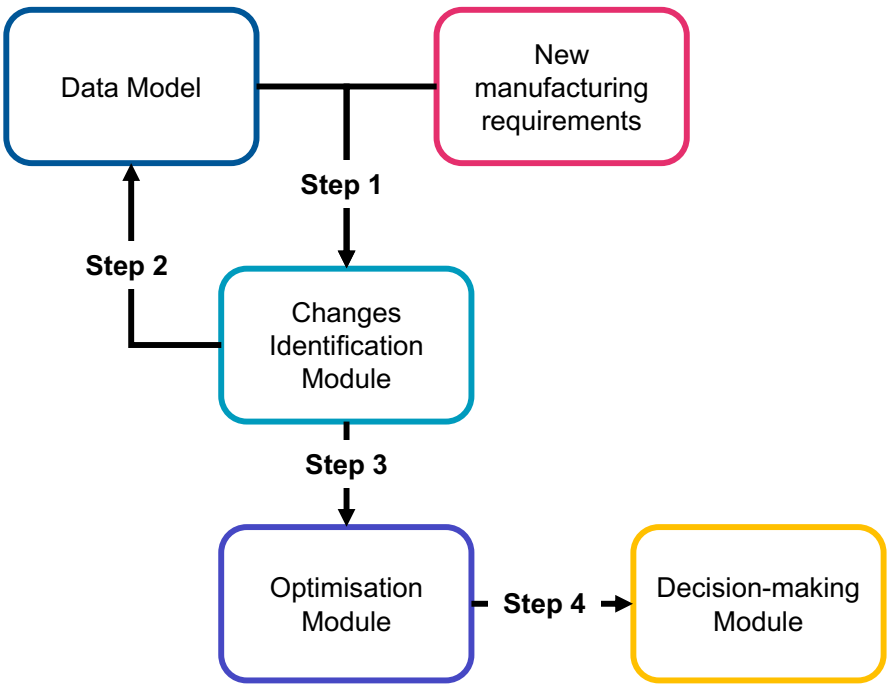

Figure 6.1: Interaction between changes identification, optimisation, and decision-making modules.

The process begins with inputting the existing data model and the new manufacturing requirements into the changes identification module. This module identifies the necessary changes and updates the data model accordingly. Next, the output from the changes identification module is fed into the optimisation module, which selects the appropriate algorithms and performs the optimisation. Finally, the output from the optimisation module is provided to the decision-making module, which effectively addresses stochastic and uncertain factors in the manufacturing environment.

These modules are designed to be robust in a dynamic manufacturing environment with stochastic fluctuations and rapidly evolving requirements. An important aspect of the proposed modules is that they are appified as was proposed in Chapter 5. This appification process is especially necessary for utilising optimisation algorithms on the fly. The next sections of this chapter go into the details of the above contributions.

6.2 Changes identification module

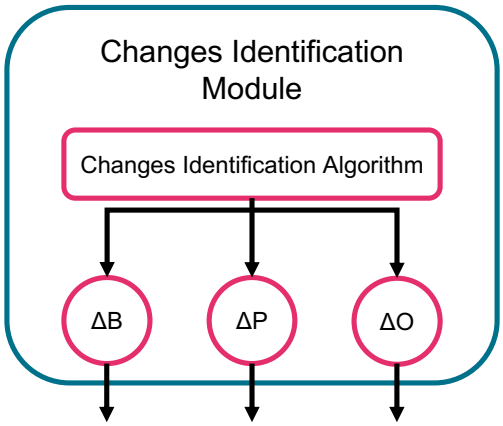

Figure 6.2: Changes Identification Module

The initial step in optimising manufacturing configurations to adapt to changing requirements involves identifying the necessary changes. Depending on the existing or empty manufacturing setup, the optimisation process must identify the required modifications to meet the new manufacturing requirements.

As discussed in Chapter 4, changes in manufacturing requirements can be systematically classified into three primary categories:

- Requirement for capability change: ΔB
- Requirement for capacity change: ΔP
- Requirement for operational parameter change: ΔO

The changes identification module, depicted in Figure 6.2, aims to autonomously determine which of the above requirements must be addressed. The implementation of this module relies on the data model and its mathematical formalisation presented in Chapter 4.

6.2.1 Changes identification algorithm

A core component of the changes identification module is the changes identification algorithm (CIDENA), presented in Algorithm 1. It is designed to compare two consecutive requirement periods, t and $t+1$, and to pinpoint the necessary modifications in manufacturing configurations.

Algorithm 1 changes identification algorithm (CIDENA)

Require: Requirement periods D_t, D_{t+1}
Ensure: $\Delta B_t, \Delta P, \Delta O$
for each $(B_{t+1,i}, P_{t+1,i})$ in D_{t+1} **do**
  if $B_{t+1,i}$ in D_t **then**
    $(B_{t,j}, P_{t,j}) \leftarrow D_{t,j}$
    if $P_{t,j} \geq P_{t+1,i}$ **then**
      $P_{t,j} \leftarrow P_{t,j} - P_{t+1,i}$
      $P_{t+1,i} \leftarrow 0$
    else
      $P_{t+1,i} \leftarrow P_{t+1,i} - P_{t,j}$
      $P_{t,j} \leftarrow 0$
    end if
  end if
end for
Filter out nonzero capacity values:
  $\Delta B_t, \Delta P_t \leftarrow [(B_{t,i}, P_{t,i})$ for i in $length(D_t)$ if $P_{t,i} > 0]$
  $\Delta B_{t+1}, \Delta P_{t+1} \leftarrow [(B_{t+1,i}, P_{t+1,i})$ for i in $length(D_{t+1})$ if $P_{t+1,i} > 0]$
Assign to ΔB and ΔP:
  $\Delta B \leftarrow [\Delta B_t, \Delta B_{t+1}]$
  $\Delta P \leftarrow [\Delta P_t, \Delta P_{t+1}]$
  $\Delta O \leftarrow False$
if $length(\Delta B_t) = 0$ and $length(\Delta B_{t+1}) = 0$ **then**
  "The new requirement period can be satisfied without any change."
else
  $\Delta O \leftarrow True$
  if $length(\Delta B_t) \neq 0$ **then**
    "Some configurations can be removed: $\Delta B_t, \Delta P_t$"
  end if
  if $length(\Delta B_{t+1}) \neq 0$ **then**
    "New configurations are needed: $\Delta B_{t+1}, \Delta P_{t+1}$"
  end if
end if
return $\Delta B, \Delta P, \Delta O$

The algorithm takes two requirement periods, D_t and D_{t+1}, as input and outputs the changes in capabilities (ΔB), capacities (ΔP), and operational parameters

(ΔO). It iteratively processes each capability-capacity pair in the requirement period D_{t+1} (Line 1). If the capability is present in the requirement period D_t (Line 2), the algorithm checks if the capacity in D_t is greater than or equal to the capacity in D_{t+1} (Line 4). If true, the capacity in D_t is reduced by the capacity in D_{t+1} (Line 5), and the capacity in D_{t+1} is set to 0 (Line 6). Otherwise, the capacity in D_{t+1} is reduced by the capacity in D_t (Line 8), and the capacity in D_t is set to 0 (Line 9).

After processing each pair, the algorithm filters out non-zero capacity values for both D_t and D_{t+1} (Lines 13-15). It then initialises ΔB, ΔP, and ΔO (Lines 16-19). ΔB is assigned to the filtered non-zero capability values for D_t and D_{t+1}, while ΔP is assigned to the corresponding filtered non-zero capacity values. ΔO is initialised as False.

The algorithm then checks various conditions based on the lengths of ΔB_t and ΔB_{t+1} to determine if any changes are needed (Line 20). If both lengths are 0, the new requirement period can be satisfied without any change. Otherwise, ΔO is set to True (Line 23). If the length of ΔB_t is non-zero, it indicates that some configurations can be removed, and the algorithm outputs the specific configurations and capacities to be removed (Lines 24-26). If the length of ΔB_{t+1} is non-zero, new configurations are needed (Lines 27-29).

The algorithm returns the values of ΔB, ΔP, and ΔO (Line 31), representing the identified changes in the capabilities, capacities, and operational parameters between the two consecutive requirement periods.

The CIDENA algorithm can be considered a tool to identify changes in the context of the MSC problem. As discussed in Chapter 5, the CIDENA algorithm can be appified and integrated into a modular architecture.

6.2.2 Demonstration of changes identification algorithm

In order to see the CIDENA algorithm in action, it is helpful to consider a simple demonstrative example.

Let there be two consecutive requirement periods D_t and D_{t+1} with different manufacturing requirements according to the data model in Figure 4.2 and Equation 3.7. The capabilities and capacities required for the requirement period D_t are given as:

$$\begin{aligned} D_t = [&(B_{t,1} = \text{Pick\&Place}, P_{t,1} = 20), \\ &(B_{t,2} = \text{Drilling}, P_{t,2} = 10)] \end{aligned} \tag{6.1}$$

Required capabilities and capacities for the requirement period D_{t+1} are given as:

$$\begin{aligned} D_{t+1} = [&(B_{t+1,1} = \text{Pick\&Place}, P_{t+1,1} = 15), \\ &B_{t+1,2} = \text{Drilling}, P_{t+1,2} = 5), \\ &(B_{t+1,3} = \text{Welding}, P_{t+1,3} = 10)] \end{aligned} \tag{6.2}$$

The CIDENA algorithm processes these two requirement periods as follows:

1. Iterate through each pair in D_{t+1}:
   (a) For $(B_{t+1,1}, 15)$, $B_{t,1}$ is in D_t. Since $P_{t,1} = 20 \geq 15$, update the capacities as follows: $P_{t,1} = 20 - 15 = 5$ and $P_{t+1,1} = 0$.
   (b) For $(B_{t+1,2}, 5)$, B_2 is in D_t. Since $P_{t,2} = 10 \geq 5$, update the capacities as follows: $P_{t,2} = 10 - 5 = 5$ and $P_{t+1,2} = 0$.
   (c) $B_{t+1,3}$ is not in D_t, so do not make any changes.
2. Filter out non-zero values:
   (a) $\Delta B_t = [B_{t,1}, B_{t,2}]$, $\Delta P_t = [5, 5]$

(b) $\Delta B_{t+1} = [B_{t+1,3}]$, $\Delta P_{t+1} = [10]$

3. Assign to ΔB and ΔP:

   (a) $\Delta B = [\Delta B_t, \Delta B_{t+1}]$

   (b) $\Delta P = [\Delta P_t, \Delta P_{t+1}]$

4. Initialize $\Delta O = False$.

5. Check the conditions based on the lengths of ΔB_t and ΔB_{t+1}:

   (a) Both lengths are non-zero, so the algorithm suggests removing configurations in ΔB_t for efficiency: $(B_{t,1}, 5)$ and $(B_{t,2}, 5)$.

   (b) The algorithm also suggests adding the configurations in ΔB_{t+1} to meet demand: $(B_{t+1,3}, 10)$.

   (c) Set $\Delta O = True$

The CIDENA algorithm returns $\Delta B = [[B_{t,1}, B_{t,2}], [B_{t+1,3}]]$, $\Delta P = [[5, 5], [10]]$ and $\Delta O = True$ for the above example. It means that the manufacturing setup needs changes in capabilities, capacities, and operational parameters to accommodate the new requirement period. It also requires decreasing the capacity of $B_{t,1}$ and $B_{t,2}$ and introducing a new manufacturing capability $B_{t+1,3}$ with a capacity of 10.

Consequently, these changes require updating the data model by adding or removing the necessary manufacturing assets and modifying the relationships between assets, manufacturing configurations, and manufacturing configuration groups in Figure 4.1.

Once the necessary changes in capabilities, capacities and operational parameters are identified using the CIDENA algorithm, these are passed to the optimisation module, as discussed in the next section.

6.3 Optimisation module

After identifying the necessary changes using the CIDENA algorithm, the optimisation and decision-making process goes to the optimisation module. The optimisation module selects the appropriate algorithms to address the identified changes in the capabilities, capacities, and operational parameters. Then, it generates solutions for the decision-making module as shown in Figure 6.3.

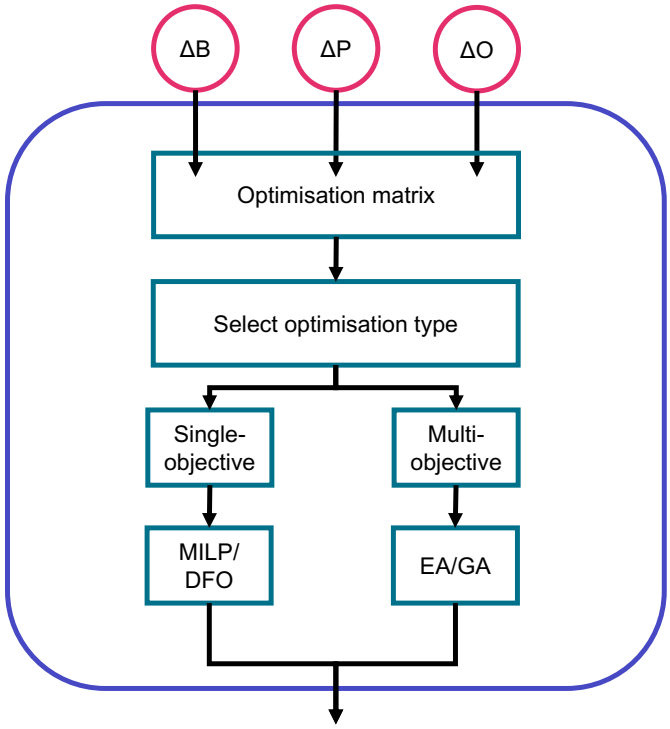

Figure 6.3: Schematic of Optimisation Module

The optimisation module consists of different parts, such as the optimisation matrix, the selection of the optimisation type, and the manufacturing-specific algorithms, as shown in Figure 6.3.

The next subsections of this section present a detailed description of each element of the optimisation module, starting from the optimisation matrix.

6.3.1 Optimisation matrix for changing manufacturing requirements

The CIDENA algorithm, although highly effective in recognising the necessary modifications in capabilities, capacities, and operational parameters, does not inherently provide a structured plan for the execution of these changes.

Here the optimisation matrix comes in. It acts as a decision-making guide, categorising the outputs of the CIDENA algorithm into four distinct scenarios based on the presence or absence of capability changes at different times, thus facilitating the selection of the most appropriate optimisation strategies. Each of these cases represents a unique situation that requires a specific response, from no changes needed to complete reconfiguration of the current manufacturing system.

It should be noted that the matrix focuses on changes in capabilities as shifts in capacities and operational parameters inherently imply changes in capabilities. Hence, the optimisation matrix streamlines the adaption to new manufacturing requirements by enabling systematic decision-making based on the identified need for change.

These four possible cases are shown in Table 6.1 and described below:

	$\Delta B_{t+1} = Null$	$\Delta B_{t+1} \neq Null$
$\Delta B_t = Null$	**Case 1.** No changes are needed.	**Case 3.** New capabilities are required with ΔP_{t+1} capacities and ΔO operational parameters.
$\Delta B_t \neq Null$	**Case 2.** Some configurations can be removed for reducing recurring costs.	**Case 4.** Complete reconfiguration of the current manufacturing system is required.

Table 6.1: Optimisation matrix for changing manufacturing requirements. This matrix guides the optimisation module for making the necessary changes and selecting optimisation algorithms.

1. **Case 1** is the most trivial case. In this case, the current manufacturing configurations do not require any changes or optimisation. For example, if an aerospace manufacturing company produces 10 aircraft per month and the demand remains constant, there is no need for adjustments in the manufacturing process. This case does not necessitate any optimisation.

2. **Case 2** occurs when current manufacturing configurations can meet the new requirements, i.e., the necessary capabilities and capacities already exist. However, some assets might be underutilised and therefore can be removed for efficiency or allocated to other manufacturing operations. For example, suppose that an aerospace manufacturing company produces 10 aircraft per month and the demand decreases to 8. In that case, they can remove or reallocate some machinery, workforce, or production lines to reduce costs and increase efficiency. This case typically requires manual removal of the necessary assets and might necessitate optimisation for utilisation.

3. **Case 3** occurs when current manufacturing configurations cannot meet the new requirements, but some of the necessary capabilities and capacities already exist. However, additional capabilities and capacities are required. It can also be the case that the demand for production increases. For example, an aerospace manufacturing company produces commercial aircraft and faces a new demand for freight aircraft with specific technology. Alternatively, it might be the case that the demand for commercial aircraft increases. They already have the equipment to manufacture the other components but need to add new machinery and facilities to produce the new specifications or meet the increased demand. This case also includes optimising operational parameters.

4. **Case 4** happens when the current manufacturing configurations completely differ from the new manufacturing requirements. This can occur when the demand for a new product is entirely distinct. For example, an aerospace

manufacturing company specialising in producing commercial aircraft faces a surge in demand for satellites or unmanned aerial vehicles (UAVs). This change in demand would require a complete reconfiguration of the manufacturing system to accommodate satellite or UAV production, including new machinery, workforce training, and potentially different suppliers. This case requires a complete reconfiguration and the selection of new optimal manufacturing configurations.

It is crucial to note that while the optimisation matrix can be appified as a subsequent step of the CIDENA algorithm, it is fundamentally designed with a human decision-maker in mind. This design consideration stems from the fact that the identified changes often require a physical reconfiguration, a process that typically requires human oversight, expertise, and discretion. In essence, the optimisation matrix bridges automated analysis and human-centric decision-making, merging both strengths to achieve efficient and effective adaptation in the manufacturing setup.

6.3.2 Algorithms selection for manufacturing optimisation

Once the optimisation cases are identified using an optimisation matrix, different optimisation approaches might be needed depending on the specific requirements of each case. The necessary optimisation approaches can be broadly classified into two categories. For each category, this work recommends algorithms based on their effectiveness and suitability in the manufacturing context, as shown in Figure 6.3 and described below.

6.3.2.1 Single-objective optimisation

This approach focuses on optimising one objective function, such as minimising cost or maximising efficiency. It is suitable for cases with only one primary goal or when multiple objectives can be combined into a single function.

For single-objective optimisation, a mixed-integer linear programming (MILP) and derivative-free optimisation (DFO) algorithms are recommended. MILP is particularly useful when dealing with problems that involve discrete variables and linear constraints, while DFO is suitable for problems where gradient information is unavailable or unreliable.

For example, in Case 2, a manufacturing company might want to minimise the recurring costs associated with underutilised assets, such as machinery or workforce. This could involve optimising the layout of the factory to minimise transportation costs or adjusting the production schedule to maximise the utilisation of available resources. In this scenario, a single-objective optimisation algorithm like MILP or DFO could be used to find the optimal solution.

6.3.2.2 Multi-objective optimisation

In this approach, multiple objective functions are considered simultaneously to find solutions that provide an optimal trade-off between the conflicting objectives. This is particularly useful when addressing problems with multiple conflicting goals, such as investments costs, recurring costs, and transition costs as discussed in Chapter 3.2.1.3.

For multi-objective optimisation problems, evolutionary algorithms (EA) and genetic algorithms (GA) are recommended. These algorithms are well-suited for handling complex problems with multiple conflicting objectives, as they can effectively explore the solution space and identify trade-offs between the objectives.

In Case 3, for example, a manufacturing company might need to add new machinery and facilities to produce a new product or meet increased demand. This would require the company to find a balance between the costs of acquiring and installing new equipment, the potential increase in recurring costs due to additional resources, and the benefits of satisfying the new demand. In this situation, a multi-objective optimisation algorithm like EA or GA could be used to identify the optimal trade-offs between these conflicting objectives.

Similarly, in Case 4, a complete reconfiguration of the manufacturing system might be necessary to accommodate a shift in demand. This could involve significant investments in new machinery, workforce training, and changes in the supply chain. The company would need to weigh these costs against the potential benefits of capturing the new market. Again, a multi-objective optimisation algorithm such as EA or GA could be applied to find the best compromise between these conflicting objectives.

The proposed methodology for selecting optimisation algorithms remains grounded in empirical validation but provides flexibility for incorporating any optimisation algorithm in theory. The recommended algorithms, as shown in Figure 6.3, are preferred based on proven efficacy in addressing similar problems within the manufacturing industry based on experimental work carried out in this research.

Following a clear definition of the methodology for choosing optimisation algorithms, the process advances to actual optimisation for optimal manufacturing configurations selection.

Optimisation generally divides into two primary requirements: first, the selection of manufacturing configurations to meet requested capabilities, and second, those to fulfil requested capacities. However, the first step in understanding which configurations to select involves finding the optimal operational parameters of manufacturing equipment or configuration, which is the goal of the next section.

6.3.3 Optimisation of operational parameters of manufacturing equipment

In a dynamic manufacturing environment, it is crucial to adapt manufacturing configurations to accommodate changing requirements in capabilities and capacities. A comprehensive understanding of the maximum limit and optimal parameters of individual manufacturing equipment is essential to achieve this optimisation because equipment performance directly influences the overall manufacturing process's efficiency and cost-effectiveness.

Consequently, optimising operational parameters becomes a prerequisite for effectively optimising capabilities and capacities within a manufacturing configuration group. However, several optimisation challenges persist in the manufacturing domain, which must be addressed to ensure the effective implementation of optimisation techniques:

- **Exact modelling of manufacturing equipment is difficult:** The manufacturing process typically involves numerous complex machines, tools, and materials. Accurately modelling the interactions and dependencies between these components is challenging due to the vast number of variables, nonlinearities, and changing conditions involved in the manufacturing process.
- **Noisy measurements and uncertainties:** Manufacturing processes often involve uncertainties related to material properties, tool wear, and environmental factors. Furthermore, measurement systems may introduce noise to the data, which can negatively impact the accuracy of optimisation algorithms. Addressing the mentioned problems requires developing robust optimisation techniques to handle noisy and uncertain data.
- **Data efficiency is required:** Collecting large volumes of data from manufacturing processes can be time-consuming and costly. Therefore, it is

essential to develop data-efficient optimisation algorithms that can identify optimal operational parameters using limited data.

Considering the above challenges, this chapter proposes a meta-algorithm online optimisation of operational parameters (O3PARAMS), shown in Algorithm 2, designed to iteratively optimise the operational parameters of manufacturing equipment, taking into account the challenges specific to the manufacturing domain and tailored to optimise the operational parameters of manufacturing equipment.

A meta-algorithm is an algorithm that can accept other algorithms as input, effectively extending or enhancing the capabilities of the input algorithms. In the case of O3PARAMS, it accepts an optimisation algorithm, denoted as Φ, as input.

Algorithm 2 Meta algorithm for online optimisation of operational parameters (O3PARAMS)

Require: An initial guess of operational parameters $\mathbf{x}_0$, the lower bound $\mathbf{l}$, the upper bound $\mathbf{u}$, the total number of iterations n, the exploration parameter α, an optimisation algorithm Φ, a data-structure for storing dataset $\mathcal{D}$

$y_0 \leftarrow O(\mathbf{x}_0)$ ▷ Measurements: energy consumption, cycle-time, etc.
$y_{best} \leftarrow y_0$ ▷ Best values so far
$\mathbf{x}_{best} \leftarrow \mathbf{x}_0$
$\mathcal{D} \leftarrow (\mathbf{x}_0, y_0)$ ▷ Set the initial dataset
$i \leftarrow 1$
while $i < n$ **do**
    Sample p from uniform distribution $U(0, 1)$
    if $p \leq \alpha$ **then**
        $i \leftarrow i + 1$
        $\mathbf{x}_i \leftarrow \Phi(i, \mathbf{l}, \mathbf{u}, \mathcal{D})$ ▷ Call the optimisation step
        $y_i \leftarrow O(\mathbf{x}_i)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, y_i)$ ▷ Extend the dataset
        if $y_i < y_{best}$ **then**
            $\mathbf{x}_{best} \leftarrow \mathbf{x}_i$
            $y_{best} \leftarrow y_i$
        end if
    end if
end while
return $\mathbf{x}_{best}, y_{best}, \mathcal{D}$

The O3PARAMS works as follows:

1. **Initialisation:** The algorithm begins by taking an initial guess of opera-

tional parameters ($\mathbf{x}_0$), lower ($\mathbf{l}$) and upper ($\mathbf{u}$) bounds, the total number of iterations (n), an exploration parameter (α), an optimisation algorithm (Φ), and a data structure for storing dataset ($\mathcal{D}$).

2. **First measurement:** It then measures the objective function ($O(\mathbf{x}_0)$) (Line 1) with the initial guess, which can include factors such as energy consumption, cycle time, etc.

3. **Iterative optimisation:** The algorithm runs through a loop for a predefined number of iterations (n) (Line 6). Each iteration samples a probability (p) (Line 7) from a uniform distribution and compares it with the exploration parameter (α). If the probability is less than or equal to the exploration parameter (Line 8), it calls the chosen optimisation algorithm (Φ) to find a new set of operational parameters ($\mathbf{x}_i$) (Line 10).

4. **Evaluation and update:** The algorithm evaluates the new operational parameters by calculating their objective function ($y_i = O(\mathbf{x}_i)$) (Line 11) and updates the dataset with the new data point. If the new objective function value is better than the current best value, the best operational parameters and their associated objective function value are updated (Line 13-16).

5. **Completion:** Once the algorithm has reached the specified number of iterations, it returns (Line 19) the best operational parameters, the best objective function value, and the dataset collected during the optimisation process.

The O3PARAMS algorithm brings the flexibility to optimise individual manufacturing equipment and a combination thereof, thanks to the black-box modelling methodology. The singular prerequisite of this algorithm is the measurability of the key performance indicator (KPI) to be optimised.

Upon optimising manufacturing configurations for maximum KPI, these configurations qualify as potential candidates for subsequent optimisation stages. These

include meeting required capabilities and capacities, which are further discussed in the following section.

6.3.4 Optimisation for capability and capacity

After fine-tuning the manufacturing equipment for optimal operational parameters, the next step is to optimise for capability, capacity, or both based on the requirements and identified changes.

Optimising capability and capacity involves multiple conflicting objectives, such as investment, recurring, and transition costs. Each cost can be further decomposed into separate dimensions as shown and modelled in Chapter 4. As a result, this process requires multi-objective optimisation to balance the competing objectives effectively.

Essentially, optimising for capabilities and capacities requires solving the problem defined in Equation 3.10 in section 3.2.1.4. There are two general methods for solving optimisation problems in this context:

- **Weighted sum scalarisation**
- **Pareto front optimisation**.

6.3.4.1 Weighted sum scalarisation optimisation

Weighted sum scalarisation is a method for solving multi-objective optimisation problems by converting them into single-objective problems. It does this by assigning a weight to each objective and then combining them into a single objective function. This method can be applied to the manufacturing requirements satisfaction problem described above. The steps for implementing weighted sum scalarisation are as follows:

1. Assign weights to the objectives: Assign a weight w_i to each of the objectives of the problem, that is, investment cost, cost of the requirement period and cost of reconfiguration. These weights must be non-negative and sum to 1, that is, $\sum_{i=1}^{3} w_i = 1$.

2. Combine the objectives: Form a single objective function by combining the weighted objectives as follows:

$$Z(x) = \sum_{t=1}^{T} (w_1 I(x_t) + w_2 L_t R(x_t) + w_3 Q(x_t, x_{t+1})) \tag{6.3}$$

3. Solve the single-objective problem: Minimise the combined objective function $Z(x)$ subject to the capacity-satisfaction constraint (Equation (3.11)) and the integrality constraint (Equation (3.12)). This can be done using traditional single-objective optimisation techniques, such as linear programming or mixed-integer programming, depending on the nature of the decision variables.

6.3.4.2 Pareto front optimisation

Pareto front optimisation is another method for solving multi-objective optimisation problems. Unlike weighted sum scalarisation, this method does not require the assignment of weights to objectives. Instead, it seeks to find a set of non-dominated solutions that represent trade-offs between the multiple objectives. A solution is considered non-dominated if it is not worse in all objectives compared to any other solution. The steps for implementing Pareto front optimisation are as follows:

1. Generate an initial set of solutions: Start with an initial set of feasible solutions which can be obtained using heuristics, random sampling, or other methods.

2. Evaluate the solutions: Calculate the objective values for each solution in the initial set using the investment cost, recurring cost, and transition cost functions.

3. Identify non-dominated solutions: For each solution in the initial set, compare it with all other solutions. A solution is considered non-dominated if there is no other solution that is better in all objectives.

4. Generate new solutions: Create new solutions by applying search and optimisation techniques, such as genetic algorithms, simulated annealing, or local search. These techniques typically involve modifying existing solutions to generate new ones that are then added to the set of solutions.

5. Update the Pareto front: Re-evaluate the new solutions and update the Pareto front by adding any new non-dominated solutions and removing any solutions that are now dominated.

6. Iterate: Repeat steps 4 and 5 until a stopping criterion is met, such as a maximum number of iterations or a convergence threshold.

The final Pareto front represents a set of trade-offs between the multiple objectives, allowing decision-makers to choose a solution based on their preferences.

The output from the weighted sum scalarisation or Pareto front optimisation is used as input to a decision-making module to further improve the adaptability and efficiency of the manufacturing system, especially in the face of stochastic events and rapidly changing manufacturing requirements.

The decision-making module learns and adapts to changes in the manufacturing environment and requirements dynamically, enabling the system to cope with uncertainties and fluctuations in demand. Therefore, the next section looks at the final module, the decision-making module.

6.4 Decision-making module

The optimisation module generates optimal manufacturing configurations based on varying capabilities, capacities, and operational parameters. However, the optimal solutions derived might require an additional decision-making step facilitated by a human expert or automated processes. This is predominantly the case when employing multi-objective optimisation techniques, as single-objective optimisation algorithms typically produce outright optimal solutions that do not need further evaluation.

Therefore, to present the decision-making process coherently, first, existing MCDM methods are discussed that were put into empirical use in this research, verifying their efficiency. Following this, a RL-based framework is proposed, which presents a novel approach to decision-making in optimisation for optimal selection of manufacturing configurations. The overall decision-making module, which consists of both MCDM and RL-based decision-making, is shown in Figure 6.4.

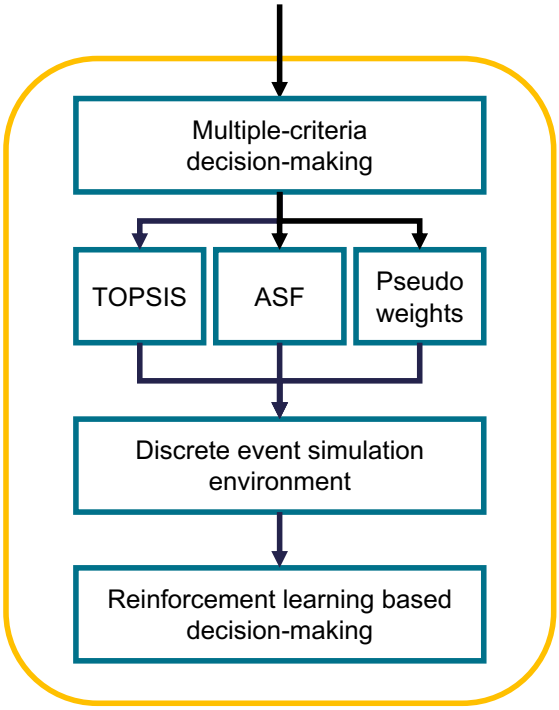

Figure 6.4: Decision-making module

6.4.1 Multiple-criteria decision-making methods

The decisions in the context of the MSC problem often involve considering multiple criteria, requiring robust decision-making methods. Three approaches that have proven effective in this context are technique for order of preference by similarity to ideal solution (TOPSIS), achievement scalarisation function (ASF), and pseudo-weights (PW). Although many multiple-criteria decision-making (MCDM) methods exist, TOPSIS, ASF, and PW were specifically chosen due to their ability to minimise subjectivity and provide robust, quantitative decision-making frameworks suited for complex engineering tasks such as manufacturing configuration selection.

6.4.1.1 TOPSIS

The TOPSIS method involves determining the ideal and negative-ideal solutions, calculating the Euclidean distance of each alternative to these solutions, and computing the relative closeness to the ideal solution. The alternative with the highest relative closeness is considered the best.

$$d_i^+ = \sqrt{\sum_{j=1}^{n}(r_{ij} - r_j^+)^2} \tag{6.4}$$

$$d_i^- = \sqrt{\sum_{j=1}^{n}(r_{ij} - r_j^-)^2} \tag{6.5}$$

$$CC_i = \frac{d_i^-}{d_i^+ + d_i^-} \tag{6.6}$$

where d_i^+ and d_i^- are the distances of the alternative i from the positive ideal solution and the negative ideal solution, respectively. r_{ij} denotes the normalized

value of criterion j for alternative i. r_j^+ and r_j^- denote the ideal and negative-ideal solutions of criterion j, respectively. CC_i is the relative closeness of alternative i to the ideal solution.

6.4.1.2 ASF

The ASF method converts a multi-objective optimisation problem into a single-objective one. The ASF for an alternative x is given by:

$$\max_{j \in J} \left(\frac{f_j(x) - z_j^*}{\lambda_j} \right) \tag{6.7}$$

where $f_j(x)$ is the value of objective function j for alternative x, z_j^* is the ideal solution for objective function j, and λ_j is the weight of objective function j. The decision-maker seeks to minimize this function to find the best solution.

6.4.1.3 Pseudo-Weights

PW offer a compromise solution for multi-objective optimisation problems. PW measure the relative importance of each objective and assist in generating a desirable trade-off surface.

$$\omega_j^* = \frac{f_j(x^*) - f_j^w}{f_j^* - f_j^w} \tag{6.8}$$

where $f_j(x^*)$ is the value of objective function j for the optimal solution x^*, f_j^w is the worst possible value of objective function j, and ω_j^* is the pseudo-weight of objective function j.

6.4.2 RL-based decision-making

The final step in decision-making for optimal manufacturing configuration selections is validating the robustness of the chosen solutions by MCDM methods. For this purpose, this thesis proposes using RL-based decision-making. The inherent capability of RL to handle complex environments characterised by stochasticity, uncertain events, and long-term planning under uncertainty makes it a suitable tool for verifying the efficacy and resilience of MCDM-based selections in real-world manufacturing scenarios.

6.4.2.1 Components of RL

RL-based decision-making consists of several inherent components. The interaction between these components is shown in Figure 6.5 and described below:

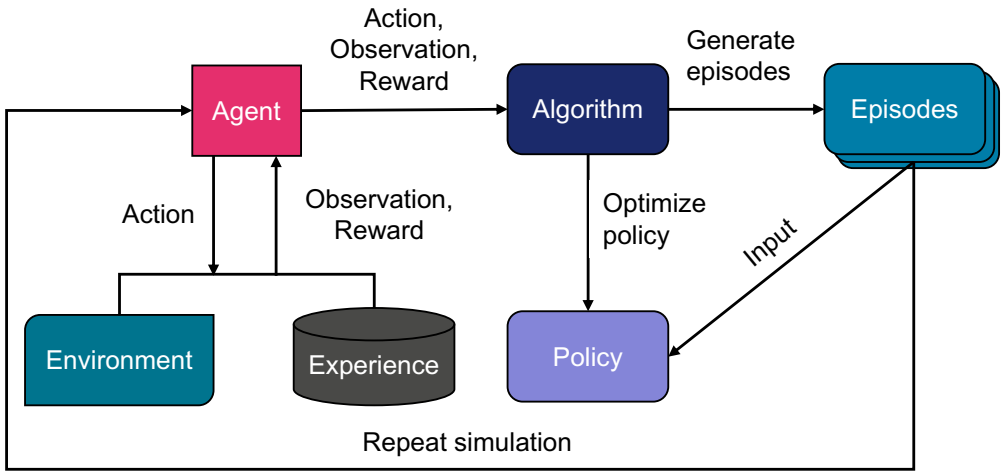

Figure 6.5: Interaction of components in RL.

- **Agent.** The agent is the decision-maker in the RL framework. It interacts with the environment, takes actions based on its current state, and learns from the feedback (rewards or penalties) received.
- **Environment.** The environment is the context in which the agent operates. It responds to the agent's actions by presenting a new state and a reward

or penalty. In manufacturing, the environment could be a production line, a warehouse, or any other system in which decisions need to be made.

- **Experience.** Experience refers to the sequence of states, actions, and rewards that an agent encounters over time. Through experience, the agent learns which actions yield the highest rewards in various states.

- **Algorithm.** The algorithm is the method by which the agent learns from its experience. There are numerous RL algorithms, each with its own approach to learning from experiences and updating the agent's policy, such as Q-Learning[148], Deep Q-Network (DQN)[149], and Proximal Policy Optimization (PPO)[150].

- **Policy.** The policy defines the behavior of the agent. It is a mapping from states to actions, directing the agent's actions at each state. The goal of the RL process is to find the optimal policy that maximises the expected cumulative reward.

- **Episodes.** Episodes represent a complete sequence of the agent interacting with the environment, from an initial state to a terminal state. Each episode provides a batch of experiences from which the agent can learn and improve its policy.

6.4.3 Manufacturing RL decision-making environment

Upon defining the components of RL, a novel manufacturing reinforcement learning environment (MFGRL) was developed. The primary objective of this environment is to provide a discrete-simulation environment for assessing the robustness of the selected solutions by replicating real decision-making scenarios as accurately as possible. The user interface of the MFGRL environment is displayed in Figure 6.6.

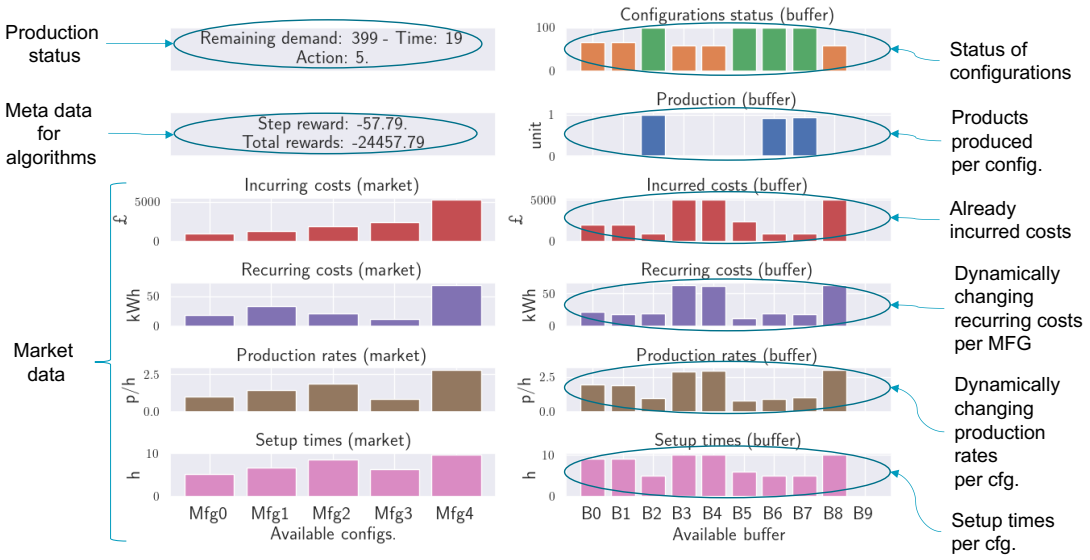

Figure 6.6: MFGRL Environment

The MFGRL environment comprises the following elements:

- **Production status:** This represents crucial details about the manufacturing process, such as:
  - *Remaining demand*: Indicates the remaining quantity of products required.
  - *Remaining demand time*: Denotes the remaining time to fulfil the demand.
- **Meta data for algorithms:** This provides information to evaluate the performance of the RL algorithm, such as:
  - *Step reward*: Indicates the reward received at the current step.
  - *Total reward*: Accumulates the rewards received over all steps.
- **Market data:** This depicts the manufacturing configurations and equipment available in the market. It provides various cost factors and production rates, all subject to stochastic changes or tied to real data.
  - *Incurring costs*: Also known as investment costs, these represent the

costs involved in purchasing new manufacturing configurations or equipment.

  - *Recurring costs*: Ongoing costs, like energy expenditures, associated with operating the manufacturing configurations.
  - *Production rates*: Denote the manufacturing capacities of the available configurations.
  - *Setup times*: Reflect the delivery times of purchased manufacturing configurations or equipment, subject to changes based on supply chain dynamics.

- **Status of current manufacturing facility:** This presents an overview of the current state of the manufacturing operations. It includes the operational status, production output, and various costs associated with the purchased manufacturing configurations or equipment.
  - *Status of currently running manufacturing configurations*: Provides real-time operational status—whether the manufacturing configuration is running, being prepared, or has failed.
  - *Products produced*: Displays the quantity of products manufactured by each configuration.
  - *Incurred costs*: Represents the total investment made for the purchased configurations or equipment.
  - *Recurring costs*: Depicts the operational expenses incurred for the purchased configurations or equipment.
  - *Production rates*: Shows the production capacities of the purchased configurations or equipment.
  - *Setup time*: Indicates the delivery time of the purchased configurations or equipment.

The **state space** in RL formally represents all the information needed for the agent to decide at any given moment. In the context of the MFGRL environment, the state includes all information available from the production data, metadata for algorithms, market data, and the status of the current manufacturing facility. It encompasses details such as remaining demand, remaining time, rewards, available manufacturing configurations in the market, their respective costs and production rates, along with the current manufacturing facility's status, production output, and cost details. These states serve as the inputs for the RL agent, enabling it to understand the current condition of the manufacturing environment.

Therefore, the state space of the proposed MFGRL environment is defined as an $(2 + 6S + 4M)$-dimensional vector. Here, S represents the space size, which essentially is the number of elements in the state space related to the current manufacturing setup. M denotes the number of different manufacturing configurations available in the market. To elaborate:

- The term 2 in the expression $2 + 6S + 4M$ accounts for the two universal parameters:
  1. Remaining demand: $D_r \in \mathbb{Z}^+$, initialized as $D_r = D$.
  2. Remaining demand time: $T_r \in \mathbb{Z}^+$, initialized as $T_r = T_D$.
- The term $6S$ accounts for six types of information for each element in the space size S. Specifically, for each S, the six types of information are:
  1. Investment costs of manufacturing configurations: $I \in \mathbb{R}^S_{>0}$.
  2. Recurring costs of purchased manufacturing configurations: $R \in \mathbb{R}^S_{>0}$.
  3. Production rates of purchased manufacturing configurations: $P \in \mathbb{R}^S_{>0}$.
  4. Setup times: $U \in \mathbb{R}^S_{>0}$.
  5. Current statuses of purchased manufacturing configurations: $C \in \mathbb{R}^S_{\geq 0, \leq 1}$.
  6. Outputs, or the number of products produced: $O \in \mathbb{R}^S_{\geq 0}$.

- The term $4M$ accounts for four types of market-related information for each of the M manufacturing configurations:

    1. Market investment costs: $\mathcal{I} \in \mathbb{R}^{M}_{>0}$.
    2. Market recurring costs: $\mathcal{R} \in \mathbb{R}^{M}_{>0}$.
    3. Market production rates: $\mathcal{P} \in \mathbb{R}^{M}_{>0}$.
    4. Market setup times: $\mathcal{U} \in \mathbb{R}^{M}_{>0}$.

In summary, the state space is a high-dimensional vector that encapsulates a comprehensive set of variables. These variables pertain to both the current manufacturing setup and the market conditions, thereby enabling the RL agent to make well-informed decisions.

The **action space** in RL represents the set of all possible actions that the agent can take at a given state. In this manufacturing context, actions could include:

- purchasing new manufacturing configurations or equipment from the market,
- starting or stopping a manufacturing process,
- changing the operating parameters of a running process, and so forth.

The RL agent determines the choice of action based on its current policy to maximise the cumulative reward over time. The action space must be well-defined to ensure the agent can interact effectively with the environment.

The proposed RL environment complements MCDM methods in decision-making for optimal manufacturing configurations. While MCDM methods provide a robust framework for making decisions based on multiple criteria, the RL environment, specifically the MFGRL, provides a testing ground to validate these decisions. It creates a discrete-simulation environment that replicates real decision-making scenarios. This environment's ability to handle stochasticity, uncertainty, and long-term planning under these scenarios makes it particularly useful.

RL adds an extra layer of robustness to the decisions derived from MCDM methods, providing an additional validation step. The RL environment also opens the door for real-time learning and dynamic decision-making, allowing the system to continually learn and improve its policy in response to the changing environment. Thus, integrating MCDM with the RL environment leads to more resilient and efficient decision-making in manufacturing configuration selection.

6.5 Chapter summary

This chapter addressed the optimisation and decision-making challenges concerning selecting optimal manufacturing configurations due to multiple and fluctuating manufacturing costs with uncertainties. To this end, three modules were developed, all of which reside on top of the proposed data model:

1. **Changes identification module**: Employing the CIDENA algorithm, this module analysed input data models to pinpoint alterations in capabilities, capacities, and operational parameters across successive manufacturing requirement periods. CIDENA ingested two requirement periods and outputted the changes in capabilities (ΔB), capacities (ΔP), and operational parameters (ΔO). The algorithm processed each capability-capacity pair, updated capacities based on availability, filtered non-zero values, and evaluated conditions to recommend configuration modifications.

2. **Optimisation Module**: This module employed an optimisation matrix to categorise the identified changes into four distinct scenarios, thereby guiding the selection of suitable optimisation algorithms. For the optimisation of operational parameters, the O3PARAMS meta-algorithm was introduced, which is tailored to address manufacturing-specific challenges such as uncertainty and noisy data. Multi-objective evolutionary or genetic algorithms

were recommended for capability and capacity optimisation to balance conflicting objectives. The module supported single and multi-objective optimisation, suggesting weighted sum scalarisation or Pareto front optimisation for multi-objective scenarios.

3. **Decision-Making Module**: This module validated the optimisation outcomes using multiple criteria decision-making (MCDM) methods, such as TOPSIS, ASF, and Pseudo-Weights. Additionally, reinforcement learning (RL) was proposed for sequential decision-making under uncertainty, facilitated by a novel manufacturing RL environment named MFGRL. The MFGRL environment was a simulation testbed for evaluating decision robustness, incorporating elements like production status, market data, current facility status, actions, and rewards.

This chapter concludes the technical contributions provided in this thesis. The following chapter will systematically validate all the technical contributions from all three technical chapters. This validation will be performed using a controlled robotic demonstrator and applying the research contributions in broader manufacturing contexts.

Chapter 7

Validation of the research contributions

7.1 Introduction

The goal of this chapter is to experimentally validate the research contributions provided in technical chapters 4, 5, and 6 following the validation methodology presented in Chapter 3.3.

Specifically, experiments in this chapter are done in a controlled robotics lab environment where use cases were developed from scratch, and the complexity of manufacturing scenarios was increased incrementally.

The validation of the research hypotheses is done sequentially in the following way:

1. Develop an object-oriented data model for given manufacturing processes that can cover disturbances such as capability change, capacity change, and operational parameter change requirements.
2. Develop manufacturing apps for each manufacturing requirement scenario to demonstrate a modular and interoperable software integration approach building on the previous step's data model.
3. Integrate developed optimisation and decision-making algorithms that use the data model and the manufacturing apps to deal with multiple objectives and uncertainties.

A **holistic solution** to the MSC problem is considered successful if all the above criteria are met, where each stage depends on the previous step's success.

With the above steps defined, the next section describes the experimental setup, including the selected representative manufacturing processes and the utilised manufacturing assets.

7.2 Experimental setup

7.2.1 Representative manufacturing processes

The experiments in this chapter focus on two representative manufacturing processes: sorting cylinders of various dimensions, as shown in Figure 7.1, and bin-picking of three different parts of industrial pipe couplers, as shown in Figure 7.2. While both sorting cylinders and bin-picking of industrial pipe couplers are similar manufacturing processes, their underlying mechanisms and challenges are different.

Sorting cylinders primarily involves a classification task. The objective is to distinguish between cylinders based on specific attributes such as diameter, height, and colour. The challenge in this task lies in the accurate differentiation of cylinders that have closely related but distinct specifications. The focus is on attribute-based categorisation, and the task is generally static, meaning the cylinders are usually presented in a uniform orientation for sorting.

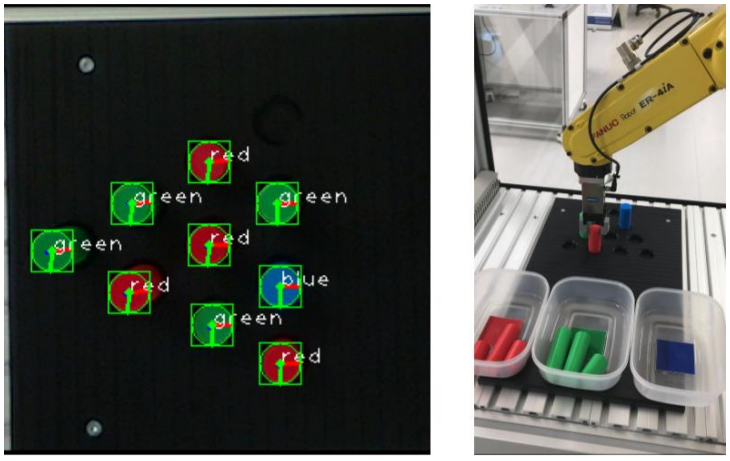

Figure 7.1: Manufacturing setup for the sorting task. The left figure shows the post-image capture state, detailing workspace and object identification. The right image depicts the robot executing the sorting procedure.

In contrast, bin-picking of industrial pipe coupler parts is a more dynamic task that involves robotic recognition and selection. The parts are often randomly

oriented within a container, necessitating advanced spatial recognition algorithms. The challenge here is identifying the correct part, determining its orientation, and planning a collision-free path for the robotic arm. This task is more complex regarding robotic manipulation and requires real-time processing to adapt to the random orientations of the parts.

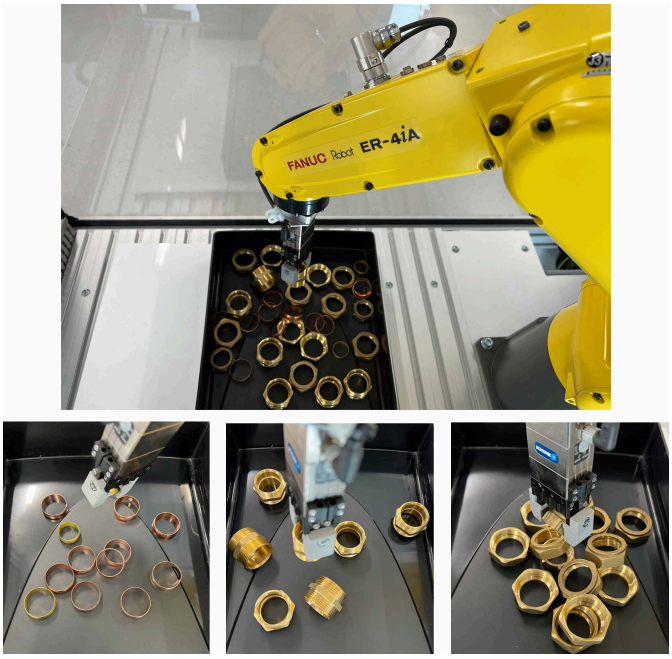

Figure 7.2: Manufacturing setup for the bin-picking process with three distinct components. The goal is to allocate them accurately to matching containers.

7.2.2 Employed manufacturing assets

The validation process employs the subsequent manufacturing equipment from the Advanced Robotics Lab at the University of Nottingham, as shown in Figure 7.3:

- **FANUC ER-4iA industrial robot:** An industrial robot designed for tasks requiring precision and reliability.
- **R-30iB Mate Plus Controller:** A FANUC robot controller that ensures operation and manoeuvring of the FANUC industrial robots.
- **Dual-arm ABB YuMi - IRB 14000 collaborative robot:** A cobot

which comes with intrinsic safety measures for tasks demanding collaboration between humans and robots.

- **ABB IRC5 - Industrial Robot Controller:** ABB robot controller which ensures operation and manoeuvring of the ABB robots.
- **Schunk parallel grippers:** Tools to provide precise and firm grasping of objects.
- **FANUC iRVision camera with black and white 2D vision:** FANUC camera for image capturing, object detection, and spatial orientation.
- **Intel RealSense LiDAR L515:** LiDAR camera for high-resolution 3D images with depth perception.
- **Raspberry PI 4 Edge device with 8GB RAM and 32GB HDD (x2):** Computational devices for hosting and deploying manufacturing software.

Figure 7.3: Manufacturing assets utilised in the validation process.

7.2.3 Introduced manufacturing requirements

Several new manufacturing requirements, as shown in Figure 7.4, are introduced during the validation steps to assess the effectiveness of the proposed research contributions:

- **Capability change requirement:** transitioning from a sorting process to a bin-picking process.
- **Capacity change requirement:** increasing the number of bin-picking operations per time unit.
- **Operational parameters change requirement:** reducing energy consumption without compromising cycle time.

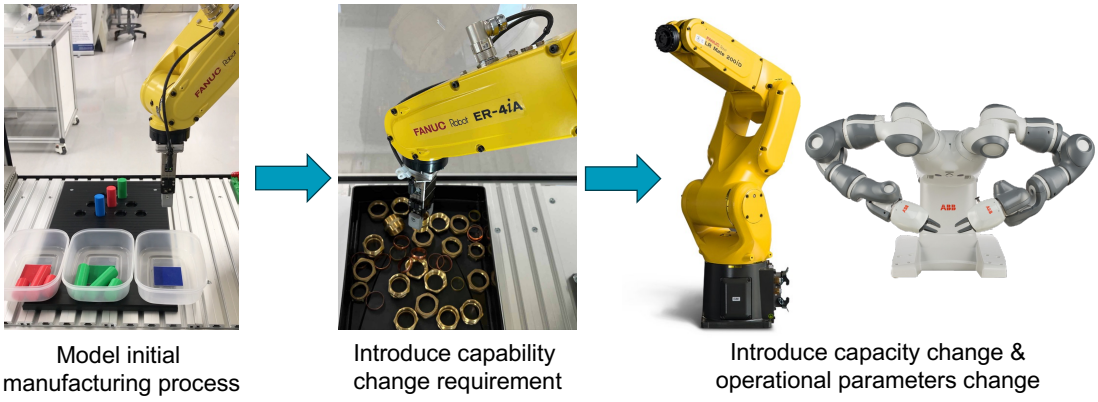

Figure 7.4: The figure shows the introduction of new manufacturing requirements as disturbances during the validation steps to assess the effectiveness of the proposed solutions.

This approach of systematically altering the manufacturing requirements during the validation steps provides a comprehensive understanding of how effectively the solutions can adapt to real-world manufacturing constraints and variations.

With the experimental setup presented, the next section begins with validating the proposed object-oriented data model, which forms the foundational basis for validating subsequent technical contributions such as manufacturing apps, optimisation, and decision-making algorithms.

7.3 Validation of the object-oriented data model

The goal of this section is to experimentally show that the technical contributions presented in Chapter 4 address

Research question 1: *What type of data models can accurately represent the complexities and dynamic nature of manufacturing systems while also suitable for software application integration?*

and validate

Research hypothesis 1: *Object-oriented data modelling can effectively capture the complexities and dynamism integral to manufacturing systems while being suitable for integrating with software applications.*

Data models form the foundation when designing complex systems such as manufacturing systems. Consequently, the first step in validating the holistic solution is to develop a data model for the underlying manufacturing process. The validation process of the data modelling is structured as follows:

1. Develop initial data model for the manufacturing process of sorting cylinders building on top of the technical contributions provided in Chapter 4.
2. Update the initial data model simulating the capability change requirement, i.e., the capability changes from sorting to bin-picking.
3. Update the data model in the previous step simulating the capacity change requirement, i.e., the number of bin-picking operations increases.

The subsequent sections delve deeper into the above steps and demonstrate the updates applied to the initial object-oriented data model sequentially.

7.3.1 Data model for the initial manufacturing process

The sorting of the cylinders is selected as an initial manufacturing process, which progresses further with different manufacturing requirements. Therefore, the initial data model for the sorting process is modelled as shown in Figure 7.5.

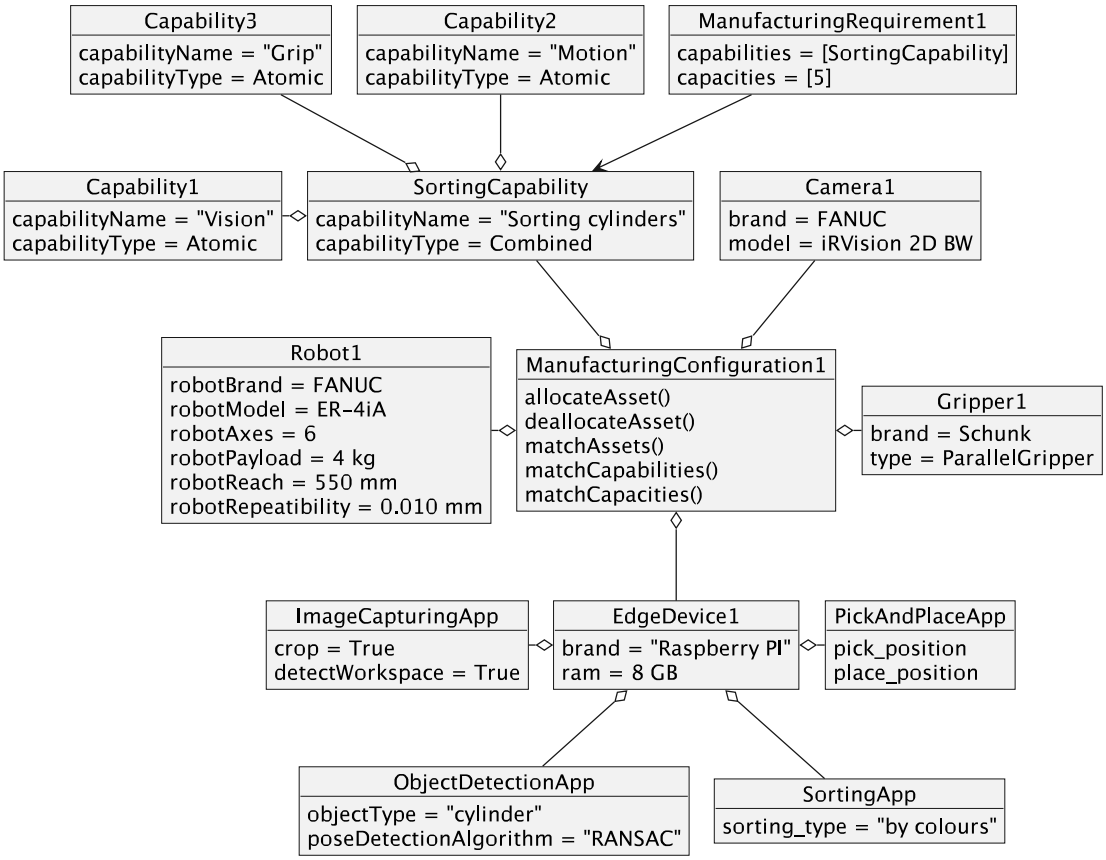

Figure 7.5: Foundational data model for the sorting process. UML object diagrams provide a visual representation of instances based on the UML class diagrams.

Figure 7.5 uses UML object diagrams to illustrate instances originating from the UML class diagrams. The objects are detailed as follows:

1. **Manufacturing configuration**: At the heart of the data model is the *ManufacturingConfiguration1* object. This object is central in coordinating the various elements of the sorting process. It establishes multiple associations with assets such as *Robot1*, *Camera1*, *Gripper1*, and *EdgeDevice1*. Additionally, the configuration is linked to the *SortingCapability*, highlighting its

sorting capability. The *ManufacturingConfiguration1* object incorporates several methods to underscore its multifunctional role in this data model.

2. **Manufacturing requirement**: The *ManufacturingRequirement1* object represents a need for the sorting capability with a specified capacity of five operations per unit of time. It establishes a directed association with the object *SortingCapability*, indicating its reliance on it.

3. **Capabilities**: The data model's design differentiates between individual capabilities and a combined capability:

   - *Capability1*, *Capability2*, and *Capability3* are individual capabilities, representing *Vision*, *Motion*, and *Grip* capabilities. Each capability embodies a distinct function vital to the sorting process.
   - *SortingCapability* is represented as a combined capability. It represents a fusion of the atomic capabilities, demonstrating the integration of multiple functions.

4. **Manufacturing assets**: Each asset has distinct attributes and contributes uniquely to the manufacturing process:

   - *Robot1*: A robotic asset from the FANUC brand, model ER-4iA, featuring six axes, a reach of 550 mm, and a repeatability of 0.010 mm.
   - *Camera1*: A FANUC's iRVision 2D BW model, serving as the visual component of the sorting operation.
   - *Gripper1*: A parallel gripper by Schunk, essential for picking cylinders.
   - *EdgeDevice1*: An edge computing device, Raspberry PI, equipped with 8 GB RAM, vital for local processing and data management.

5. **Manufacturing apps**: These software components, hosted on *EdgeDevice1*, are tailored for specific tasks:

- *ImageCapturingApp*: Designed for visual data collection, with features like cropping and workspace detection.
- *ObjectDetectionApp*: Specialised in cylinder detection, it utilises the RANSAC algorithm for advanced pose detection.
- *SortingApp*: As implied, it sorts objects, primarily by colour.
- *PickAndPlaceApp*: This app facilitates the pick-and-place operations, bridging the software and hardware components.

The detailed structure of this data model encapsulates the complexity of the manufacturing system and the sorting process. Each object, relationship, and attribute is designed to provide a comprehensive representation, ensuring seamless operation.

Modelling the manufacturing process in an object-oriented way helps to encapsulate the complexities and provides modularity. Therefore, the next section demonstrates how the data model is minimally affected when there is a capability change requirement.

7.3.2 Modelling capability change requirement

In this setup, the data model earlier designed for the sorting process is adjusted for the bin-picking of three distinct parts of pipe couplers. The primary objective of this section is to demonstrate how few changes to the initial data model are needed to introduce a new capability. The alterations made to the initial data model from the previous section are presented in Figure 7.6.

Several changes are indispensable to transition from sorting cylinders to bin-picking of pipe couplers. Firstly, the capability requirements differ; bin-picking demands enhanced 3D vision to navigate densely packed environments, necessitating introducing a “3D Vision” capability. The camera hardware must also be

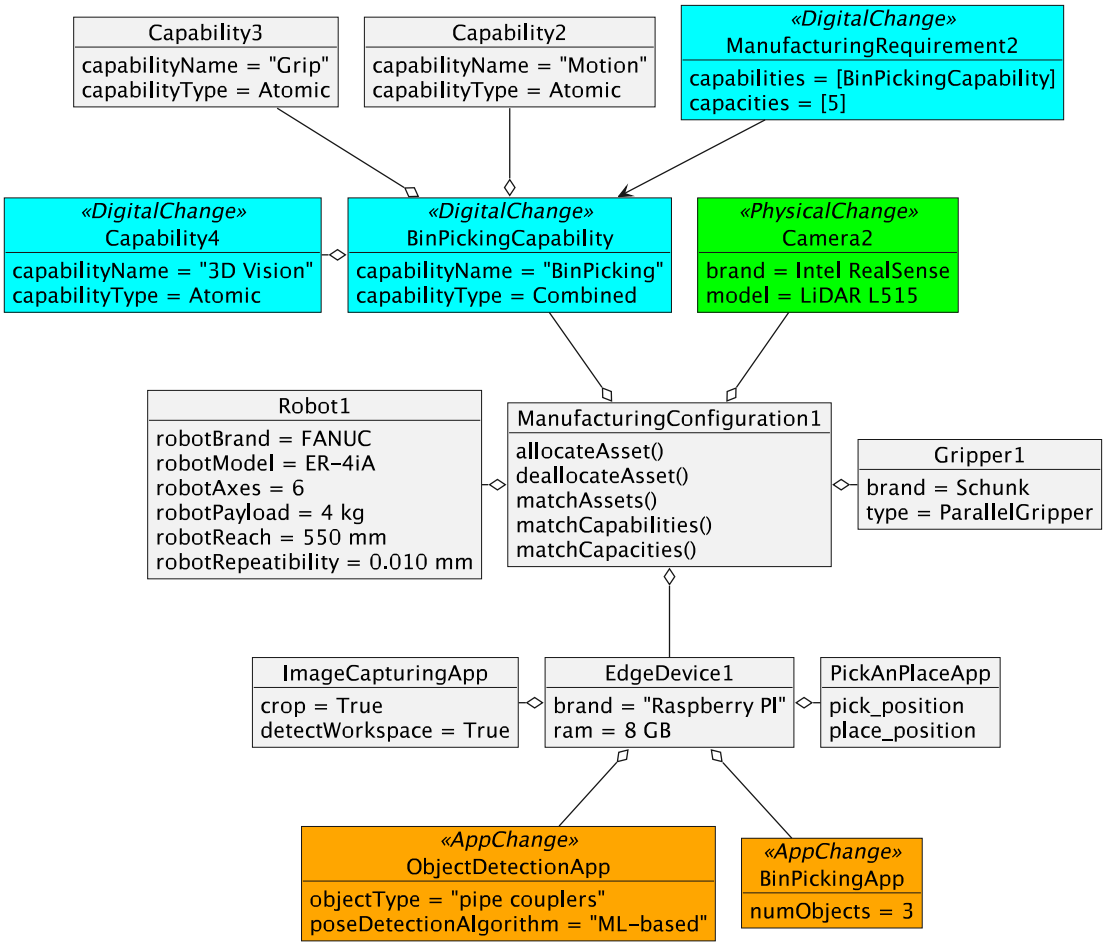

Figure 7.6: Modelling capability change requirement. Capability changes from sorting to bin-picking process. That data model presented in Figure 7.5 is changed with new coloured elements.

updated to support this, leading to the shift from a 2D camera to a more advanced LiDAR-based camera. The software components also need to be changed to accommodate new software capabilities.

The changes, highlighted using different colours, provide a clear visual representation of the modifications. The cyan elements represent digital changes, the lime elements indicate physical changes, and the orange elements signify software adjustments. Key modifications include the introduction of *Capability4* for "3D Vision", transitioning from *Camera1* to *Camera2*, and software updates like the *ObjectDetectionApp* and the new *BinPickingApp*.

Despite these alterations, a significant portion of the original data model remained

the same, showcasing the model's inherent flexibility. Elements such as *Robot1*, *Gripper1*, and *EdgeDevice1* were retained, indicating the model's versatility across different manufacturing processes. The capability structure was particularly modular, allowing a seamless transition from *SortingCapability* to *BinPickingCapability*, integrating existing atomic capabilities and introducing only one new capability.

On the hardware side, the singular change from *Camera1* to *Camera2* underscores the model's adaptability with minimal physical alterations. Similarly, software changes were direct and process-specific without affecting the broader software architecture. These focused updates in hardware and software highlight the model's ability to evolve with precision.

As the volume or scale of operations expands, it becomes important to integrate additional manufacturing equipment to adapt to changes in the capacities of manufacturing operations. Therefore, the next section will demonstrate how the data model adapts to the capacity change requirements requiring new manufacturing equipment.

7.3.3 Modelling capacity change requirement

In this section, the focus is on addressing the increased demand in the capacity of bin-picking operations. The revised capacity, set at ten bin-picking operations per minute, surpasses the capabilities of the current robot, necessitating the integration of an additional robotic arm. This is a significant jump from the previous capacity of five operations per minute.

The data model from the previous section undergoes very few modifications to accommodate this change, as illustrated in Figure 7.7.

A new robotic arm, *Robot2*, from the ABB brand with the model YuMi IRB

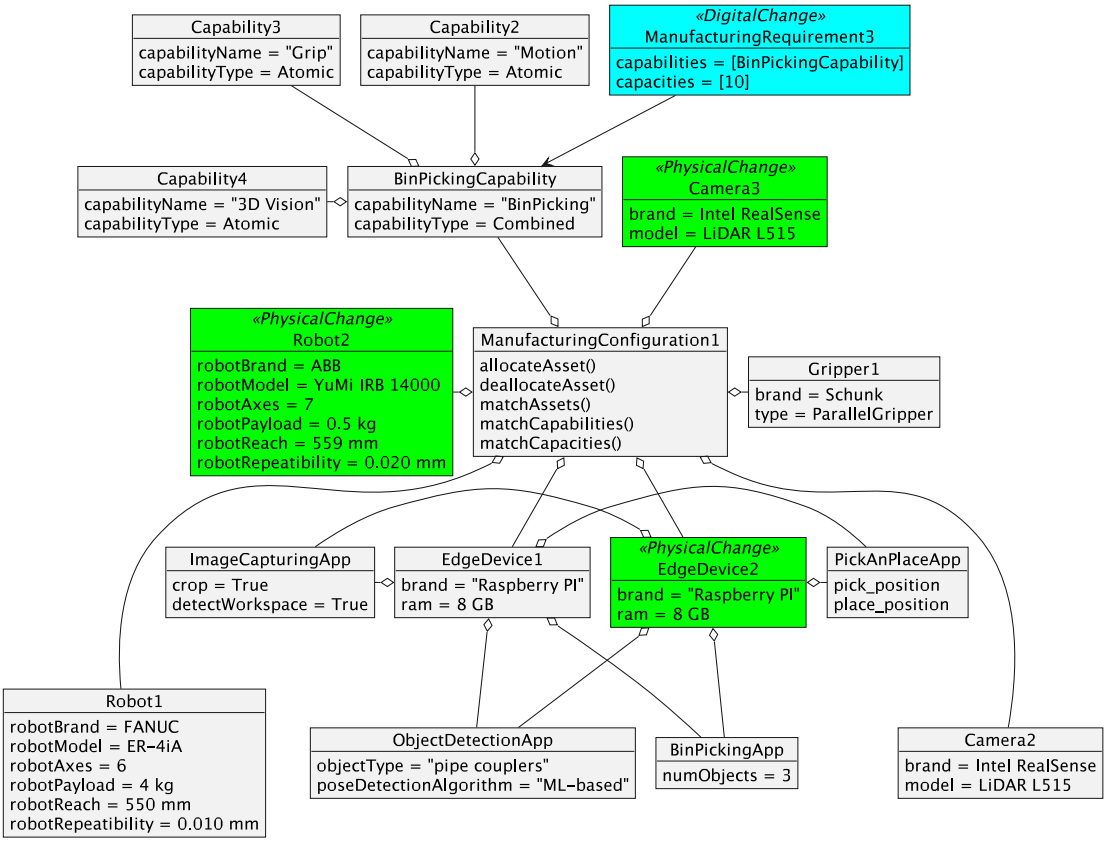

Figure 7.7: Modelling capacity change requirement. The capacity has been adjusted from 5 bin-picking operations to 10 bin-picking operations per minute. The data model, as presented in Figure 7.6, has been updated with new coloured elements to reflect these changes.

14000, has been introduced to address the increased capacity requirements. With its seven axes, a payload of 0.5 kg, a reach of 559 mm, and a repeatability of 0.020 mm, this robot complements the existing *Robot1* to ensure the system can handle the increased operations.

Additionally, another camera, *Camera3*, identical to *Camera2*, has been added to support the new robot's vision needs. To manage the computational demands of the additional equipment, an extra edge device, *EdgeDevice2*, mirroring the specifications of *EdgeDevice1*, has been integrated. This device runs the necessary applications, such as *ImageCapturingApp*, *ObjectDetectionApp*, *BinPickingApp*, and *PickAnPlaceApp*, ensuring seamless operations.

The adjustments to accommodate the increased bin-picking capacity highlight the

changes' essential yet minimal nature. This minimalism underscores the robustness and adaptability of the original data model. Rather than a comprehensive overhaul, the model can seamlessly integrate an additional robotic arm and associated equipment. The foundational structure of the data model remains largely consistent, with the introduction of *Robot2*, *Camera3*, and *EdgeDevice2* adhering to the same framework and associations as their predecessors.

Furthermore, integrating the new robotic arm and the additional camera does not require redefinition or adjustment of the existing capabilities or applications. This reuse of software applications for the new edge device exemplifies the model's versatility. Emphasising its modular design, the data model effortlessly facilitates the addition of new equipment. Each new component, be it the robotic arm, camera, or edge device, is incorporated as a distinct entity, ensuring no disruption to the existing components.

In addition to capability and capacity change, operational parameters in manufacturing systems often require adjustments due to factors like evolving product specifications or technological advancements. These operational parameters can be various, such as changing the speed of the conveyor belt or adjusting the velocity and acceleration parameters of robotic arms. Therefore, the next section illustrates how the data model adapts when there is a need for optimising the parameters of the manufacturing equipment.

7.3.4 Modelling operational parameters change requirement

In this section, the focus is on a pressing issue in sustainable manufacturing: reducing the energy consumption of robotic systems without compromising their operational efficiency.

The FANUC ER-4iA robot is the exemplar for this validation step, chosen for its widespread use and representative features. However, it is worth noting that the approach delineated here is modular and can be seamlessly applied to other robotic systems, such as those manufactured by ABB.

The imperative for energy optimisation stems from both environmental considerations and the drive to improve cost-efficiency. As energy costs escalate and sustainability becomes a corporate priority, understanding how the data model adapts to these energy-saving requirements gains paramount importance. Therefore, the modifications to the previous data model are illustrated in Figure 7.8. Only the affected objects are showcased, avoiding unnecessary clutter.

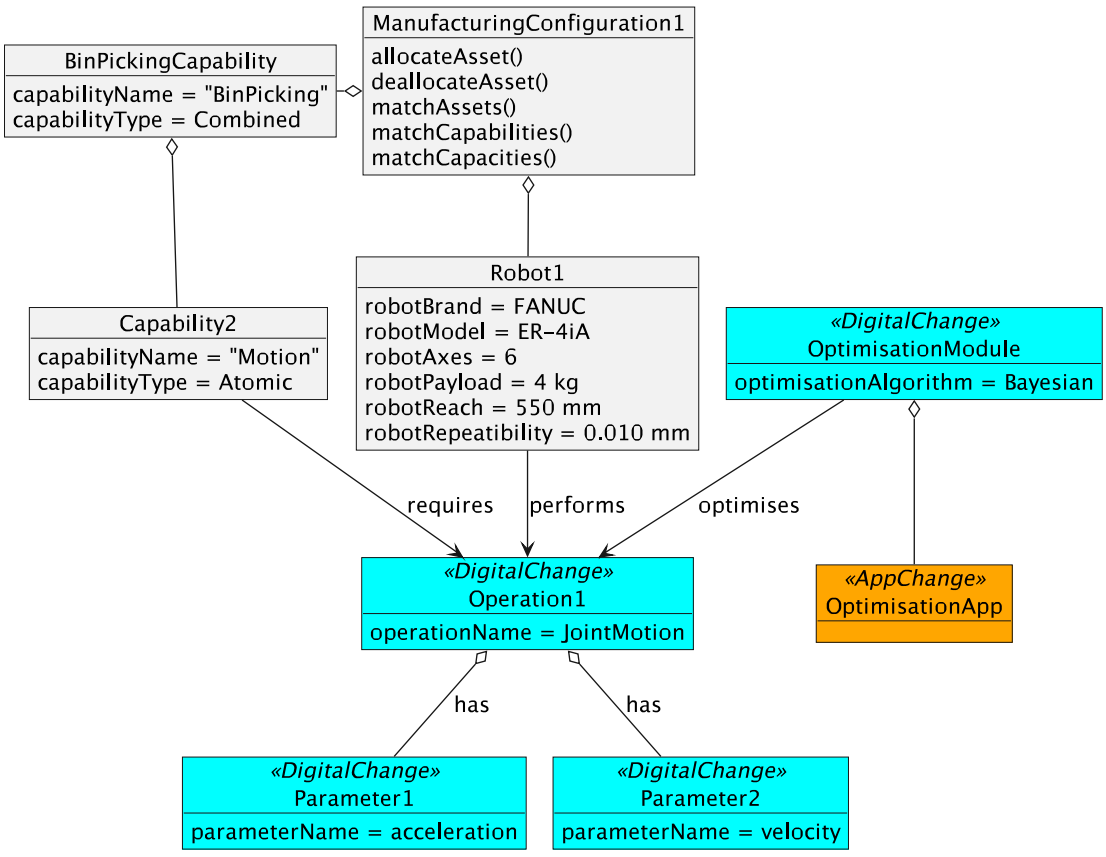

Figure 7.8: Modelling operational parameter change requirement. The new requirement is to reduce the energy consumption of the FANUC ER-4iA industrial robot.

The data model introduces new digital changes, primarily focusing on the operations and parameters associated with the robot's motion. The *Operation1* object,

labelled *JointMotion*, represents the specific operation that the robot performs and will be the optimisation target. This operation has associated parameters, *Parameter1* (acceleration) and *Parameter2* (velocity), which are crucial determinants of the robot's energy consumption and cycle time.

The *OptimisationModule* object, employing a Bayesian optimisation algorithm, is introduced to fine-tune these parameters. This module interacts directly with the *Operation1* object, indicating its role in optimising the robot's joint motion. An associated *OptimisationApp* is also introduced, suggesting that there might be a software interface or application layer where the optimisation can be monitored or controlled.

The associations between the robot, its capabilities, and its operations remain consistent. The robot performs the *JointMotion* operation, which requires the *Motion* capability. The optimisation module's direct link to the operation signifies its pivotal role in ensuring the robot's energy-efficient performance without compromising its operational speed or accuracy.

The object-oriented nature of the data model is especially beneficial in addressing dynamic manufacturing requirements, such as the one at hand. Each component of the manufacturing process, be it the robot, its operations, or the optimisation module, is encapsulated as an individual object. This encapsulation ensures that each object can be modified, optimised, or replaced without disrupting the system.

Furthermore, the relationships and associations between objects provide an intuitive understanding of how different components interact. In this scenario, the direct association between the *OptimisationModule* and *Operation1* indicates the module's role in fine-tuning the robot's motion. This clarity not only aids in comprehending the system's functioning but also simplifies implementing changes.

7.3.5 Analysis

In Chapter 3.3.2, the following success metrics were established for the validation of the proposed object-oriented data model:

1. **Adaptability without extensive modifications.** Throughout the validation process, the data model showcased its adaptability by responding efficiently to various changes in requirements. For instance, when the need arose from sorting to bin-picking, the model was adjusted by introducing a new capability and some software and hardware updates. Furthermore, when there was an increase in bin-picking operations capacity, the model could incorporate an additional robotic arm and its associated equipment without undergoing a major change.

2. **Representation of static and dynamic aspects.** The data model can capture the manufacturing process's static and dynamic elements. On the static side, components such as manufacturing assets (robots, cameras, grippers, and edge devices) are well-defined with their respective attributes and associations. The model effectively represents operations, capabilities, and associated parameters on the dynamic front. A notable instance of this dynamic representation is when the model was adjusted to optimise the robot's energy consumption, which was achieved by introducing new objects related to operations and parameters, further emphasised by adding the *OptimisationModule*, which directly interacts with the robot's operation.

In light of the above arguments, the first research hypothesis is successfully validated, affirming the robustness and versatility of the object-oriented data model in the context of manufacturing systems.

7.4 Validation of the manufacturing apps

The goal of this section is to experimentally show that the technical contributions presented in Chapter 5 address

Research question 2: *What software development approaches can reduce the interoperability challenges posed by utilising diverse equipment in manufacturing systems while enhancing plug-and-produce capabilities?*

and validate

Research hypothesis 2: *Developing manufacturing software as modular manufacturing apps, incorporating a modular approach to integrate various equipment and communication protocols, can enhance a manufacturing system's interoperability and plug-and-produce capabilities.*

Building upon the foundational data model and its subsequent modifications discussed in the previous section 7.3, this section delves into the validation of the manufacturing apps. The validation process is structured as follows:

1. The development of manufacturing apps development kit (MAPPDK), as described in Chapter 5.3.
2. The development of manufacturing apps for each identified scenario, leveraging the capabilities of the MAPPDK.
3. The development of a modular architecture that aids in deploying and orchestrating the manufacturing apps, as detailed in Chapter 5.4.

The subsequent sections delve deeper into each of these steps, detailing the intricacies of the MAPPDK, the nuances of the manufacturing apps developed using this kit, and the details of the modular architecture that serves these apps, ensuring an interoperable and plug-and-produce manufacturing system.

7.4.1 Manufacturing apps development kit

The MAPPDK was developed through a structured process, combining theoretical insights and practical requirements. This process is detailed as follows:

1. **Requirement analysis:** The requirements from Chapter 5.3.1 were reviewed to ensure the MAPPDK addressed the main challenges in the manufacturing sector.

2. **Development environment choice:** Python programming language was selected due to its focus on object-oriented data modelling, rich libraries, and strong community backing.

3. **Framework design:** The base framework was made modular and adaptable, allowing for the addition of new functions or modules specific to equipment without major changes.

4. **Incorporation of vendor SDKs:** Modules for FANUC and ABB were developed to tackle the diversity of robots and vendor-specific standards and to combine into a standard MAPPDK interface.

5. **Core functionality development:** Equipment-related methods were developed and thoroughly tested in virtual settings before real-world application.

6. **Documentation:** Detailed documentation was produced, explaining the features, usage instructions, and best practices for the MAPPDK.[1]

7. **Iterative refinement:** The MAPPDK was refined over time. User feedback, technological advancements in manufacturing, and emerging industry challenges led to regular updates and improvements.[2]

[1]MAPPDK source code: https://github.com/torayeff/mappdk
[2]User feedback and documentation: https://github.com/torayeff/fanucpy

One of the challenges in developing MAPPDK was developing a robot-specific driver and communication protocol.

Therefore, a proof-of-concept communication protocol was developed between the MAPPDK and the robot controller to validate the proposed idea, utilising socket communication. The design of this protocol is depicted in Figure 7.9.

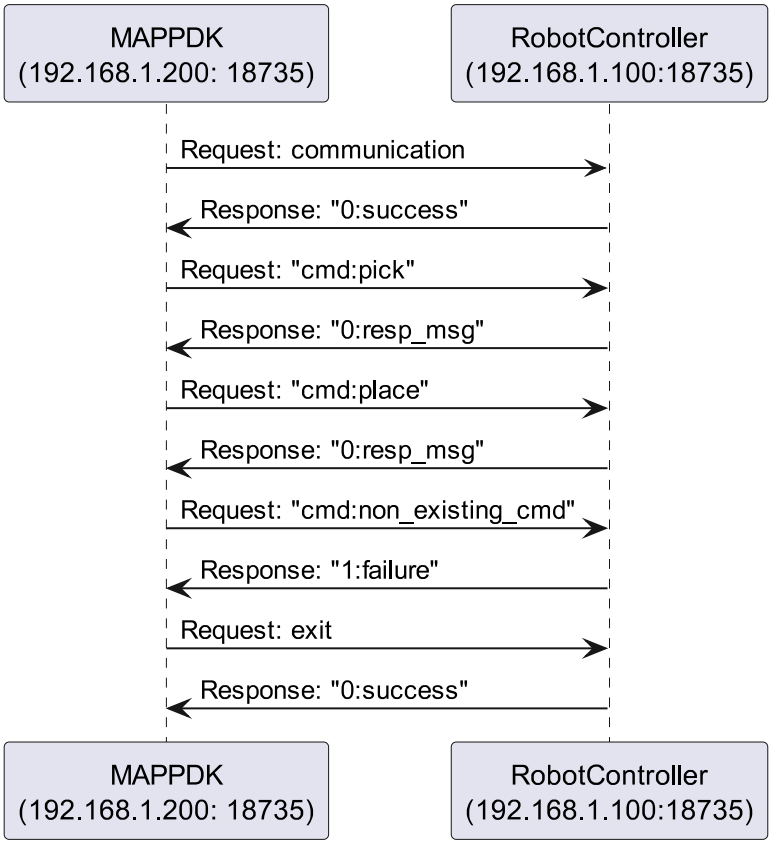

Figure 7.9: MAPPDK communication protocol with robot controllers using socket communication over TCP/IP.

The communication protocol serves as a structured method for interaction between MAPPDK and a robot controller. Initiated by the software, a request is sent to the hardware to establish communication. Once acknowledged, a series of commands can be relayed from the software to the hardware, which executes these commands and provides feedback. The protocol is designed to handle standard operations and errors, ensuring that the system responds with an error message if an invalid command is sent.

7.4.2 Developed manufacturing apps

Building upon the methodology proposed in Chapter 5.2.3, several manufacturing apps were developed using the MAPPDK. Each of these apps, as illustrated in Figure 7.10, is structured around the following components:

1. Functional interface
2. Control logic
3. Configuration parameters
4. Data management
5. Error handling

The object-oriented data modelling's inheritance property was leveraged to inherit these properties from the abstract *ManufacturingApp* class, as defined in Chapter 5.2.3 and depicted in Figure 5.4.

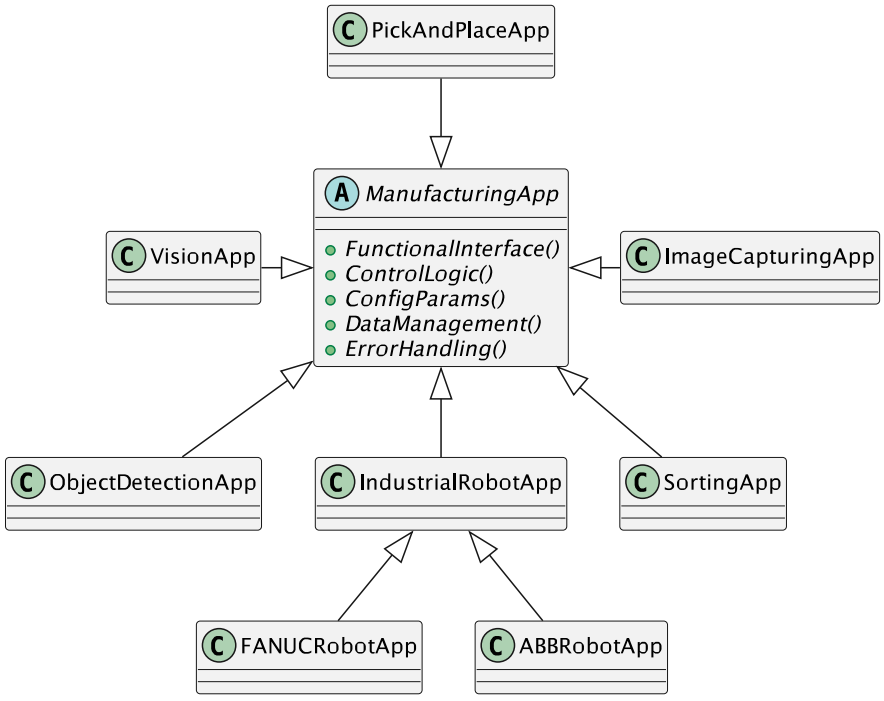

Figure 7.10: Developed manufacturing apps for the validation.

The developed apps, their functionalities, and their components are detailed as follows:

1. **VisionApp:**
    - *FunctionalInterface*: Interfaces with camera hardware, such as FANUC iRVision 2D BW model or the LiDAR-based camera.
    - *ControlLogic*: Contains image processing and analysis algorithms.
    - *ConfigParams*: Parameters related to image resolution, focus, zoom.
    - *DataManagement*: Manages the storage and retrieval of captured images.
    - *ErrorHandling*: Addresses camera malfunction or image corruption issues.
2. **PickAndPlaceApp:**
    - *FunctionalInterface*: Interfaces with the robotic arm and gripper.
    - *ControlLogic*: Contains algorithms for determining pick-and-place locations.
    - *ConfigParams*: Parameters related to robot speed and gripper.
    - *DataManagement*: Manages the storage and retrieval of pick-and-place sequences.
    - *ErrorHandling*: Addresses issues like robot movement or object drop.
3. **ImageCapturingApp:**
    - *FunctionalInterface*: Interfaces with the camera hardware.
    - *ControlLogic*: Contains algorithms for capturing and cropping images.
    - *ConfigParams*: Parameters related to image capture settings.
    - *DataManagement*: Manages the storage and retrieval of captured images.

- *ErrorHandling*: Addresses camera malfunction or storage issues.

4. **ObjectDetectionApp:**

   - *FunctionalInterface*: Interfaces with captured images.
   - *ControlLogic*: Contains algorithms for detecting objects in images, using the RANSAC and deep learnings algorithms.
   - *ConfigParams*: Parameters related to detection sensitivity and object size.
   - *DataManagement*: Manages the storage and retrieval of detection results.
   - *ErrorHandling*: Addresses issues like false detections or missed detections.

5. **IndustrialRobotApp:**

   - *FunctionalInterface*: Interfaces with the robotic arm hardware.
   - *ControlLogic*: Contains algorithms for robot movement and operation.
   - *ConfigParams*: Parameters related to robot speed and movement sequences.
   - *DataManagement*: Manages the storage and retrieval of robot operation logs.
   - *ErrorHandling*: Addresses robot malfunction or movement issues.

6. **SortingApp:**

   - *FunctionalInterface*: Interfaces with the robotic arm and vision system.
   - *ControlLogic*: Contains algorithms for sorting objects based on criteria like colour.
   - *ConfigParams*: Parameters related to sorting criteria and bin locations.
   - *DataManagement*: Manages the storage and retrieval of sorting results.

- *ErrorHandling*: Addresses issues like mis-sorting or object drop.

7. **FANUCRobotApp and ABBRobotApp:**

   - *FunctionalInterface*: Specific interfaces for FANUC and ABB robotic arms.
   - *ControlLogic*: Contains brand-specific algorithms for robot movement and operation.
   - *ConfigParams*: Brand-specific parameters related to robot speed and movement sequences.
   - *DataManagement*: Manages the storage and retrieval of brand-specific robot operation logs.
   - *ErrorHandling*: Addresses brand-specific robot malfunction or movement issues.

These apps, developed using the MAPPDK, demonstrate the potential of the proposed methodology in addressing interoperability challenges in manufacturing systems. The modular and object-oriented approach ensures that each app can be developed, deployed, and maintained independently, enhancing the plug-and-produce capabilities of the system.

While the MAPPDK and the developed manufacturing apps enhance plug-and-produce capabilities, ensuring seamless integration and interoperability of diverse equipment in manufacturing systems, they alone are not sufficient for a holistic solution. For effective deployment and synchronised operations, a modular architecture is essential. This architecture not only supports dynamic app modifications without system disruptions but also ensures optimal resource utilisation and real-time responses. Therefore the next section delves into the details of this modular architecture.

7.4.3 Modular architecture

The modular architecture, as introduced in section 5.4, facilitates holistic communication and deployment between manufacturing apps and equipment. The realised architecture is illustrated in Figure 7.11.

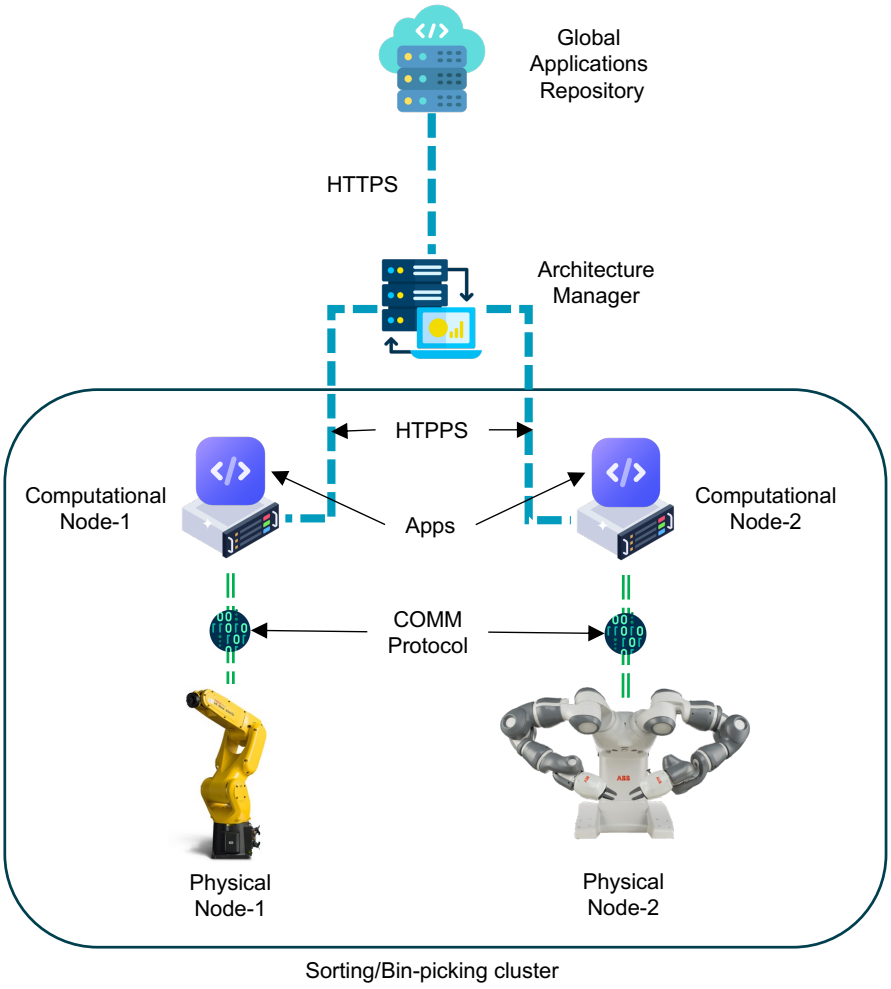

Figure 7.11: Schematic of the modular architecture tailored for sorting and bin-picking processes.

7.4.3.1 Elements of a modular architecture

Building upon the definitions from Chapter 5.4, the alignment between the physical and digital components in this validation setup is detailed below:

- **Atomic devices:** The system integrates the following atomic devices:

  - FANUC ER-4iA industrial robot
  - Dual-arm ABB YuMi - IRB 14000 collaborative robot
  - Schunk gripper
  - FANUC iRVision camera
  - Intel RealSense LiDAR L515

- **Physical nodes:** Two nodes are present:
  - *Physical node-1:* Combines FANUC ER-4iA robot, R-30iB Mate Plus Controller, Schunk gripper, and Intel RealSense LiDAR L515.
  - *Physical node-2:* Combines Dual-arm ABB YuMi robot, ABB IRC5 Controller, Schunk gripper, and Intel RealSense LiDAR L515.
- **Computational nodes:** The system has two computational nodes:
  - *Computational node-1* operates with Raspberry PI 4 Edge device for *Physical node-1*.
  - *Computational node-2* utilises Raspberry PI 4 Edge device for *Physical node-2*.
- **Manufacturing process cluster:** The final system architecture consists of the sorting and bin-picking process clusters.

7.4.3.2 Architecture manager

The architecture manager is an important component designed to streamline the deployment and management of manufacturing apps. The architecture manager ensures that each manufacturing app operates in an isolated environment by leveraging Docker, a platform that packages an application and its dependencies together in a container. This isolation is crucial to prevent conflicts arising from differing software dependencies or versions.

Kubernetes, a powerful container orchestration tool, is integrated to manage these Docker containers. It not only automates the deployment of containers but also handles tasks like scaling and failover for the applications. This significantly enhances system reliability and resilience, as Kubernetes can automatically replace or reschedule containers when necessary.

Furthermore, the architecture manager incorporates Node.js for its backend processes. Node.js, known for its non-blocking, event-driven architecture, is particularly suitable for real-time applications. This makes it a suitable choice for managing real-time data flows and communication in the manufacturing setup. Moreover, its vast library ecosystem accelerates the development of network applications, making the system more adaptable and extensible.

7.4.3.3 Global applications repository

The global applications repository is the centralised storage and management system for all manufacturing apps. Built on Ubuntu Linux, a widely-recognised and robust operating system, it ensures a stable and secure foundation for the repository. Ubuntu's consistent updates and vast community support make it a reliable choice for such critical infrastructure.

Data within the repository is managed using an SQL-based database. This relational database system offers structured storage, essential for coherently organising vast amounts of data. Its querying capabilities ensure efficient data retrieval, making it easier for users to access and manipulate the stored information.

A dedicated router is employed to ensure that all system components can communicate effectively. This router provides a stable connection and ensures the integrity of the data being transferred. Its capability to support wireless data transfer is crucial, allowing for flexibility in the placement of components and ensuring seamless data flow between them.

7.4.4 Analysis

In Chapter 3.3.2, the following success metrics were established for the validation of the manufacturing apps:

1. **Ease of integration with diverse equipment.** The manufacturing apps, developed using the MAPPDK, demonstrated their adaptability by seamlessly interfacing with a range of equipment, such as FANUC and ABB robotic arms, Schunk grippers, and various camera models. The modular design of the MAPPDK and the object-oriented approach ensured that each app could be tailored to specific equipment needs without extensive customisation. This was particularly evident in the development of brand-specific apps like *FANUCRobotApp* and *ABBRobotApp*, which catered to the unique requirements of each robot brand while maintaining a consistent interface for the broader system.

2. **Enabling plug-and-produce capabilities.** The modular architecture, combined with the architecture manager's use of Docker and Kubernetes, played an important role in enhancing the plug-and-produce capabilities of the manufacturing apps. This setup allowed for the rapid deployment and scaling of apps in response to changing manufacturing needs. Furthermore, the global applications repository, ensured that apps could be easily accessed, modified, and redeployed, facilitating quick setup in diverse manufacturing environments. The real-time capabilities of Node.js in the architecture manager further streamlined the deployment process, ensuring that apps could respond promptly to real-time manufacturing events.

Given the above arguments, the second research hypothesis is successfully validated, highlighting the effectiveness of the manufacturing apps in addressing interoperability challenges and enhancing plug-and-produce capabilities in manufacturing systems.

7.5 Validation of the optimisation and decision-making algorithms

The goal of this section is to experimentally show that the technical contributions presented in Chapter 6 address

Research question 3: *What types of algorithms can effectively manage uncertainties and facilitate sequential decision-making in the dynamic and evolving multi-objective environment of manufacturing processes?*

and validate

Research hypothesis 3: *An optimisation and decision-making framework that integrates traditional optimisation and machine learning algorithms can effectively manage uncertainties and facilitate sequential decision-making in dynamic, multi-objective manufacturing environments.*

The validation of the third research hypothesis relies on the data model introduced in section 7.3, the requirements and data model illustrated in Figure 7.8, and the MAPPDK developed in the previous section 7.4. The structure of the validation process is as follows:

1. Development of energy monitoring app and modelling of the energy consumption.
2. Optimisation of pick-and-place operation and development of an optimisation app.
3. Optimisation of the whole bin-picking process using the developed apps.

The following sections delve into the details of each of the above steps.

7.5.1 Monitoring and modelling energy consumption

The FANUC ER-4iA industrial robot was chosen to optimise the operational parameters to minimise the energy consumption of the bin-picking process. The FANUC robot was chosen primarily because of its constrained software and data collection features, especially compared to ABB robots. Thus, a secondary aim is to show that MAPPDK can compensate for these shortcomings, acting as an interface between manufacturing hardware and optimisation algorithms. An energy monitoring app was developed, and the robot's energy consumption was modelled, as elaborated in the subsequent sections.

7.5.1.1 Monitoring energy consumption

Energy optimisation of operational parameters of any manufacturing equipment requires monitoring the energy data. Therefore, the first task was to monitor the energy consumption of the FANUC ER-4iA during a bin-picking manufacturing process. Unlike ABB robots, which allow energy data export to CSV files, FANUC controllers restrict this information to the user interface of the teach pendant.

A comprehensive analysis of the FANUC robot controller was done to understand how the energy data was stored in the FANUC robot controllers. It was identified that the specific system variable is responsible for storing energy data. This system variable was integrated into the MAPPDK, which allowed monitoring and collecting energy data using Python programming language.

The sampling rate is another important aspect when collecting the energy data and using the data for optimisation and decision-making. High sampling rates are essential for capturing the motion profiles of industrial robots. However, they can be cumbersome for real-time data preprocessing, especially in low-resource computing edge devices such as Raspberry PI. Conversely, low sampling rates risk

omitting crucial information like the start and end of operation cycles, which is necessary for the accuracy of the optimisation process.

In this validation experiment, the sampling rate was capped at 50 millisecond intervals due to controller limitations and network latency, the maximum rate supported by the FANUC controller. Post-acquisition, the data underwent resampling to align timestamps and was stored in a time-series database, InfluxDB.

As shown in Figure 7.12, the energy monitoring and visualisation app was developed to be robot-agnostic, fitting into a modular architecture developed in 7.4.3.

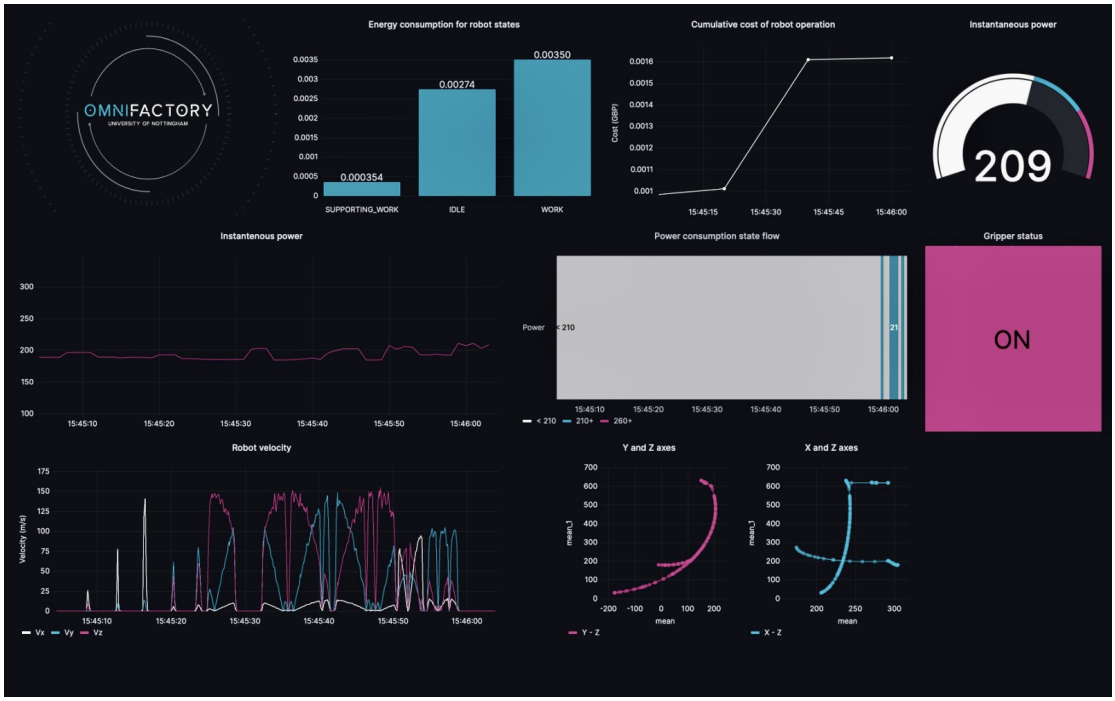

Figure 7.12: Energy visualisation app developed using MAPPDK, InfluxDB, and Grafana. This app was showcased at the opening of the Omnifactory facility at the University of Nottingham. User interface credits to Karol Niewiadomski.

After developing a robot-agnostic energy monitoring and visualisation app, the next step is to model the energy consumption for subsequent optimisation and decision-making. The energy monitoring and visualisation facilitated an understanding of the key parameters affecting the energy consumption in the bin-picking process. The key parameters' optimisation is discussed in detail in the next section.

7.5.1.2 Modelling the energy consumption

Following monitoring, visualising and collecting the energy data, the next step involves modelling the energy consumption of the industrial robot during the bin-picking process.

A black-box modelling approach was used on a real robot instead of explicitly modelling the industrial robot with all its intrinsic details, such as kinematics and dynamics, or using simulation-based methods. The advantage of black-box modelling is that it allows treating the whole system without knowing its internal workings and transferring it to different robots and manufacturing processes. Therefore, a black-box model was developed, integrating an industrial robot with a bin-picking manufacturing process. The black-box model can be represented as below function:

$$f : \mathbb{R}^d \rightarrow \mathbb{R} \tag{7.1}$$

The above black-box function f receives a d-dimensional vector of parameter values $\mathbf{x} \in \mathbb{R}^d$. The input vector $\mathbf{x}$ to the black-box function f is constrained using the lower and upper bound parameter values vectors $\mathbf{l} \in \mathbb{R}^d$, and $\mathbf{u} \in \mathbb{R}^d$ respectively. In this experimental scenario, $\mathbf{x}$ is the vector of adjustable operating parameters such as velocity and acceleration of the robot's arm. The lower and upper values, $\mathbf{l}$ and $\mathbf{u}$, are the minimum and maximum values of the parameters allowed.

Robotic manufacturing systems are complex, and different factors can affect the energy consumption of an industrial robot. Also, energy consumption measurements can be noisy depending on energy monitoring solutions. Thus, the methodology developed should be robust to such noisy outputs. To deal with the measurement noise directly inside the optimisation loop, the output of the black-box model with a noise value and the observed energy consumption of a black-box

model $f(\mathbf{x})$ were measured as below:

$$E(\mathbf{x}) = f(\mathbf{x}) + \epsilon \tag{7.2}$$

where the noise parameter ϵ is normally distributed:

$$\epsilon \sim \mathcal{N}(0,\, \sigma^2) \tag{7.3}$$

Considering the above model, the energy consumption optimisation of this bin-picking manufacturing process is formulated as below:

$$\min_{\mathbf{x} \in P} E(\mathbf{x}) \tag{7.4}$$

where

$$\mathbf{P} = \{\mathbf{x} \in \mathbb{R}^d \mid l_i \leq x_i \leq u_i \; \forall i = 1..d\} \tag{7.5}$$

is the feasible set of solutions.

The optimisation problem can be further broken down into smaller optimisations, which makes the method scalable concerning the length of the robot program. The black-box optimisation reasoning can be applied to the bin-picking process by considering each pick-and-place of each type of part separately and independently optimising energy consumption.

The modelled optimisation problem becomes as below:

$$\min_{\mathbf{x}_1 \in \mathbf{P}_1} E(\mathbf{x}_1) + \ldots + \min_{\mathbf{x}_N \in \mathbf{P}_N} E(\mathbf{x}_N) \tag{7.6}$$

where N is the number of separable and independent operations.

7.5.2 Pick-and-place optimisation

Before optimising the entire bin-picking process, the effect of an industrial robot's operational parameters on individual pick-and-place operations that comprise the whole bin-picking process was optimised and analysed. A single pick-and-place operation was optimised for energy efficiency by fine-tuning an industrial robot's operational parameters, such as velocity and acceleration.

7.5.2.1 Constraints for pick-and-place

Each pick-and-place operation in the bin-picking process was divided into the following operations to analyse the effect of run time operational parameters:

- **M1.** Move to the "pick approach" pose.
- **G1.** Open the gripper.
- **M2.** Move to the "pick" pose.
- **G3.** Close the gripper.
- **M3.** Move to the "pick retract" pose.
- **M4.** Move to the "place approach" pose.
- **M5.** Move to the "place" pose.
- **G3.** Open the gripper.
- **M6.** Move to the "place retract" pose.
- **M7.** Move to the "home" pose.

The labels M and G refer to different types of actions, and there are 7 motions labelled as M and 3 gripping actions labelled as G. The motion parameters can be adjusted to affect energy consumption.

In the optimisation process for setting the operating parameters of each motion sequence, the dual objectives are to minimise energy consumption while maintaining a cycle time that does not exceed a specified threshold. The latter is particularly crucial, as manufacturing operations' cycle time is a key performance indicator.

In this experiment, a human operator established a maximum allowable cycle time of 6 seconds for a single pick-and-place operation, aligning with the throughput requirement of 10 sorting operations per minute in the capacity adaptation scenario.

In a real manufacturing scenario, there are some challenging situations where the energy consumption is demanding, and the robot performing tasks requires a given velocity and acceleration. Therefore, each parameter was separately constrained in addition to a maximum execution time constraint to meet such requirements as shown in Table 7.1.

Motion type	**Velocity range**	**Acceleration range**	**Configs #**
Motion 1	$[50, 100]$	$[50, 100]$	$50 * 50 = 2500$
Motion 2	$[5, 40]$	$[5, 40]$	$35 * 35 = 1225$
Motion 3	$[5, 40]$	$[5, 40]$	$35 * 35 = 1225$
Motion 4	$[50, 100]$	$[50, 100]$	$50 * 50 = 2500$
Motion 5	$[5, 40]$	$[5, 40]$	$35 * 35 = 1225$
Motion 6	$[50, 100]$	$[50, 100]$	$50 * 50 = 2500$
Motion 7	$[50, 100]$	$[50, 100]$	$50 * 50 = 2500$

Table 7.1: Minimum and maximum allowed velocity and acceleration values for motions M1-M7.

Table 7.1 also shows the number of possible configurations for each type of motion. These values are fixed for this specific manufacturing process, and the parameter values are displayed in percentages of the robot's maximum possible velocity and acceleration.

7.5.2.2 Optimisation using O3PARAMS algorithm

Pick-and-place operation was optimised using the O3PARAMS algorithm in the following way:

1. Initially, energy data was collected via an energy monitoring app.
2. The energy data was then resampled to align with the timestamps and to create regular time intervals.
3. Data were stored in the database to calculate total power consumption, which was then incorporated into the model.
4. Subsequently, the optimisation loop continuously fine-tuned the operational parameters of an industrial robot.
5. Lastly, the algorithm recommended new parameters for the robot, and the robot parameters were updated accordingly.

Robot energy consumption was measured with minimum and maximum allowed parameter configurations for the described pick-and-place operation. For clarity, these configurations are referred to as $B_{\min}$ and $B_{\max}$ respectively. The same pick-and-place operation was repeated ten times for each configuration, considering the noisy nature of energy consumption measurements.

A key feature of the O3PARAMS algorithm is its modularity and capability for online optimisation. Therefore, the modularity of the O3PARAMS algorithm and its suitability for an industrial environment were tested using three different well-known algorithms: Random Search (RS)[151], Generalised Simulated Annealing (SA)[152], and Bayesian Optimisation (BO)[153]. The selection of these algorithms was dependent on the task.

Various algorithms can be found in the literature, yet not all are appropriate for solving black-box optimisation problems. In this context, the selection of

optimisation algorithms was restricted to derivative-free optimisation algorithms due to the absence of derivative information.

Random Search, Simulated Annealing, and Bayesian Optimisation are nondeterministic algorithms, which means the output differs in every run, even with the same input parameters. Therefore, each algorithm was run 100 iterations ten times to ensure a fair comparison.

7.5.2.3 Analysis of optimisation results

The box plots and related statistics of the optimisation results can be seen in Figure 7.13 and Table 7.2, respectively.

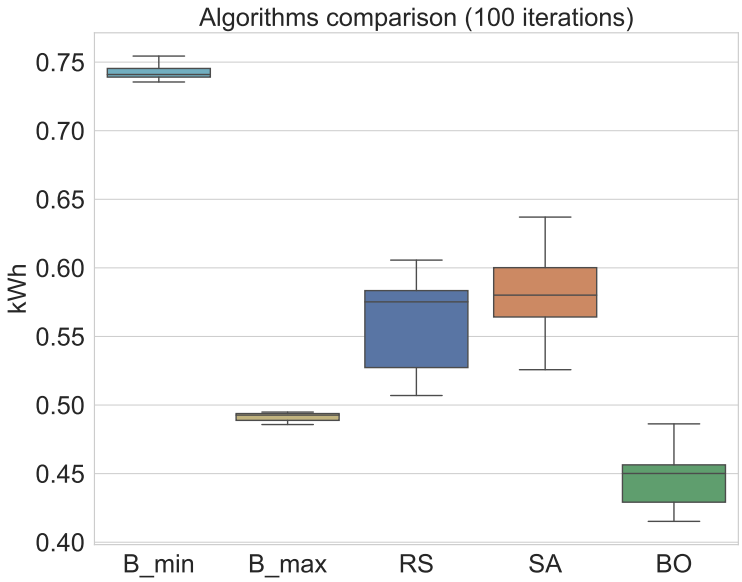

Figure 7.13: Comparing algorithms and baselines for the energy consumption optimisation using O3PARAMS algorithm.

Figure 7.13 indicates that the maximum allowed parameters configuration results in less energy consumption than the minimum allowed parameters configuration. However, generally, the relationship is non-linear, and the maximum allowed parameters configuration does not always yield optimal energy consumption. Various authors have demonstrated this [154–157] and is supported by the results obtained

in this experiment.

Method	Min.	Max.	Median	Mean	Std.
$B_{\min}$	0.736	0.754	0.741	0.743	0.006
$B_{\max}$	0.486	0.495	0.493	0.491	0.003
Simulated Annealing	0.507	0.606	0.575	0.559	0.033
Random Search	0.526	0.637	0.580	0.579	0.033
Bayesian Optimisation	**0.415**	**0.486**	**0.450**	**0.446**	0.020

Table 7.2: Experiment statistics. Values are in kWh.

From Figure 7.13 and Table 7.2, it can be seen that all optimisation algorithms outperform the $B_{\min}$ baseline. However, only Bayesian optimisation outperforms $B_{\max}$. Bayesian optimisation outperforms $B_{\max}$ on average by 9%, and the worst case of Bayesian optimisation is similar to the best case of $B_{\max}$, which proves that the fastest possible option is not the most energy optimal motion.

O3PARAMS algorithm was run with three previously chosen optimisation algorithms starting from the same initial parameters configuration for 200 more iterations to check the possibility of further improvement. Figure 7.14 shows the energy savings achieved by the optimisation algorithms compared to the baseline results.

It can be observed from Figure 7.14 that the benchmarked algorithms Random Search, Simulated Annealing, and Bayesian Optimisation outperform the $B_{\min}$ by 28.38%, 29.73%, and 44.59%, respectively. However, Random Search and Simulated Annealing methods do not outperform the $B_{\max}$ because usually, Random Search and Simulated Annealing methods require thousands of iterations to find the optimal solution. Bayesian optimisation outperforms $B_{\max}$ by 16.32% in as few as 40 iterations.

The parameter values found by three optimisation algorithms are shown in Table 7.3, and the discovered velocity and acceleration values are not the fastest or slowest motion values. These results are consistent with the literature's related work and show the non-linear relationship between operating parameters and energy

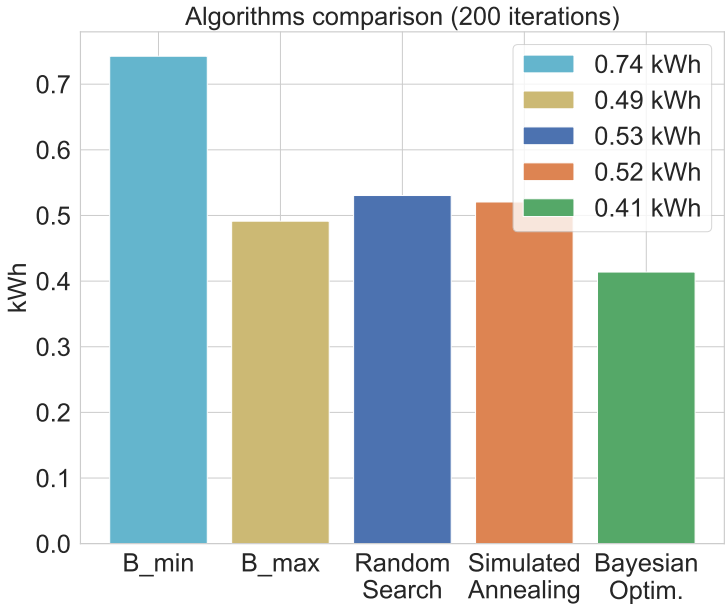

Figure 7.14: The energy consumption optimisation results for pick-and-place operation.

consumption.

	RS		SA		BO	
Motion type	v.	a.	v	a.	v.	a.
Motion 1	100	52	96	50	98	64
Motion 2	13	30	19	31	19	40
Motion 3	6	32	29	30	19	39
Motion 4	46	68	88	92	50	100
Motion 5	26	34	13	31	12	40
Motion 6	57	91	96	88	75	85
Motion 7	95	63	66	63	71	72

Table 7.3: Optimised velocity and acceleration values. This table shows velocity (v.) and acceleration (a.) values found by three optimisation algorithms: Random Search (RS), Simulated Annealing (SA), Bayesian Optimisation (BO).

Figure 7.15 displays the cumulative minimum curves for each optimisation algorithm for 200 iterations. Cumulative minimum curves are useful to visualise the speed of finding solutions. As shown in Figure 7.15, the Simulated Annealing algorithm finds its solution in 21 iterations. However, the algorithm does not improve any more after that. The Random Search algorithm finds its best solution in 166 iterations. However, the solution found is not better than the solution

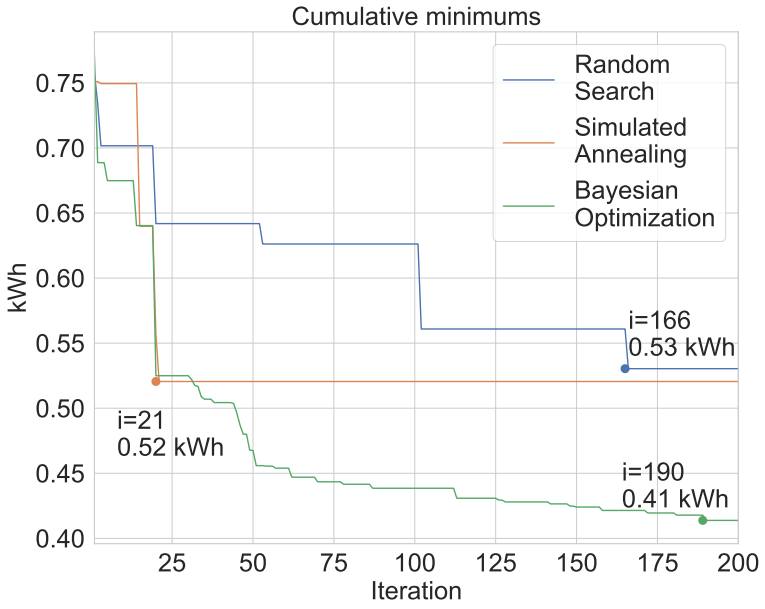

Figure 7.15: The cumulative minimum values for three different optimisation results.

found by the Simulated Annealing algorithm. The Bayesian optimisation finds its best solution in 190 iterations. However, after 30 iterations, it outperforms the Simulated Annealing algorithm, and after 40 iterations, it outperforms all the baselines. Consequently, it can be concluded that Bayesian optimisation is a relatively data-efficient algorithm.

Figure 7.16 shows the outcome of energy consumption as a function of the velocity and acceleration parameters. The figure on the left shows the energy consumption when all parameters are fixed except the velocity parameter of motion M7. It can be observed that the relationship between velocity and energy consumption is non-linear, i.e., the minimum energy consumption is achieved when the velocity is around 70%. Similarly, the right sub-figure shows the energy consumption as a function of the acceleration parameter of motion M7. The non-linearity property also holds for this parameter. Figure 7.17 shows the contour plot of energy consumption as a function of the velocity and acceleration parameters together. This plot also reveals how the optimisation algorithm reaches the optimal point. Initially, the parameters are sampled from different regions. However, as the opti-

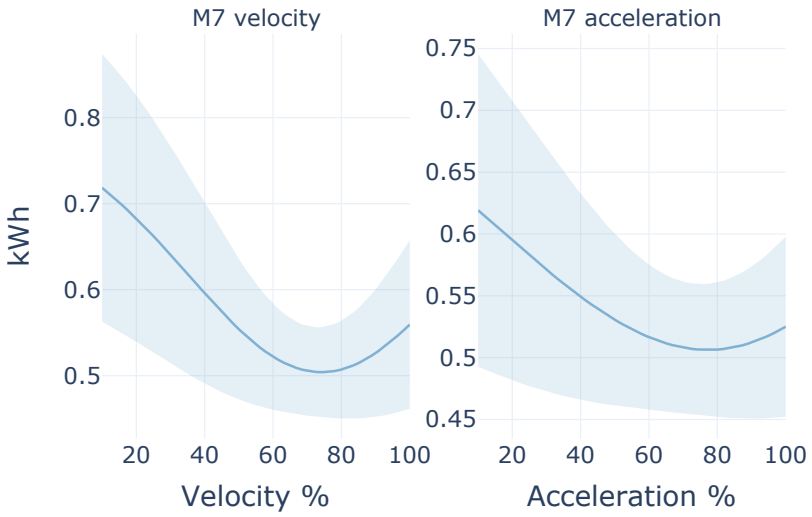

Figure 7.16: Slice plot for motion 7. (Left) Velocity varies; other parameters are fixed. (Right) Acceleration varies; others are fixed.

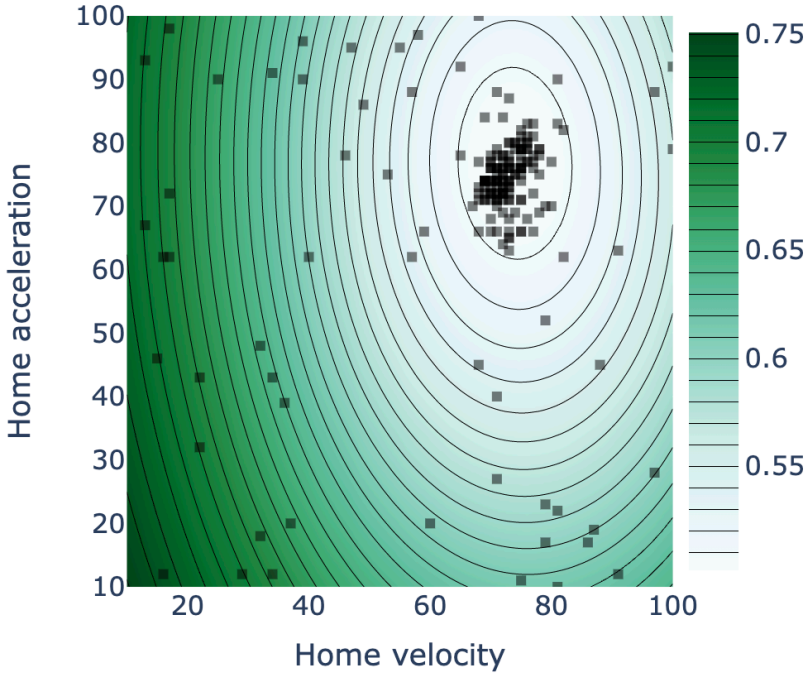

Figure 7.17: Contour plot for motion M7 as a function of velocity and acceleration. Other parameters are fixed.

misation progresses, more and more parameters are sampled in the neighbourhood of the optimal parameters.

7.5.3 Optimisation of bin-picking process

After optimising and analysing the pick-and-place operation, the optimisation app was created and integrated into the global applications repository. Then the goal shifted towards optimising the whole bin-picking process by reusing the results and optimisation apps.

In the process of picking and placing each of the 3 parts of the pipe coupler, 7 distinct motions are involved. However, the parameters associated with these movements differ for each of the 3 parts, referred to as Part-1, Part-2, and Part-3, with distinct lower and upper bounds.

The optimisation applications previously developed were adapted for each part according to their specific lower and upper bounds. In this experimental setup, the optimisation was carried out over 200 iterations, a number that is the same as the number of iterations employed in optimising pick-and-place operation. The outcomes of each optimisation scenario were benchmarked against $B_{\min}$ and $B_{\max}$ baselines, as shown in Figure 7.18.

The results from the optimisation procedure applied in this experimental setup are shown in Figure 7.18. Optimisation of each part facilitated by the optimisation app achieved a marked improvement in parameter values compared to the $B_{\max}$ baseline. The optimisation process may be discontinued upon the determination of satisfactory motion parameters. The optimised parameter values discovered can then be leveraged in subsequent iterations. This particular optimisation approach yielded an average energy savings of 25%.

Optimisation apps developed for optimising the energy consumption of a single pick-and-place operation proved effective in optimising the whole bin-picking process.

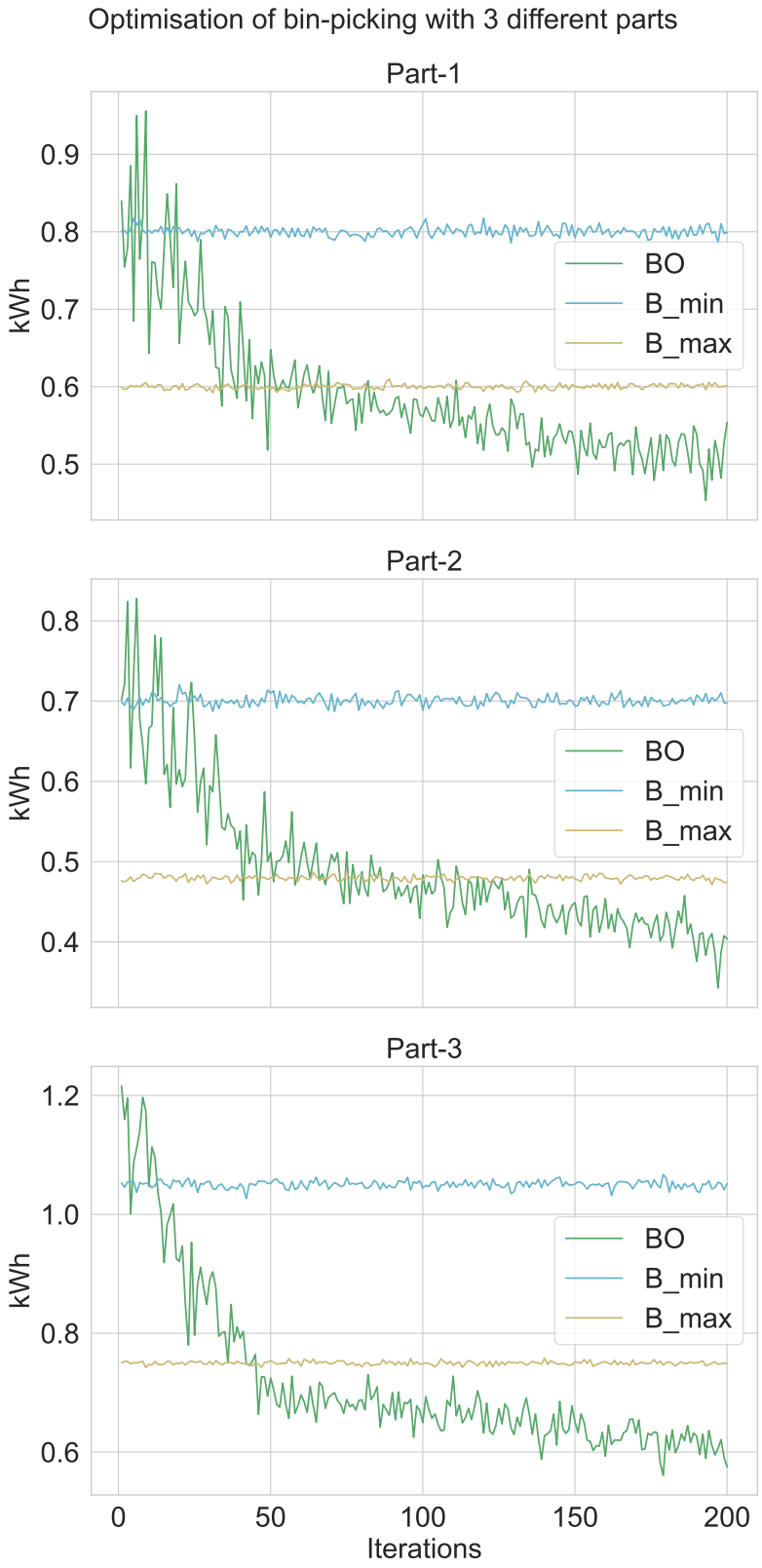

Figure 7.18: The plot shows the optimisation results of bin-picking for three different parts separately.

7.5.4 Analysis

In Chapter 3.3.2, the following success metrics were established for the validation of optimisation and decision-making algorithms:

1. **Effective management of uncertainties and multiple objectives**: The optimisation algorithms were efficient at managing uncertainties and conflicting objectives. For example, O3PARAMS algorithm minimises energy consumption while adhering to a maximum allowable cycle time of 6 seconds. This optimisation was achieved through real-time adaptation to manufacturing environment changes, meeting the criteria for effective uncertainty management.

2. **Integration with the developed data models and apps:** The optimisation algorithms were fully integrated with the object-oriented data model and the manufacturing apps. The energy monitoring app, for instance, provided real-time energy data that the optimisation algorithm used to fine-tune operational parameters. This seamless integration validates the algorithms' capability to work with the developed data models and apps.

In summary, the optimisation and decision-making algorithms successfully met the established criteria. The O3PARAMS algorithm was notably effective in real-time parameter tuning, contributing to the system's adaptability to dynamic conditions. These algorithms managed uncertainties and multiple objectives effectively and integrated seamlessly into the developed data models, utilising the modular architecture of the manufacturing apps for real-time adjustments. Therefore, the algorithms effectively address the challenges posed by dynamic and uncertain manufacturing environments, thereby validating the research contributions made in Chapter 6.

7.6 Chapter Summary

This chapter aimed to experimentally validate the research contributions delineated in the preceding technical chapters, specifically those related to object-oriented data models, manufacturing apps, and optimisation and decision-making algorithms.

The validation methodology adhered to the framework outlined in Chapter 3.3. Two manufacturing processes served as the experimental backdrop: sorting cylinders and bin-picking parts of industrial pipe couplers. These processes were chosen for their distinct challenges and mechanisms, thereby providing a comprehensive testing ground for the proposed solutions.

The validation process was structured to sequentially assess three research hypotheses:

1. The object-oriented data model's adaptability and robustness in handling changes in manufacturing requirements.
2. The manufacturing apps' effectiveness in achieving interoperability and plug-and-produce capabilities.
3. The optimisation and decision-making algorithms' efficiency in managing uncertainties and multiple objectives while integrating with the developed data models and apps.

For the first hypothesis, the object-oriented data model demonstrated its adaptability by efficiently responding to changes in manufacturing requirements, such as transitioning from sorting to bin-picking and scaling the number of operations. The model also effectively represented both static and dynamic aspects of the manufacturing process, thereby validating its robustness and versatility.

The second hypothesis focused on the manufacturing apps developed using the

MAPPDK. These apps showcased seamless integration with diverse equipment and facilitated rapid deployment and scaling, thereby confirming their effectiveness in enhancing interoperability and plug-and-produce capabilities.

Lastly, the optimisation and decision-making algorithms proved adept at managing uncertainties and multiple objectives. They integrated seamlessly with the object-oriented data models and manufacturing apps, demonstrating their capability to adapt in real-time to dynamic manufacturing conditions.

Each research hypothesis was successfully validated, and a holistic solution to the MSC problem was achieved. This holistic solution met all the predefined criteria, confirming that each stage of the research was dependent on the successful validation of the preceding stage. Therefore, the chapter successfully validates the research contributions and presents a comprehensive solution to the challenges posed by modern manufacturing systems.

While the validation process conducted in this chapter provides strong evidence for the efficacy of the proposed solutions, it is imperative to extend this validation to real-world industrial settings. Therefore, the subsequent chapter aims to bridge this gap by presenting case studies where the validated object-oriented data models, manufacturing apps, and optimisation algorithms are deployed in real industrial use-cases.

These case studies will not only confirm the findings of this chapter but also provide actionable insights into how the proposed solutions can be tailored to meet the unique challenges of different manufacturing environments. The industrial validations will substantiate the research contributions further and provide a robust framework for their practical implementation.

Chapter 8

Industrial validations

8.1 Introduction

In the preceding chapter, the focus was primarily on the experimental validation of the research contributions, specifically in object-oriented data models, manufacturing apps, and optimisation and decision-making algorithms. The experiments were carried out in a controlled environment, targeting two specific manufacturing processes: sorting cylinders and bin-picking parts of industrial pipe couplers. While the results are promising and substantiate the research hypotheses, the question which remains is how these findings translate to real-world, complex industrial settings.

Therefore, this chapter addresses the above question by extending the validation process to more intricate and diverse manufacturing environments. The objective is to assess the generalisability and adaptability of the proposed methodology, thereby providing a more robust validation of its efficacy. Therefore, two industrial validations of the proposed research contributions were carried out.

The first industrial validation, conducted at the Centre for Aerospace Manufacturing at the University of Nottingham, delves into the aerospace sector. This industry is characterised by stringent quality requirements and complex assembly processes, making it an ideal setting for evaluating the proposed research contributions for the MSC problem.

The second industrial validation was conducted at Mondragon University, Spain, as part of the DiManD project. This validation aims to demonstrate the applicability of the research contributions in custom product manufacturing, particularly focusing on machining process planning. Custom manufacturing often involves high variability and requires flexible planning and optimisation strategies.

The next sections delve deeper into the details of the above industrial validations.

8.2 Hinged product assembly in aerospace manufacturing

8.2.1 Experimental setup

The experiments in this section were conducted at the Centre for Aerospace Manufacturing at the University of Nottingham. The validation involves assembling the "generic hinged product", which represents the family of small-box hinged products, such as rudders and elevators. These components share similar dimensions and construction principles, allowing them to be assembled using shared capabilities.

Employing a flexible assembly system makes it possible to adapt to different sizes and specific requirements of various aircraft and product types. The task is to find optimal manufacturing configurations for changing and multiple conflicting manufacturing costs.

The experimental setup of the assembly process used in this experiment is shown in Figure 8.1, and Table 8.1 summarises the process as having 19 sequential operations classified into 4 distinct capabilities: (a) Empty jig frame (b) Upper beam, lower beam and two skin locations are loaded, with hingeline, ribs and spars assembled and measured (Op 1-5), (c) Upper skin assembled with three profile board supports (Op 6-10), (d) Back view of (c) and lower beam to be removed (Op 11), (e) Lower skin assembled, three profile boards and two skin locations are to be removed (Op 12-18), (f) Final inspection (Op 19).

In contrast to earlier experiments that used UML object diagrams to depict the object-oriented data model, this experiment opted for a tabular presentation of data to avoid excessive clutter that stems from many connections between manufacturing assets, capabilities, and requirements.

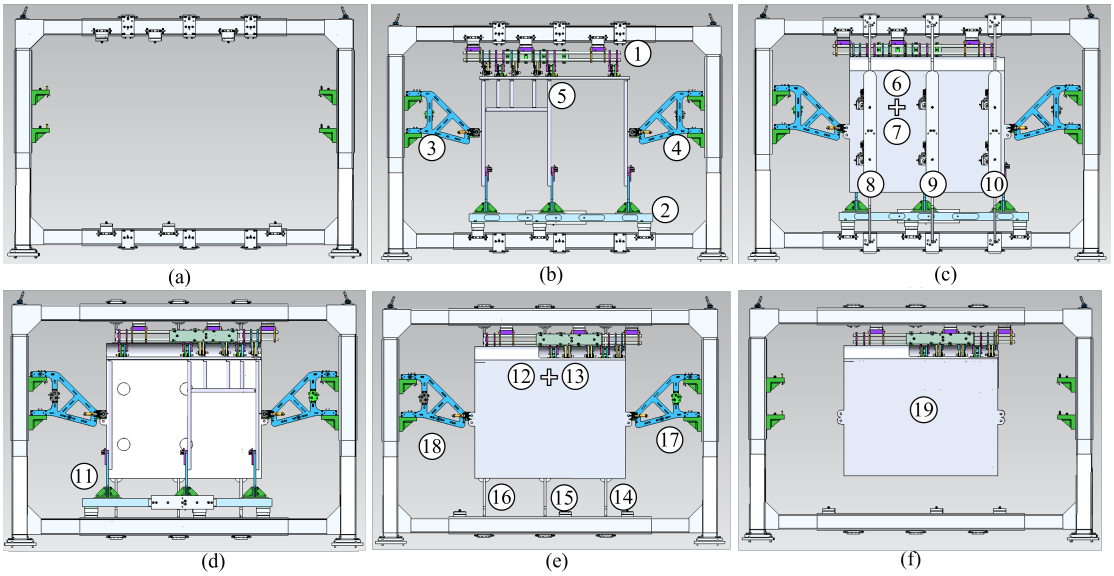

Figure 8.1: Hinged product assembly sequence.

Operation	Capability ID	Capability name
1-4	B1	Pick and place accurate
5	B3	Inspection
6	B2	Pick and place compliant
7	B4	Drilling
8-11	B1	Pick and place accurate
12	B2	Pick and place compliant
13	B4	Drilling
14-18	B1	Pick and place accurate
19	B3	Inspection

Table 8.1: Assembly process description and required capabilities

Detailed information about the available manufacturing assets can be found in Table 8.2. In total, there are seven base assets and eight auxiliary assets.

Table 8.3 presents information on the available manufacturing configurations. Each row of the table represents the capability and capacity of a specific manufacturing configuration, as indicated by the corresponding column. Furthermore, Table 8.3 also shows the normalised unitless recurring costs associated with each manufacturing configuration and the assets involved.

Using Tables 8.1, 8.2, and 8.3, it can be observed that the production of a single product requires 13 operations with B1 capability, 2 operations with B2 capability, 2 operations with B3 capability and 2 operations with B4 capability.

Asseet ID	Asset name	Asset type
A1	ABB IRC6700	base
A2	FANUC M900	base
A3	KUKA KR270	base
A4	FANUC M800iA	base
A5	KUKA Titan	base
A6	Drilling end effector	auxiliary
A7	Pick and place end effector	auxiliary
A8	Photogrammetry inspection end effector	auxiliary
A9	Skin pick and place end effector	auxiliary
A10	AGV	base
A11	VSTARS cameras	auxiliary
A12	Reconfigurable floor	base
A13	Tool changer set 3200Nm	auxiliary
A14	Tool changer set 5000Nm	auxiliary
A15	End effector tool storage	auxiliary

Table 8.2: Available manufacturing assets

	M1	**M2**	**M3**	**M4**	**M5**
B1	15	20	16	-	21
B2	15	20	16	-	21
B3	20	20	20	-	20
B4	14	20	14	12	22
B5	12	18	13	10	20
B6	6	6	6	-	6
B7	30	30	30	-	30
B8	10	15	10	-	10
B9	1	1	1	2	1
Recurr. costs	89.0	88.0	87.0	29.0	90.0
Assets	A1, A6 - A13, A15	A2, A6 - A13, A15	A3, A6 - A13, A15	A4, A7, A12	A5 - A13, A15

Table 8.3: Available manufacturing configurations

The manufacturing requirements based on the data modelling presented in Chapter 4.3 are given as follows.

In the first manufacturing requirement scenario, the demand is to produce 10 parts

per hour for 1000 hours, i.e., based on Equation 3.7:

$$D_1 = [(B_{1,1} = \text{“}B1\text{”}, 140), (B_{1,2} = \text{“}B2\text{”}, 20), \\ (B_{1,3} = \text{“}B1\text{”}, 20), (B_{1,4} = \text{“}B4\text{”}, 20)] \tag{8.1}$$

In the second manufacturing requirement period, the capacity requirement is doubled, i.e., the requirement is to produce 20 parts per hour for 1000 hours, Equation 3.7 becomes:

$$D_2 = [(B_{2,1} = \text{“}B1\text{”}, 280), (B_{2,2} = \text{“}B2\text{”}, 40), \\ (B_{2,3} = \text{“}B1\text{”}, 40), (B_{2,4} = \text{“}B4\text{”}, 40)] \tag{8.2}$$

Then manufacturing requirement periods with a sequence of demands and demand period lengths are represented as below according to the Equation 3.8:

$$\mathcal{R} = [(D_1, 1000), (D_2, 1000)] \tag{8.3}$$

The overall aim is to discover the optimal manufacturing configurations, taking into account investment, recurring, and transition costs as outlined in Chapter 3.2.1.3, allowing smooth transitions between manufacturing setups or shifts in capacities.

Traditional methods would simply duplicate the optimal initial manufacturing setup once the capacity increases twofold. However, this experiment suggests that a more intelligent approach is feasible.

8.2.2 Optimisation and decision-making for capabilities and capacities

According to the mathematical formalisation of the manufacturing systems configuration (MSC) problem defined in Chapter 3.2.1.4, this problem has three con-

flicting objectives: investment, recurring, and transition costs, and there are two types of constraints in this problem: demand-satisfaction constraints, Equation 3.11, and integrality constraints, Equation 3.12.

This experiment used NSGA-II: Non-dominated Sorting Genetic Algorithm[158] to find Pareto efficient solutions. The pymoo[159] library implementation of the NSGA-II algorithm was used, which resulted in 1000 non-dominated Pareto efficient solutions, depicted in Figure 8.2. Since objectives have different scales, solutions were normalised by approximate ideal and nadir points.

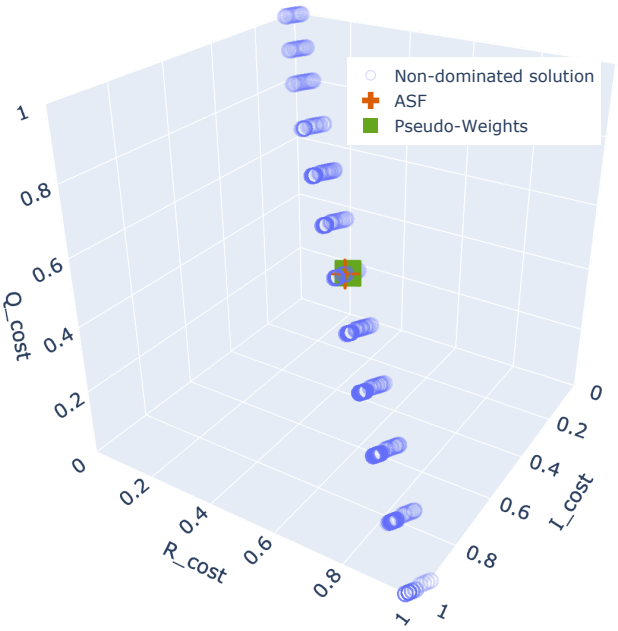

Figure 8.2: Normalised Pareto efficient solutions consisting of 1000 non-dominated solutions.

Achievement scalarisation function (ASF)[160] decomposition and pseudo-weights (PW)[161] were analysed to select a solution from the Pareto front in this experiment. ASF and PW require subjective weights for each objective cost. The weights represent the importance given to each objective cost and must sum to 1. In this scenario, the results for the importance of investment, recurring and

transition costs are weighted as $[1/3, 1/3, 1/3]$.

Table 8.4: Selected solutions by the ASF decomposition and PW methods

$D_{t,i}$	ASF	PW
$D_{1,1}$	M2 (x6) M5 (x1)	M2 (x6) M5 (x2)
$D_{1,2}$	M2 (x2) M3 (x1)	M2 (x1)
$D_{1,3}$	M2 (x1)	M2 (x1) M5 (x1)
$D_{1,4}$	M2 (x2) M4 (x4)	M2 (x1) M3 (x1) M4 (x4)
$D_{2,1}$	M2 (x9) M3 (x1) M5 (x4)	M2 (x9) M3 (x1) M5 (x4)
$D_{2,2}$	M2 (x1) M5 (x1)	M2 (x1) M5 (x1)
$D_{2,3}$	M2 (x1) M5 (x1)	M2 (x1) M5 (x1)
$D_{2,4}$	M4 (x4)	M4 (x4)

The final solutions selected by the ASF decomposition and PW are shown in Table 8.4. As seen in Figure 8.2 and Table 8.4, the solutions selected by ASF and PW are very close. In particular, the selected configurations only change for the first demand period, although the number of configurations is the same for both solutions. Closeness in solutions can be explained by having the same weight vector for ASF and PW.

Figure 8.2 and Table 8.4 show how a decision-maker can benefit from the output of the optimisation and decision-making algorithms by having variability in possible manufacturing configurations while distilling 1000 solutions into 2 by choosing preference vectors.

8.2.3 Optimisation of operational parameters

Following the determination of optimal manufacturing configurations that satisfy the required capabilities and capacities with respect to costs, the subsequent objective was to optimise the operational parameters of one of the manufacturing assets involved, thus ensuring the validity of the whole optimisation methodology.

Within the context of this aerospace manufacturing experiment, a FANUC M800iA industrial robot and a drilling end effector, controlled by a CNC machine, were used to optimise energy-efficient drilling of holes through layers of 6mm thick aluminium and 6mm thick cast acrylic.

The primary objective of this experiment was to determine the optimal spindle speed and feed rate parameters to minimise energy consumption during the drilling process.

The setup for this experiment is depicted in Figure 8.3.

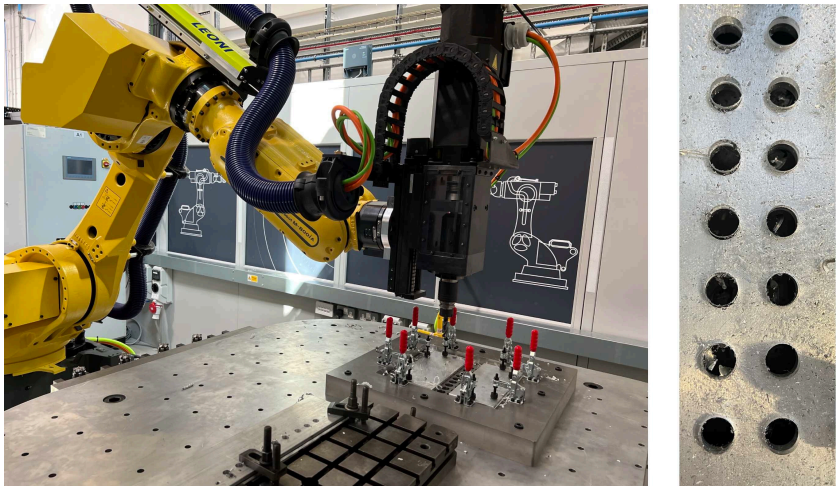

Figure 8.3: Optimisation of drilling. The aim is to energy efficiently drill holes through 6mm thick aluminium and 6mm thick cast acrylic layers.

The approach and retract positions of the robotic arm were kept constant and the total drilling time was kept within a 20-second constraint. The maximum spindle speed was allowed to fluctuate between 1500 and 2500 rpm, while the feed rate was set to vary from 150 to 180 mm/min, as detailed in Table 8.5.

Parameter	Range
Spindle speed	1500 - 2500 rpm
Feed rate	150 - 180 mm/min

Table 8.5: Parameter ranges for the drilling experiment

During this experiment, optimisation apps developed in previous experiments were re-used with minimal modifications to the drilling experiment. In particular, the O3PARAMS algorithm was used to optimise the drilling process using a black-box model approach. Bayesian optimisation was used in real time on the actual equipment to achieve this goal.

Total energy consumption, including spindle and servo motors, was measured and subsequently fed into the O3PARAMS algorithm. This algorithm suggested new parameter values, and the optimisation loop continued until convergence was reached. The convergence was determined when no further improvements were made, that is, the process was stopped when the algorithm repeatedly suggested the same parameters.

After only 25 iterations of the optimisation process, energy consumption savings of 17.64% were achieved. The cumulative minimum graph for the Bayesian optimisation results is shown in Figure 8.4.

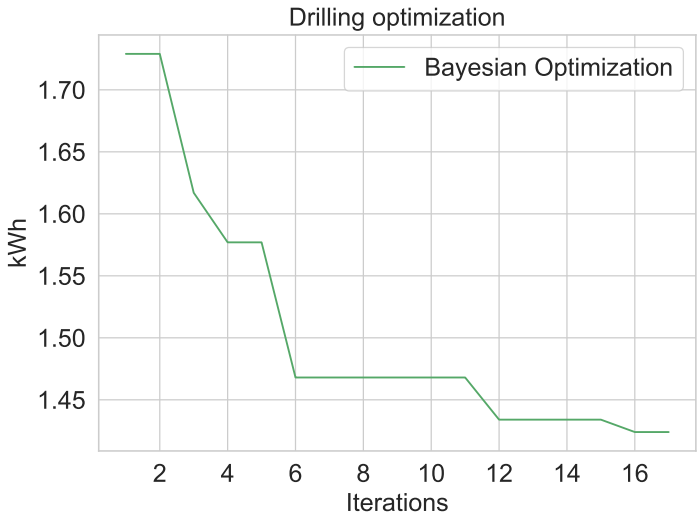

Figure 8.4: The cumulative minimum values for drilling optimisation results.

Figure 8.5 presents the slice graphs of energy consumption in relation to the spindle speed and feed rate parameters.

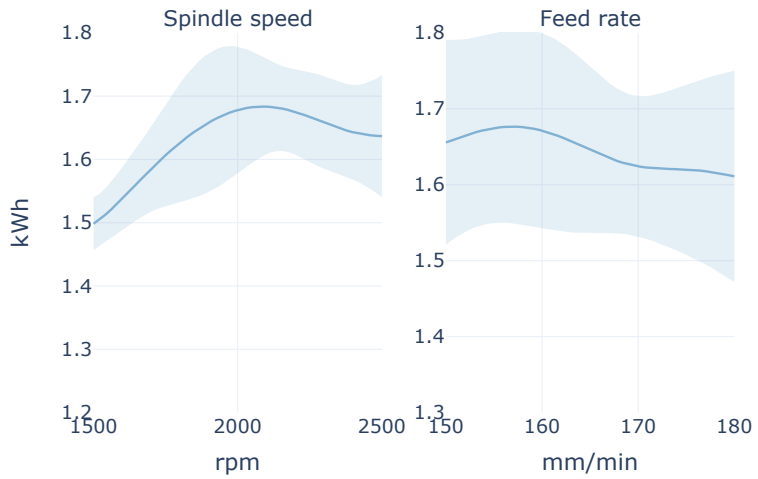

Figure 8.5: Slice plot for drilling. (Left) Spindle speed varies, and the feed rate is fixed. (Right) The feed rate varies, the spindle speed is fixed.

The slice plots in Figure 8.5 show the non-linear relationship between spindle speed and feed rate for the optimal energy consumption in the drilling operation.

These results are consistent with previous experiments, demonstrating the non-linear relationship between process parameters and energy consumption and highlighting the importance of optimisation for energy-efficient manufacturing processes.

This experiment of optimisation of operational parameters of FANUC M800iA robot for energy efficiency demonstrates the applicability of the O3PARAMS algorithm developed in Chapter 6.3.3 to a more advanced industrial setting than bin-picking process and how the same optimisation app can be re-used.

8.2.4 Analysis

The results of the experiments conducted at the Centre for Aerospace Manufacturing at the University of Nottingham offer evidence supporting the research objectives and hypotheses outlined in this thesis. The validation process was comprehensive, covering the optimisation of manufacturing configurations and operational parameters.

All the manufacturing assets were successfully modelled using the proposed object-oriented data model presented in Chapter 4 and integrated into manufacturing configurations, which shows how object-oriented data modelling can model the manufacturing process in industrial scenarios.

The developed object-oriented data model was used to optimally select manufacturing configurations using the NSGA-II algorithm. The results indicate that these algorithms are not only effective but also versatile. They allow for a nuanced approach to manufacturing configuration. Applying ASF and PW methods for decision-making further refined these choices, providing actionable insights for manufacturing setups.

Regarding operational parameters, the experiment optimised the drilling process, achieving a significant 17.64% reduction in energy consumption using O3PARAMS algorithm. The results validate the overall optimisation methodology, confirming its applicability to real-world manufacturing scenarios.

Also, the optimisation apps from the previous experiments were re-used, further providing evidence for the holistic approach: combining object-oriented data models, manufacturing apps, optimisation and decision-making algorithms.

8.3 Decision-making for machining process planning

8.3.1 Experimental setup

The validation detailed in this section was carried out at Mondragon University, Spain, as part of a secondment programme of the DiManD project. The validation necessitated the planning of a machining process that included the balance of both investment and recurring costs. The data for this validation were provided by the Software and Systems Engineering and High-Performance Machining groups at Mondragon University.

Figure 8.6 depicts the target design of the product. Accomplishing this design involves a three-stage process:

1. **Machining phase 1:** Right-side turning operation
2. **Machining phase 2:** Left-side turning operation
3. **Machining phase 3:** Centre-side milling operation

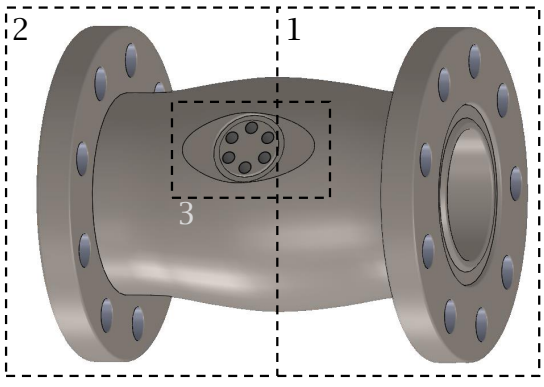

Figure 8.6: Desired shape of the product.

The first phase requires a turning lathe capable of drilling, boring, turning, facing, and slotting operations. First, the centre hole is drilled, bored, and finished.

Second, slotting and rough and finish-facing operations are performed, followed by drilling the 9 front holes. At the end of the phase, rough turning is performed from the border to the centre of the workpiece.

The second phase is performed similarly on the turning lathe. The phase starts with slotting and rough and finish-facing operations, then drilling the other 9 front holes. After that, rough turning, external finish profiling, and threading operations are executed.

Finally, the third phase requires a milling centre capable of performing face and slot milling, drilling, and threading. First, the cylindrical part is face-milled and the six holes are drilled. Then, the 6 holes are threaded and the last slot milling operation is performed.

Manufacturing requirements are provided as below:

- Capability requirements
  - Turning
  - Milling
- Capacity requirements, D: 2000 products
- Constraints:
  - Time constraint, T_D: 100 hours
  - Space constraint, S: 10 manufacturing configurations (combinations of manufacturing assets)

The available manufacturing assets are provided as:

- **Asset A0**: a CNC lathe,
- **Asset A1**: a CNC milling centre,

- **Asset A2**: a multitask CNC lathe,
- **Asset A3**: a dual-spindle CNC turning centre,
- **Asset A4**: a twin-spindle twin-turret turning centre.

The capabilities of the provided assets, along with their normalised manufacturing specifications, are detailed in Table 8.6.

Manufacturing asset	Turning lathe	Milling centre	Investment cost	Recurring cost	Setup time
A0	X		300	12	2
A1		X	700	8	3
A2	X	X	2000	20	9
A3	X		1700	5	3
A4	X	X	5000	65	10

Table 8.6: Normalised manufacturing data of the assets. The investment and recurring costs are normalised to make the costs unitless. Setup times represent the number of hours that a manufacturing asset requires to set up.

Given the target design, manufacturing requirement, and manufacturing assets, the problem is to find the optimal combination of assets, that is, optimal manufacturing configurations that meet the required demand of $D = 1000$ products over $T_D = 100$ hours.

8.3.2 Data modelling for machining process planning

The experimental data supplied had to first be converted into an object-oriented data model, following the guidelines outlined in Chapter 4 and visualised in Figure 8.7. This object-oriented data model enhances the reusability of previous research contributions, and in particular, it facilitates the usage of optimisation apps.

The specifications of the target design dictate that the minimal requirements for fulfilling demand D would involve the use of a turning lathe, a milling centre, or a combination of the two. However, in the context of the time frame demanded

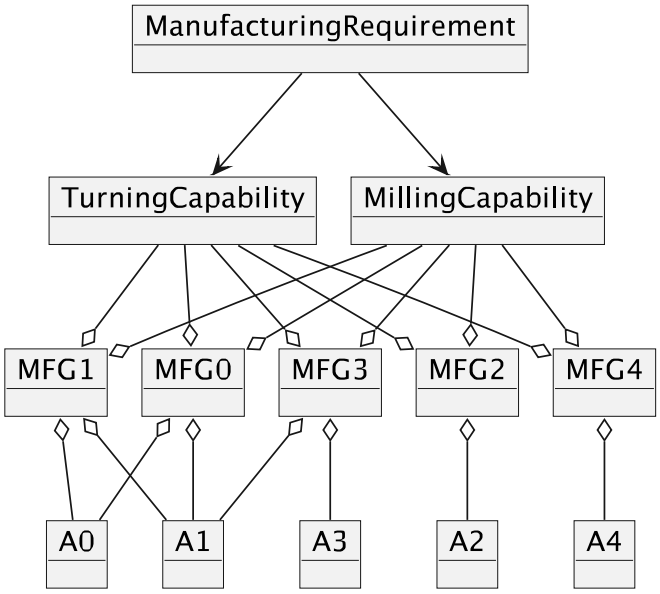

Figure 8.7: Data model for machining process planning experiment. Details of the manufacturing configurations and manufacturing assets are provided in Table 8.6, Table 8.7, and Figure 8.8

T_D, a single manufacturing asset could be insufficient. Consequently, multiple machines have been merged into manufacturing configurations, as prescribed in Chapter 4.4.1, to meet the criteria T_D adequately.

Five machine configurations have been identified from the available assets as capable of satisfying D. These configurations consist of one or more assets, and their respective sequential operational production routes are depicted in Figure 8.8.

Cfg.	Incur. cost	Recur. cost	Prod. rate	Setup time
***MFG*0**	1000.0	20.0	1.0	5
***MFG*1**	1300.0	32.0	1.5	7
***MFG*2**	2000.0	20.0	2.0	9
***MFG*3**	2400.0	13.0	0.75	6
***MFG*4**	5000.0	65.0	3.0	10

Table 8.7: Normalised manufacturing data of the machine configurations.

8.3.3 RL-based problem formulation

The problem of planning the machining process requires a formulation such as MSC problem. This formulation follows the principles of RL, facilitating the ap-

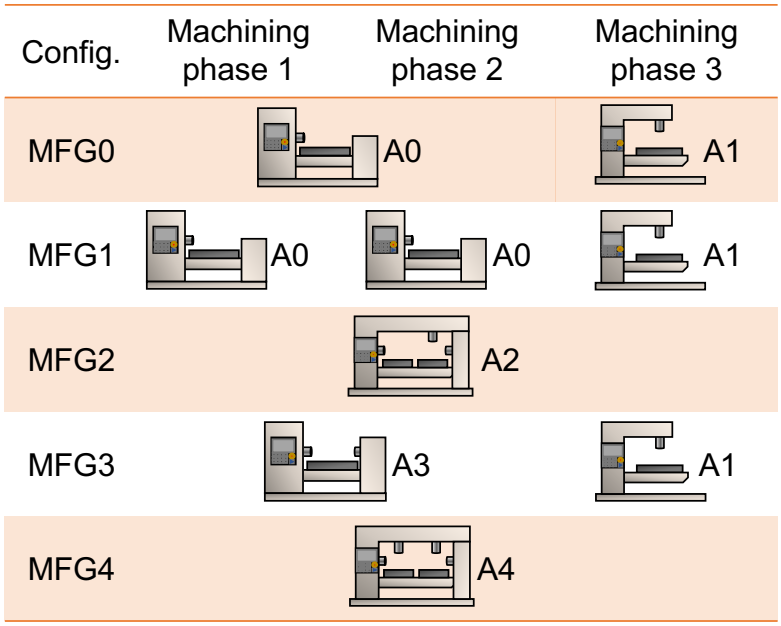

Figure 8.8: Manufacturing configurations capable to produce the desired product.

plication of RL-based sequential optimisation and decision-making. It is essential to transform manufacturing scenarios into RL states, map strategies for satisfying requirements to RL actions, and design a reward system that promotes the most efficient configurations to meet manufacturing requirements. Consequently, the following problem formulation is given which adheres to the methodologies detailed in Chapter 4 and Chapter 6:

Given:

- **Demand,** D: Number of products required.
- **Demand time,** T_D: Maximum time allowed to produce the requested products.
- **Set of manufacturing configurations,** $\mathbb{M}$: The total number of unique manufacturing configurations is $M = |\mathbb{M}|$. Each element of the set $\mathbb{M}$ is represented as MFG has the following attributes:
  - **Investment cost**: Cost of purchasing MFG.
  - **Recurring cost**: Cost of running a MFG for 1 unit of time.

  - **Production rate**: Number of products produced by the *MFG* per 1 unit of time.
  - **Setup time**: Time required to set up *MFG*.

- **Space size,** S: Maximum number of allowed manufacturing configurations to purchase.

Problem: Find the multiset of manufacturing configurations that can meet the given demand in the given demand time with minimum cost, where the cost is the sum of the incurred and recurred costs.

Since investment and recurring costs conflict in this problem, they were scalarized using the strategies discussed in Chapter 6.3.4.1.

The manufacturing RL environment and the state space for this problem were defined using the methodology and notations proposed in Chapter 6.4.2. The action space in the environment is represented as an integer between 0 and M inclusive, that is, $a \in [0, M]$ and is formally defined as

$$Step(a) = \begin{cases} \text{“buy configuration } a\text{”, if } 0 \leq a < M \\ \text{“continue production”, otherwise} \end{cases} \tag{8.4}$$

where $Step(a)$ makes an episode step in the environment.

Selected actions by an agent affect the dynamics of the environment as follows:

- Action “buy configuration a” adds a configuration a into the production space. It also pauses the remaining demand time, T_r, in the environment. Counter-intuitively, stopping the remaining demand time resembles a decision-making process in the real world where purchasing decisions can be made while production is still running.
- Action “continue production” decreases the remaining demand time and

updates the produced products.

- An agent can make purchase decisions until the space is full. As soon as the space is full, an agent exceeds all its action choices, and the environment advances independently until the termination criteria are reached.
- The environment terminates when the condition "$D_r \leq 0$ OR $T_r \leq 0$" is met.

Two main **learning principles** were defined for agents:

- **P1:** Demand must be met at any cost.
- **P2:** Total cost must be minimised.

Learning principle P1 gives a high penalty if D is not met within T_D. The penalty is defined as a function of the remaining demand and a K penalty coefficient as

$$J(D_r) = \begin{cases} -D_r K, & \text{if } D_r > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8.5}$$

where $K = R_{max} + 1$ is a penalty coefficient, and

$$R_{max} = \sum_{i=1}^{S} (\max_{1 \leq j \leq M} \mathcal{I}_j + T_D \max_{1 \leq j \leq M} \mathcal{R}_j) \tag{8.6}$$

The equation (8.5) is derived from inequality $DK > R_{\max} + (D-1)K$.

Learning principle P2 gives two negative rewards, i.e. the rewards are multiplied by a negative one. The action "buy configuration a" rewards with investment cost $\mathcal{I}_a$ of *MFG* a. The action "continue production" rewards the sum of the recurring costs of running manufacturing configurations $\sum_a \mathcal{R}_a$.

8.3.4 Analysis

The data in Table 8.7 are provided to the RL environment for manufacturing, and optimisation is performed using the Proximal Policy Optimisation[150] algorithm. Although the data in Table 8.7 are fixed, the RL environment is stochastic, that is, the investment cost, the recurring cost, the production rate, and the setup times change by $+/-10\%$ at every simulation step, resembling real industrial fluctuations. Moreover, the environment provides the possibility of defining and experimenting with different fluctuations.

Six experiments have been defined to validate the robustness of the methodology to changing demands. The demand D and the demand time T_D defined for each experiment are presented in Table 8.8, as well as the results obtained.

On the one hand, experiments E1 and E3 have enough T_D for accomplishing D (D is at least 10 times T_D). On the other hand, experiments E2, E4, and E5 have few T_D for accomplishing D (D is at least 16 times T_D). Experiment E6 represents a scenario where T_D is greater than D (D is almost one fourth of T_D). All experiments are capable of satisfying D, using the full space size S except E6 which uses only one, with remaining time T_r and usually with excess production D_r. Additionally for E1 to E5, the unit cost (Mean reward/D) has a decreasing behavior as demand rises, due to economies of scale. E6 has a particular cost behavior, as only one configuration is needed and the agent does not need to purchase more than one configuration.

Exp.	D	T_D	Cost	D_r	T_r	Purchased MFGs	Cost per part
E1	2000	150	46432	-4	44	9	23.17
E2	2000	80	93806	-25	3	10	46.32
E3	1000	100	28573	0	7	10	28.57
E4	1000	48	63545	-5	3	10	63.22
E5	400	24	55438	-7	1	10	136.21
E6	24	100	2589	-1	79	1	103.56

Table 8.8: Experiments performed to validate the RL-based methodology.

Given the stochastic nature of the proposed environment, the decision-making procedure for E3, using RL, is shown in Figure 8.9. E3 is chosen as it shows purchasing decisions throughout T_D. As seen from the figure, the decisions to buy manufacturing configurations are not done at once. The trained agent learns to buy the necessary manufacturing configurations only when it decides that demand will not be satisfied with current options demonstrating that sequential decision-making algorithms are necessary for robust MSC problem.

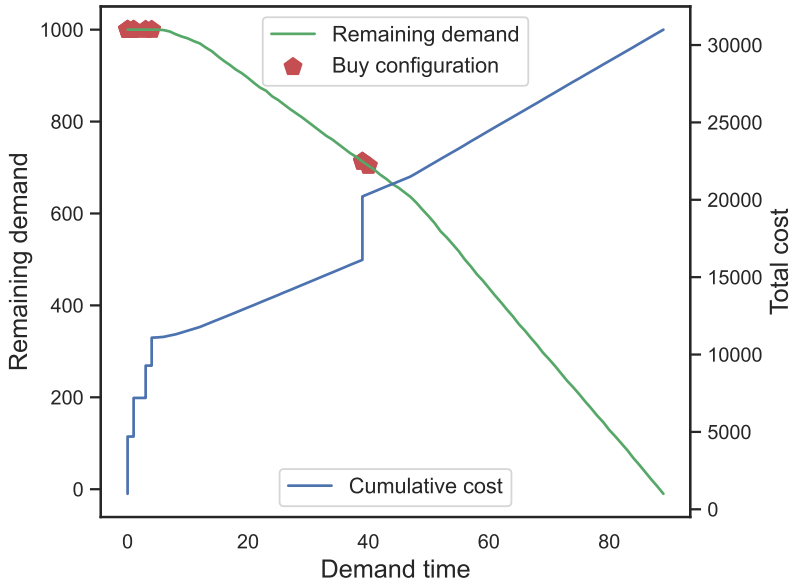

Figure 8.9: The sequential decision making process by trained agent for $D = 1000$ and $T_D = 100$

In summary, the validation experiments present a comprehensive methodology for optimising machining process planning using RL.

The methodology is validated through a series of experiments that consider various demands and time constraints. The results demonstrate the approach's efficacy in selecting optimal manufacturing configurations that meet demand within a given time frame while minimising costs.

Overall, this industrial validation contributes a valuable computational tool that can significantly aid in the complex task of machining process planning.

8.4 Chapter Summary

This chapter aimed to extend the validation of the research contributions to complex, real-world industrial settings. Two distinct manufacturing environments were chosen: aerospace manufacturing at the Centre for Aerospace Manufacturing at the University of Nottingham and custom product manufacturing at Mondragon University, Spain.

The first validation focused on the aerospace sector, known for its stringent quality requirements and complex assembly processes. The experiments successfully demonstrated the applicability of the proposed object-oriented data models and optimisation algorithms in solving the MSC problem. Notably, a significant reduction in energy consumption was achieved, validating the efficacy of the proposed methodology in real-world scenarios.

The second validation was conducted in custom product manufacturing, particularly in machining process planning. The validation demonstrated the methodology's robustness in handling high variability and complex cost structures. Proximal Policy Optimisation in a stochastic RL environment proved effective in optimising manufacturing configurations under various demand and time constraints.

Both validations substantiate the research hypotheses and demonstrate the proposed methodology's generalisability and adaptability. The results indicate that the research contributions are not merely theoretical constructs but are applicable and beneficial in tackling real-world manufacturing challenges. Therefore, this chapter validates the research contributions, significantly enhancing their credibility and practical relevance.

Chapter 9

Conclusions

9.1 Introduction

This thesis aimed to tackle the manufacturing systems configuration (MSC) problem, a problem that has become increasingly relevant due to the dynamic nature of the manufacturing sector. This research identified three core knowledge gaps from existing literature: the underutilisation of data models, software integration and interoperability challenges, and the inadequacy of traditional optimisation and decision-making methods in managing multiple uncertain objectives for the MSC problem. Importantly, the study posited that these limitations are interconnected and should be addressed holistically. To this end, the research employed a holistic approach that integrated data models, software development methodologies, and advanced optimisation and decision-making algorithms to develop a methodology for addressing the MSC problem.

The research objectives were clearly defined to guide the development of a holistic methodology for the MSC problem:

1. **Development of comprehensive and adaptable data models:** The research successfully developed an object-oriented data model that encapsulates the complexities inherent in manufacturing systems. This data model presented in Chapter 4 effectively represents tangible and intangible manufacturing assets, manufacturing configuration and other manufacturing resources capturing the complexities of the manufacturing processes.

2. **Development of plug-and-produce manufacturing software solutions:** Chapter 5 introduced the concept of "manufacturing apps", providing a theoretical and practical foundation for modular software solutions specifically tailored for the manufacturing domain. The development of manufacturing apps and a modular reference architecture advanced the state of software integration and interoperability in manufacturing systems.

3. **Development of adaptive optimisation algorithms and decision-making algorithms:** Chapter 6 tackled the optimisation and decision-making aspects of the MSC problem. Three modules were developed: changes identification, optimisation, and decision-making. These modules were built on top of the proposed data model and used the concept of manufacturing apps for deployment. Novel algorithms were developed, such as changes identification algorithm (CIDENA), online optimisation of operational parameters (O3PARAMS), and reinforcement learning-based optimisation and decision-making methodology to address multiple, conflicting, and uncertain manufacturing costs.

Each objective was validated through rigorous experimental setups presented in Chapter 7 and Chapter 8. The validation process confirmed the data models' adaptability and ability to capture the complexities of manufacturing processes, the manufacturing apps' effectiveness in achieving plug-and-produce functionality, and the efficiency of the optimisation and decision-making algorithms in managing uncertainties and multiple manufacturing objectives.

9.2 Research dissemination

The knowledge contributions in this research have resulted in several distinct and noteworthy contributions to digital manufacturing. These contributions were disseminated through eight peer-reviewed journal articles and conference publications. Furthermore, five work packages were completed as part of the European Union (EU) deliverables, and two software packages were developed. These scholarly and technical outputs are directly relevant to the thesis's various technical chapters. For ease of reference, these outputs have been comprehensively catalogued in Table 9.1 (peer-reviewed journal and conference publications), Table 9.2 (WPs), and Table 9.3 (software libraries).

Table 9.1: Journal and conference publications

#	Title	Research output	Contrib. level	Chapter relevance
1	Online and modular energy consumption optimization of industrial robots [146]	Journal publication	First author	6
2	Towards modular and plug-and-produce manufacturing apps [145]	Conference publication	First author	5
3	Optimal manufacturing configuration selection: sequential decision making and optimization using reinforcement learning [147]	Conference publication	First author	6
4	Optimal selection of manufacturing configurations using object-oriented and mathematical data models [143]	Conference publication	First author	4
5	Multi-criteria decision-making for optimal manufacturing configuration selection using an object-oriented data model and mathematical formalization [144]	Conference publication	First author	4
6	Integration of cutting-edge interoperability approaches in cyber-physical production systems and Industry 4.0 [118]	Book chapter	Co-author	5
7	Big data life cycle in shop floor – trends and challenges [162]	Journal publication	Co-author	6
8	A maturity model for the autonomy of manufacturing systems [163]	Journal publication	Co-author	6

Table 9.2: DiManD WPs

#	Title	Research output	Contrib. level	Chapter relevance
1	Definition of industrial barriers and challenges to context-aware autonomous systems with respect to current practice	EU deliverable	Main contributor	6
2	Development of a data model for proactive intelligent products	EU deliverable	Main contributor	4
3	Development of adaption strategies for context-aware autonomous systems	EU deliverable	Main contributor	6
4	State-of-the-art on integration of cutting edge interoperability approaches in manufacturing and cyber-safety and security requirements	EU deliverable	Contributor	5
5	Using big data in shop-floor – challenges and approaches	EU deliverable	Contributor	6

Table 9.3: Software libraries

#	Title	Research output	Contrib. level	Chapter relevance
1	fanucpy: Python package for FANUC industrial robots	Software library	Main developer	5
2	Manufacturing reinforcement learning environment	Software library	Main developer	6

9.3 Future work

While significant progress was made in holistically addressing the MSC problem, several challenges still exist for future works.

9.3.1 Data models

In this research, object-oriented data modelling proved to be highly effective in addressing the complexities of manufacturing processes for addressing the MSC problem. The model demonstrated adaptability, easily accommodating new requirements and scaling with additional hardware components. Furthermore, it provided a comprehensive representation of static and dynamic elements, from manufacturing assets like grippers to operational capabilities, confirming its versatility and applicability in diverse manufacturing scenarios.

However, one significant limitation lies in its inability to perform complex reasoning, a capability where semantic data models have an edge. Semantic models can capture intricate relationships and dependencies, allowing for nuanced queries and inferences. This reasoning functionality is particularly important in environments where understanding causal relationships or predicting future states based on existing data is crucial. Object-oriented models, while flexible and descriptive, may not offer the same depth of reasoning, making them less suitable for applications requiring complex logical inferences.

A hybrid approach could be highly beneficial to mitigate this limitation. By integrating elements of semantic data modelling into the object-oriented framework, the model could gain advanced reasoning capabilities without sacrificing its inherent adaptability and representational strengths. This hybrid model offers a more comprehensive solution capable of handling a broader range of tasks, from simple data representation to complex logical reasoning.

9.3.2 Manufacturing apps

The concept of "manufacturing apps" introduced in this research offers a promising pathway for enhancing software integration and interoperability in today's manufacturing systems. These apps are designed as modular solutions, allowing for quick deployment, scalability, and adaptability to meet the ever-changing demands of modern manufacturing. However, the modular architecture was tested in a controlled lab environment with modern equipment, and the manufacturing apps developed for validation experiments were limited to robotic apps and for the digital aspects of manufacturing processes. This limitation points to the need for further research to extend the modular architecture and app framework to other types of manufacturing equipment and scenarios. While the architecture has proven effective in a lab setting, its scalability, adaptability, and cyber-security in real-world, multi-equipment manufacturing environments remain unexplored.

One significant obstacle to the widespread adoption of manufacturing apps is the presence of legacy equipment in many industrial settings. These older systems frequently lack the necessary connectivity and computational capabilities for seamless integration with contemporary software solutions. The technical and economic challenge lies in retrofitting legacy equipment for compatibility with manufacturing apps, which can be complex and costly. Future research could focus on creating cost-effective solutions, such as specialised adapters or middleware, to bridge the technological divide between these legacy systems and modern manufacturing apps.

Another challenge that remains untested but crucial for manufacturing operations across multiple sites or countries is scaling the developed modular architecture for hosting manufacturing apps in different locations. Future work could focus on developing strategies for geographical scaling, possibly through cloud-based solutions or distributed computing frameworks, to ensure that the modular archi-

tecture retains its effectiveness in a more expansive setting. By addressing these challenges, the applicability and impact of manufacturing apps could be significantly broadened, making them a more viable solution for diverse manufacturing environments.

9.3.3 Optimisation and decision-making algorithms

Optimisation and machine learning algorithms, particularly black-box optimisation methods such as Bayesian optimisation and sequential decision-making algorithms like reinforcement learning (RL), have shown great promise in tackling the MSC problem. Despite their proven effectiveness, their successful deployment on edge devices poses a significant challenge due to their required computational resources. These resource-intensive algorithms, especially RL, are limited to high-power devices with GPUs.

For example, the RL environment developed in Chapter 6.4 and validated in 8.3 was performed using high computational resources which might not be available for the real-time decision-making using edge devices.

A possible avenue for future research can be developing "lightweight" versions of these algorithms that maintain the core functionality but are stripped of extraneous computational elements. This streamlined version would be more suitable for deployment on edge devices with lower computational power. These lightweight algorithms must be carefully designed to balance their resource requirements with their predictive performance.

Furthermore, deploying advanced machine learning algorithms on edge devices may benefit from future research into distributed computing and federated learning techniques. These techniques allow multiple devices to train a model, effectively pooling their computational resources collaboratively. This approach should be particularly beneficial for large-scale manufacturing systems with multiple edge

devices, allowing them to solve the MSC problem jointly. This would, however, require further research into effective methods for distributed data processing and model synchronisation and also raise new questions around data privacy and security that would need to be addressed.

9.4 Chapter summary

In summary, this research aimed to develop a holistic approach to the manufacturing systems configuration (MSC) problem by integrating data models, software solutions, and optimisation and decision-making algorithms in contrast to the existing literature, where it is done in a fragmented way.

The empirical validation of this holistic approach, as elaborated in Chapter 7 and 8, was conducted within the confines of a controlled laboratory settings. While the results are promising, it is important to acknowledge that the holistic solution presented has limitations, as discussed above. Addressing these limitations is essential for fully realising the holistic approach's potential and applicability in diverse manufacturing environments. Future research endeavours should focus on these areas to enhance the robustness and versatility of the proposed solution.

Bibliography

[1] Morteza Ghobakhloo and Ng Tan Ching. Adoption of digital technologies of smart manufacturing in SMEs. *Journal of Industrial Information Integration*, 16:100107, 2019.

[2] Hoda ElMaraghy, Laszlo Monostori, Guenther Schuh, and Waguih ElMaraghy. Evolution and future of manufacturing systems. *CIRP Annals*, 70 (2):635–658, 2021.

[3] Nigel Slack, Stuart Chambers, and Robert Johnston. *Operations management*. Pearson education, 2010.

[4] DN Sormaz and B Khoshnevis. Process planning knowledge representation using an object-oriented data model. *International Journal of Computer Integrated Manufacturing*, 10(1-4):92–104, 1997.

[5] Jie Zhao, Wai Ming Cheung, and RIM Young. A consistent manufacturing data model to support virtual enterprises. *International Journal of Agile Management Systems*, 1(3):150–158, 1999.

[6] Bing-hai Zhou, Shi-jin Wang, and Li-feng Xi. Data model design for manufacturing execution system. *Journal of Manufacturing Technology Management*, 16(8):909–935, 2005.

[7] David Goerzig, Dominik Lucke, Juergen Lenz, Timo Denner, Michael Lickefett, and Thomas Bauernhansl. Engineering Environment for Production System Planning in Small and Medium Enterprises. *Procedia CIRP*, 33:

111–114, 2015. 9th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '14.

[8] Naresh Ganesh Nayak, Frank Dürr, and Kurt Rothermel. Software-defined environment for reconfigurable manufacturing systems. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 122–129. IEEE, 2015.

[9] Pablo González-Nalda, Ismael Etxeberria-Agiriano, Isidro Calvo, and Mari Carmen Otero. A modular CPS architecture design based on ROS and Docker. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 11(4):949–955, 2017.

[10] Joao Rufino, Muhammad Alam, Joaquim Ferreira, Abdur Rehman, and Kim Fung Tsang. Orchestration of containerized microservices for IIoT using Docker. In *2017 IEEE International Conference on Industrial Technology (ICIT)*, pages 1532–1536. IEEE, 2017.

[11] Md Abdul Wazed, Shamsuddin Ahmed, and Yusoff Nukman. Uncertainty factors in real manufacturing environment. *Australian Journal of Basic and Applied Sciences*, 3(2):342–351, 2009.

[12] Wenqiang Xiao, Vernon N Hsu, and Qiaohai Hu. Manufacturing capacity decisions with demand uncertainty and tax cross-crediting. *Manufacturing & Service Operations Management*, 17(3):384–398, 2015.

[13] Md Abdul Wazed, Shamsuddin Ahmed, and N Nukman. A review of manufacturing resources planning models under different uncertainties: state-of-the-art and future directions: feature articles. *South African Journal of Industrial Engineering*, 21(1):17–33, 2010.

[14] B Denkena, J Henjes, and LE Lorenzen. Adaptive process chain optimisation of manufacturing systems. In *Enabling Manufacturing Competitiveness and Economic Sustainability: Proceedings of the 4th International Conference*

on Changeable, Agile, Reconfigurable and Virtual production (CARV2011), Montreal, Canada, 2-5 October 2011, pages 184–188. Springer, 2012.

[15] YC Liang, X Lu, WD Li, and S Wang. Cyber Physical System and Big Data enabled energy efficient machining optimisation. *Journal of cleaner Production*, 187:46–62, 2018.

[16] Yuqian Lu, Hongqiang Wang, and Xun Xu. ManuService ontology: A product data model for service-oriented business interactions in a cloud manufacturing environment. *Journal of Intelligent Manufacturing*, 30:317–334, 2019.

[17] Paul Valckenaers, Hendrik Van Brussel, Luc Bongaerts, and Jozef Wyns. Holonic manufacturing systems. *Integrated Computer-Aided Engineering*, 4 (3):191–201, 1997.

[18] Radu F Babiceanu and F Frank Chen. Development and applications of holonic manufacturing systems: a survey. *Journal of Intelligent Manufacturing*, 17:111–131, 2006.

[19] Vicente Botti and Adriana Giret. *Holonic manufacturing systems*. Springer, 2008.

[20] Regina Frei, Jose Barata, and Mauro Onori. Evolvable production systems context and implications. In *2007 IEEE International Symposium on Industrial Electronics*, pages 3233–3238. IEEE, 2007.

[21] Jose Barata, Mauro Onori, Rregina Frei, and P Leitão. Evolvable production systems: enabling research domains. In *2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2007); Toronto, Ontario, Canada 22-24 July 2007*, 2007.

[22] JC Chaplin, OJ Bakker, L De Silva, D Sanderson, E Kelly, B Logan, and SM Ratchev. Evolvable assembly systems: a distributed architecture for intelligent manufacturing. *IFAC-PapersOnLine*, 48(3):2065–2070, 2015.

[23] M Estela Peralta and Víctor M Soltero. Analysis of fractal manufacturing systems framework towards industry 4.0. *Journal of manufacturing systems*, 57:46–60, 2020.

[24] Kwangyeol Ryu and Mooyoung Jung. Goal-orientation mechanism in a fractal manufacturing system. *International Journal of Production Research*, 42 (11):2207–2225, 2004.

[25] Moonsoo Shin, Jungtae Mun, and Mooyoung Jung. Self-evolution framework of manufacturing systems based on fractal organization. *Computers & Industrial Engineering*, 56(3):1029–1039, 2009.

[26] Norio Okino. Bionic manufacturing systems-modelon based approach. In *CAM-I 18th Annual International Conference, New Orleans, Lousiana*, pages 485–492, 1989.

[27] Kanji Ueda. A concept for bionic manufacturing systems based on DNA-type information. In *Human Aspects in Computer Integrated Manufacturing*, pages 853–863. Elsevier, 1992.

[28] Wenbin Gu, Dunbing Tang, Kun Zheng, and Lei Wang. A neuroendocrine-inspired bionic manufacturing system. *Journal of Systems Science and Systems Engineering*, 20(3):275–293, 2011.

[29] N Singh. Design of cellular manufacturing systems: an invited review. *European journal of operational research*, 69(3):284–291, 1993.

[30] Ronald G Askin. Contributions to the design and analysis of cellular manufacturing systems. *International Journal of Production Research*, 51(23-24): 6778–6787, 2013.

[31] Fantahun M Defersha and Mingyuan Chen. A comprehensive mathematical model for the design of cellular manufacturing systems. *International Journal of Production Economics*, 103(2):767–783, 2006.

[32] Malte Schönemann, Christoph Herrmann, Peter Greschke, and Sebastian Thiede. Simulation of matrix-structured manufacturing systems. *Journal of Manufacturing Systems*, 37:104–112, 2015.

[33] Patrick Schumacher, Christian Weckenborg, and Thomas S Spengler. The impact of operation, equipment, and material handling flexibility on the design of matrix-structured manufacturing systems. *IFAC-PapersOnLine*, 55(2):481–486, 2022.

[34] Christian Petersson Nielsen. Matrix-structured manufacturing systems: From design to operations. *PhD Dissertation*, 2023.

[35] Hoda A ElMaraghy. Flexible and reconfigurable manufacturing systems paradigms. *International journal of flexible manufacturing systems*, 17:261–276, 2005.

[36] Yoram Koren. General RMS characteristics. Comparison with dedicated and flexible systems. In *Reconfigurable manufacturing systems and transformable factories*, pages 27–45. Springer, 2006.

[37] Marcel T Michaelis and Hans Johannesson. From dedicated to platform-based co-development of products and manufacturing systems. In *Enabling Manufacturing Competitiveness and Economic Sustainability: Proceedings of the 4th International Conference on Changeable, Agile, Reconfigurable and Virtual production (CARV2011), Montreal, Canada, 2-5 October 2011*, pages 196–202. Springer, 2012.

[38] Joseph Kimemia and Stanley B Gershwin. An algorithm for the computer control of a flexible manufacturing system. *AIIE Transactions*, 15(4):353–362, 1983.

[39] Anupma Yadav and SC Jayswal. Modelling of flexible manufacturing system: a review. *International journal of production research*, 56(7):2464–2487, 2018.

[40] Peter Kostal and Karol Velisek. Flexible manufacturing system. *International Journal of Industrial and Manufacturing Engineering*, 5(5):917–921, 2011.

[41] G Zhang, R Liu, L Gong, and Q Huang. An analytical comparison on cost and performance among DMS, AMS, FMS and RMS. In *Reconfigurable manufacturing systems and transformable factories*, pages 659–673. Springer, 2006.

[42] Yoram Koren, Uwe Heisel, Francesco Jovane, Toshimichi Moriwaki, Gumter Pritschow, Galip Ulsoy, and Hendrik Van Brussel. Reconfigurable manufacturing systems. *CIRP annals*, 48(2):527–540, 1999.

[43] Mostafa G Mehrabi, A Galip Ulsoy, and Yoram Koren. Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent manufacturing*, 11:403–419, 2000.

[44] Zhuming M Bi, Sherman YT Lang, Weiming Shen, and Lihui Wang. Reconfigurable manufacturing systems: the state of the art. *International journal of production research*, 46(4):967–992, 2008.

[45] Mostafa G Mehrabi, A Galip Ulsoy, Yoram Koren, and Peter Heytler. Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent manufacturing*, 13:135–146, 2002.

[46] Hoda A ElMaraghy. *Changeable and reconfigurable manufacturing systems*. Springer Science & Business Media, 2008.

[47] Alessia Napoleone, Alessandro Pozzetti, and Marco Macchi. A Framework to Manage Reconfigurability in Manufacturing. *International Journal of Production Research*, 2018. doi: 10.1080/00207543.2018.1437286.

[48] Ann-Louise Andersen, Thomas D. Brunoe, and Kjeld Nielsen. Reconfigurable Manufacturing on Multiple Levels: Literature Review and Research

Directions. In Shigeki Umeda, Masaru Nakano, Hajime Mizuyama, Nironori Hibino, Dimitris Kiritsis, and Gregor von Cieminski, editors, *Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth*, pages 266–273, Cham, 2015. Springer International Publishing. ISBN 978-3-319-22756-6.

[49] Amro M. Farid. Measures of Reconfigurability and Its Key Characteristics in Intelligent Manufacturing Systems. *Journal of Intelligent Manufacturing*, 2014. doi: 10.1007/s10845-014-0983-7.

[50] Amro M. Farid. Facilitating Ease of System Reconfiguration Through Measures of Manufacturing Modularity. *Proceedings of the Institution of Mechanical Engineers Part B Journal of Engineering Manufacture*, 2008. doi: 10.1243/09544054jem1055.

[51] Yoram Koren, Xi Gu, and Weihong Guo. Reconfigurable manufacturing systems: Principles, design, and future trends. *Frontiers of Mechanical Engineering*, 13:121–136, 2018.

[52] Shokraneh K Moghaddam, Mahmoud Houshmand, Kazuhiro Saitou, and Omid Fatahi Valilai. Configuration design of scalable reconfigurable manufacturing systems for part family. *International Journal of Production Research*, 58(10):2974–2996, 2020.

[53] Ayman MA Youssef and Hoda A ElMaraghy. Optimal configuration selection for reconfigurable manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 19:67–106, 2007.

[54] Filip Skärin, Carin Rösiö, and Ann-Louise Andersen. Considering Sustainability in Reconfigurable Manufacturing Systems Research–A Literature Review. In *SPS 2022-Proceedings of the 10th Swedish Production Symposium: SPS2022 Proceedings of the 10th Swedish Production Symposium*, pages 781–792. IOS Press, 2022.

[55] Kapil Gumasta, Santosh K. Gupta, Lyes Benyoucef, and Manoj Kumar Tiwari. Developing a Reconfigurability Index Using Multi-Attribute Utility Theory. *International Journal of Production Research*, 2011. doi: 10.1080/00207540903555536.

[56] M. Maniraj, V. Pakkirisamy, and R. Jeyapaul. An Ant Colony Optimization–based Approach for a Single-Product Flow-Line Reconfigurable Manufacturing Systems. *Proceedings of the Institution of Mechanical Engineers Part B Journal of Engineering Manufacture*, 2015. doi: 10.1177/0954405415585260.

[57] Lars Dürkop, Henning Trsek, Jens Otto, and Jürgen Jasperneite. A field level architecture for reconfigurable real-time automation systems. In *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, pages 1–10. IEEE, 2014.

[58] Hichem Haddou Benderbal and Lyes Benyoucef. Machine Layout Design Problem Under Product Family Evolution in Reconfigurable Manufacturing Environment: A Two-Phase-Based AMOSA Approach. *The International Journal of Advanced Manufacturing Technology*, 2019. doi: 10.1007/s00170-019-03865-1.

[59] Ayman Youssef and Hoda A. ElMaraghy. Modelling and Optimization of Multiple-Aspect RMS Configurations. *International Journal of Production Research*, 2006. doi: 10.1080/00207540600620955.

[60] Sergey A Yerofeyev, Oleg S Ipatov, Sergey A Markov, Vyacheslav V Potekhin, Angelina S Sulerova, and Viacheslav P Shkodyrev. Adaptive Intelligent Manufacturing Control Systems. *Annals of DAAAM & Proceedings*, 26(1), 2015.

[61] Ayman Youssef and Hoda A ElMaraghy. Assessment of manufacturing sys-

tems reconfiguration smoothness. *The International Journal of Advanced Manufacturing Technology*, 30(1):174–193, 2006.

[62] Kapil Kumar Goyal, Pramod Kumar Jain, and Madhu Jain. A novel methodology to measure the responsiveness of RMTs in reconfigurable manufacturing system. *Journal of Manufacturing Systems*, 32(4):724–730, 2013.

[63] Hichem Haddou Benderbal, Mohammed Dahane, and Lyes Benyoucef. A new robustness index formachines selection in reconfigurable manufacturing system. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1019–1026. IEEE, 2015.

[64] Sihan Huang, Guoxin Wang, Yan Yan, and Jia Hao. Similarity coefficient of RMS part family grouping considering reconfiguration efforts. *IEEE Access*, 6:71871–71883, 2018.

[65] Durga Prasad and SC Jayswal. Reconfigurability consideration and scheduling of products in a manufacturing industry. *International Journal of Production Research*, 56(19):6430–6449, 2018.

[66] Muhammed Ameer and Mohammed Dahane. Reconfiguration effort based optimization for design problem of Reconfigurable Manufacturing System. *Procedia Computer Science*, 200:1264–1273, 2022.

[67] George Q Huang, XY Zhang, and L Liang. Towards integrated optimal configuration of platform products, manufacturing processes, and supply chains. *Journal of operations management*, 23(3-4):267–290, 2005.

[68] Jianping Dou, Xianzhong Dai, and Zhengda Meng. Optimisation for multi-part flow-line configuration of reconfigurable manufacturing system using GA. *International Journal of Production Research*, 48(14):4071–4100, 2010.

[69] Kapil Kumar Goyal, PK Jain, and Madhu Jain. Optimal configuration selection for reconfigurable manufacturing system using NSGA II and TOPSIS. *International Journal of Production Research*, 50(15):4175–4191, 2012.

[70] Abderrahmane Bensmaine, Mohammed Dahane, and Lyes Benyoucef. A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment. *Computers & Industrial Engineering*, 66(3):519–524, 2013.

[71] Faisal Hasan, PK Jain, and Dinesh Kumar. Optimum configuration selection in reconfigurable manufacturing system involving multiple part families. *Opsearch*, 51:297–311, 2014.

[72] Kamal Kumar Mittal, Dinesh Kumar, and Pramod Kumar Jain. A systematic approach for optimum configuration selection in reconfigurable manufacturing system. *Journal of The Institution of Engineers (India): Series C*, 99:629–635, 2018.

[73] Shokraneh K Moghaddam, Mahmoud Houshmand, and Omid Fatahi Valilai. Configuration design in scalable reconfigurable manufacturing systems (RMS); a case of single-product flow line (SPFL). *International Journal of Production Research*, 56(11):3932–3954, 2018.

[74] Masood Ashraf and Faisal Hasan. Configuration selection for a reconfigurable manufacturing flow line involving part production with operation constraints. *The international journal of advanced manufacturing technology*, 98:2137–2156, 2018.

[75] Carlos Alberto Barrera Diaz, Tehseen Aslam, and Amos HC Ng. Optimizing reconfigurable manufacturing systems for fluctuating production volumes: a simulation-based multi-objective approach. *IEEE Access*, 9:144195–144210, 2021.

[76] I Drstvensek, Mirko Ficko, and J Balic. Relational database as a cogitative part of an intelligent manufacturing system. *Journal of materials processing technology*, 157:114–122, 2004.

[77] Angelo Corallo, M Espostito, Andrea Massafra, and Salvatore Totaro. A relational database management system approach for data integration in manufacturing process. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–7. IEEE, 2018.

[78] Boppana V Chowdary, KSP Rao, and Arun Kanda. A study on manufacturing flexibilities using entity-relationship models. *International Journal of Risk Assessment and Management*, 7(4):569–588, 2007.

[79] Samira Alvandi, Georg Bienert, Wen Li, and Sami Kara. Hierarchical modelling of complex material and energy flow in manufacturing systems. *Procedia CIRP*, 29:92–97, 2015.

[80] Yuhang Yang, Davis J McGregor, Sameh Tawfick, William P King, and Chenhui Shao. Hierarchical data models improve the accuracy of feature level predictions for additively manufactured parts. *Additive Manufacturing*, 51:102621, 2022.

[81] Sebastiano Gaiardelli, Stefano Spellini, Michele Lora, and Franco Fummi. A hierarchical modeling approach to improve scheduling of manufacturing processes. In *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, pages 226–232. IEEE, 2022.

[82] Lin Lin, Xin-Chang Hao, Mitsuo Gen, and Jung-Bok Jo. Network modeling and evolutionary optimization for scheduling in manufacturing. *Journal of Intelligent Manufacturing*, 23:2237–2253, 2012.

[83] Till Becker, Mirja Meyer, and Katja Windt. A manufacturing systems network model for the evaluation of complex manufacturing systems. *International Journal of Productivity and Performance Management*, 63(3):324–340, 2014.

[84] Yuyun Zhang, Shaw C Feng, Xiankui Wang, Wensheng Tian, and Ruirong Wu. Object oriented manufacturing resource modelling for adaptive process

planning. *International Journal of Production Research*, 37(18):4179–4195, 1999.

[85] Liu Chengying, Wang Xiankui, and He Yuchen. Research on manufacturing resource modeling based on the O–O method. *Journal of Materials Processing Technology*, 139(1-3):40–43, 2003.

[86] Hang-Wai Law and Tak-Man Woo. Quality control information representation using object-oriented data models. *International Journal of Computer Integrated Manufacturing*, 16(3):192–209, 2003.

[87] Hans Ehm, Stefan Heilmayer, Thomas Ponsignon, and Tim Russland. A discussion of object-oriented process modeling approaches for discrete manufacturing on the example of the semiconductor industry. In *Proceedings of the 2010 Winter Simulation Conference*, pages 2553–2562. IEEE, 2010.

[88] TT Pullan, M Bhasi, and G Madhu. Object-oriented modelling of manufacturing information system for collaborative design. *International journal of production research*, 50(12):3328–3344, 2012.

[89] Richard Hedman, Robin Sundkvist, Peter Almström, and Anders Kinnander. Object-oriented modeling of manufacturing resources using work study inputs. *Procedia CIRP*, 7:443–448, 2013.

[90] KW Young, R Piggin, and P Rachitrangsan. An object-oriented approach to an agile manufacturing control system design. *The International Journal of Advanced Manufacturing Technology*, 17:850–859, 2001.

[91] Alfredo Anglani, Antonio Grieco, Massimo Pacella, and Tullio Tolio. Object-oriented modeling and simulation of flexible manufacturing systems: a rule-based procedure. *Simulation Modelling Practice and Theory*, 10(3-4):209–234, 2002.

[92] A Garcia-Crespo, Belén Ruiz-Mezcua, JL Lopez-Cuadrado, and Juan Miguel Gómez-Berbís. Conceptual model for semantic representation of industrial manufacturing processes. *Computers in Industry*, 61(7):595–612, 2010.

[93] Elisa Negri, Luca Fumagalli, Marco Garetti, and Letizia Tanca. Requirements and languages for the semantic representation of manufacturing systems. *Computers in Industry*, 81:55–66, 2016.

[94] Jiayi Zhang, Bilal Ahmad, Daniel Vera, and Robert Harrison. Ontology based semantic-predictive model for reconfigurable automation systems. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 1094–1099. IEEE, 2016.

[95] Constantin Hildebrandt, André Scholz, Alexander Fay, Tizian Schröder, Thomas Hadlich, Christian Diedrich, Martin Dubovy, Christian Eck, and Ralf Wiegand. Semantic modeling for collaboration and cooperation of systems in the production domain. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[96] Jiafu Wan, Boxing Yin, Di Li, Antonio Celesti, Fei Tao, and Qingsong Hua. An ontology-based resource reconfiguration method for manufacturing cyber-physical systems. *IEEE/ASME Transactions on Mechatronics*, 23(6): 2537–2546, 2018.

[97] Giuseppe Landolfi, Andrea Bami, Gabriele Izzo, Elias Montini, Andrea Bettoni, Marko Vujasinovic, Alessio Gugliotta, António Lucas Soares, and Henrique Diogo Silva. An ontology based semantic data model supporting a MaaS digital platform. In *2018 International Conference on Intelligent Systems (IS)*, pages 896–904. IEEE, 2018.

[98] Aljosha Köcher, Constantin Hildebrandt, Luis Miguel Vieira da Silva, and Alexander Fay. A formal capability and skill model for use in plug and

produce scenarios. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1663–1670. IEEE, 2020.

[99] Dorsaf Elheni-Daldoul, Julien Le Duigou, Benoit Eynard, and Sonia Hajri-Gabouj. Enterprise information systems' interoperability: Focus on PLM challenges. In *Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services: IFIP WG 5.7 International Conference, APMS 2012, Rhodes, Greece, September 24-26, 2012, Revised Selected Papers, Part II*, pages 184–191. Springer, 2013.

[100] Fahim Ahmed and Soonhung Han. Interoperability of product and manufacturing information using ontology. *Concurrent Engineering*, 23(3):265–278, 2015.

[101] David Sanchez-Londono, Giacomo Barbieri, and Luca Fumagalli. Smart retrofitting in maintenance: a systematic literature review. *Journal of Intelligent Manufacturing*, 34(1):1–19, 2023.

[102] Aydin Nassehi, RD Allen, and ST Newman. Application of mobile agents in interoperable STEP-NC compliant manufacturing. *International Journal of Production Research*, 44(18-19):4159–4174, 2006.

[103] Hendro Wicaksono, Fabian Jost, Sven Rogalski, and Jivka Ovtcharova. Energy efficiency evaluation in manufacturing through an ontology-represented knowledge base. *Intelligent Systems in Accounting, Finance and Management*, 21(1):59–69, 2014.

[104] Didem Gürdür and Katja Tasala Gradin. Interoperable toolchains in cyber-physical systems with a sustainability perspective. In *2017 IEEE conference on technologies for sustainability (SusTech)*, pages 1–8. IEEE, 2017.

[105] Lampropoulos Georgios, Siakas Kerstin, and Anastasiadis Theofylaktos. Internet of things in the context of industry 4.0: An overview. *IJEK*, 2019.

[106] Helbert Da Rocha, Reza Abrishambaf, João Pereira, and Antonio Espirito Santo. Integrating the IEEE 1451 and IEC 61499 standards with the industrial internet reference architecture. *Sensors*, 22(4):1495, 2022.

[107] Takero Arai, Y Aiyama, Yusuke Maeda, Masao Sugi, and J Ota. Agile assembly system by "plug and produce". *CIRP annals*, 49(1):1–4, 2000.

[108] Nikolas Antzoulatos, Elkin Castro, Daniele Scrimieri, and Svetan Ratchev. A multi-agent architecture for plug and produce on an industrial assembly platform. *Production Engineering*, 8(6):773–781, 2014.

[109] Sebastian Hjorth, Casper Schou, Elias Ribeiro da Silva, Finn Tryggvason, Michael Sparre Sørensen, and Henning Forbech. A case study of plug and produce robot assistants for hybrid manufacturing workstations. In *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems: Proceedings of the 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021), Aalborg, Denmark, October/November 2021 8*, pages 242–249. Springer, 2022.

[110] Artem A Nazarenko, Joao Sarraipa, Luis M Camarinha-Matos, Christian Grunewald, Marc Dorchain, and Ricardo Jardim-Goncalves. Analysis of relevant standards for industrial systems to support zero defects manufacturing process. *Journal of Industrial Information Integration*, 23:100214, 2021.

[111] Shirley Cavin and Niels Lohse. Multi-level skill-based allocation methodology for evolvable assembly systems. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 532–537. IEEE, 2014.

[112] José Dias, Johan Vallhagen, José Barbosa, and Paulo Leitão. Agent-based reconfiguration in a micro-flow production cell. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 1123–1128. IEEE, 2017.

[113] Mattias Bennulf, Fredrik Danielsson, and Bo Svensson. A Method for Configuring Agents in Plug & Produce Systems. In *10th Swedish Production Symposium, SPS 2022; Conference date: 26 April 2022 through 29 April 2022; Conference code: 179964*, volume 21, pages 135–146. IOS Press, 2022.

[114] Xun Ye, Tae Yang Park, Seung Ho Hong, Yuemin Ding, and Aidong Xu. Implementation of a production-control system using integrated automation ML and OPC UA. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 1–6. IEEE, 2018.

[115] Jonathan Falk, Heiko Geppert, Frank Dürr, Sukanya Bhowmik, and Kurt Rothermel. Dynamic QoS-aware traffic planning for time-triggered flows in the real-time data plane. *IEEE Transactions on Network and Service Management*, 19(2):1807–1825, 2022.

[116] Christoph Hinze, David A Tomzik, Armin Lechler, Xun W Xu, and Alexander Verl. Control architecture for industrial robotics based on container virtualization. In *Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter*, pages 64–73. Springer, 2019.

[117] Francisco Ponce, Gastón Márquez, and Hernán Astudillo. Migrating from monolithic architecture to microservices: A Rapid Review. In *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7. IEEE, 2019.

[118] Luis Alberto Estrada-Jimenez, Terrin Pulikottil, Nguyen Ngoc Hien, Agajan Torayev, Hamood Ur Rehman, Fan Mo, Sanaz Nikghadam Hojjati, and José Barata. Integration of cutting-edge interoperability approaches in cyber-physical production systems and industry 4.0. In *Design, Applications, and Maintenance of Cyber-Physical Systems*, pages 144–172. IGI Global, 2021.

[119] Aydin Homay, Alois Zoitl, Mário de Sousa, and Martin Wollschlaeger. A survey: Microservices architecture in advanced manufacturing systems. In

2019 IEEE 17th International Conference on Industrial Informatics (INDIN), volume 1, pages 1165–1168. IEEE, 2019.

[120] Vrani Ibarra-Junquera, Apolinar González, Carlos Mario Paredes, Diego Martínez-Castro, and Rubi A Nuñez-Vizcaino. Component-Based Microservices for Flexible and Scalable Automation of Industrial Bioprocesses. *IEEE Access*, 9:58192–58207, 2021.

[121] Thomas Goldschmidt, Stefan Hauck-Stattelmann, Somayeh Malakuti, and Sten Grüner. Container-based architecture for flexible industrial control applications. *Journal of Systems Architecture*, 84:28–36, 2018.

[122] Kleanthis Thramboulidis, Danai C Vachtsevanou, and Ioanna Kontou. CPuS-IoT: A cyber-physical microservice and IoT-based framework for manufacturing assembly systems. *Annual reviews in control*, 47:237–248, 2019.

[123] Daniel Nüst, Dirk Eddelbuettel, Dom Bennett, Robrecht Cannoodt, Dav Clark, Gergely Daróczi, Mark Edmondson, Colin Fay, Ellis Hughes, Lars Kjeldgaard, et al. The rockerverse: packages and applications for containerization with r. *arXiv preprint arXiv:2001.10641*, 2020.

[124] Richard Senington, Balazs Pataki, and Xi Vincent Wang. Using docker for factory system software management: Experience report. *procedia CIRp*, 72:659–664, 2018.

[125] Nikolaos Nikolakis, Richard Senington, Konstantinos Sipsas, Anna Syberfeldt, and Sotiris Makris. On a containerized approach for the dynamic planning and control of a cyber-physical production system. *Robotics and computer-integrated manufacturing*, 64:101919, 2020.

[126] Christoph Gröger, Stefan Silcher, Engelbert Westkämper, and Bernhard Mitschang. Leveraging apps in manufacturing. A framework for app technology in the enterprise. *Procedia CIRP*, 7:664–669, 2013.

[127] Aoyu Chen, Mahmoud Dinar, Thomas Gruenewald, Max Wang, Justinian Rosca, and Thomas R Kurfess. Manufacturing apps and the Dynamic House of Quality: Towards an industrial revolution. *Manufacturing letters*, 13:25–29, 2017.

[128] Shaurabh Singh, Atin Angrish, James Barkley, Binil Starly, Yuan-Shin Lee, and Paul Cohen. Streaming machine generated data to enable a third-party ecosystem of digital manufacturing apps. *Procedia Manufacturing*, 10:1020–1030, 2017.

[129] Jianping Dou, Chun Su, and Xia Zhao. Mixed integer programming models for concurrent configuration design and scheduling in a reconfigurable manufacturing system. *Concurrent Engineering*, 28(1):32–46, 2020.

[130] Andreas Hees, Christina Bayerl, Brian Van Vuuren, Corné SL Schutte, Stefan Braunreuther, and Gunther Reinhart. A production planning method to optimally exploit the potential of reconfigurable manufacturing systems. *Procedia Cirp*, 62:181–186, 2017.

[131] Ming Liu, Lijie An, Jiantong Zhang, Feng Chu, and Chengbin Chu. Energy-oriented bi-objective optimisation for a multi-module reconfigurable manufacturing system. *International Journal of Production Research*, 57(19): 5974–5995, 2019.

[132] Jarosław Wikarek, Paweł Sitek, and Peter Nielsen. Model of decision support for the configuration of manufacturing system. *IFAC-PapersOnLine*, 52(13): 826–831, 2019.

[133] Imen Khettabi, Lyes Benyoucef, and Mohamed Amine Boutiche. Sustainable multi-objective process planning in reconfigurable manufacturing environment: adapted new dynamic nsga-ii vs new nsga-iii. *International Journal of Production Research*, 60(20):6329–6349, 2022.

[134] Manel Houimli, Laid Kahloul, and Mohamed Khalgui. Multi-objective optimization and formal specification of reconfigurable manufacturing system using adaptive nsga-ii. In *2017 First International Conference on Embedded & Distributed Systems (EDiS)*, pages 1–6. IEEE, 2017.

[135] Anuch Chaube, Lyés Benyoucef, and Manoj Kumar Tiwari. An adapted nsga-2 algorithm based dynamic process plan generation for a reconfigurable manufacturing system. *Journal of Intelligent Manufacturing*, 23:1141–1155, 2012.

[136] Abderahmane Bensmaine, Mohammed Dahane, and Lyes Benyoucef. Simulation-based nsga-ii approach for multi-unit process plans generation in reconfigurable manufacturing system. In *ETFA2011*, pages 1–8. IEEE, 2011.

[137] Shengluo Yang, Junyi Wang, Liming Xin, and Zhigang Xu. Real-time and concurrent optimization of scheduling and reconfiguration for dynamic reconfigurable flow shop using deep reinforcement learning. *CIRP Journal of Manufacturing Science and Technology*, 40:243–252, 2023.

[138] Bogdan I Epureanu, Xingyu Li, Aydin Nassehi, and Yoram Koren. Self-repair of smart manufacturing systems by deep reinforcement learning. *CIRP Annals*, 69(1):421–424, 2020.

[139] Jiecheng Tang and Konstantinos Salonitis. A deep reinforcement learning based scheduling policy for reconfigurable manufacturing systems. *Procedia CIRP*, 103:1–7, 2021.

[140] Zsolt J Viharos and Richard Jakab. Reinforcement learning for statistical process control in manufacturing. *Measurement*, 182:109616, 2021.

[141] Clemens Zimmerling, Christian Poppe, Oliver Stein, and Luise Kärger. Optimisation of manufacturing process parameters for variable component ge-

ometries using reinforcement learning. *Materials & Design*, 214:110423, 2022.

[142] Junzheng Li, Dong Pang, Yu Zheng, Xinping Guan, and Xinyi Le. A flexible manufacturing assembly system with deep reinforcement learning. *Control Engineering Practice*, 118:104957, 2022.

[143] Agajan Torayev, Zi Wang, Giovanna Martínez-Arellano, Jack C Chaplin, David Sanderson, and Svetan Ratchev. Optimal selection of manufacturing configurations using object-oriented and mathematical data models. In *Industrial Engineering and Applications*, pages 3–12. IOS Press, 2023.

[144] Agajan Torayev, Zi Wang, Giovanna Martínez-Arellano, Jack C Chaplin, David Sanderson, and Svetan Ratchev. Multi-criteria decision-making for optimal manufacturing configuration selection using an object-oriented data model and mathematical formalization. In *International Manufacturing Science and Engineering Conference*, volume 87240, page V002T07A012. American Society of Mechanical Engineers, 2023.

[145] Agajan Torayev, Giovanna Martínez-Arellano, Jack C Chaplin, David Sanderson, and Svetan Ratchev. Towards modular and plug-and-produce manufacturing apps. *Procedia CIRP*, 107:1257–1262, 2022.

[146] Agajan Torayev, Giovanna Martínez-Arellano, Jack C Chaplin, David Sanderson, and Svetan Ratchev. Online and Modular Energy Consumption Optimization of Industrial Robots. *IEEE Transactions on Industrial Informatics*, 2023.

[147] Agajan Torayev, Jose Joaquin Peralta Abadia, Giovanna Martínez-Arellano, Mikel Cuesta, Jack C Chaplin, Felix Larrinaga, David Sanderson, Pedro-José Arrazola, and Svetan Ratchev. Optimal manufacturing configuration selection: Sequential decision making and optimization using reinforcement learning. *Procedia CIRP*, 120:986–991, 2023.

[148] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[149] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[150] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[151] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[152] Y Xiang, DY Sun, W Fan, and XG Gong. Generalized simulated annealing algorithm and its application to the Thomson model. *Physics Letters A*, 233 (3):216–220, 1997.

[153] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[154] Jihong Yan and Mingyang Zhang. A transfer-learning based energy consumption modeling method for industrial robots. *Journal of Cleaner Production*, 325:129299, 2021.

[155] Michele Gadaleta, Giovanni Berselli, Marcello Pellicciari, and Federico Grassia. Extensive experimental investigation for the optimization of the energy consumption of a high payload industrial robot with open research dataset. *Robotics and Computer-Integrated Manufacturing*, 68:102046, 2021.

[156] Mingyang Zhang and Jihong Yan. A data-driven method for optimizing the energy consumption of industrial robots. *Journal of Cleaner Production*, 285:124862, 2021.

[157] Michele Gadaleta, Marcello Pellicciari, and Giovanni Berselli. Optimization of the energy consumption of industrial robots for automatic code generation. *Robotics and Computer-Integrated Manufacturing*, 57:452–464, 2019.

[158] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

[159] J. Blank and K. Deb. pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

[160] Andrzej P Wierzbicki. The use of reference objectives in multiobjective optimization. In *Multiple criteria decision making theory and application*, pages 468–486. Springer, 1980.

[161] Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011.

[162] Terrin Pulikottil, Luis A Estrada-Jimenez, José Joaquín Peralta Abadía, Angela Carrera-Rivera, Agajan Torayev, Hamood Ur Rehman, Fan Mo, Sanaz Nikghadam-Hojjati, and Jose Barata. Big data life cycle in shop-floor–trends and challenges. *IEEE Access*, 2023.

[163] Fan Mo, Fabio Marco Monetti, Agajan Torayev, Hamood Ur Rehman, Jose A Mulet Alberola, Nathaly Rea Minango, Hien Ngoc Nguyen, Antonio Maffei, and Jack C Chaplin. A maturity model for the autonomy of manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 126(1-2):405–428, 2023.