NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ANTÓNIO LUÍS TEIXEIRA DA SILVA MONTE PEGADO

BSc Degree in Electrical and Computer Engineering

ENHANCING APPLICATION AVAILABILITY AND INTEGRATION IN INDUSTRY

DEVELOPING A MODULAR OPERATING SYSTEM WITH A MARKETPLACE

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

ENHANCING APPLICATION AVAILABILITY AND INTEGRATION IN INDUSTRY

DEVELOPING A MODULAR OPERATING SYSTEM WITH A MARKETPLACE

ANTÓNIO LUÍS TEIXEIRA DA SILVA MONTE PEGADO

BSc Degree in Electrical and Computer Engineering

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

Examination Committee

Chair: Bruno João Nogueira Guerreiro
Assistant Professor, NOVA School of Science and Technology, NOVA University Lisbon

Rapporteur: Filipa Alexandra Moreira Ferrada
Assistant Professor, NOVA School of Science and Technology, NOVA University Lisbon

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

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

Enhancing Application Availability and Integration in Industry
Developing a Modular Operating System with a Marketplace

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.27) [1].

To everyone who showed me the way.

Acknowledgements

These past five years have been nothing short of incredible, passing by in what feels like the blink of an eye. When I first embarked on this journey, I imagined that I would find answers to many of my questions. Yet, I've discovered that the true gift of this experience has been the new questions it has sparked in me. My curiosity has deepened, leading me down a path of innovation and exploration that I now realize is a way of life. Along the way, I've had the privilege of meeting extraordinary people who have left a lasting impact on both my academic and personal life.

I am deeply grateful to the professors at NOVA FCT, who have shaped my academic journey with their knowledge and guidance. I owe a special thanks to my supervisor, Professor André Dionísio Rocha, who has become more than just a mentor, a friend. His influence has changed my path, perhaps without him even realizing it. His ability to balance his role as a university professor while maintaining a strong relationship with students is truly inspiring. He has consistently encouraged us to explore more, and always keep learning, all while managing all his responsibilities with a calm demeanor. I am also profoundly thankful to Professor José Barata, and my colleagues at RICS, who have taught me so much and continue to inspire me to strive for more. Thank you, João Amorim, for your incredible friendship and for being my work buddy over these past five years. We've been through so much together, and your friendship is one I know I'll carry with me for life.

Thank you, NOVA FCT, for not only providing me with knowledge and helping me grow, but also for reigniting my passion for learning, discovery, and the desire to do more.

I must also express my deepest gratitude to my parents and sister whose support has been the foundation of this journey. Without them, I would not have reached this point. To my amazing girlfriend, Marta Casquilho - thank you for helping find a balance between my personal life and work, for being my best friend, for reminding me to live my life and enjoy every moment, and for our lunches grabbing a "*panado maior que o pão*".

This journey has been a wild ride, and as it comes to a close, I realize that it has only just begun, opening the door to a multitude of new possibilities.

Abstract

In the age of rapid technological advancement and digital transformation, Small and Medium-sized Enterprises (SMEs) frequently face challenges in keeping up with innovation due to budgetary limitations and complex integration processes. In response to the shift from Industry 4.0 (I4.0) to Industry 5.0 (I5.0), both manufacturers and software developers are increasingly adopting a modular approach to software acquisition and distribution. This approach not only enhances flexibility and cost-effectiveness but also emphasizes sustainability and the importance of collaborative networks. The Industry Modular Operating System (IMOS) is proposed as a solution positioned to enhance application availability, execution, and integration, while fostering collaboration between software developers and industrial site managers. Designed to tackle the inherent challenges of I5.0, the author of IMOS aims to lay a robust foundation for the proposed ecosystem, emphasizing modularity, accessibility, and collaboration. This work presents an OS-like application that streamlines industry-specific resources by offering containerized software solutions through an integrated marketplace. The implemented platform allows users to efficiently manage a variety of applications and services both on-premise and on-cloud, through an Open Containerization Platform (OCP). By offering this dynamic ecosystem, IMOS bridges the gap between development and production, promoting seamless symbiosis, problem-solving, and resource-sharing within a vibrant community of industry experts.

Keywords: Application, Cloud, Collaborative Manufacturing, Community, Containerization, IMOS, Industry 5.0, Integration, Marketplace

Resumo

Numa era marcada por rápidos avanços tecnológicos e pela transformação digital, pequenas e médias empresas lutam frequentemente para acompanhar com o ritmo da inovação devido a restrições orçamentais e desafios de integração. Como resposta à transição da Indústria 4.0 para a Indústria 5.0, tanto gestores de manufatura como desenvolvedores de software estão cada vez mais a adotar uma abordagem modular para a aquisição e distribuição de software. Esta abordagem não só aumenta a flexibilidade e a relação custo-eficácia a longo prazo, como também enfatiza a sustentabilidade e a importância das redes colaborativas. O IMOS surge como uma solução posicionada para melhorar a disponibilidade, a execução e a integração de aplicações, promovendo ao mesmo tempo a colaboração entre programadores de software e gestores de espaços industriais. Concebido para enfrentar os desafios inerentes à Indústria 5.0, o presente projeto tem como objetivo estabelecer uma base robusta para o ecossistema proposto, enfatizando a modularidade, a acessibilidade e a colaboração. Este trabalho apresenta uma aplicação protótipo de um sistema operativo que facilita a disseminação de recursos específicos do sector, oferecendo soluções de software conteinerizadas através de um marketplace integrado. A plataforma desenvolvida permite aos seus utilizadores gerir eficazmente uma variedade de aplicações e serviços, tanto localmente como na cloud, através de um OCP. Com este ecossistema dinâmico, o IMOS vem colmatar uma lacuna presente entre as áreas de desenvolvimento e de produção, promovendo a simbióse entre beneficiários, a resolução de problemas e a partilha de recursos numa comunidade dinâmica, composta por especialistas na área.

Palavras-chave: Aplicação, Comunidade, Conteinerização, Cloud, IMOS, Indústria 5.0, Integração, Manufatura Colaborativa, Marketplace

Contents

List of Figures

Acronyms

AAS Asset Administration Shell *(pp. 20, 21, 80, 81)*
AI Artificial Intelligence *(pp. 2, 4, 5, 11, 12, 15, 27, 28, 31, 41, 48, 73–75, 78, 79)*
API Application Programming Interfaces *(pp. 11, 13, 14, 16, 22, 52, 55, 57, 59, 70)*

CE Circular Economy *(p. 17)*
CIN Collaborative Innovation Network *(pp. 36, 37, 48)*
CNs Collaborative Networks *(pp. 35–37, 81)*
CPS Cyber-Physical Systems *(pp. 4, 5)*

DL Deep Learning *(pp. 27, 28)*

EF Enabler Framework *(p. 16)*
EHS Environment, Health and Safety *(p. 27)*
ERP Enterprise Resource Planning *(pp. 19, 21)*
ESA Enhanced Self-organizing Agent *(p. 18)*

FaaS Factory-as-a-Service *(pp. 10, 11)*

HRC Human-Robot Collaboration *(p. 26)*

I4.0 Industry 4.0 *(pp. iv, 1, 3–6, 13, 15, 16, 20, 21, 23, 30, 31, 35, 62, 80)*
I5.0 Industry 5.0 *(pp. iv, 1, 3–8, 13, 30, 31, 35, 39, 51, 62, 80)*
IaaS Infrastructure-as-a-Service *(p. 11)*
IIRA Industrial Internet Reference Architecture *(p. 21)*
IMOS Industry Modular Operating System *(pp. iv, v, viii, ix, 3, 31–33, 35, 36, 38–41, 43–58, 60–62, 64–67, 69, 70, 72, 73, 76, 78–81)*
IoT Internet of Things *(pp. 5, 12, 13, 15, 16, 21, 26)*
IPC Inter-Process Communication *(pp. 43, 44, 61, 70)*
IS Industrial Symbiosis *(p. 17)*

IT Information Technology *(p. 21)*

MES Manufacturing Execution Systems *(pp. 12, 19, 21, 22)*
ML Machine Learning *(pp. 5, 15, 27, 28, 76)*

NIST National Institute of Standards and Technology *(pp. 16, 21)*
NSIS Nullsoft Scriptable Install System *(pp. ix, 65–67, 72, 73)*

OCP Open Containerization Platform *(pp. iv, v, viii, 31, 33, 34, 44, 45, 47, 53, 54, 69, 70, 72, 81)*
OPC UA Open Platform Communications Unified Architecture *(pp. 20, 22)*
OT Operational Technology *(p. 21)*

PaaS Platform as a Service *(p. 11)*
PLM Product Lifecycle Management *(p. 12)*
PML Physical Manufacturing Layer *(pp. 10, 11)*
PPE Personal Protective Equipment *(pp. ix, 3, 43, 73, 75, 76, 78)*
PVCs Professional Virtual Communities *(p. 36)*

RAMI 4.0 Reference Architectural Model Industrie 4.0 *(pp. 20, 21, 80, 81)*

SaaS Software-as-a-Service *(p. 11)*
SDL Software Definition Layer *(pp. 10, 11)*
SLM Service Lifecycle Management *(p. 12)*
SMEs Small and Medium-sized Enterprises *(pp. iv, 1–3, 5, 7, 11, 17, 18)*
SOA Service Oriented Architecture *(pp. 14, 16, 21)*

UML Unified Modeling Language *(pp. 20, 32, 33, 60, 69)*

vApps Virtual Applications *(pp. 13–15)*
VBEs Virtual organizations Breeding Environments *(p. 36)*
vf-OS Virtual Factory Open Operating System *(pp. viii, 13–16, 80)*
VOs Virtual Organizations *(pp. 36, 37, 48)*
VTs Virtual Teams *(pp. 36, 37, 48)*

XR Extended Reality *(pp. 6, 15)*

YOLO You Only Look Once *(pp. 29, 74)*

ZDMP Zero Defect Manufacturing Platform *(pp. 15, 80)*

1 Introduction

This chapter provides the foundation for this thesis document by addressing its scope, motivation, and objectives, explaining the purpose and goals of the research. It also outlines the structure of the dissertation, offering a clear roadmap for the chapters that follow.

1.1 Scope, Motivation and Objectives

This Master's dissertation thesis marks the culmination of five years of academic pursuit in the field of Electrical and Computer Engineering, with a focus on Integrated Manufacturing and Digital Systems. This introductory chapter outlines the key themes that will be explored in greater detail throughout this document. The aim is to identify existing gaps in related work and to propose a solution that not only addresses these gaps but also has the potential to contribute meaningfully to advancing the current state of the art.

The emergence of cutting-edge technologies and the modularization movement are transforming production processes in the rapidly shifting industrial landscape. At the same time, I4.0, and the transitioning towards I5.0, push for a more adaptable, collaborative, and human-centered approach to manufacturing. Still, one significant challenge prevails: the distribution and integration of software applications into manufacturing processes [2, 3].

As the industry continues to embrace digitalization, the demand for scalable and specialized solutions becomes critical, not only streamlining integration and accessibility, but also enhancing collaboration in response to evolving market dynamic [4]. Manufacturers are increasingly seeking sustainable, holistic approaches that allow them to build processes incrementally or integrate new modules into existing workflows [5]. This trend also creates opportunities for smaller developers to innovate in niche areas, offering specialized solutions that are easily distributed and monetized.

Although software modularity promises more responsiveness to market dynamics and simpler operations, it also comes with drawbacks, especially for SMEs. The fast developments in technology and short lifespans of modern solutions, driven by these

evolving paradigms, present significant setbacks for small manufacturers.

SMEs often battle to keep up with the rapid advancements in industry and struggle with the expenses and complications of integrating new technologies. Because of market competitiveness, existing software applications frequently lack seamless integration and holistic solutions, which forces companies to invest a significant amount of time and money in the development of custom-made solutions [6, 7].

SMEs and manufacturers face several challenges, [6]:

1. **Lack of Integration and Modularity**. Existing software applications often do not integrate seamlessly with one another due to market competitiveness, making it difficult to create comprehensive and adjustable solutions;

2. **Fast-Growing Solutions**. It is challenging for SMEs to keep up with the ever-growing software solutions introduced to the community on a daily basis;

3. **Limited Holistic Options**. Manufacturers are constrained by a narrow range of software applications that fully meet their unique necessities, requiring a lot of time and resource allocation for research and state-of-the-art technology findings;

4. **High Development Costs**. Developing and integrating custom software for specific tasks is expensive and time-consuming.

In response to these challenges following modularization and sustainability of systems, significant efforts have focused on developing containerized applications tailored to industrial needs and processes.

Manufacturers can significantly enhance their operations by integrating a range of advanced solutions [8]. From quality control solutions - combining image-based software with sensor-based anomaly detection to ensure products meet quality standards -, to safety monitoring - elevated through the use of image recognition for detecting helmets, vests, masks, and critical proximity areas -, predictive maintenance - utilizing sensor data and machine learning to anticipate equipment needs and automate safety warnings with inference results -, and on-site support - by delivering real-time maintenance instructions and repair guidance through Artificial Intelligence (AI) applications, in the form of chatbots and large language models, and augmented reality solutions.

These applications cover a diverse range of functionalities, including AI-driven quality control, safety monitoring, robot control libraries, predictive maintenance, machine learning analytics, and real-time inventory management. However, there is a notable absence of a centralized platform to facilitate the distribution, standardization, and monetization of these solutions, while bridging the gap between various stakeholders.

Such a platform allows users to easily discover distributed applications or identify gaps for new tailored solutions, while giving developers greater exposure and access to valuable data and real-world scenarios. This creates not only a robust software marketplace

but also a community-driven ecosystem that connects both development and production industry stakeholders.

After exploring these concepts, the discussion leads to the following research question: **How might a collaborative ecosystem be developed to promote software distribution and integration in industrial/manufacturing settings?** By designing a platform as a community marketplace for SMEs, we may create an ecosystem in which all stakeholders can share resources and create new solutions, supporting innovation and collaboration.

1.2 Dissertation Structure

The dissertation is organized into seven interconnected chapters, each contributing to a cohesive narrative that progresses from foundational concepts to practical applications and future directions. It begins with Chapter 2, which explores the state of the art, positioning the research within the broader context of technological evolution from I4.0 to I5.0. This chapter examines key concepts such as industrial marketplaces, collaborative ecosystems, and manufacturing integration, culminating in a gap analysis that identifies areas requiring further investigation.

Building on this groundwork, Chapter 3 introduces the conceptual framework, emphasizing collaborative innovation, co-creation, and the IMOS Framework as a response to the challenges identified. This theoretical foundation is translated into practice in Chapter 4, which focuses on the design and development of the IMOS platform. This chapter highlights its architectural components, the role of marketplaces, cloud computing options, and its potential as an open-source ecosystem fostering collaboration.

The practical implementation and evaluation of the IMOS platform are detailed in Chapter 5, covering aspects such as development, server setup, and application execution. It also addresses the challenges and limitations encountered during this process, providing insights into the platform's functionality and areas for improvement. The platform's versatility is further illustrated in Chapter 6, which presents real-world application cases, including solutions for recycling, safety PPE, data visualization, and AI-driven automation.

Finally, Chapter 7 offers a comprehensive conclusion, synthesizing the research findings, reflecting on their implications, and outlining potential avenues for future work. This structure ensures a seamless transition from theory to practice, delivering a thorough exploration of the research topic.

2

State of the Art

In this chapter, the author delves into research areas pertinent to this thesis, which supported the development of the proposed project. The author divided this state-of-the-art research into six sections.

Initially, the transition from I4.0 to I5.0 is examined, along with the concepts and ideation influencing the presented work. Following this, the marketplace concept for manufacturing is showcased, emphasizing modularity and software distribution. The discussion then moves to concepts applied to industrial symbiosis and co-innovation, setting the stage for exploring collaborative networks. Architectural models for integrating with smart manufacturing systems are presented next, supporting communication principles and standards. The modularization of software applications is also explored, including available containerization tools. Finally, real-world examples of technological solutions transforming manufacturing are presented, showcasing useful application examples.

These insights provide a strong foundation for understanding the current state of smart manufacturing and the marketplace concept applied to manufacturing, setting the stage for the implementation of the proposed project in this thesis.

2.1 From Industry 4.0 to Industry 5.0

The rapid evolution of digital technologies and the proliferation of AI-driven solutions, as driven by the latest industrial revolution, have introduced new complexities into the manufacturing landscape. In this section the author will address these complexities and examine the current challenges to overcome them as we transition from the I4.0 era to the emerging I5.0.

The Fourth Industrial Revolution represents a paradigm shift in manufacturing and industrial practices through the integration of digital technologies, as detailed by [2]. This revolution builds upon the advancements of the third industrial revolution, which introduced automation and computing into production processes. I4.0 extends these innovations by connecting physical systems with cyber systems, creating Cyber-Physical Systems (CPS) that enable intelligent, interconnected production environments. At its

core, I4.0 is characterized by several foundational concepts [9, 10]:

- **Cyber-Physical Systems**. CPS consist of interconnected systems that integrate computational algorithms with physical processes. These systems enable real-time monitoring, control, and optimization of industrial processes;
- **Internet of Things (IoT)**. The IoT allows for the seamless communication between machines, products, and humans. IoT devices collect and exchange data across networks, facilitating the automation and optimization of manufacturing processes;
- **Big Data and Analytics**. The vast amount of data generated by CPS and IoT devices necessitates the use of big data analytics. By analyzing this data, industries can gain insights into process efficiency, product quality, and predictive maintenance, leading to more informed decision-making;
- **Cloud Computing**. Cloud technologies provide the computational power and storage needed to manage the large volumes of data generated by I4.0 systems. Cloud computing also enables the flexibility and scalability required for modern manufacturing operations;
- **AI and Machine Learning (ML)**. AI and ML algorithms play a critical role in analyzing data, optimizing processes, and enabling autonomous decision-making in real-time;
- **Smart Factories**. These factories integrate the above technologies to create highly adaptive, efficient, and responsive production environments. They can self-optimize performance, self-adapt to new production requirements, and learn from new conditions.

Despite its progress, I4.0 still faces a number of obstacles and limitations. These include the initial cost associated with updating legacy systems, particularly for SMEs [6]; problems with interoperability and communication arising from the integration of multiple systems and technologies from different vendors [7]; and cybersecurity risks arising from the increased number of potential entry points for cyberattacks within the industrial infrastructure [11].

On the other hand, I5.0 holds new concepts that point the way forward, envisioning a manufacturing system characterized by its human-centricity, sustainability and resilience [3]. This notable evolution, often termed the "*first industrial evolution led by humans*", is described by the 6R rules of industrial cycle: Recognize, Reconsider, Realize, Reduce, Reuse and Recycle. These principles form a systematic approach to waste prevention, logistics efficiency and the creation of high-quality, custom-made products, as articulated by Michael Rada [12]. An alternative definition, as proposed by [13], underscores the crucial importance of human creativity and cognitive capacities within the factory environment. In this perspective, I5.0 envisions a partnership between the human

workforce and machines, with the shared goal of elevating process efficiency by seamlessly integrating workflows with intelligent systems.

I5.0 will build upon the foundational concepts established during I4.0, with a distinct focus on human-centered, resilient and sustainable design [4]. In this evolution process, a spectrum of additional features that promise to reshape the manufacturing landscape is presented [4, 8]:

- **Smart Additive Manufacturing**. This sustainable approach to industrial production creates product components layer by layer, departing from the traditional solid structures. The result is lighter yet more robust parts, contributing to resource efficiency and architectural sustainability;
- **Predictive Maintenance**. I5.0 leverages cutting-edge prediction tools to systematically process data, extracting valuable information and delineating uncertainties. This empowers decision-makers with insights, facilitating smarter and proactive maintenance strategies;
- **Hyper Customization**. I5.0 offers highly customized goods, services and content to every single consumer by leveraging real-time data and cutting-edge technologies. This encourages a more adaptable and customer-focused strategy;
- **Collaborative Robots**. These are pivotal to amplify human capabilities, simplifying automation for individuals and small businesses [14]. While robots excel in high-volume product manufacturing, their collaboration with humans adds a layer of critical thinking and adaptability to industrial processes;
- **Extended Reality (XR)**. Through the smooth integration of the virtual and physical domains, XR technologies improve human-computer interactions. I5.0 is moving toward interconnected digital assistants for manufacturing operators as a result of the fusion of virtual reality, augmented reality and mixed reality. These new applications include solutions like remote assistance [15], assembly line monitoring [16], and virtual training and maintenance [17].

This industrial movement is driving the emergence of numerous start-up companies worldwide, specializing in custom manufacturing solutions encompassing both hardware and software. This surge in innovative technologies and applications holds great promise for I5.0, enabling increased production capacities, the delivery of personalized products, the establishment of human-machine interoperability, the enhancement of production quality control and even the monitoring of workplace safety.

However, this accelerated digital evolution and application development comes with a distinct challenge: the rapid pace at which cutting-edge manufacturing solutions are being created far outpaces the ease with which companies can build and integrate diverse applications from various developers within their existing manufacturing systems. The deficiency in interoperability and integration between different software solutions presents

an obstacle interfering with companies swiftly customizing their manufacturing processes into a more sustainable ecosystem.

Consequently, this challenge constitutes the central focus of this thesis, prompting the introduction of a marketplace tailored for the manufacturing industry. This concept establishes a community-based platform, offering a multitude of applications developed by diverse software providers. These applications can be readily acquired and seamlessly integrated into an existing manufacturing system, ultimately bridging the gap between the rapid development of manufacturing technologies and the ability of companies to adapt, evolve, and effectively enrich their processes.

2.2 Marketplace Concept

In the realm of smart manufacturing, the transition towards I5.0 is encouraging a fundamental shift in how businesses approach technology adoption and system's sustainability. The traditional model of acquiring custom-made, unique and often immutable solutions is giving way to a more dynamic, adaptable paradigm.

Just as additive manufacturing revolutionizes the physical production of goods, a similar philosophy is appearing in the digital ground. **Rather than investing in monolithic and inflexible solutions, companies are increasingly inclined to build their manufacturing systems, application by application, creating a modular architecture that aligns seamlessly with the specific needs of their operations**. This paradigm shift towards modularity highlights the driving principles of I5.0 and Smart Manufacturing, offering a multitude of advantages for SMEs, both from manufacturing client's and software developer's perspectives, and making the acquisition of tailored applications more cost-effective and accessible for SMEs.

The notion of app-based manufacturing was introduced by Gröger *et al.* [5], envisioning a single application development platform that would enable users to design customized tools for different production manufacturing levels. These task-specific tools could be anything from company-specific solutions, to tools with broader applicability, such as interactive diagnostic maintenance aids for troubleshooting equipment on the production floor. The authors propose the need for a unified platform where these manufacturing apps can be developed, akin to a repository similar to the Google Play Store, facilitating the acquisition and submission of applications by end users from diverse companies.

To fully realize the potential of the contextualized "*right place, right time*" digital tools in manufacturing, Gröger *et al.* emphasize the necessity of implementing standards for input/output protocols of these applications [5] and highlight security and user interface design as crucial elements to ensure a seamless and secure user experience. Although the idea has the potential to greatly speed up development, integration and maintenance activities, the lack of a unified or standard platform now stands in the way of the concept's widespread adoption and realization.

The marketplace concept for manufacturing solutions and software distribution closely aligns with the I5.0 principles, showcasing key attributes such as hyper customization and smart manufacturing. This innovative concept unlocks a vast array of software applications and services designed to fulfill I5.0's requirements, encompassing crucial areas like predictive maintenance, robot collaboration, safety issues detection or the integration of augmented reality technologies, all examples of application areas that we will explore in greater detail.

In this section, we dive into the current state of the art concerning the presented marketplace concept, examining how it presents a solution for the manufacturing ecosystem and providing different approaches to software distribution.

2.2.1 Influences from Mainstream Marketplaces

The architecture and design of mainstream digital marketplaces, such as the App Store and Google Play Store, have significantly influenced the conceptualization of the proposed industry-oriented marketplace. These widely-used platforms have set a benchmark for user-friendly experiences, flexibility and accessibility. Examining their key features offers valuable insights into the potential evolution of industrial application ecosystems.

The intuitive user interfaces of mainstream app marketplaces have inspired our emphasis on a user-oriented experience. Just as these platforms allow users to seamlessly discover, install and manage applications, an industry-oriented marketplace would strive for a similarly straightforward approach, where accessibility is a key factor, ensuring that manufacturing professionals can easily navigate and leverage the available applications.

While drawing inspiration from mainstream marketplaces might be very beneficial from a user interaction perspective, it's crucial to acknowledge the unique characteristics that set the industrial marketplace concept apart.

Unlike general-purpose app stores, such marketplace is purpose-built for the industrial sector. Tailored applications address specific challenges in manufacturing and introduce cloud computing options, providing solutions that may not find a home in broader consumer-focused platforms. Such ecosystem aims to seamlessly integrate with manufacturing systems, aligning applications with the specific needs of production environments, recognizing the critical nature of industrial processes, and emphasizing standardization in both application development and integration. This level of exigency and requirements distinguishes itself from mainstream counterparts that primarily focus on standalone consumer applications.

2.2.2 Flexible Software-based Ecosystems

In regards to flexible software platforms, the authors of [18] defend the open software ecosystem approach as it "*differs from traditional outsourcing techniques in that the initiating actor does not necessarily own the software produced by contributing actors and does not hire the contributing actors*". Manikas and Hansen consider a network of developers working, rather than a singular entity delivering the final product to the customer, as the defining characteristic of software ecosystems. This approach is consistent with the notion of transforming the process of developing manufacturing systems through the gradual integration of several software modules from various providers into a single adaptable system.

Starting from more theoretical foundations, Jan Bosch *et al.* [19] define an integration-centric approach in software development that can be summarized as follows: Project requirements are first allocated to specific software components and development teams who put those needs into practice. Following these components' completion, the integration phase starts, during which all of the components are brought together to create the entire system. Finally, system-level testing is also carried out, where the majority of integration problems are found and fixed.

However, the integration-centric approach becomes increasingly problematic with greater subcontractor involvement, particularly in complex systems. This publication [20] identifies a number of reasons why this strategy is less competitive, especially in cases involving significant outsourcing, much like a marketplace ecosystem:

- **Process Control Challenges**. Ensuring control and synchronization over processes becomes difficult when external developers may not adhere to the same standards. Even when subcontractors use similar methodologies, the impact of process control on project efficiency is often marginal;

- **Misalignment of Release Cycles**. Imposing the hardware-manufacturing-based release cycle on software development results in outdated software features. Mechanical and hardware development typically have longer lead times compared to software, resulting in desynced newly implemented features;

- **Rapidly Growing Feature Content**. Adding complexity to traditional development processes, with the content of software features growing exponentially, negatively affects software effort, quality and cycle time, as new features introduce interdependencies. Inadequate architectural control may result in delayed integration of individual software components.

It is becoming more and more beneficial to move away from the integration-centric software development strategy in light of these difficulties. Adopting an open software ecosystem is a viable alternative, according to Jansen *et al.* [21]. Using federated embedded systems, Papatheocharous *et al.* [22] identify important architecture features. While the

focus of their study is embedded systems, the majority of the ideas are applicable to software ecosystems. According to [22], the following characteristics are thought to be necessary for a reference architecture:

- **Composability**. Application decoupling should be possible thanks to the design, allowing for independent application development, integration and validation. Developers of applications and platforms should provide this separation;
- **Deployability**. Applications ought to be able to be installed one after the other and behave regardless of the sequence in which they are installed. A deployment infrastructure meeting integrity requirements should be in place;
- **Stability**. The architecture should provide stability over time, enabling ecosystem parties to maintain sustainable processes while remaining extensible for future feature additions;
- **Configurability**. The platform should support variability in software configuration, particularly in products that vary within a product family or for different types of processes to be handled.

A. Yadav and S.C. Jayswal [23] conduct an in-depth review on Flexible Manufacturing Systems modeling. The authors investigate a range of modeling approaches, such as Petri Nets and simulation environments, hierarchical and multi-criteria decision-making, and mathematics and artificial intelligence models. According to their findings, flexibility in manufacturing systems is defined as the ability to accommodate certain variations, in part styles and process variations, without disrupting the production line. The authors identify four distinct flexibility tests used to evaluate a manufacturing system's flexibility: process variety, schedule change, error recovery and new part testing.

The marketplace ecosystem being studied breaks away from the rigidity of traditional, integration-centric approaches by offering a module-based platform where different software components and solutions can coexist seamlessly. In doing so, it promotes an ecosystem where flexibility is a practical reality, enabling manufacturing systems to respond to products' changing demands, accommodate new and updated technologies, adapt to different process variations and setups, and enhance their overall efficiency.

2.2.3 Cloud Manufacturing

Many of the following presented works on manufacturing software applications advocate for a Factory-as-a-Service (FaaS) approach, promoting seamless collaboration in both the physical and software realms. This collaborative architecture is divided into two layers: the Physical Manufacturing Layer (PML) and the Software Definition Layer (SDL), holding particular benefits for smaller companies and businesses, as it offers a pay-as-you-go model for utilizing both software and manufacturing resources. This, in turn, minimizes expenses associated with hardware maintenance and the acquisition of software updates.

Following the FaaS trend, the authors of [24] introduce a novel concept: a software-defined manufacturing framework that transcends traditional boundaries, transforming it into an open and collaborative ecosystem. This approach partitions the industry ecosystem into two key layers: the PML and the SDL. Both of these layers are made accessible to and shared by all practitioners, fostering a more inclusive and dynamic manufacturing environment. The adoption of these technologies and methodologies reveals a clear trajectory for the future of manufacturing, characterized by two fundamental components identified by the authors: resource sharing and collaboration. This trend aims to bridge the gap, enabling SMEs to enter the manufacturing domain and break the barriers that have upheld larger practitioners in monopoly positions.

In the realm of Cybermanufacturing, an "app" represents a sophisticated data processing workflow executed across distributed resources managed through Cloud Application Programming Interfaces (API). Prior to being scaled in the public cloud, these applications are developed and validated locally by stakeholders who provide their knowledge to clients. Manufacturing apps have been transformed into cloud-compatible environments, which has resulted in the appearance of apps in a number of app marketplaces. The FaaS notion in the context of Cybermanufacturing includes the following important cloud computing solutions: Platform as a Service (PaaS), (also called App Hosting), Infrastructure-as-a-Service (IaaS), (which includes Cloud Computing resources), and Software-as-a-Service (SaaS), (also known as App Chaining).

To showcase the management of these "apps", this paper by A. Akula *et al.* [25], envisions a comprehensive framework with a crucial capability – an **App Runtime Management**. Customers may easily acquire these programs thanks to the app developers' ability to perform tasks like create, add, publish and delete within the marketplace. Specifically, RESTful web services - a widely-used web service mechanism -allow the framework's components to connect with each other seamlessly. These are comparable to the APIs that are offered by well-known public cloud service providers like Microsoft Azure, IBM Bluemix and Amazon Web Services.

Within the App Runtime Management, **the applications are not only executable but also serve as building blocks for the creation of new apps**. For instance, one could train an AI model on top of another, utilizing different geometries. This approach significantly reduces design, development and deployment cycles. This dual benefit extends to developers, who contribute to the marketplace, and customers, who may lack specific expertise but have distinct use cases for advanced manufacturing. They can easily find pre-implemented solutions within the marketplace, streamlining their integration into manufacturing processes.

Initially, the purposed marketplace concept was more oriented towards SDL collaboration, providing SaaS solutions to existing manufacturing companies. However, as the carried research expanded, it became evident that incorporating PML collaboration, as seen in the Cybermanufacturing paradigm, was essential. In this context, resources are provisioned from a cloud infrastructure. Manufacturing software developers, equipped

with their hardware for testing purposes, can leverage such marketplace platform to offer comprehensive services or products. These offerings integrate not only their software applications or models, but also their dedicated hardware, enabling functionalities like data processing, model training using client data, 3D printing and more.

Cloud manufacturing, as a paradigm, deviates from the traditional approach of each factory maintaining a dedicated computational infrastructure. Instead, most computation and storage tasks are centralized within a shared cloud computing infrastructure, which can be accessed by multiple factories owned by different entities. Inheriting the benefits of cloud computing, including heightened flexibility, availability, scalability and a pay-as-you-go service model, cloud manufacturing presents a more collaborative and resource-efficient model. Moreover, cloud manufacturing opens the door to a broader community for acquiring explicit knowledge within a less formal and more dynamic context.

2.2.4 Siemens Insights-Hub

Siemens' Insights Hub (formerly known as MindSphere) [26] exemplifies a sophisticated industrial IoT application suite. Designed to accelerate production processes and enhance adaptability, people-centricity, and holistic integration, Insights Hub addresses the ever-evolving demands of the industry.

This platform spans from edge to cloud, a feature underscored by V. Kishore Annanth *et al.* [27], making it a prominent solution in the industrial IoT landscape. By collecting data from products, systems, and plants, Insights Hub drives innovation in business models, optimizes operations, and enables advanced analytics and AI capabilities. Furthermore, the platform facilitates low-code application development through Mendix and provides seamless integration with critical systems such as Service Lifecycle Management (SLM), Product Lifecycle Management (PLM), and Manufacturing Execution Systems (MES).

Despite its impressive capabilities, Insights Hub has not garnered substantial research interest due to its closed-platform nature and Siemens' market strategy, exclusively facilitating compatibility with their hardware systems. This restrictiveness limits broader academic exploration, creating a barrier to open innovation and collaboration.

However, Insights Hub remains a compelling starting point for ideation on this topic, particularly in demonstrating integration capabilities and facilitating the marketplace concept within the industry. By showcasing how diverse systems can interoperate efficiently, Insights Hub sets a precedent for future industrial IoT applications distributed in a community marketplace, highlighting the potential benefits of comprehensive integration and seamless data flow across various industrial processes.

2.2.5 vf-OS: Virtual Factory Open Operating System

Following the previous example, we delve into a compelling case study within the realm of marketplace ecosystems, exemplified by the vf-OS. vf-OS is an exceptional initiative, funded and supported by the H2020 Programme of the European Commission, with the primary objective of establishing an Open Operating System customized to meet the specific needs of Virtual Factories [28].

This broad system consists of a kernel, an API, and a specialized middleware. These components are specially made to meet the demands of progressive factories. Through this exemplary case, we gain invaluable insights into the practical realization of a marketplace ecosystem, showcasing its immense potential to revolutionize and enhance the manufacturing landscape.

To tackle the challenges presented by the industrial digitalization brought by I4.0 and the transition to I5.0, solutions like edge computing and edge analytics have emerged as essential tools. These approaches involve the **decentralization of computing, storage and networking resources, bringing these capabilities closer to the data sources**. This distributed computing paradigm offers a range of compelling advantages, including reduced latency, faster response times, minimized network traffic, decreased strain on cloud infrastructure and improved energy efficiency.

Within the context of the vf-OS ecosystem for IoT analytics a modular and extensible stack of components, presented by Víctor Anaya *et al.* [29], complements the core operating system, creating a thriving **marketplace of Virtual Applications (vApps) and vf-OS services**. This design serves manufacturing enterprises by simplifying the development of robust and secure manufacturing and logistics applications. It offers a streamlined path for generic application developers to harness enterprise assets and resources aligning seamlessly with the dynamic landscape of I4.0, as represented by Figure 2.1.

Figure 2.1: vf-OS ecosystem representation by [29].

According to Danny Pape *et al.* [30], vf-OS uses a Service Oriented Architecture (SOA) methodology in which different vf-OS components implement different functionalities. The entire vf-OS architecture is made up of the arrangement of these connected components. With this method, all project participants implement and publish a REST interface, facilitating smooth data interchange with the message bus—a crucial component. Event Driven SOA characteristics, which enable components to identify their patterns of interaction and react to both internal and external events, are another element of vf-OS.

The fundamental components of the architecture include:

- **Environment Block**. Functions as the operational framework for the vf-OS structure, serving as a encompassing vessel that houses all integral components;
- **Application Development Block**. Encompasses diverse vf-OS elements essential for the creation of vf-OS manufacturing assets and vApps;
- **Application Services and Middleware Block**. Acts as the hosting platform for vf-OS components utilized by assets and vApps when seeking vf-OS resources;
- **Application-Deployment Block**. Incorporates the entirety of vf-OS components taken into account during the activation of the vf-OS environment.

vf-OS documentation has also meticulously defined and described the API endpoints and data models for their platform, offering comprehensive guidance for developers. Each endpoint is introduced with a brief description of its purpose and associated parameters are detailed for successful response generation. This information sets forth specific guidelines and standards for developers to follow in order to build applications ready for seamless integration.

The environment created by the vf-OS platform architecture offers the setting for installing, using and accessing vf-OS assets. These resources include tools, services and applications. Visible applications communicate with one another through web-based technologies including web-based GUIs, REST-services, and contemporary message buses. Because data generated within factories and end-users' legacy systems may be leveraged within cloud apps, this platform provides a heterogeneous paradigm that supports both "*InCloud*" and "*OnPremise*" deployment [31]. This matches with the particular requirements of the manufacturing sector.

vf-OS assets adhere to a specified structure and packaging format that is dependent on Docker images in order to maintain consistency. A wrapper structure that contains metadata, security signatures, access rules and dependency information shells every Docker image. According to the work of L. M. D. Cunha, L. Stellingwerff and A. Stam [31], these assets are available for storage in the vf-OS marketplace, making it easier for users to get and install them into the execution environment that the platform offers.

According to F. Fraile *et al.* [32], the value offer for the various client groups on the multi-sided platform includes a number of important advantages, this includes: integrating

manufacturing and logistics operations and promote collaboration throughout the value chain (where manufacturing users can choose and employ vApps from the marketplace or request for software developers to create new solutions); access to a rapidly expanding market of applications (serving I4.0 and future factories); connecting with clients and working together with other developers to create value (providing goods and services to manufacturing beneficiaries); support from the vf-OS ecosystem offering new services and computational options (including cloud hosting and storage).

2.2.6 Zero Defect Manufacturing Platform

Introducing another prominent European project that serves as a reference case study for software interoperability, it is imperative to acknowledge the critical role of Zero Defect Manufacturing Platform (ZDMP) within the context of I4.0 technologies. Giving manufacturers the necessary tools and skill sets to draw significant insights from the huge amounts of data that are constantly being collected from shop floors is crucial in this era of data-driven manufacturing. Because quality assurance greatly depends on I4.0 technologies, it has a big influence on how sustainable manufacturing systems are conceptualized [33].

ZDMP is a comprehensive methodology that seeks to guarantee process and product quality by reducing errors using preventative, corrective and predictive methods. To ensure that no defective items leave the production site and reach the client, it primarily uses data-driven technologies, which improves manufacturing sustainability. It is essential to underscore the pivotal role of software interoperability in achieving a ZDMP system through detection, prediction, prevention and repair strategies [34].

Given the growing complexity of modern manufacturing systems and the diverse heterogeneous industrial software systems responsible for generating and consuming data, integration and processing become indispensable. Some of the most data-intensive technologies, including AI, ML, IoT, XR and Digital Twins, all hinge on data integrity and interoperability as fundamental requirements for the successful realization of the I4.0 vision and beyond.

In the realm of marketplace ecosystems, the importance of mitigating interoperability challenges cannot be overstated. By embracing a standardized approach from the inception of a system or platform design, organizations can significantly ease the process of integrating various software applications and systems. One of the key strategies to address these challenges involves the adaptation or development of a reference architecture. This architecture is purposefully designed to align with the prevalent standards and widely adopted methodologies within the specific application domain. Such reference architectures serve as a blueprint that outlines the structure and design of the ecosystem, offering a standardized framework for integrating software applications, devices and data [35, 36].

Reference architectures, which serve as crucial building blocks for marketplace ecosystems, include the Industrial Internet Reference Architecture, the Reference Architecture

Model for I4.0, IBM's I4.0 Reference Architecture, and the National Institute of Standards and Technology (NIST) Service-Oriented Architecture for Smart Manufacturing. They provide a well-defined structure that facilitates the coexistence of diverse elements, enabling the smooth integration of software applications, throughout the entire ecosystem, and creating standardized shell-based metadata for each module, providing information rules for users and developers, as we will see in more detail in the next section.

Following the SOA presented for vf-OS, the framework outlined in this work by J. Giao *et al.* revolves around a SO-based conceptual architecture that delivers functionalities capable of integrating and standardizing enablers for use by software applications. Enablers, which are open-source software modules, serve as modular components that execute specific services applicable across multiple domains, designed for seamless integration with third-party applications and offering various functionalities [37].

The **Enabler Framework (EF)** component depicted in Figure 2.2 is utilized to address this difficulty, which is made possible by the multiplicity of protocols in the IoT sector. Applications can use a single REST API for all registered enablers by using EF as an interface proxy. **These enablers serve as middleware for storing and retrieving data when they are connected to sensors, actuators, or data storage.** Enablers, by virtue of its service-oriented architecture, make it easier for applications and real-world devices to communicate dynamically. This helps to improve communication with the physical system and integrate updated or new software modules.

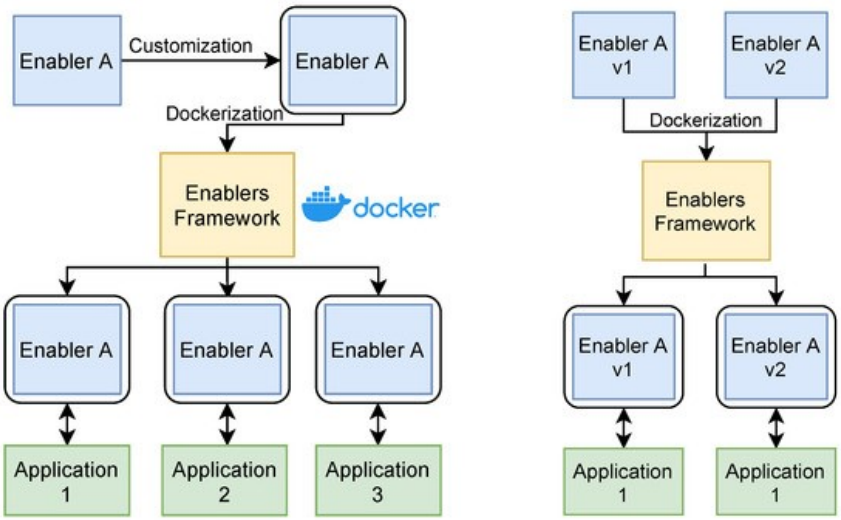


Figure 2.2: Enabler Framework in a Service-Oriented architecture by [37].

Manufacturing facilities are increasingly utilizing several IoT platforms to gather sensor data, enhance production procedures and reduce production expenses. Using a SOA while designing and building software to support IoT-based production processes is one way to handle this move towards efficiency. By providing flexibility in the use and re-purposing of loosely linked IoT services at the middleware layer, SOA-based design aims to reduce integration complexity.

2.3 Collaborative Ecosystems and Industrial Symbiosis

The previously discussed industry-oriented marketplace ecosystems provided valuable insights into common approaches and their benefits for SMEs in the industrial sector. These ecosystems do more than just enhance operational efficiency, they foster the creation of a community by bringing together diverse industry stakeholders. This convergence creates opportunities for collaboration and industrial symbiosis, paving the way for a more sustainable and interconnected industrial landscape.

Unlike traditional manufacturing, sustainable manufacturing incorporates the triple bottom line framework, which evaluates environmental, economic, and social dimensions simultaneously. Abubakr, M. *et al.* [38] identify key challenges in implementing sustainable manufacturing within collaborative and intelligent networks: addressing safety, privacy, and ethical concerns is crucial as these networks evolve; additionally, there is a significant disconnect between industrial practices and academic research in sustainable and smart manufacturing; the complexity of these advanced systems necessitates a compelling economic justification for their adoption; moreover, integrating new smart, sustainable systems with existing infrastructures poses considerable compatibility challenges.

By focusing on collaborative efforts and sustainability, the industry can work towards bridging these gaps, fostering innovation, and creating manufacturing processes that are not only efficient but also environmentally and socially responsible. According to the authors of [38], **this holistic approach promises to transform the manufacturing landscape, ensuring long-term viability and integrity**.

From the perspective of the Circular Economy (CE), Industrial Symbiosis (IS) is conceptualized by [39], as a business model archetype that enhances resource efficiency and creates value from waste through the shared use of infrastructures and by-products. **The foundation of a circular business rests on three pillars: technical innovation, collaboration, and sustainable business model innovation**.

Technical innovation in IS involves exchanging waste, resources, and energy across multiple production processes. This pillar focuses on developing and implementing technologies that facilitate these exchanges, ultimately optimizing resource use, minimizing waste, and contributing for a CE. Collaboration is crucial for IS, requiring the identification and cooperation of various stakeholders. According to the authors of [39], effective collaboration ensures the successful implementation and operation of an IS cluster by fostering synergies between different industries and sectors.

Sustainable business model innovation involves crafting a value proposition centered on waste elimination. This includes designing activities for value creation and delivery that leverage cross-industry partnerships to minimize life cycle waste. Additionally, it encompasses value capture mechanisms that transform waste into valuable resources, conserving raw materials and energy.

According to the work of Zhang, H., *et al.* [40], sustainability management in small and medium-sized manufacturing enterprises often involves separate decision-making

processes for environmental impact, social responsibility, and financial performance. Environmental and social policies are frequently viewed as mere compliance costs necessary to satisfy customer demands and regulatory requirements. This fragmented approach prevents managers from fully comprehending the integrated nature of business sustainability within manufacturing systems, as the economic, environmental, and social dimensions are inherently interconnected.

In response to all of these factors, policies, and their impact on manufacturing sustainability, this paper [40] presents a dynamic model that functions as a decision support tool, aiding managers and engineers in overcoming the limitations of a narrow focus and enabling holistic decision-making that considers the three pillars of sustainability as an interconnected whole.

Furthermore, the authors of [41] believe that **embracing system modularity and innovative collaboration tools enables smaller developers to thrive by allowing them to focus on smaller goals and projects**. This, in turn, provides SMEs with a wider range of options and enhances their competitiveness against larger industry stakeholders, fostering a more sustainable, competitive, and innovative environment for all involved.

Considering a more specific and technical approach to sustainable and collaborative ecosystems, this paper [42], introduces an innovative Enhanced Self-organizing Agent (ESA) within the sustainable Shared Factory framework, designed to facilitate cross-sharing of manufacturing resources through self-organizing communication and negotiation mechanisms. One of the key advantages of ESAs is their ability to dynamically integrate various shared resources without constraints, encompassing not only physical resources but also environmental factors and algorithms. Results indicate that ESAs outperform traditional manufacturing processes by enabling seamless access and sharing of resources, thereby enhancing collaboration between manufacturers and customers. The Shared Factory concept extends beyond physical boundaries, allowing customers to participate in the manufacturing process and ensuring their specific needs are met.

2.4 Frameworks for Seamless Manufacturing Integration

In this section, we delve into the complexities of integrating software with established manufacturing systems, communication standards and asset data exchange. Exploring practical frameworks, the author examines their role in shaping manufacturing architectures and fostering seamless integration with execution systems. The goal is to gain insights into how these frameworks contribute to a unified and efficient approach within the smart manufacturing landscape.

2.4.1 Reference Architectural Models for Smart Manufacturing

Within the realm of smart manufacturing many parts and services can be connected both inside a factory, like smart shop floor equipment, and outside of it, like combining a

cloud-based service with a production cell. This dual integration is generally referred to as horizontal when it extends outside of the factory limits and vertical when it is contained within them.

As emphasized by A. Zeid *et al.* [43], smart manufacturing necessitates the integration of diverse and dispersed cloud-based services, enterprises, smart factories, smart devices and smart processes, in contrast to the traditional "*automation pyramid*" that is common in manufacturing control. Figure 2.3 represents this transition, where integration challenges primarily arise within intra-enterprise hierarchical structures like Enterprise Resource Planning (ERP), or MES. These disparate systems must seamlessly communicate data with one another while adhering to a wide range of communication protocols as a result of this integration. A number of reference models and architectures have been created recently to address the interoperability and integration issues in the field of smart manufacturing.

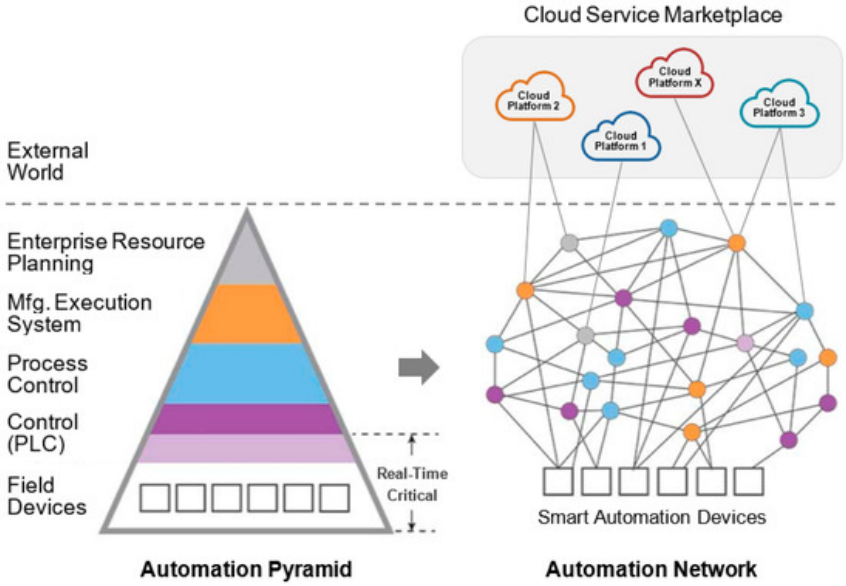


Figure 2.3: Transitioning "*automation pyramid*" towards "*automation network*" by [43].

The complexity of manufacturing processes is reflected in the variety of interoperability challenges that arise. The National Institute of Standards and Technology's Manufacturing Interoperability Program highlights several key variables that affect the effectiveness of interoperability. These include facilitating data transfer between systems, whether commercially similar or dissimilar, and addressing data transfer obstacles within software from the same vendor but with different versions on the same system. Ensuring compatibility across various software versions, encompassing both newer and older releases, is essential. Additionally, mitigating the risk of misinterpreting terminology used for data or information exchange, tackling issues from using non-standardized documentation for data processing or formatting, and overcoming the challenge of not testing applications deemed conformant due to a lack of means for cross-system testing are critical factors in achieving effective interoperability in manufacturing.

The **Reference Architectural Model Industrie 4.0 (RAMI 4.0)**, developed by a consortium of industry partners in Germany, presents a service-oriented architecture that simplifies complex processes into manageable units. As outlined by Lin *et al.* [44], RAMI 4.0 introduces a structured framework with abstract interoperability layers for the manufacturing industry:

- **Business layer**. Including business concepts and details about the elements of the company, like its services and goods;
- **Functional layer**. Characterizing and detailing the functions within the architecture, establishing their relationships. These functions are designed as autonomous entities, independent of processes or specific utilizations in the architecture;
- **Information layer**. Controlling the information and data utilized in the architecture, evaluating the information shared, and guaranteeing the accuracy of the data;
- **Communication layer**. Describing and highlighting interoperability in the communication between levels, systems and components;
- **Integration layer**. Providing connectivity between all architecture layers and physical components, addressing network and software integration;
- **Asset Layer**. Encompassing physical systems like shop floor machines, human interactions with systems and other tangible objects.

In the context of RAMI 4.0, every technical asset within a factory is considered a distinct entity that can be digitally represented, forming an integral part of I4.0 components. This conceptualization is embodied in the layered reference model presented. However, RAMI 4.0 goes beyond by introducing specifications for the Asset Administration Shell (AAS), a critical element of I4.0. **The administration shell encapsulates all essential information corresponding to an asset, including its technical functionalities and ultimately creating a digital representation of this same object in the RAMI 4.0 context**.

At the core of I4.0 communications is the Open Platform Communications Unified Architecture (OPC UA) serving as the standard for AAS communication. Rooted in the ISO 13584 standard, the conceptual definitions for assets leverage AutomationML (mapping to OPC UA during product development) and interoperability ensured with XML and JSON schemas. RAMI 4.0 administration shell meta-models manage data exchange between assets through Unified Modeling Language (UML) class diagrams representing data elements and service interfaces [36].

Xun Ye and Seung Ho Hong [45] present meaningful insights into an implementation of the AAS within a real manufacturing use case, bridging the gap between physical assets and their digital representations, while setting the stage for demonstrating the practical applicability of the AAS framework applied to multiple stakeholders involved in the same ecosystem. This structured approach, encompassing digital representation,

standardized communication and comprehensive data management, underscores the robustness of RAMI 4.0 in facilitating the integration of manufacturing assets within the I4.0 framework.

The **Industrial Internet Reference Architecture (IIRA)**, detailed by Lin [44], is designed as an open architecture applicable across various industries, providing a framework for developing systems based on IoTs without imposing constraints on adherence to specific standards or protocols. This architecture includes enterprise-level business-related data and managing systems, such as MES and ERP, within the business domain. The information area is responsible for data analysis and quality assessment to understand every component of the system. The application domain builds upon the information domain by applying logic to improve system performance and operational efficiency. The operations domain assesses ongoing runtimes, conducting scheduled maintenance and diagnostics to ensure optimal performance. Finally, the control layer features mechanisms dedicated to monitoring and controlling processes within the physical system, including feedback mechanisms.

IBM introduced the I4.0 Architecture, featuring a reference model structured into three layers, each hosting distributed functions [46]. Located at the "*edge*" of the industrial environment or right on the manufacturing floor, the Edge Layer is the lowest level in the architecture, including sensors, devices, and edge computing resources that collect and process data at the source. It plays a crucial role in real-time data collection, local analytics, and immediate control of industrial processes. Above the Edge Layer, the Plant Layer manages operations within the factory or industrial facility, involving supervisory control, process optimization, and coordination of various production processes, often incorporating other systems such as MES, and serving as the bridge between the Edge Layer and the higher-level Enterprise Layer. The highest layer, known as the Enterprise Layer, is dedicated to the organization's broader business facets, connecting industrial processes with business operations to enable data-driven decision-making and strategic planning, often including systems like ERP and Business Intelligence tools.

A manufacturing service bus connects Operational Technology (OT) and Information Technology (IT) in a SOA developed by the NIST for smart manufacturing [47]. It also includes a Business Intelligence service to help with stakeholder communication. All IT operations, from the system level to the enterprise level (ERP, MES, etc.), are included in the IT domain services. The OT domain services, on the other hand, concentrate on tasks and procedures that are assigned at the physical level, particularly shop floor components.

The exploration of various architectures in this section provides valuable insights into the intricacies of smart manufacturing systems and their integration challenges. As the chosen architecture for the development phase of this thesis project, RAMI 4.0's AAS will be instrumental in describing every software module in our architecture. This not only streamlines the integration of applications but also provides a clear framework for development standards, aligning with the envisioned marketplace concept.

2.4.2 Manufacturing Execution System Integration

Within manufacturing organizations, one important class of information and management are MES systems. They can be defined as an essential instrument for manufacturing management, serving as a link between enterprise-level systems and modern manufacturing's shop floor. Resource allocation, production dispatching, data acquisition, quality management, maintenance control, performance analysis, operations scheduling, document control, labor management, and product tracking are just a few of the modules that make up MES systems, according to the MES Association and the ISA SP95 standard [48].

Achieving successful software integration with a MES system involves the implementation of several key strategies. Standardized protocols, such as OPC UA, MQTT or RESTful APIs, play a pivotal role in ensuring seamless communication between MES and an external platform or software [49]. These protocols establish a common language for data exchange, facilitating interoperability while adhering to industry norms. By adopting these standardized protocols, manufacturers can exchange data with their MES in a consistent and efficient manner, reducing the risk of communication errors.

Another key integration strategy is the implementation of a middleware that acts as an intermediary layer that simplifies data exchange and routing between the marketplace ecosystem and MES. It serves as a bridge between the two systems, enhancing flexibility and adaptability in data transfer [50]. Middleware solutions, such as message brokers and service buses, are instrumental in orchestrating data flows, translating data formats and ensuring that information is appropriately directed to its intended destination.

Data mapping and transformation constitute another valid strategy for MES integration. These processes involve the harmonization of data formats and structures, ensuring that data exchanged between systems is accurately interpreted and utilized. Data may need to be transformed to meet specific requirements or standards. Tools like ETL (Extract, Transform, Load) can help in data transformation [51]. Through effective data mapping we can ensure that data is correctly interpreted.

In addition to the already mentioned strategies, APIs provide a set of rules and protocols that allow different software applications to interact seamlessly. By defining well-documented APIs and services, any platform can offer MES systems a straightforward and standardized way to access its functionalities and data, ensuring that information flows smoothly in both directions [52].

A deep understanding of the effective integration and communication strategies just presented is pivotal for the successful implementation of the discussed marketplace platform with existing MES systems. These resources serve as a foundation for improving interoperability, data exchange and streamlined communication, which are essential for enhancing the overall efficiency of manufacturing processes. By leveraging the framework options available, future development efforts concerning data protocols and integration challenges will be substantially reduced, simplifying the software implementation process and promoting seamless communication with MES systems.

2.5 Modular Software Applications

The contemporary manufacturing industry is transitioning from mass production to mass customization of both products and processes, and faces challenges arising from poor interoperability among various original equipment manufacturers. This hinders seamless integration into manufacturing systems, necessitating the development of modular "*plug-and-produce*" applications aligning with I4.0 standards [53]. Current approaches, using a monolithic structure, limit the system's modularity and integration ability.

Torayev *et al.* [53] investigate the modularization of industrial systems using software, with a particular emphasis on containerization and microservices technologies. The suggested approach entails creating a **plug-and-produce, modular and interoperable manufacturing app architecture as well as a manufacturing app development kit**. The idea of "*appification*" makes it easier to distinguish between physical and digital implementations, allowing for modular and reusable solutions. This solution features a web interface for managing various manufacturing apps and is written in NodeJS.

The conceptual architecture presented in Figure 2.4 incorporates atomic devices, physical and computational nodes, manufacturing process clusters, an architecture manager and a global applications repository.

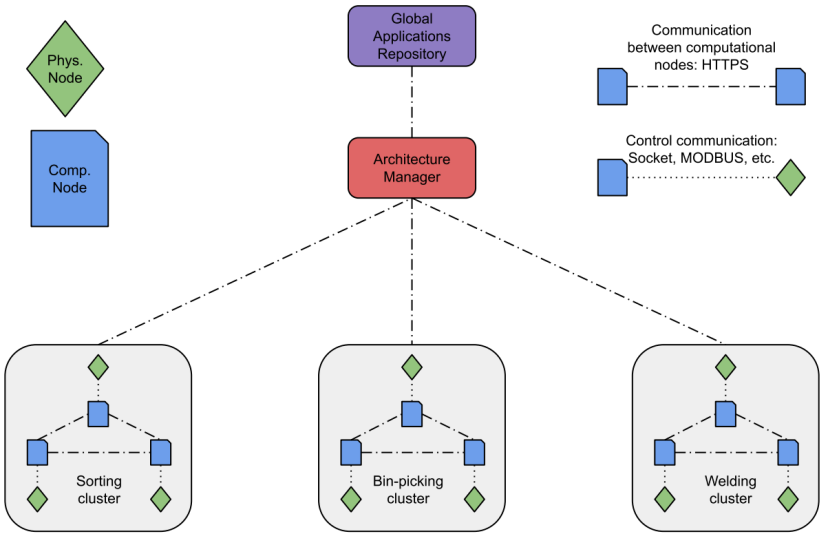


Figure 2.4: Proposed architecture by [53] focusing on microservices and containerization technologies.

To improve the modularity and interoperability of manufacturing systems, advanced management tools are implemented and containerization is achieved through the use of Docker. However, more complex orchestration technologies, such as Kubernetes, can be used to orchestrate computational nodes.

Providing manufacturing engineers access to advanced management tools and even an interface that allows developers to concentrate on software solutions rather than

management-related issues, the architecture manager enables the plug-and-produce property. By centralizing the publishing and delivery of manufacturing apps, the global applications repository promotes process reuse for analogous operations.

This study [54] is another pertinent example using online, cloud and containerization technologies. It offers a unique architecture for virtual commissioning of control software. The suggested method deploys IEC 61499 runtime environments and control programs by using Docker and a cloud computing platform. The cloud-based architecture involves **deploying virtual images of devices constituting the distributed automation system as containers in the cloud**. These containers include details such as the operating system, runtime environment and specific composition of function block libraries. A cloud-based virtual commissioning website facilitates system configuration, offering off-the-shelf virtual devices and allowing users to customize configurations without extensive engineering work.

The scalability of runtime containers is achieved through Docker, enabling users to deploy dozens to hundreds of runtimes to various devices, as described by [54]. The website, integrated with Docker technology, provides container management services for end-users to customize virtual devices and runtimes according to their needs, using the system configuration file of IEC 61499 as input.

The work of T. Goldschmidt *et al.* [55] provides another architectural example, examining practical use cases that highlight the importance of implementing a **container-based solution for industrial control as a basis for the next automation system architectures**.

The suggested architecture for a multi-purpose controller uses lightweight container solutions like LXC or Docker, which are inspired by the virtualization trend in cloud systems. **Multiple segregated user-space instances can be hosted by the operating system kernel thanks to software container technology**, which is typified by operating-system-level virtualization. Containers ensure the isolation of applications, preventing them from accessing each other's resources, while the container host has the flexibility to limit resource usage for individual containers, such as CPU, memory, disk I/O, and network. The paper emphasizes the dynamic nature of containers, which can be added, removed, started and stopped based on system requirements.

Figure 2.5 presents the discussed architecture presented by [55], designed to facilitate adaptable deployment scenarios. The Information Model serves as a crucial repository, encompassing the system goals, computing capacities, execution state and requirements for reconfiguration. For instance, a system goal might dictate that no controller should be overloaded beyond a specified threshold. In these situations, some containers need to be moved to different controllers that are available.

The Monitoring sub-component of the Deployment Coordinator component connects with controllers by gathering status data from them and updating the Information Model accordingly. Container migrations are carried out by the Deployment Executor sub-component when it receives migration commands from the Planner component.

Like other container systems, the Container Registry is the central repository for

sharing and storing containers. If required security measures are met, it might be located in the factory or be available online through the manufacturer's site. The Deployment Executor uses the Container Registry to perform re-deployment.

The Analyzer component consists of several algorithms that use the Information Model to analyze the condition of the system at that moment and determine whether the goals that were set forth are being achieved. If not, the Planner is prompted by the Analyzer to create a strategy for re-deployment.

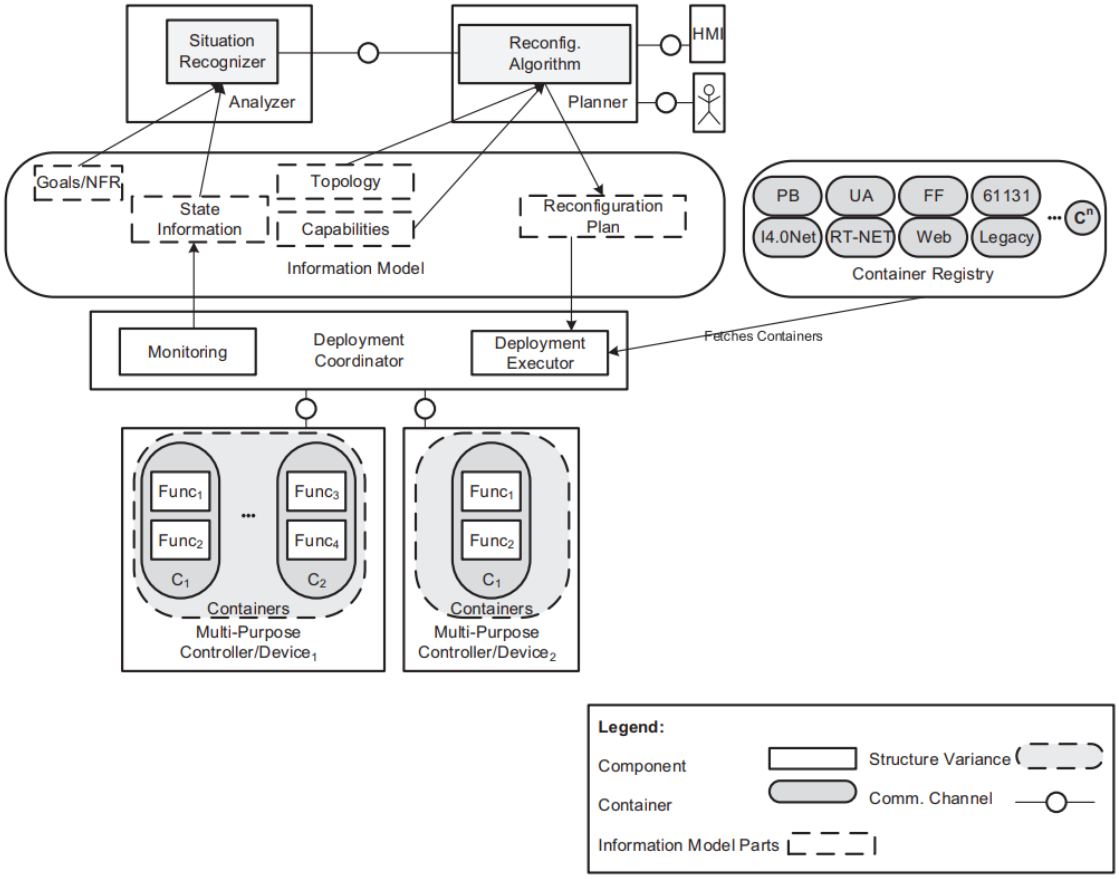


Figure 2.5: Architecture of the container-based solution solution proposed by [55].

In conclusion, the exploration of modular software applications and research in software containerization strategies provides a solid foundation for advancing the integration capabilities of a marketplace platform, particularly one geared towards the manufacturing domain. Understanding the significance of modularization in manufacturing systems, as exemplified by the proposed "*appification*" approach and the utilization of microservices, underscores the potential for enhanced reusability in software development. The focus on Information Model-centric architectures further highlights the importance of a data-driven approach, empowering dynamic decision-making in deploying software components.

The outlined strategies, rooted in real-world use cases and architectural frameworks facilitate the integration of modular and plug-and-produce software solutions. Whether it

be through the cloud-based deployment of virtual components, the adoption of containerization tools like Docker, or the development of architecture managers to streamline the orchestration of manufacturing apps, these approaches collectively contribute to a more interoperable and efficient software ecosystem.

Overall, this state of the art section serves as a guide for navigating the intricacies of software development, integration and containerization, paving the way for a modular and efficient platform.

2.6 Manufacturing Software Applications

The proposed platform and the marketplace ecosystem for smart manufacturing is set to influence various facets of industrial environment, offering a diverse array of applications that respond to the evolving needs of the manufacturing landscape. This hub encompasses a multitude of areas, each contributing to the seamless integration of cutting-edge technologies into industrial workflows. While exploring the diverse landscape of this marketplace, we delve into the concepts of human-robot collaboration, predictive analytics, advanced image recognition technologies, fault detection and safety applications, all of which converge to shape the future of smart manufacturing. This section sets the stage for a comprehensive exploration of the various applications in the domain of smart manufacturing.

In this survey, Charith Perera *et al.* [56] discuss the IoT from the standpoint of the industrial market. Their analysis indicates that third-party application developers, IoT cloud service and platform providers and device makers form the IoT marketplace. Voice, gesture, and touch are just a few of the user engagement modalities used in the world of IoT solutions. Notably, advances in natural language processing and semantic technologies have drawn a lot of attention to voice-activated solutions. Examples of voice-activated personal assistants are Ubi and Amazon Echo.

The significance of the suggested Sensing-as-a-Service model [56], which aims to create a market for buying and selling data supported by the use of about 30–40 distinct types of sensors, is highlighted by the diversity of insights obtained from data in different fields.

The key advancements envisioned for future smart factories include the establishment of an open workspace facilitating autonomy for both humans and robots. This entails collaborative decision-making and action optimization, leveraging the complementary skills of humans and robots to achieve tasks. Regarding Human-Robot Collaboration (HRC), J. Arents *et al.* [57] present a review on this subject and its trends in the context of smart manufacturing. The focus is on exploring how human-robot interaction methods play a pivotal role in fostering secure and efficient HRC.

Shared autonomy is another crucial aspect, featuring a collaborative workspace where tasks are dynamically scheduled between humans and robots, allowing a robot to share control optimally in various degrees of freedom with a human.

The research outcomes detailed by Linn D. Evjemo *et al.* [58] encompass enhanced flexibility, productivity and conditions related to Environment, Health and Safety (EHS) for human workers, while robots are expected to possess learning capabilities achieved through seamless and natural communication with humans.

S. Robla-Gómez *et al.* [59] go into a thorough discussion covering methods for estimating and evaluating injuries in human-robot collisions after the EHS conditions for workers. The study looks at software and mechanical approaches to reduce the impact of humans on robots. This comprises impact detection systems and collision avoidance or reduction tactics. A safety mechanism is triggered when an impact danger is identified, generating alternate trajectories to steer the industrial robot away from people.

The study also covers current research on gadgets like RGB-D devices and ToF/PMD cameras that simplify the extraction of 3D information from scenes and provide 3D information to improve safety in robotic environments. In order to reduce casualties in the event of a human-robot accident, the paper also describes alternative mechanical systems and safety measures for collision detection, acknowledging that collision avoidance is not always guaranteed in human-robot collaboration.

Another very interesting research utilizing hand gestures for interaction with an industrial robot is the one presented by Ahmed R. Sadik *et al.* [60]. According to the authors, operators can communicate with the robot through intuitive hand gestures, fostering a seamless and cooperative working environment. The integration of gesture-based interaction enhances the flexibility of manufacturing processes, allowing operators to convey commands, instructions, or preferences to the robot with natural hand movements. This innovation promotes a more ergonomic and user-friendly collaboration between humans and robots.

Regarding another domain for application development, the concept of self-asserting robot-skills for manufacturing is presented in [61]. It involves treating individual robot skills as isolated applications that can be integrated into an existing system. These skills are akin to modular blocks, providing the flexibility to selectively choose and activate specific functionalities based on operational requirements. By breaking down robot-skills into distinct, self-contained applications, the overall system becomes more versatile, scalable and efficient in addressing diverse manufacturing tasks. This opens the door for the development, integration and commerce of modular robot-skills as applications.

A thorough review of Deep Learning (DL) techniques and their applications in smart manufacturing is given by [62], which explores AI applied to manufacturing software applications. Convolutional neural networks, restricted Boltzmann machines, auto encoders and recurrent neural networks are among the DL architectures that are discussed and are essential for manufacturing intelligence. Specific applications are highlighted, including descriptive analytics for product quality inspection, diagnostic analytics for fault assessment and predictive analytics for defect prognosis.

ML is another dynamic field within the vast subject of AI. These surveys [63, 64] explore how this technology is developing in the context of smart manufacturing and

provide insights into its rapidly expanding applications and current research.

Deng and Yeh [65] offer a method to characterize the cost space and ascertain a distinct optimal cost estimator for a product by using a Least Squares technique with Support Vector Machines; A decision support system that uses genetic algorithms to explore and optimize a collection of elemental heuristics while being directed by a predetermined set of hyper-heuristics is proposed by Woodward and Gindy [66]. Yusof *et al.* [67] concentrate on using ML models to address flexible manufacturing systems, specifically in scheduling machine time for production line operations; Wu *et al.* [68] use ML and Neural Networks to predict the percentage of useful life left in rotating equipment.

Computer vision-based part inspection and process monitoring is a highly influential use of ML in manufacturing. Rai *et al.* [64] have underlined the significance of utilizing ML techniques in conjunction with affordable sensors such as RGB cameras to facilitate successful high-throughput part inspection.

Chen *et al.*'s paper, "*A Data-Driven Method for Enhancing IC Wire Bonding Defect Inspection*" [69], employs advanced ML and data-driven algorithms to improve the accuracy and efficiency of automatic inspection. The focus is on identifying and classifying defects in IC wire bonding through image analysis. Glaeser *et al.* [70] focus on industrial cold forging, employing DL for fault detection. This research highlights the adaptability of ML in addressing faults in the manufacturing system. Zangaro *et al.* [71] contribute with a supervised ML approach for assembly line optimization. This research showcases ML's capacity to empower personnel for informed decision-making and process enhancement in assembly line contexts.

An important breakthrough in safety management for businesses is the creation of a AI-assisted Safety System and Industrial Safety Apps. To improve safety detection skills, these two studies [72, 73] provide a system that combines specialized picture recognition technology and DL algorithms. Businesses can quickly detect abnormal production behaviors thanks to the system's real-time monitoring and early warning capabilities, which reduces the risk of production accidents and significantly boosts the effectiveness of safety management. Numerous applications have been found to meet particular needs in terms of safety:

- **Warning and Detection App**. Focuses on identifying unsafe behaviors in high-risk operations. An application example is the case of [74], that focuses on developing a system that utilizes advanced DL algorithms to identify whether workers are wearing the required safety gear, such as helmets and masks, in real-time;

- **Personnel Positioning App**. Aims to track and manage the location of personnel for safety purposes. An example of Human Detection in manufacturing sites is the research carried out by [75]. By leveraging advanced DL algorithms, the study aims to enhance safety enforcement through accurate and efficient detection of human presence in industrial settings;

- **Safety Training App**. Facilitates the management of safety training programs and the simulation of real case scenarios helping to prepare workers for emergency situations;

- **Intelligent Inspection App**. Makes use of intelligence to conduct effective and efficient safety checks. The researchers suggest a real-time object detection system for early-warning substation security in this paper [76]. You Only Look Once (YOLO)v5 is a Deep Neural Network that forms the basis of the system;

- **MSDS Querying App**. Provides quick access to Material Safety Data Sheets (MSDS) for better information on materials used. Manages equipment integrity and predicts maintenance needs;

- **Major Hazards App**. Monitors safety parameters like liquid level, temperature, pressure and gas concentration for potential major hazards.

Radio-Frequency Identification (RFID) is another prominent technology that enables the tracking and identification of objects in real-time through the use of radio-frequency communication. In a flexible manufacturing system, this technology is employed to monitor and manage the movement of components, products, or equipment throughout the manufacturing process as described in [77]. By doing so, the system can ensure efficient quality control measures within the manufacturing environment.

Finally, [78] aims to systematically organize knowledge in the field of Industrial Wearables, evaluating the relevance of their implementation in enterprises as a technological means to uphold occupational safety. Industrial wearables refer to devices or technology integrated into clothing or accessories worn by industrial workers, often designed to enhance safety, productivity, or communication. Ekaterina Svertoka *et al*. seek to contribute to a better understanding of the role and significance of industrial wearables in promoting occupational safety within diverse work environments trough the implementation or integration of software applications that monitor the information provided by these devices in a specific way.

In summary, the diverse array of software applications explored in this section provide a comprehensive overview of the multifaceted landscape of manufacturing. From human-robot collaboration and predictive analytics to advanced image recognition technologies, fault detection, safety applications and industrial wearables, these examples highlight the possibilities within smart manufacturing. These applications not only address current challenges but also pave the way for future innovations in the industry. Importantly, these identified application ideas, alongside numerous others, can serve as a foundational guide for the integration and development examples of diverse software applications for the marketplace of the discussed platform, that fosters collaboration, innovation and seamless access to a spectrum of smart manufacturing solutions.

2.7 Gap Analysis

The current state of the manufacturing industry highlights several limitations, particularly in the adoption of a marketplace approach to software distribution and the establishment of collaborative innovation networks for knowledge and resource sharing. This section provides a review of the existing literature, identifying the gaps that motivated this thesis' work and leading to the research question presented in Chapter 1.

This literature review began with an exploration of I4.0, emphasizing its digitalization impact and the ongoing transition towards I5.0, where human-centric innovation and sustainability become central. The discussion then shifts towards the potential of revolutionizing how software solutions are accessed and integrated within marketplace ecosystems. Additionally, the review examines the role of collaborative ecosystems and industrial symbiosis, fostering innovation through shared resources and knowledge, and the modularization of software applications, highlighting the importance of enabling flexible, scalable solutions that can adapt to the evolving needs of the industry.

The research conducted has revealed several critical limitations and gaps within the industry. **A major challenge is the lack of robust software distribution solutions that can effectively keep up with the rapid pace of technological advancement.** Additionally, industrial environments continue to face significant integration challenges, primarily due to the complexity and non-modularity of existing systems, which impede the smooth integration of new technologies into established workflows. Furthermore, **the competitive nature of the market, coupled with the inherent complexity of the solutions needed, often results in a reluctance among industry stakeholders to engage in collaborative efforts**.

The following presented work and implementation results were undertaken to develop a solution aimed at addressing these challenges and mitigating the identified difficulties. This research seeks to advance a framework that not only alleviates the complexities of software distribution and system integration but also fosters greater collaboration among industry stakeholders. In doing so, it directly responds to the initial research question, providing a pathway towards more effective and sustainable industrial practices.

3

Conceptual Framework

To properly address the identified needs and challenges in the literature review, the author proposes the creation of a comprehensive solution, the **Industry Modular Operating System**. The conceptual framework presented in this chapter is aligned with both I4.0 and I5.0 principles, allowing users to adapt, integrate, reuse, and upgrade their processes in a more sustainable way. The chapter emphasizes on software modularity and integration, and introduces collaborative aspects applied to co-innovation and industrial software symbiosis. Through this framework, the author aims to provide a robust, adaptable solution that meets the evolving demands of modern industry.

IMOS, as its name suggests, is an operating system built on top of an **OCP**, and designed to deliver a modular approach to software integration and acquisition in industrial environments. This solution aims to offer a set of built-in applications and tools tailored to the diverse needs and interests of various industry stakeholders. By doing so, it serves as a unifying platform that connects different industrial sectors.

IMOS empowers users to progressively construct their manufacturing systems application by application, while also promoting collaborative innovation and resource sharing across the industry. The platform features **three distinct and interconnected modules** that provide a collection of tools to its users. Figure 3.1 depicts these built-in applications and their relationships.

IMOStore focuses on distributing containerized applications and development tools, providing an industry-specific marketplace. This platform provides assets that address specific industry concerns, appealing to both manufacturing and software stakeholders and assuring compliance with IMOS integration standards. Users can access and download a wide range of resources tailored to their needs, including software applications, libraries, AI models, and datasets. Developers are encouraged to publish and monetize their solutions, while collaborating with clients and other developers to innovate and contribute to larger, yet modular software systems. Through these efforts, IMOStore aims to bridge the gap between industry stakeholders, facilitating the monetization of assets and fostering the creation of innovative, industry-driven solutions.

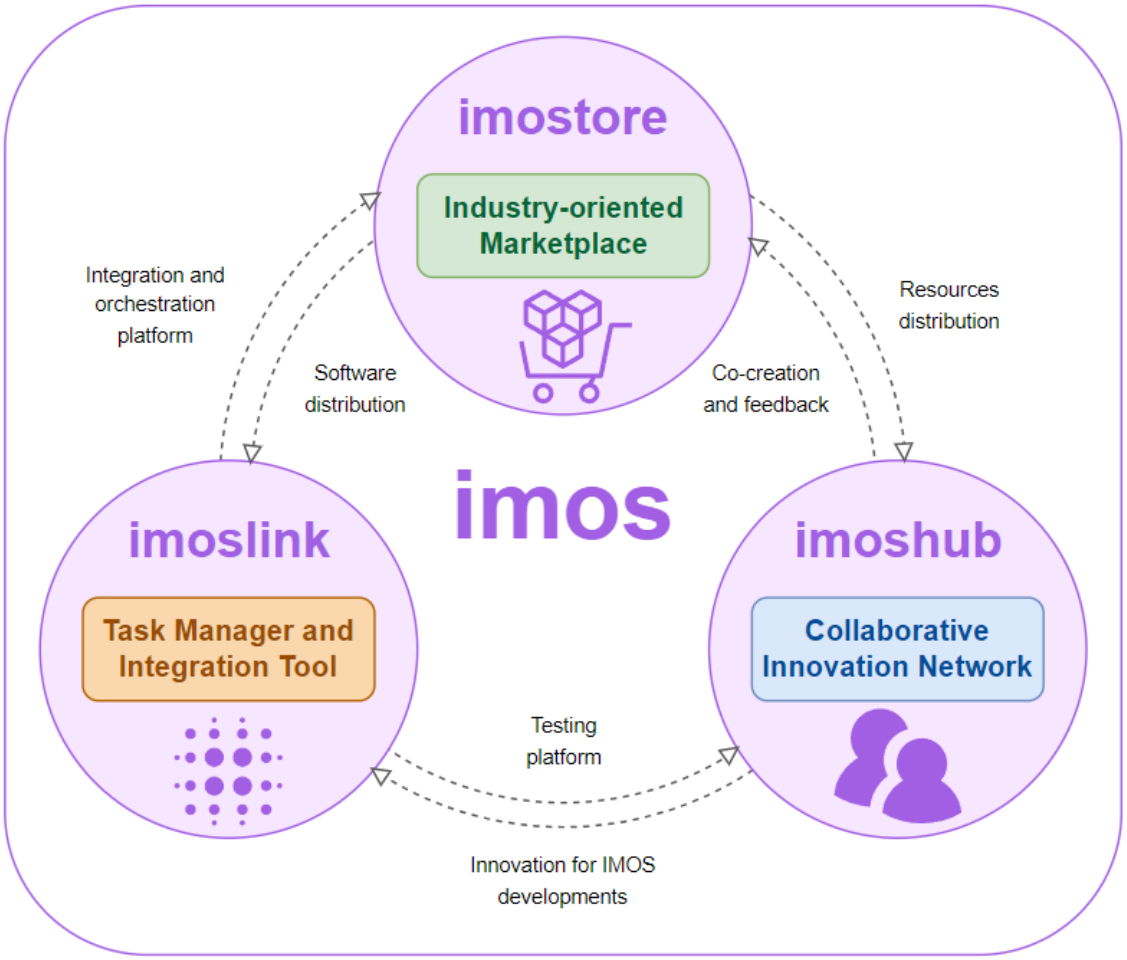


Figure 3.1: Industry Modular Operating System modules.

IMOStore's UML Use Case diagram depicted in Figure 3.2 offers a clear representation of the functionalities and options available from the user's perspective. It distinguishes between two types of users - Clients and Developers -, adapting the marketplace UI and functionalities to each user preferences and roles.

Accessing the IMOStore module requires registration or login. Once inside the application, Clients can browse, acquire, or download apps, while Developers, with their verified roles, have extended permissions and actions, such as publishing new applications or updates, and accessing submission reviews and user feedback.

IMOSlink automates application execution and software integration. It serves as an application environment, centralizing all assets of IMOS, allowing users to quickly access and orchestrate them. IMOSlink, like a task manager or integration tool, is a smaller, more specialized version of the operating system backbone. While it retains the essential functionalities of IMOS as an operating system, its emphasis on visualization aids user decision-making and the orchestration of additional processes. This interface improves usability and efficiency, allowing users to easily manage and integrate software components within the IMOS framework.

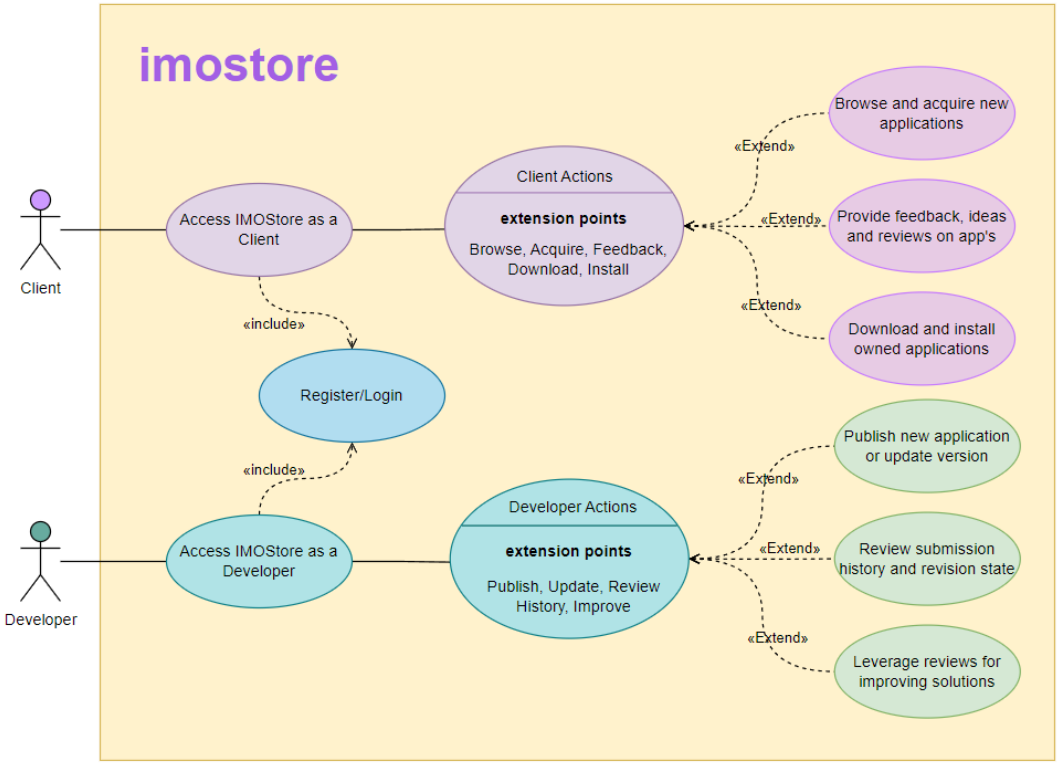


Figure 3.2: IMOStore use case diagram demonstrating user's interaction with the system.

IMOSlink's UML Use Case diagram presented on Figure 3.3 showcases the user's available actions within IMOSlink to manage a set of local apps. When booting the application, an OCP such as Docker is also started, or required to be previously running, in order for the module to properly work. Once the application is ready, IMOSlink's console is prompted, displaying a set of actions and commands to be executed in the IMOS OCP.

Similarly to IMOSlink, Figure 3.4 presents a set of actions available for the user to manage verified applications (published and acquired on IMOStore) on IMOS' cloud service: **IMOScloud**. When an application is published on IMOStore its replica is made available on the server's cloud OCP, allowing every logged-in user to create, start or stop that same application through a set of functions communicated to the server's endpoints, ultimately managing and orchestrating software modules on the cloud.

Finally, the **IMOShub** app stands out as a distinctive component of the proposed ecosystem. It acts as a community-driven platform that supports IMOS by addressing current industry concerns. It enables software developers and industrial engineers to interact and innovate. IMOShub promotes high levels of cooperation by discovering new industry concerns, sharing constructive feedback, promoting real-data exchange, and encouraging joint efforts for integrated solutions. This strategy ensures that IMOS not only tackles current manufacturing challenges, but also fosters continued industry collaboration and innovation.

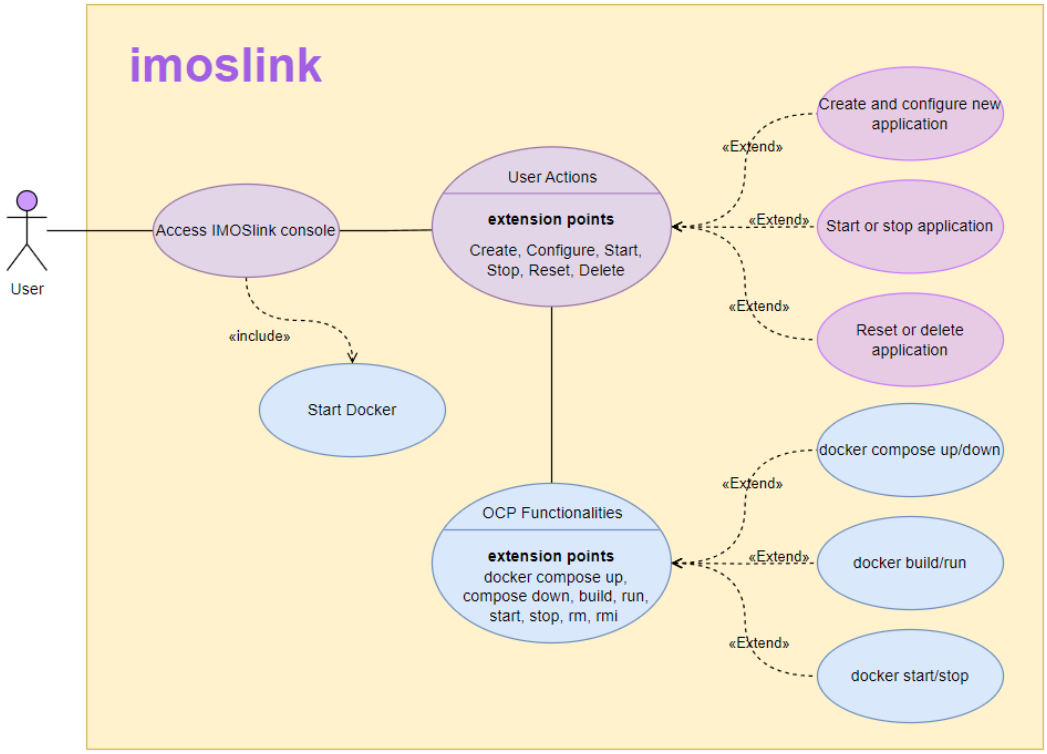


Figure 3.3: IMOSlink use case diagram demonstrating user's available functionalities within the OCP.

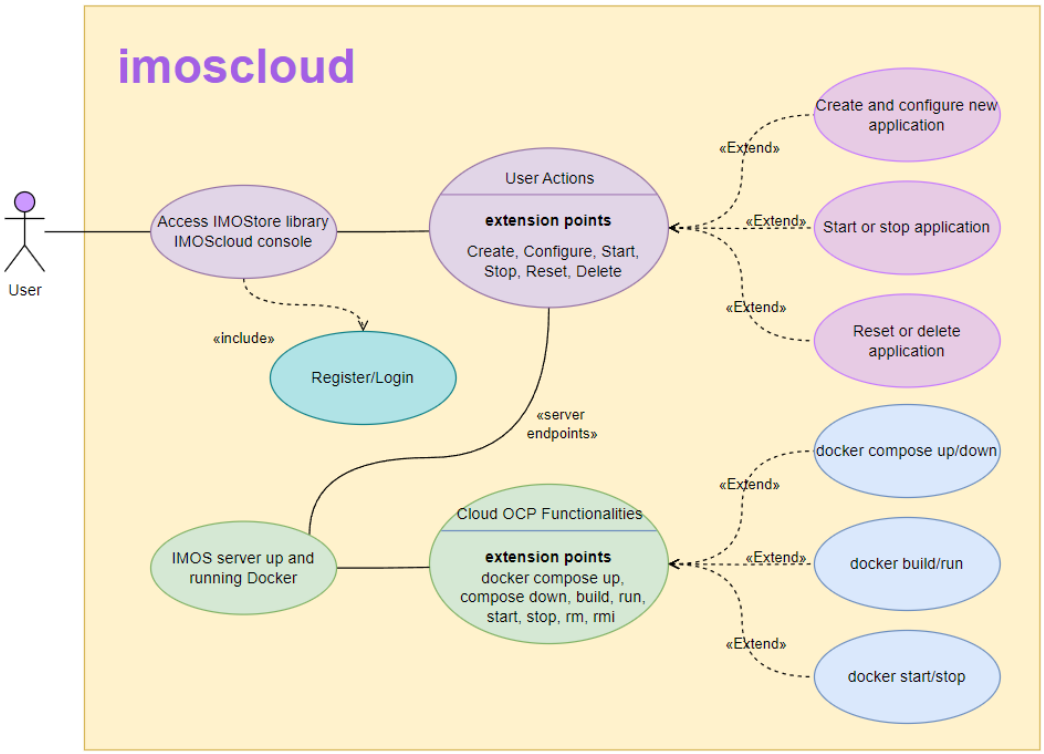


Figure 3.4: IMOScloud use case diagram demonstrating user's available functionalities within the OCP.

These three interconnected modules, while separate, work effortlessly together as essential components of the IMOS ecosystem, fostering software integration, distribution, and collaboration. IMOSlink acts as a framework for integrating new modules obtained from the marketplace, as well as a stage for testing and developing solutions proposed by community. In contrast, IMOStore supports the distribution of software applications and other resources that can be used on IMOS or other platforms. Finally, IMOShub functions as a community forum, offering user support and a platform for co-creation, encouraging collaboration on both marketplace distributed assets and new solutions. Through this framework, IMOS promotes synergy among its modules, increasing its effectiveness in meeting industry's evolving goals.

A notable surge in networking and technology over the past decades has made it possible for a significant rise in collaborative activities facilitated by computer networks and newly developed platforms. According to Camarinha-Matos L.M. and Afsarmanesh H., this trend marks the beginning of what is known as Collaborative Networks (CNs) 4.0, defined by characteristics that align with the goals of the IMOS ecosystem. These characteristics include managing massive amounts of data, monetizing collaborative initiatives, and co-creating value through new business models [79].

In order to address the modularity and sustainability requirements of both I5.0 and CNs 4.0, the dimensioning and development of the proposed IMOS platform were done by these established goals and guidelines. The UN Agenda 2030, which set forth 17 goals for sustainable development, calls on manufacturing to "*build resilient infrastructures, promote inclusive and sustainable industrialization and foster innovation*" [80] is a good representation of this need.

3.1 Collaborative Innovation and Co-creation

A challenge CNs face is the emergence of data-rich environments, which calls for a reexamination of prior design assumptions and new architectures and mechanisms. This same challenge calls for I4.0's higher levels of integration, requiring the involvement of various knowledge domains and, ultimately pointing to the need for new collaboration platforms [81]. However, data-richness also raises concerns about data security, access, and quality, necessitating the verification of platform beneficiaries and monitoring their activities.

The co-creation process facilitates collaboration among various beneficiaries, allowing them to generate new values and resources that contribute to innovation. Value co-creation is seen as a business model that centers on an active **consumer-producer interaction in which consumers can function as the organization's employees or designers** [82], fostering new synergies and relationships between its beneficiaries and allowing both consumers and producers to optimize their outputs.

Esposito De Falco et al. characterize a collaborative digital platform as a "*stable, centralized core of a distributed innovation network*", aiming to foster collective, synergized, and

decentralized breakthroughs, while capturing future market opportunities and capitalizing on external creativity [83]. **By storing ideas in the cloud for easy access, these platforms accumulate knowledge and value over time**, driving a continuous process of long-term value creation. Smorodinskaya et al. take it a step further, describing these innovative ecosystems as "*dynamic collaborative networks of people and organizations formed around projects with an innovation objective*" [84].

Another key concept in collaborative ecosystems and sustainable manufacturing is industrial symbiosis, which is the process by which the wastes or by-products of an industry or industrial process become resources for another. This notion means transitioning from a linear paradigm to a circular one in which waste from certain activities is repurposed as a resource for others.

Industrial symbiosis enables entities and enterprises that are normally isolated from one another to collaborate in resource sharing, which adds to increased sustainability with environmental, economic, and social advantages [85].

The article [41] points out some relevant initiatives, linking product life cycles to collaborative models and implementing collaboration standards to better support the circular economy and industrial symbiosis. These concepts were followed as a means to develop IMOS' framework, answering the proposed research challenges.

Resource sharing drives collaborative innovation in software and technology. Platforms like IMOS leverage co-creation to unite producers and consumers, facilitating feedback, knowledge sharing, and problem-solving. Furthermore, the author proposes **a new vision for the application of industrial symbiosis, extending it from sharing physical resources to sharing remains of software modules or manufacturing data** that otherwise would not have value. This approach aims to facilitate technological advancements towards industry through the collective contribution of community members and platform users.

3.2 IMOS Framework in Collaborative Networks 4.0

In this section, the author presents the proposed IMOS Framework in the context of CNs 4.0, trying to best categorize the platform based on the classes and definitions of CNs, and following the challenges and principles previously discussed in this dissertation document.

IMOS community forum can be classified as a **Collaborative Innovation Network (CIN)**, a sub-class of both Virtual organizations Breeding Environments (VBEs) and Professional Virtual Communities (PVCs), since its membership type can include both organizations/enterprises and people/freelancers [86]. IMOShub supports the creation of collaborative groups, both Virtual Organizations (VOs) or Virtual Teams (VTs), working towards a common goal or joint project, through computer networks.

These groups usually have a life cycle consisting of three main stages: creation, operation, and dissolution. In the creation stage, members unite and establish shared

goals. Subsequently, they work together to achieve these objectives through collaboration and resource sharing. Finally, once their goals are met, the network may disband or undergo transformation to adapt or pursue new initiatives [86].

A collaborative project's success and course of action is usually determined during the construction and planning of a collaborative network. The author set the goal to build **IMOShub as a platform and tool for facilitating the creation stage in CNs life cycles**. It will encourage early connections, talks, and experience exchanges, and facilitate partners search and selection. The presented framework in Figure 3.5 intends to bring entities together in a common space, combining their competencies, sharing resources and working toward common goals.

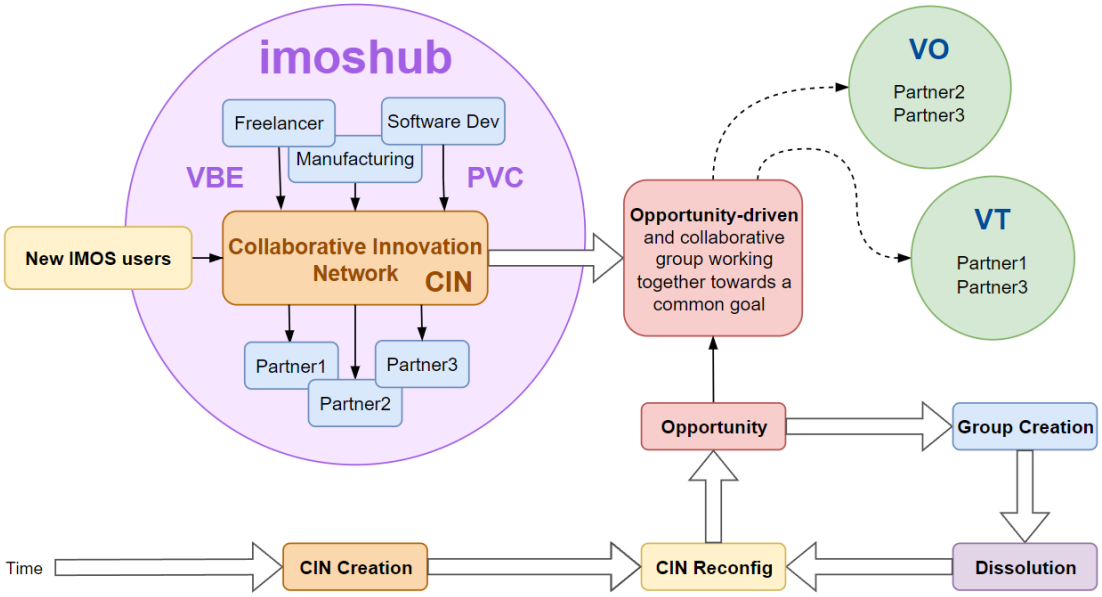


Figure 3.5: IMOShub as a Collaborative Innovation Network.

The majority of CNs created and supported by IMOShub are opportunity-drive and focused on specific projects. This often leads to the formation of VOs or VTs, primarily comprising manufacturing companies (consumers) seeking problem-solving solutions, and one or more software stakeholders (producers) developing applications to address these needs. IMOShub acts as a bridge between these entities, facilitating contact and resource-sharing to foster collaboration.

During the operational phase, the platform offers community-centered features such as problem-solving discussions, application feedback, and knowledge-sharing forums, along with open-sourced dataset publishing and other assets for developers to build upon. Inner-group collaboration is facilitated through GitHub integration offering repositories and workspaces, supporting data interchange, code implementation, and communication.

In the dissolution phase of CNs, IMOShub focuses on extracting value and gathering user feedback to optimize the CIN as a whole. This involves refining collaboration tools and enhancing the user interface based on insights from user interactions and outcomes.

IMOShub, as part of the IMOS ecosystem, works alongside IMOSlink and IMOStore. This all-in-one strategy makes it easier to promote collaboration since the platfor provides so many different tools. From software distribution, to asset integration and orchestration, IMOS is bringing together industry stakeholders with different backgrounds, roles and interests, ultimately allowing users to effortlessly connect, exchange resources, and collaborate on common objectives within a platform that does so much more.

4

IMOS Platform

In response to the identified challenges and the proposed framework, the author worked on the development of a new concept and software solution: IMOS, presenting software integration tools, a marketplace ecosystem with a range of containerized software applications and cloud computing services, and a community of industry experts, fostering collaborative innovation and bridging the gap between production and development. This chapter provides an overview of IMOS as well as the four major phases of platform building and ideation.

As previously stated, the manufacturing industry is undergoing a significant transformation, driven by mass digitalization and rapid technology breakthroughs. This transformation demonstrates an increasing reliance on modular and adaptable solutions to meet the demand for more efficient processes and to effectively respond to changing market conditions.

These tendencies frequently result in issues related to integration, distribution, and flexibility. For instance, integration challenges arise as a result of ongoing changes in work settings and changing standards. The lack of an industry-oriented software marketplace affects distribution and accessibility, resulting in high development costs and other difficulties, specially for SMEs. Furthermore, the persistent gap between manufacturing companies and software stakeholders, along with the lack of industry-oriented collaborative tools, limits adaptability and sets back innovation.

IMOS was proposed and created to address both I5.0 objectives and previously identified challenges. As a result, the author anticipates that the platform as a whole will provide several key benefits:

- **Integration and Synergy**. Manufacturers can choose from a diverse range of containerized applications, selecting those that best fit their needs. These applications are designed to easily integrate on IMOSlink and work cohesively with other apps;
- **Modularity and Customization**. Users can choose from a customized suite of application modules on IMOStore or orchestrate task execution with IMOSlink, optimizing their automation systems for efficiency, modularity, and safety;

- **Distribution**. IMOStore facilitates the distribution of containerized applications, making it easier for users to access, deploy, and share software or digital assets across different environments, reducing the complexities and costs associated with traditional software distribution;

- **Monetization**. Software providers can monetize their applications or services, creating a sustainable revenue stream and using the marketplace to gain visibility and to grow their solutions. Conversely, manufacturing companies can select and pay for only the applications they need, reducing unnecessary expenses;

- **Standardization**. Containerization ensures that software applications adhere to a standardized format, simplifying integration and compatibility across various manufacturing systems, and streamlining asset distribution with IMOStore;

- **Community**. Industry stakeholders contribute to the growth of IMOS as a community platform. This includes providing feedback for each application on the marketplace, participating in discussions about industry challenges and problems, or proposing ideas and solutions for implementation, all of this through IMOShub;

- **Innovation and Co-creation**. Stakeholders are incentivized to innovate in collaboration with other users, including manufacturers, developers and researchers, addressing specific manufacturing challenges proposed by the IMOShub community. This approach to knowledge and resource sharing between the different platform's beneficiaries leads to the continuous development of cutting-edge solutions;

- **Industrial Symbiosis**. By promoting a collaborative ecosystem, IMOS enables different industrial sectors to share resources and knowledge, leading to industry specific and oriented solutions and increased efficiency in the development process leveraging symbiotic relationships;

- **Feedback Loop**. Feedback and data generated by application usage can be supplied back to the marketplace software providers, assisting developers in improving their solutions based on real-world usage, ensuring ongoing improvement and adaption to customer needs;

- **Open-sourced**. The modularity and open-source nature of IMOS allows users to modify, enhance, and create their own modules and applications. This encourages a collaborative environment in which each user can personalize the platform to their specific needs, while contributing to the improvement of IMOS itself. A platform built by the industry, for the industry.

The next sections of this chapter outline the evolution of the proposed platform in a way that best achieves the main objective: to provide an industry-oriented solution capable of providing modularity, integration, collaboration, and software distribution.

4.1 Marketplace as an Enabler of IMOS

With advancements in technology, particularly recent breakthroughs in AI, industry-specific software solutions are emerging at a very fast rate. This rapid expansion frequently outpaces companies' ability to fully adapt or remain at the forefront of innovation in their production processes. As a result, there is a great opportunity to create a platform that distributes these cutting-edge apps while also encouraging collaboration among industry players.

The framework in Chapter 3 presents a software marketplace, central to the development of IMOS. This marketplace offers a wide range of containerized software applications and other assets supporting the development of new solutions. This opens up the possibility of establishing a community hub where manufacturers, engineers, developers, and academics can collaborate to solve new problems by sharing knowledge and resources.

Figure 4.1 depicts the two main beneficiaries in this marketplace ecosystem: manufacturers - SMEs - that gain access to the IMOStore marketplace solutions; and software providers - DEVs - distributing their developed applications, and leveraging the collaboration and data sources provided by the IMOShub community.

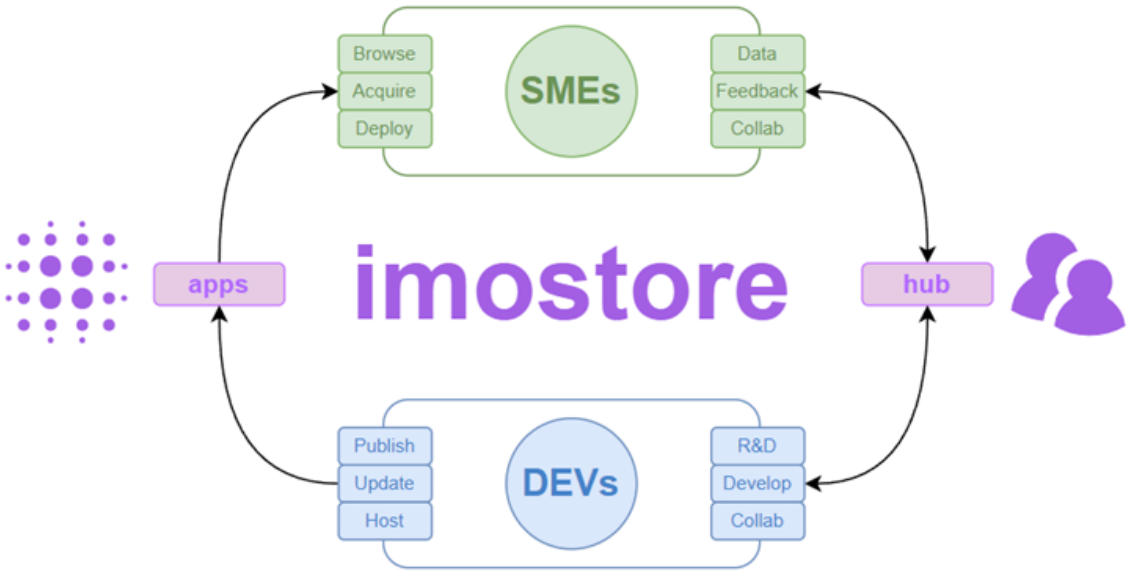


Figure 4.1: IMOStore marketplace concept, bridging the gap between SMEs and DEVs.

Manufacturing companies access the IMOStore, a digital hub that hosts a diverse array of containerized applications, assets and services. These applications are standardized, ensuring seamless compatibility across various manufacturing environments. Clients can explore the marketplace, selecting the software applications or services that best align with their specific requirements.

Software providers play a pivotal role in this ecosystem. Developers create and containerize their applications, ensuring they meet the marketplace's standardized format. These providers showcase their products in the IMOStore, where their solutions gain

visibility and can be accessed by potential clients, ultimately supporting their development efforts. The containerization process ensures that applications are interoperable, creating an holistic ecosystem.

Additionally, SMEs and DEVs can use IMOStore as a community hub, where users provide app-related feedback, share data and experience, and leverage other resources for a boost in development. Manufacturers are encouraged to identify industry needs, review app performance, and share collected data, all of which are beneficial to developers, who may use this direct interaction with manufacturing organizations to create new solutions or improve existing processes.

The author provides the conceptualized flow of activities and decision-making for IMOStore in Figure 4.2. This representation, although simplified, aims to best illustrate the module's usage from a user's perspective.

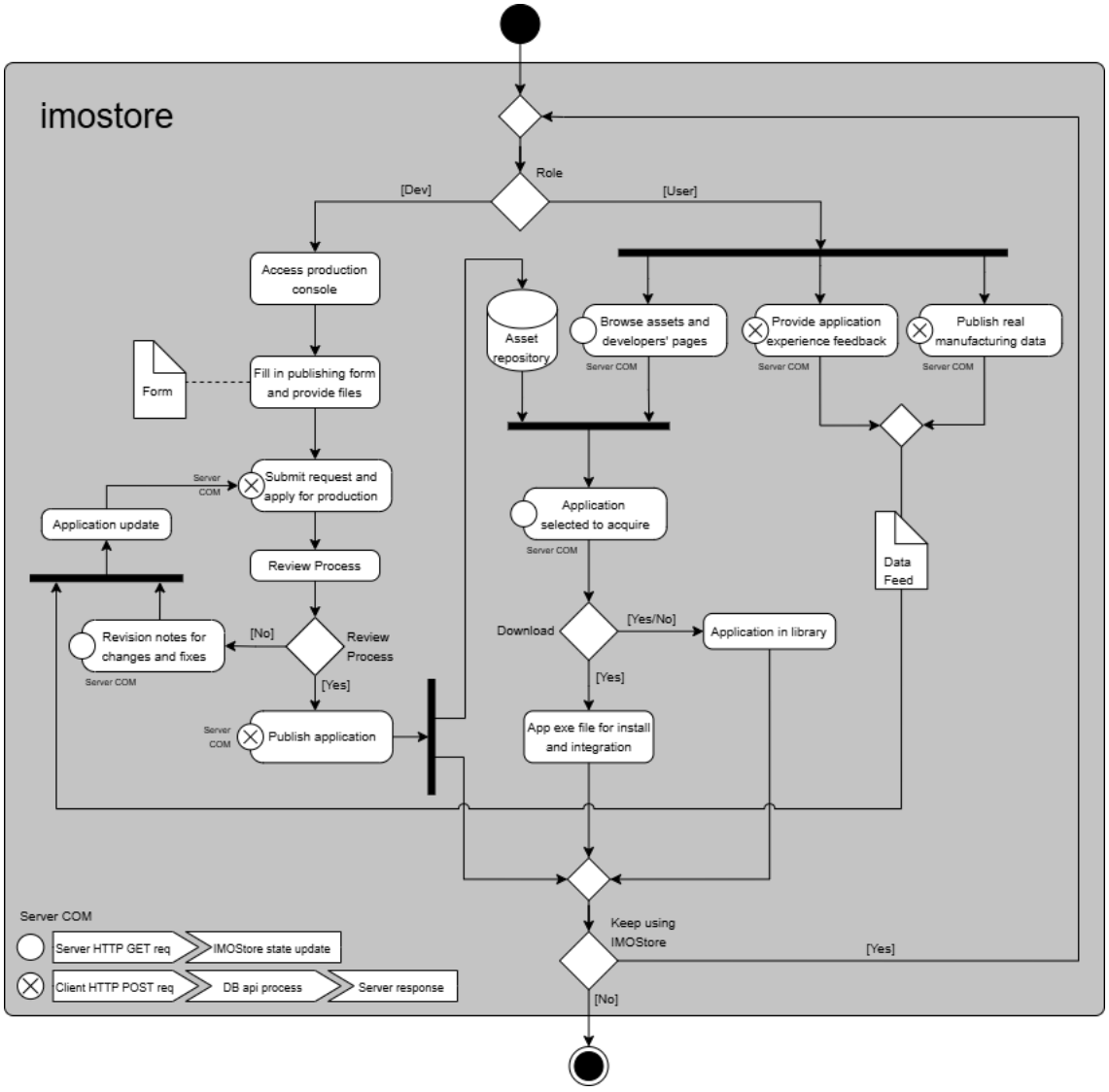


Figure 4.2: Activity diagram representing IMOStore's flow of execution and relations.

When registering on the platform, users can select their role as either a client or a developer, a choice designed to tailor the user experience to their specific interests and preferred level of participation. Developers gain access to a software submission console, enabling them to publish or update applications, while clients are directed toward browsing, acquiring, and downloading applications of interest. In addition, clients can provide feedback on specific application pages, offering valuable insights that developers can use to refine and enhance their solutions.

A compelling example illustrating the benefits of IMOStore and its user relations is the acquisition and integration process of a new safety solution for a manufacturing site. Consider a small or medium-sized company looking to invest in a safety system that uses cameras to detect various PPE and identify areas of risk for safety prevention. **Instead of acquiring an all-in-one solution, the company can achieve the same result by building its system application by application, module by module, at its own pace and need and from different providers. Together, these components form a complete solution that is more affordable, scalable, and sustainable**.

From the developer's perspective, IMOStore enables smaller software publishers to succeed. By focusing on specific and niche solutions, developers end up collaborating with each other, since their applications might be complementary, as described in the previous example. IMOStore grants their solutions more visibility and leverages IMOS's modularity to facilitate distribution and promote interoperability. In addition, manufacturing companies can share real on-site data, enabling developers to improve their work.

This collaborative ecosystem benefits both manufacturers and developers, driving innovation and improving modularity and scalability of manufacturing processes. Ultimately, IMOStore aims to bridge the gap between manufacturing enterprises and software developers, facilitating the software distribution process and supporting an environment for sharing knowledge and resources.

4.2 First Iteration of the IMOS Architecture

In response to the purposed marketplace came the need to develop a solution capable of actually integrating all the acquired resources and applications and orchestrating their execution based on the user's decision-making. **The goal is to create a core platform dedicated to supporting the integration, execution, and orchestration of every containerized asset, while hosting its own built-in apps and tools**.

IMOS was designed to be the central unit (parent) of all other applications (childs), being able to communicate with them through Inter-Process Communication (IPC) and execute commands for process management. Once acquired, containerized applications integrate effortlessly into their manufacturing system flow, allowing operators to easily swap between different application setups for each station thanks to an enabler agent.

Figure 4.3 illustrates the initial iteration of the IMOS architecture. This architecture features the IMOStore module, which handles communication with the platform's server for application distribution, along with the newly introduced IMOSlink module, which operates locally and runs in parallel with the selected OCP. IMOS facilitates the instantiation of new application instances via OCP, while effectively managing the parallel runtimes of each module through IPC.

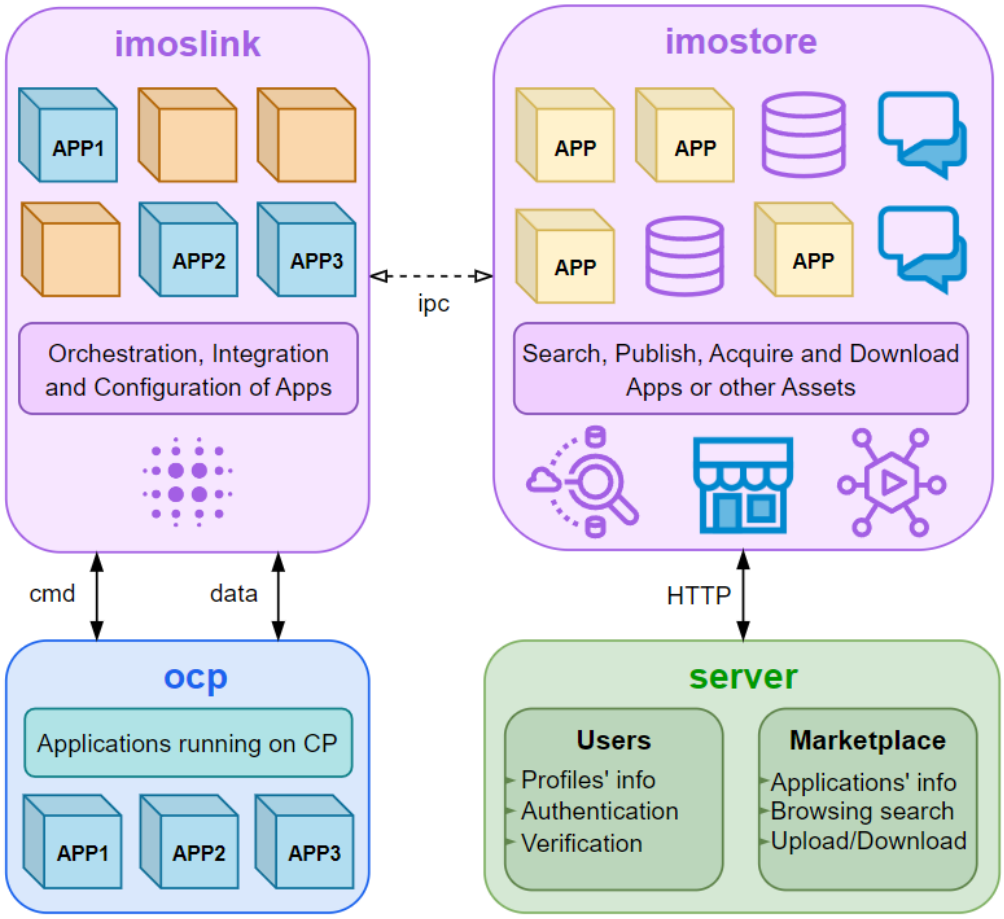


Figure 4.3: First envisioned architecture for the IMOS platform.

Using IMOSlink, users can integrate their own applications or acquire/distribute solutions from/to IMOStore within the platform, configuring and connecting them to hardware endpoints. **IMOSlink serves as the bridge between an OCP and the user**, leveraging its capabilities for seamless integration, distribution, and modularity, thereby fostering a holistic system of interconnected containers with a central orchestrating component - the user. This in-built app offers its users the ability to start, stop and configure each installed application, all through one graphical interface.

The IMOS servers host one dedicated cluster of services for managing user information, authentication routes, marketplace actions and data sharing through HTTP requests. Data generated from users' profile information, marketplace's application files, and community's interactions are stored in a simple database infrastructure.

As depicted in Figure 4.4, IMOSlink operates locally and in parallel with an OCP, functioning both as a container manager and an integration tool for applications acquired through IMOStore. When accessing IMOSlink, users can configure new applications by setting the correct inputs, settings, and endpoints as described by the app's metadata. They can also start previously configured modules, reset task executions, stop applications, or delete any module directly from the OCP. These tools empower users to act as decision-makers and orchestrators of multiple modules, facilitating the integration and management processes.

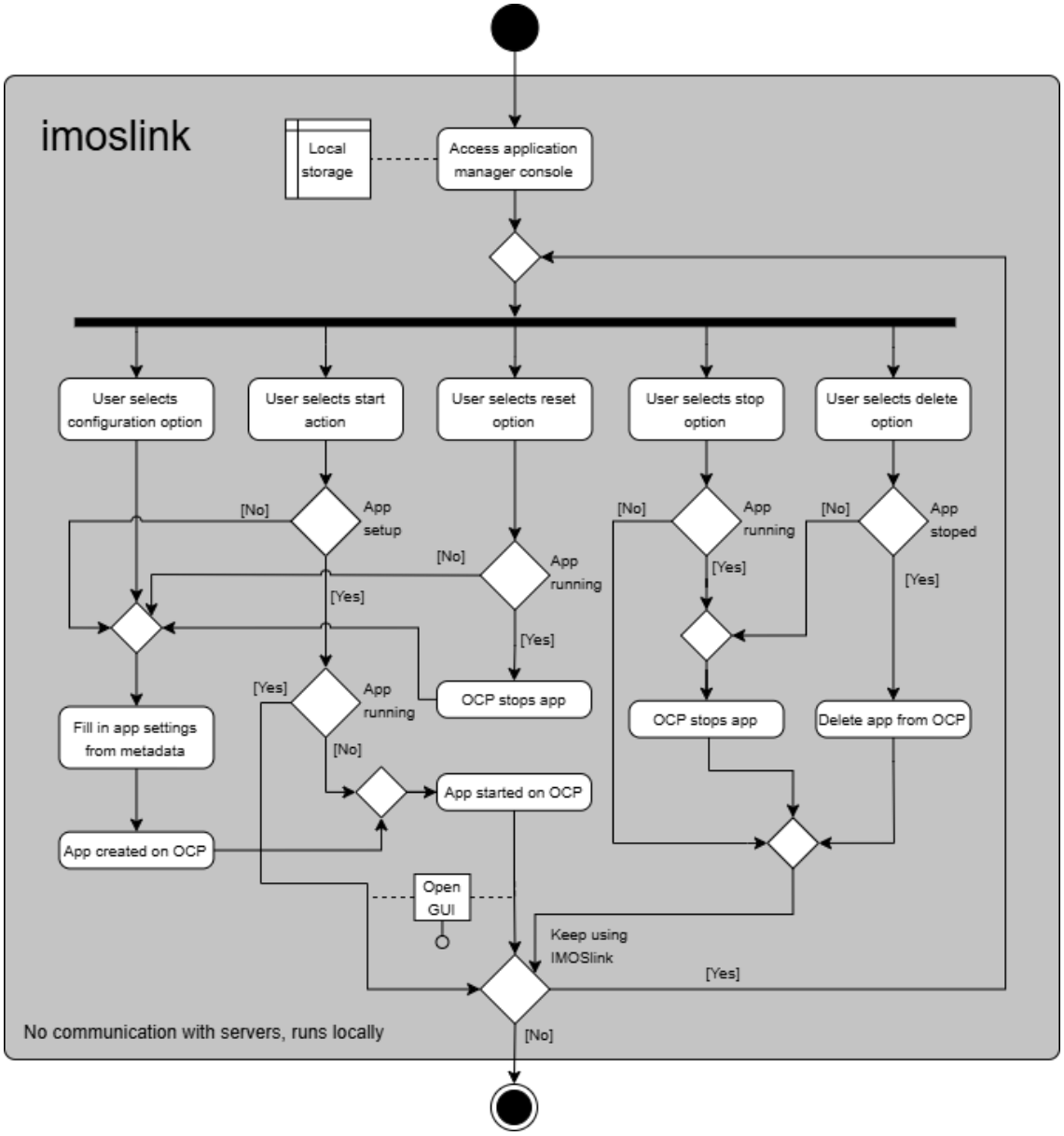


Figure 4.4: Activity diagram representing IMOSlink's flow of execution and OCP managing functionalities.

This phase of the IMOS architecture's development has resulted in a robust initial iteration of the platform's structure and functionalities, effectively meeting the objectives

set during the project's preparation stage. The author successfully developed a marketplace ecosystem for the distribution of software modules within an industrial context, as well as an Enabler Agent, known as IMOSlink, designed to centrally orchestrate and manage these acquired applications. However, this marks only the start of the platform's potential, as further advancements and possibilities will be explored in the subsequent sections.

4.3 Cloud Computing Option for Application Execution

As discussed in the state of the art chapter of this thesis, Cloud Computing is increasingly becoming a preferred option for companies aiming to reduce costs and avoid the complexities of maintaining their own on-premise software infrastructure. This technology offers significant cost savings by eliminating the need for companies to invest in their own hardware and computational resources. This is particularly attractive for SMEs, which often lack the financial and technical resources to manage extensive IT infrastructure.

As the IMOS architecture evolved, the author integrated the concept of Cloud Computing, **enabling users to run their applications both on-premise and on-cloud**, as depicted in Figure 4.5. This integration allows IMOStore to offer its applications as SaaS. Users can simply acquire their desired applications and choose to run them on the cloud without the need for local installation or execution.

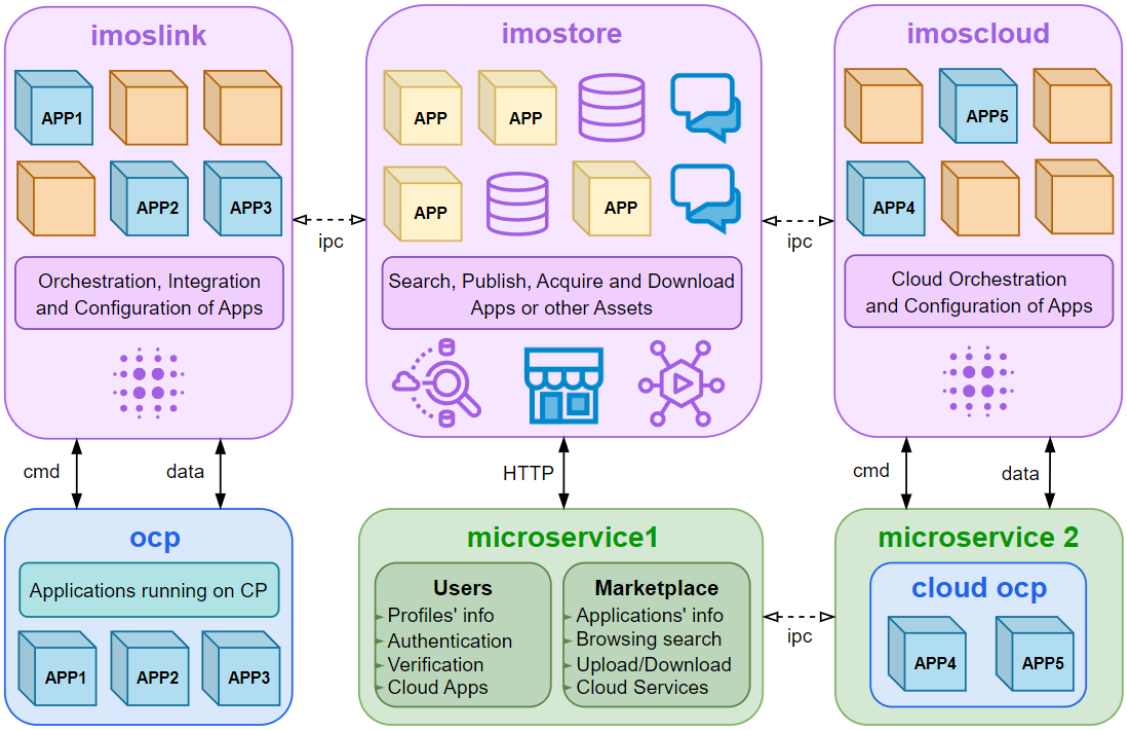


Figure 4.5: IMOS architecture incorporating IMOScloud and a micro-services approach.

Similar to IMOSlink, IMOScloud operates parallel to an OCP, but in-cloud. The IMOS servers now incorporate two distinct micro-services that communicate with each other and provide external routes. The existing IMOStore micro-service has been scaled to include the details and status of cloud applications. Consequently, the IMOScloud micro-service can focus entirely on cloud applications execution, receiving orchestration commands from users, and providing endpoints for accessing generated data.

The behavior described by the activity diagram in Figure 4.6 is very close to IMOSlink's but with a few changes, mostly regarding the fact that the OCP runs on the platform servers, which means that IMOScloud is in fact a cloud instance of IMOSlink. Accessing IMOStore the users will find every acquired asset on their library, allowing them to either download the app for local installation or manage an application cloud execution.

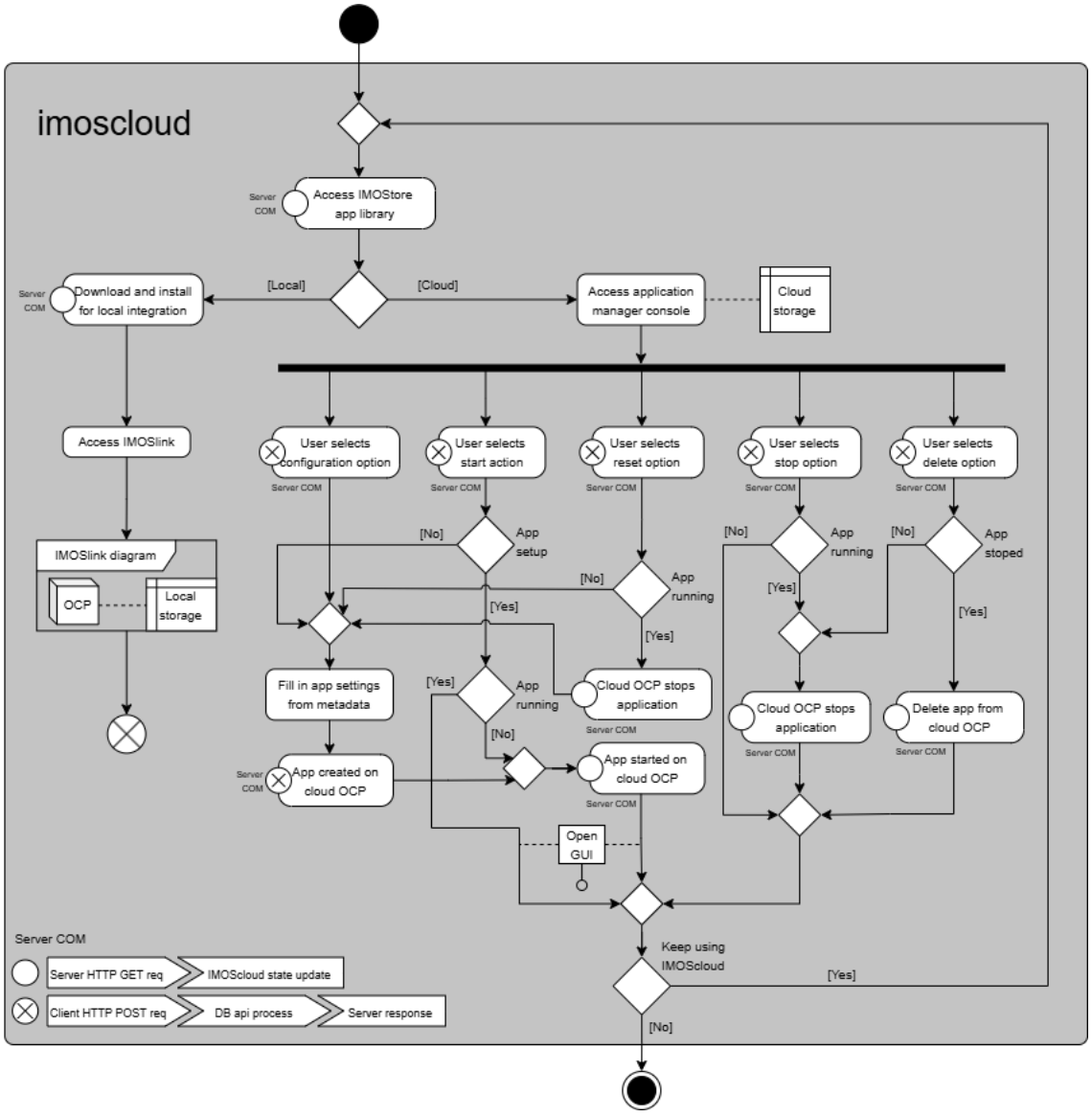


Figure 4.6: Activity diagram representing the flow of execution between IMOScloud and the cloud OCP instance.

This architecture ensures seamless coexistence between client/marketplace logic and cloud-based processes by separating these instances into micro-services. This setup optimizes data responses and reduces latency issues, as the communications between cloud application execution and the user's local instance of IMOS are data-intensive. Additionally, this allows for easier scalability, essential for these types of solutions. For instance, a load balancer can be used to manage and distribute networking traffic across a scalable number of IMOScloud clusters, ensuring efficient performance and reliability.

IMOScloud was introduced as a comprehensive cloud computing solution that prioritizes providing execution environments for certified and trusted applications available on IMOStore, rather than merely offering computational power. It mirrors the orchestration capabilities of IMOSlink, utilizing endpoints provided by IMOS servers, and enables users to efficiently leverage cloud-based resources while retaining the flexibility to run applications on-premise as required. This architectural stage marks the conclusion of the implemented features and the IMOS prototype results presented on Chapter 5.

4.4 IMOShub Collaborative Platform

Finally, as presented in the proposed framework, **IMOShub was introduced to the IMOS architecture, serving as a community forum** that enables industry experts from diverse backgrounds to contribute to innovation and problem-solving within the industry.

IMOShub is the new integrated module within the IMOS ecosystem, functioning as a CIN. It facilitates the creation of new collaborative initiatives, such as VOs and VTs. The forum hosts industry-oriented challenges, questions, and brainstorming discussions, enabling companies and individuals to network and embark on new projects with partners who share similar interests.

Manufacturers are encouraged to identify industry needs, review application performance, and share collected data. This direct interaction is invaluable for developers, who can use the feedback and data from manufacturing organizations to create new solutions or improve existing processes. **IMOShub provides a space for monetizing knowledge, resources and technology, or open-source these contributions** (including datasets, libraries, research papers, scripts, and AI models). This collaborative environment is designed to accelerate development and ultimately enhance production efficiency.

Ultimately, IMOShub aims to bridge the gap between manufacturing enterprises and software developers by providing a data-rich environment. Figure 4.7 consists of the final implemented architecture of IMOS and includes a new application with networking and sharing functionalities, interfacing with an additional micro-service within the servers. This integration ensures distinct communication spaces for users, along with tools for data publishing and resource sharing.

The activity diagram in Figure 4.8 outlines the activities and intended user experience flow for IMOShub. The diagram, though simplified, emphasizes the freedom users have to engage with the community as they see fit.

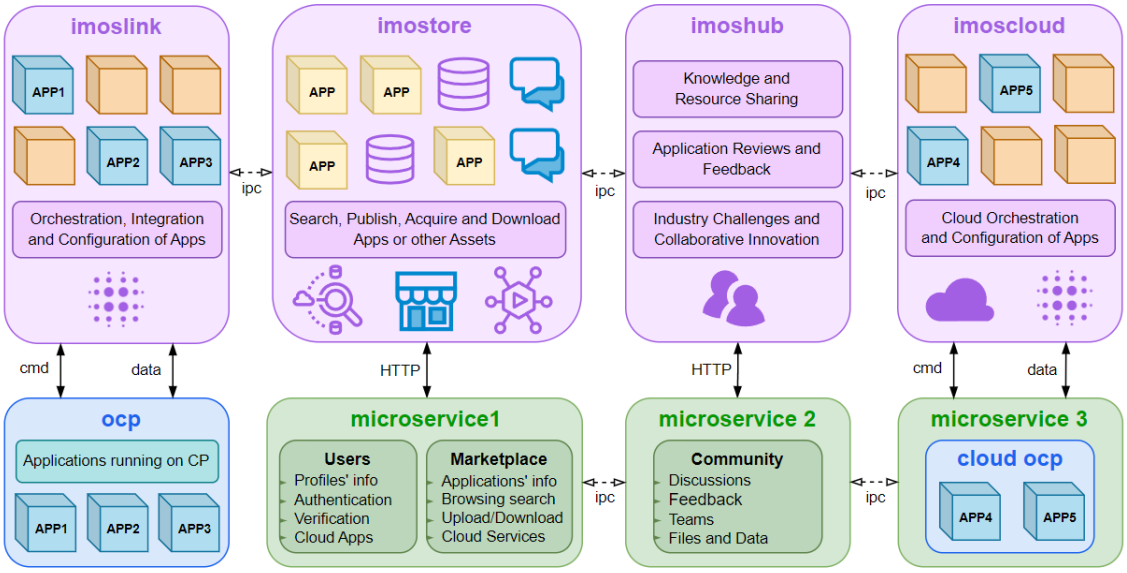


Figure 4.7: IMOShub introducing the final layer of complexity to the IMOS architecture.

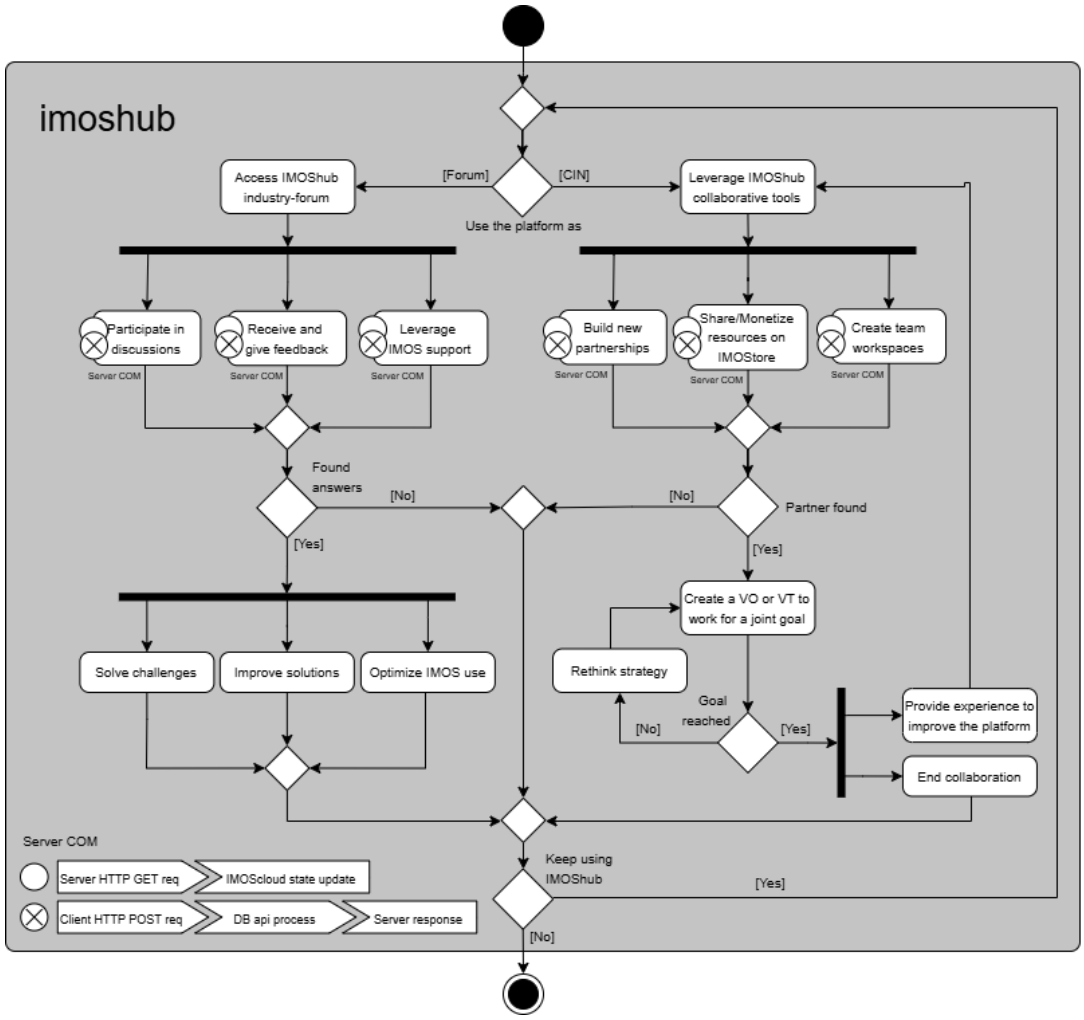


Figure 4.8: Activity diagram representing IMOShub's flow of execution and interactions.

While the module has been conceptualized, it has not yet been implemented. Further developments are needed, focusing on studying and enhancing user experience, refining collaboration tools, and optimizing workspaces to fully realize IMOShub's potential.

Finally, the component diagram provided in Figure 4.9 offers a comprehensive overview of the various modules discussed throughout this chapter and illustrates the culmination of the IMOS ecosystem's evolution. This diagram effectively displays all modules along with their dependencies and connections, particularly describing connections and required interfaces for their inter-operation.

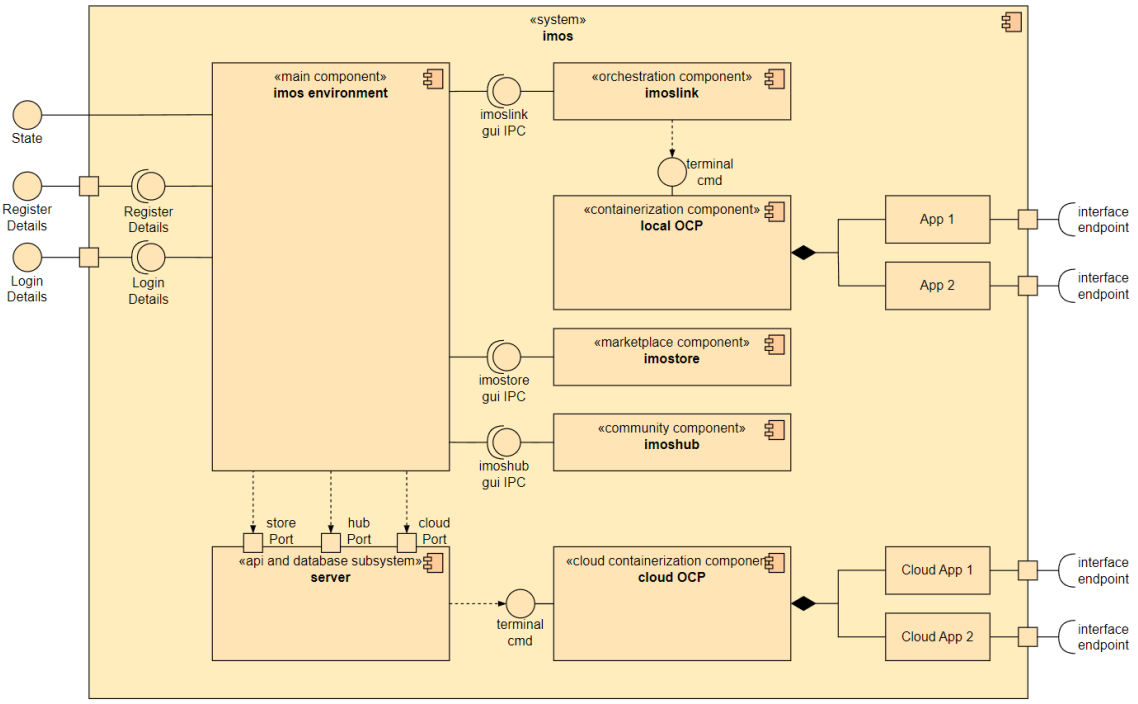


Figure 4.9: Component diagram describing every module as part of the IMOS ecosystem.

This representation aligns with the architectural diagram presented earlier containing the final implemented architecture of this master's dissertation project. The scalable and modular nature of this solution is designed to address the challenges identified in the state of the art and the conceptual framework formulation. These challenges include the scarcity of software distribution solutions that keep pace with rapid technological advancements, the integration issues posed by overly complex and non-modular systems, and the lack of collaborative perspectives in industry environments due to market competitiveness.

In general, the proposed architecture for the IMOS platform was designed to establish a robust foundation for further exploration of the IMOScloud and IMOShub modules. This initiative aims to scale the current solution and address emerging challenges, providing an industry-oriented platform. IMOS promotes software distribution within an integration-friendly environment and prioritizes collaborative tools that foster innovation in line with technological advancements, paving the way for continued advancements that mainly benefit SMEs.

4.5 IMOS as an Open-Source Ecosystem

IMOS was designed as an **open-source platform that fosters innovation and collaboration, rather than monopolizing the industry with another centralized solution** as depicted from Figure 4.10. While providing a robust structure with integrated tools like IMOSlink, IMOStore, and IMOShub, the platform remains flexible and open for users to modify and expand. Each user can create their own unique version of IMOS adapting to their needs and leveraging comprehensive documentation for local application development, without relying solely on IMOStore for growing their ecosystem. This open-source approach encourages the development of new modules and continuous improvement of the platform, aligning with the dynamic nature of I5.0.

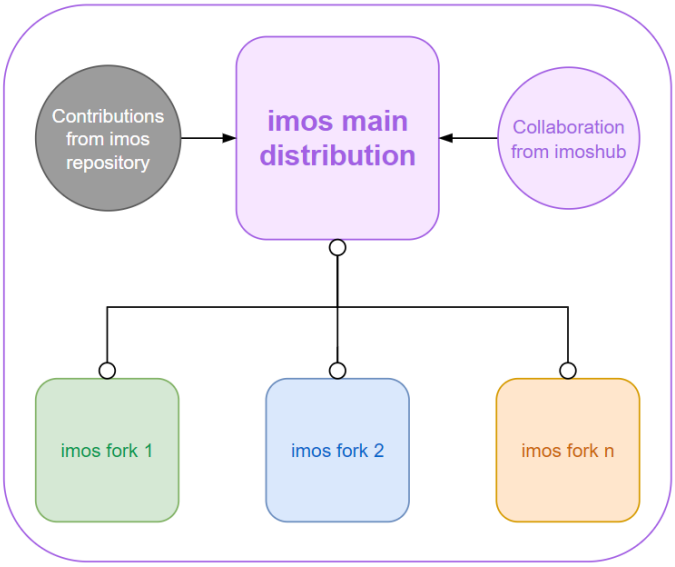


Figure 4.10: IMOS ecosystem as and open-source research project.

Community repository contributors are encouraged to propose and work on changes and improvements for the main distribution of IMOS. To effectively drive advancements, these contributions or issues should be properly handled, **fostering a platform built by the industry, for the industry**.

On the other hand, users can simply access and work on the main IMOS distribution, contributing to a thriving marketplace of community-built, industry-oriented applications. By utilizing tools like IMOSlink, IMOStore, and IMOShub, users are empowered to optimize their processes and find new solutions. IMOShub, in particular, serves as a community hub that promotes collaboration, knowledge sharing, and resource exchange, driving innovation not only within the IMOS platform but also across the broader industry.

5

Implementation and Platform Results

This chapter provides a detailed account of the IMOS' development, including the technologies used, the challenges encountered, and the final platform results. The implementation process was divided into three main stages for better understanding and testing segmentation. Each new stage commenced only after ensuring the required functionalities of the previous one were met satisfactorily.

The three implementation stages identified are: IIMOStore Development and Server Setup; IMOSlink and App Use Cases; and Application Execution on IMOScloud. This chapter presents each of these stages, discussing their implementation work, difficulties encountered, areas for improvement, and the results achieved, aiming to provide a comprehensive understanding of the platform, and a structure to make it reproducible with any suitable technology.

In general, the platform was built in a Windows environment using NodeJS with Electron, supporting the development of the IMOS structure, and Docker, providing a containerization solution that simplifies application administration and scalability. The backend server was constructed as an Express API with a MongoDB database, offering a robust and flexible architecture for data storage and communication. This setup is well-suited for IMOS as a NodeJS application.

Figure 5.1 illustrates a class diagram depicting the entities and data model within the IMOS database structure. This diagram includes the user class, which can be either a client or a developer. Each user can be associated with acquired applications from the marketplace and applications running on the cloud, along with their execution states. For users classified as developers, the diagram also includes associations with app submission requests for publications and updates. This structure ensures a comprehensive overview of user interaction and application management within the IMOStore.

A sequence diagram illustrating user interaction with the IMOS platform and its core modules is shown in Figure 5.2. The diagram is divided into three main sequence lanes, representing user interactions with IMOStore, IMOSlink, and IMOScloud. It details how various platform components — such as application use cases, the IMOS server, local and cloud systems, and the containerization platform — interact with each other and the user.

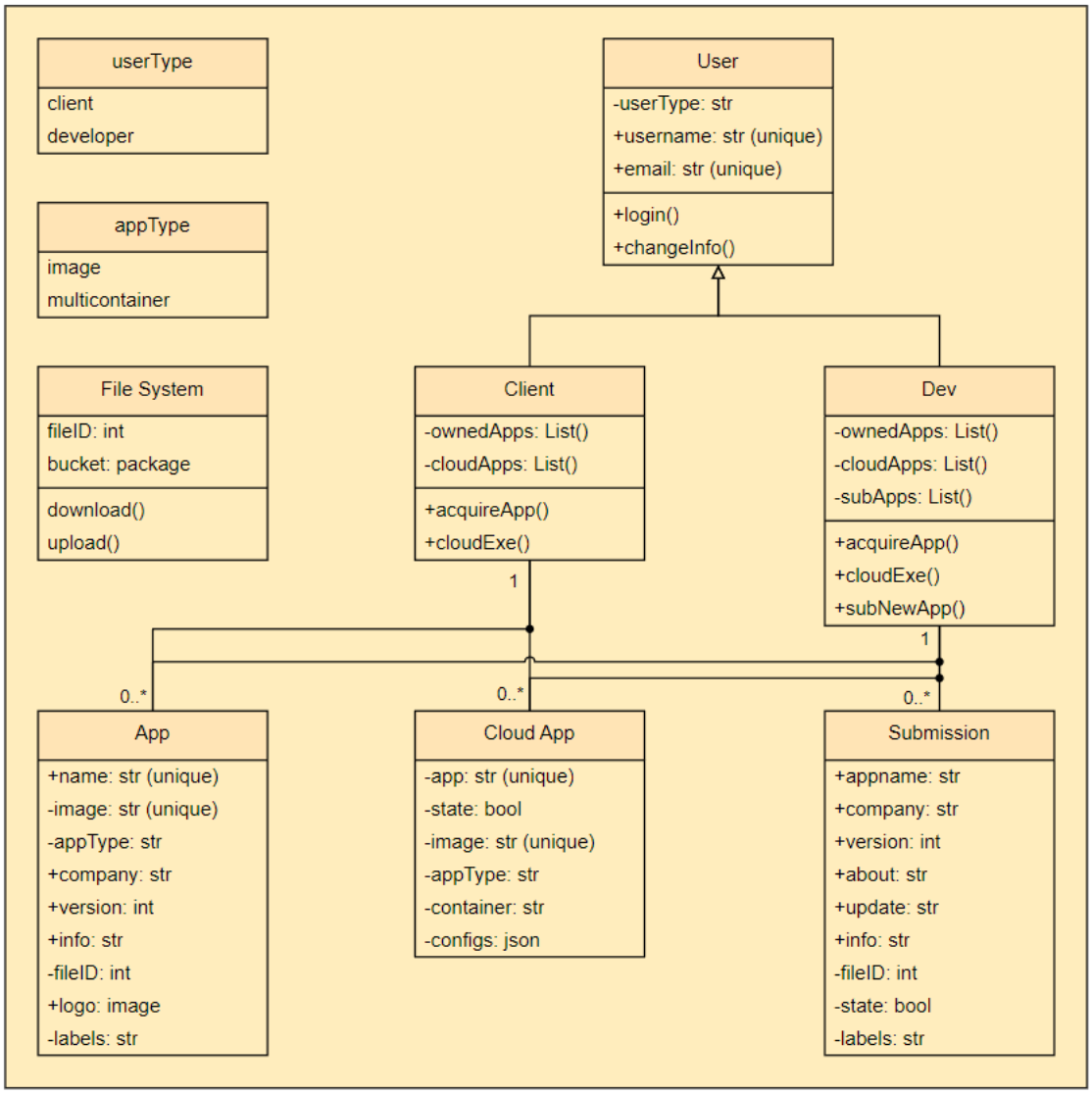


Figure 5.1: IMOS ecosystem class diagram.

It is important to note that all three sequences depend on the user's prior registration and login, which involve communication with the platform's server to validate access and retrieve user-specific data.

In the IMOStore sequence lane, the diagram outlines the relationship between the user and the marketplace module. Here, the user is prompted to search for new applications, add suitable assets to their app library, and download and install these applications locally on IMOS and the local OCP.

Conversely, the IMOSlink sequence lane details the operations of this locally-running module, which requires server communication only for registration and login. As previously described, the user can manage and orchestrate all applications installed from IMOStore, as well as those developed outside the IMOS ecosystem, provided they meet the integration requirements outlined in Section 5.2. User actions within IMOSlink include

configuring apps, starting and stopping applications, and restarting or deleting modules. These commands are communicated to the local OCP, which executes them and updates the application's state within the IMOSlink interface.

Lastly, the IMOScloud sequence lane illustrates the user's interactions with this cloud-based alternative to IMOSlink. Integrated into the IMOStore library interface, IMOScloud allows the user to manage and orchestrate any application acquired through IMOStore, with the same set of actions available in IMOSlink. However, for security reasons, IMOScloud only runs applications published on IMOStore. User commands are sent as requests to the server, which forwards them for execution on the cloud OCP, returning the application state to be updated in the IMOScloud interface.

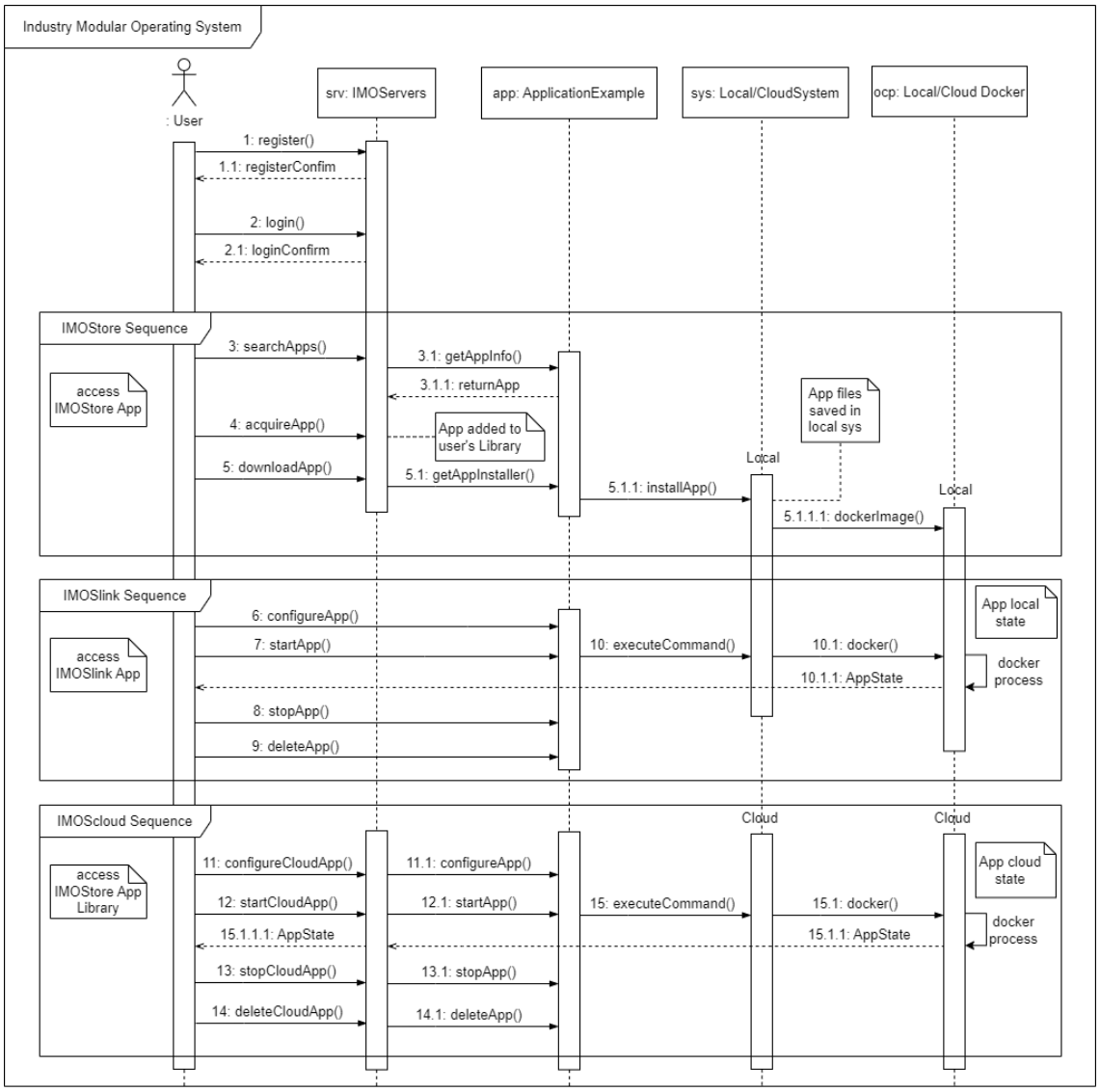


Figure 5.2: IMOS sequence diagram of user interaction with the in-built modules.

Every implementation process begins with defining the most suitable technology stack for achieving the desired goal — in this case, developing the proposed platform as a proof of concept for the envisioned IMOS ecosystem. The implementation was successful, as the selected technologies and approach did not require any reformulation. Table 5.1 outlines the chosen technology stack and their roles within the overall structure, referencing academic work that has applied and evaluated these technologies. They were selected for their capability to support the IMOS platform's modular architecture while ensuring seamless integration and scalability with existing systems.

Table 5.1: List of selected technology stack description.

Technology	Version	Specification
NodeJS [87]	20.11.1	NodeJS serves as the runtime environment for the IMOS platform, offering a non-blocking, event-driven architecture that supports scalable, high-performance backend services. Its efficiency in handling multiple connections makes it ideal for real-time data processing and user interactions [88].
Electron [89]	28.1.0	Electron is used to build the cross-platform desktop application interface for IMOS (parent) and its in-built applications (child). By leveraging web technologies like HTML, CSS, and JavaScript, Electron ensures a consistent user experience across different operating systems while maintaining a single code-base [90].
Docker [91]	26.0.0	Docker was chosen for its widespread use and simplicity. It provides application packaging to ensure consistency across environments, simplifies application management, enables seamless scaling, and supports IMOS's modular architecture through the execution of various applications and services [92].
Express [93]	4.18.2	Express is the web application framework for IMOS's backend API. It simplifies the development of scalable web services, handling routing, middleware, and HTTP requests, and facilitating communication between the IMOS client applications and the MongoDB database [94].
MongoDB [95]	6.3.0	MongoDB, a NoSQL database, is used for its simplicity and scalability through the integration with a data warehouse solution. Its document-oriented structure efficiently manages the data models of IMOS, handling user information, marketplace assets, application states, and other data [96].
NSIS [97]	3.9.0	Nullsoft Scriptable Install System creates the installation package for the IMOS desktop application and the developed application use cases, simplifying the setup process. It supports advanced scripting for customized installations, ensuring a correct installation on various operating systems [98].

5.1 IMOStore Development and Server Setup

As previously stated, the major objective for implementing IMOS was to distribute containerized apps as modules. Following the framework of the preceding Chapter 4, the implementation phase began with the building of IMOStore, which eventually led to the formation of the IMOS ecosystem. This early focus on IMOStore established the framework for the larger platform, which will be discussed in the next sections as well.

Before detailing the development specifics and final results of IMOStore, the author provides a Component diagram, illustrated in Figure 5.3, to describe the structure and functionality of this module, and for the reader to better follow some of the implementation details on the project's repository [99].

It is important to notice the hierarchical relationship between the components, where the primary component, initialized in the `main.js` file, spawns several child components. The main interface component offers three distinct views: the home page, the all applications page, and the library page. Additionally, three child components correspond to specific functionalities: the application-specific page, the software publishing form, and the submission history page. Overall, this structure is straightforward and adheres to a clear organizational logic, with each component systematically extending from the central initialization.

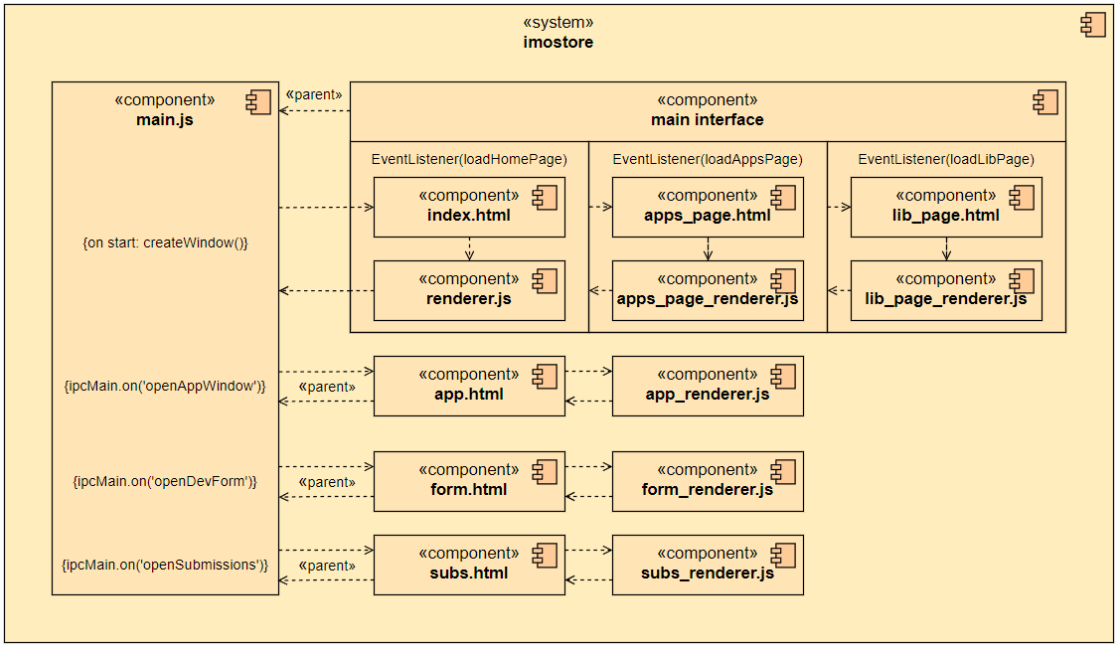


Figure 5.3: IMOStore component diagram describing project's files and components.

The IMOStore platform's development began with the establishment of its fundamental structure and the implementation of the minimal essential functions. Initially, a simple interface was designed to highlight various fabricated application samples. In addition, the database structure and server endpoints were designed to allow users to interact with

the platform. Figure 5.4 shows the activity on MongoDB Atlas, where the IMOS database was deployed, providing a centralized location to store data. This arrangement provides connectivity and handling of HTTP requests through a local Express server (available on the project's files), resulting in effective data administration.

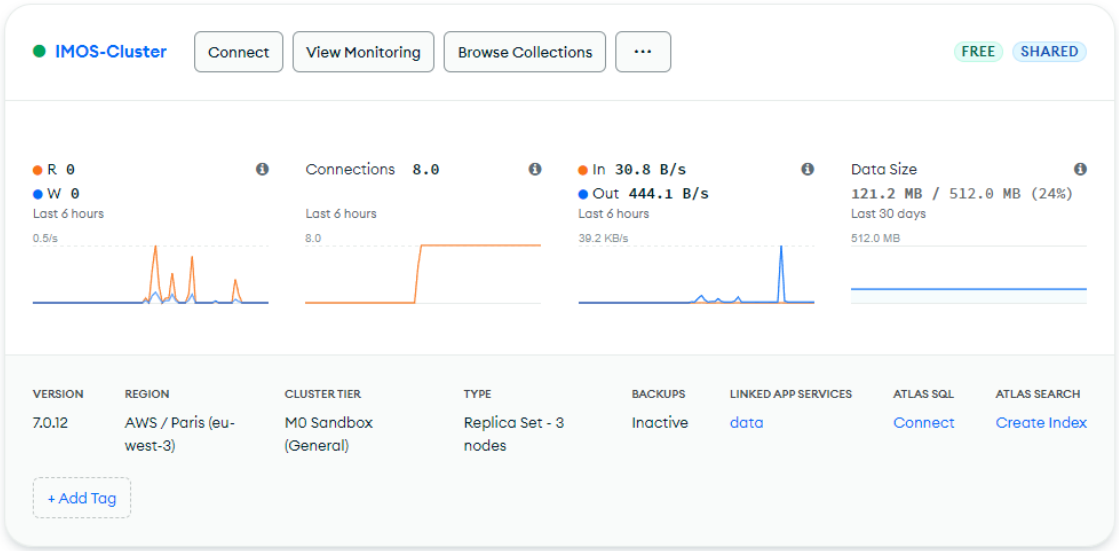


Figure 5.4: IMOS cluster running on MongoDB Atlas.

After creating the appropriate endpoints on the Express API server and generating data models for user registration, login, application acquisition, downloads, uploads, and submission requests, the attention switched to developing the marketplace's front-end. This component is critical for improving the user experience and making it easier to browse and acquire newly published items.

The platform's basis was built using NodeJS and Electron, which ensured compatibility and scalability. Figure 5.5 shows how the responsive and intuitive front-end, built with CSS and HTML, allows users to easily explore and interact with available programs, inspired in modern mainstream marketplaces' design.

From the IMOStore home page, users can access news, highlights, and featured applications. The left-side panel includes links to different pages such as the home page, all applications page, and the user's library. For users registered as developers, additional buttons appear for software submission and a console to track submission history and review status. A documentation link at the bottom left directs users to the IMOS GitHub community repository and documentation. Each application card is interactive, displaying specific details like distributor, version, requirements, rating, and other relevant information.

As previously stated, following registration, a user can create an account as a client or a developer, depending on their function within the ecosystem. This differentiation is significant since various users have unique interests and needs. As a result, the marketplace adapts to their individual requirements, creating a more personalized user experience.

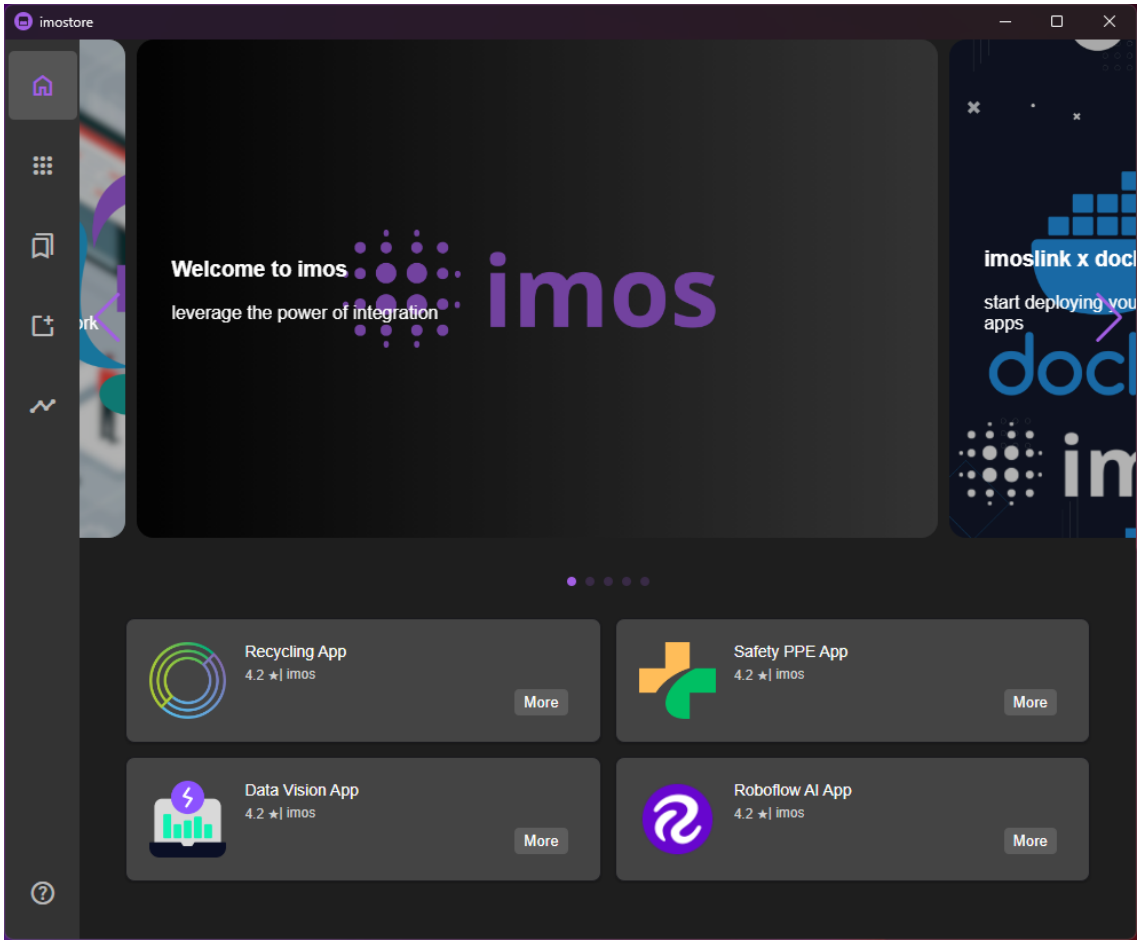
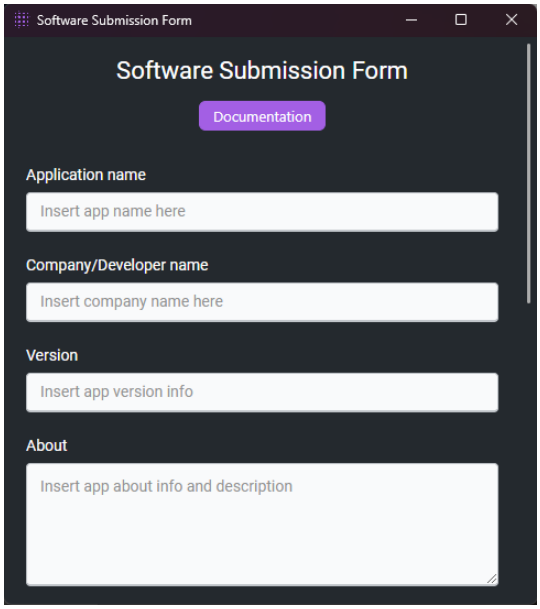


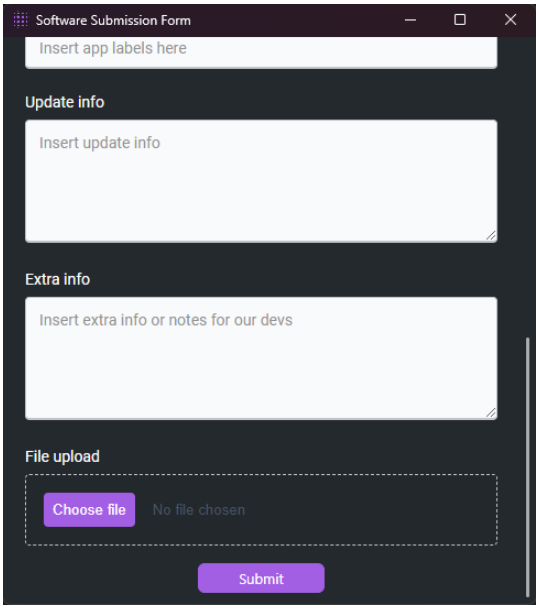
Figure 5.5: IMOStore marketplace home page, with available apps.

One example of this adaptation is the software submission capability, which is exclusively available to developers. For a platform like IMOS, it is crucial to ensure the quality and security of the applications offered on the marketplace, making sure that all developed solutions meet the platform's integration standards. When developers submit their solutions, they are required to provide comprehensive details about the application, as illustrated in Figures 5.6a and 5.6b. Once a submission is made, an IMOS reviewer carefully examines each aspect and tests the application within the IMOS environment to verify its compatibility. The submission history and review status are presented as shown in Figure 5.6c.

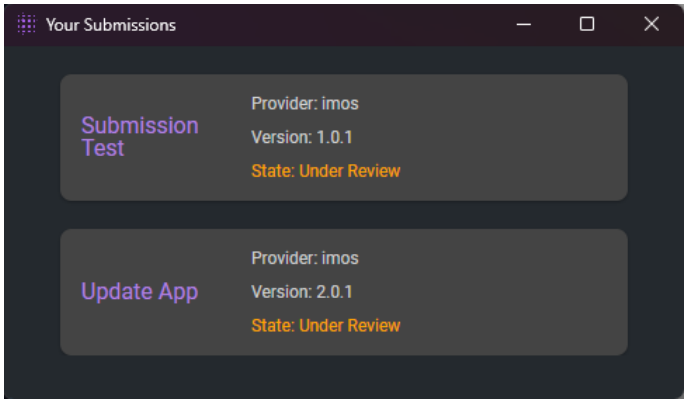
After the review process is complete, developers receive feedback outlining any necessary changes or adjustments. Depending on whether the application meets all the specified requirements, it will either be approved or rejected. If the application is accepted, it is then distributed on the marketplace by an IMOS administrator using the platform's back-office system. To efficiently manage and store large files, the server leverages MongoDB's GridFSBucket feature, which allows for the smooth handling of containerized applications. This ensures that users can easily access and deploy software directly from the platform without any complications.



(a) Software submission form 1.



(b) Software submission form 2.



(c) Submissions history and state of review.

Figure 5.6: IMOStore developer submission process.

Overall, this stage of development progressed smoothly, with the primary objective being the creation of the IMOStore marketplace and the establishment of the database and API structure for the entire platform. The main challenges faced included implementing GridFSBucket for efficient file upload and download, as well as designing a user-friendly graphical interface for the marketplace to create an intuitive software distribution ecosystem. Despite these challenges, the development phase successfully laid the foundation for the platform's core functionalities.

5.2 IMOSlink and App Use Cases

Following the development of the IMOStore marketplace, a natural need emerged to provide an integration-oriented platform, compatible with the containerized solutions available for acquisition. This posed an integration challenge: developing the Industry Modular Operating System as the foundational structure supporting all other applications, including IMOStore, and implementing IMOSlink as an in-built package manager and orchestration tool.

This section delves into the implementation of IMOS and IMOSlink, offering insights into the packaging and integration specifics of the application use cases. Further discussion of these use cases, along with their results and specifications, is provided in Chapter 6.

Once again, before presenting the implementation approach and results of IMOS and IMOSlink, the author provides Figures 5.7 and 5.8 as an overview of this section, providing UML Component diagrams of IMOSlink and IMOS showcasing files' hierarchical relations and project organization.

Figure 5.7 presents a component diagram of IMOSlink from an implementation perspective, aiming to facilitate the reader's interest in exploring the project's repository [99] and better understanding the implementation details and code structure provided ahead.

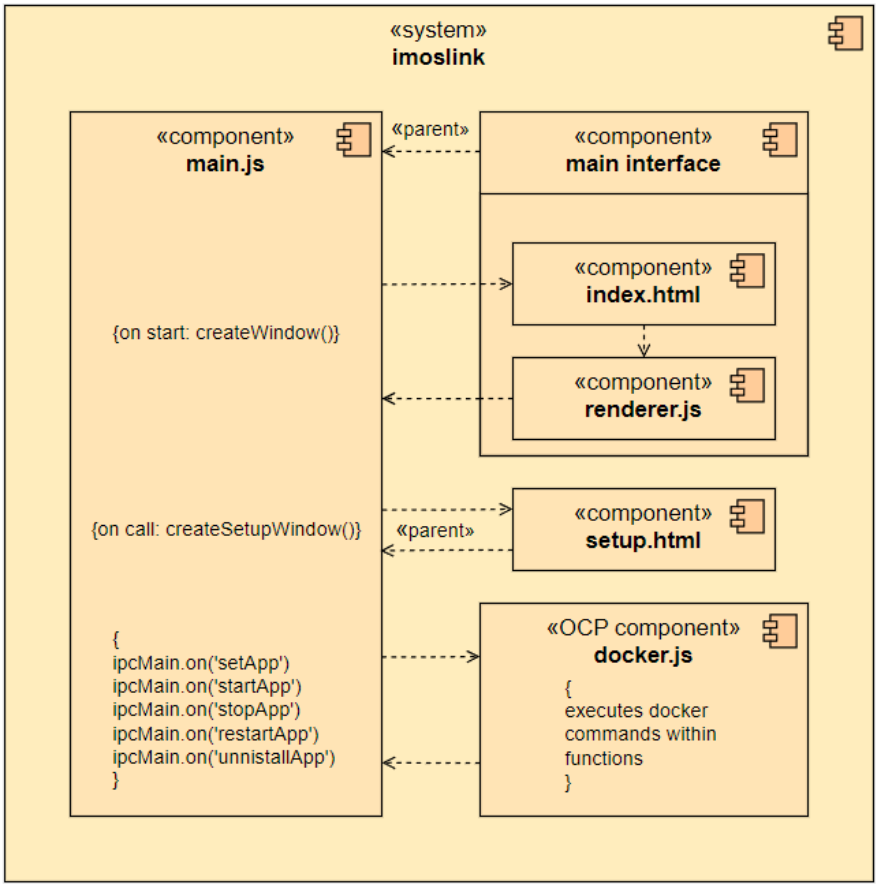


Figure 5.7: IMOSlink component diagram describing project's files and components.

IMOSlink features a primary interface that serves as the logical backbone for managing and orchestrating user applications. It includes a child application setup form that is launched for configuring or reconfiguring each module. Additionally, it's important to note that the connection between the `main.js` file and Docker is indirect; this interaction is managed by the `docker.js` file, which handles Docker commands via IPC requests.

Figure 5.8 presents the IMOS project's component diagram, providing a comprehensive overview of the entire system by integrating all core components within the ecosystem. This diagram represents the highest level of abstraction, where IMOS acts as the parent to both the IMOStore and IMOSlink components, as well as to all installed applications (as depicted in the application repository component).

The main interface of IMOS displays either the authentication console - where users are prompted to register or login -, or the desktop environment - from which users can launch any child component and its subsequent elements. Upon successful registration and login, users gain access to the IMOS desktop environment, enabling them to interact with any module or installed application. IMOS itself establishes a direct IPC connection to the `docker.js` component, which functions as a middleware for accessing the application repository.

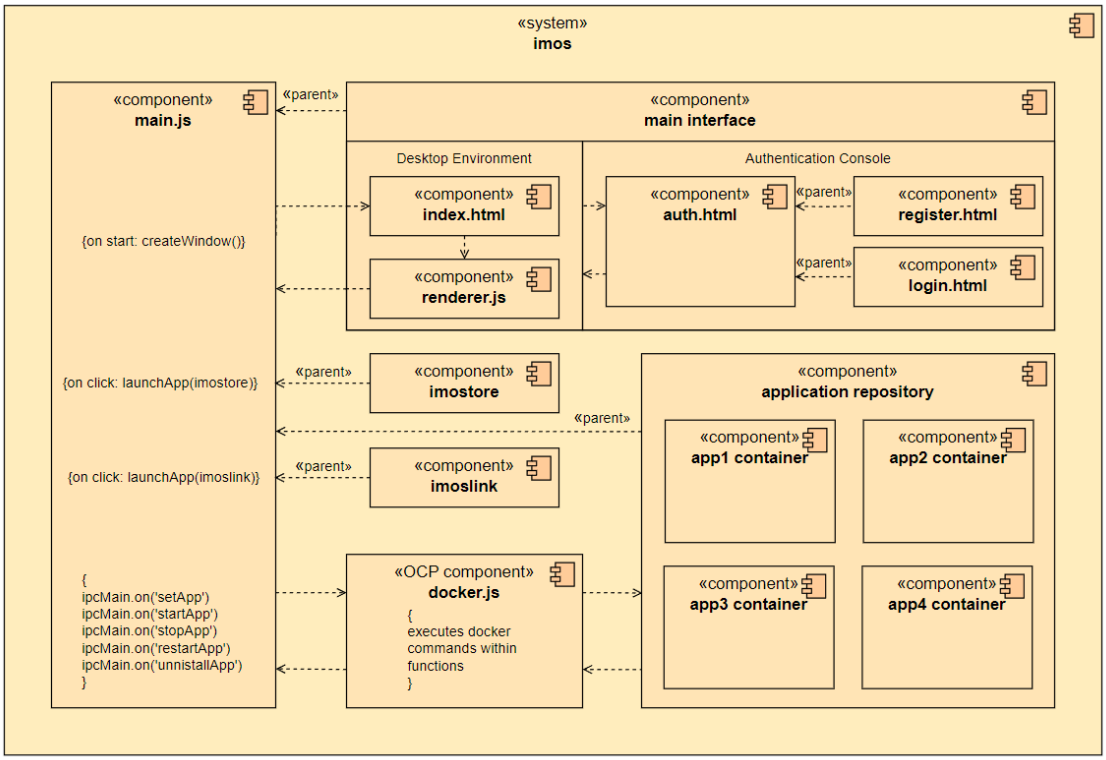


Figure 5.8: IMOS component diagram describing project's files and components.

The principles of I4.0 and the shift toward I5.0 inspired the creation of a scalable, modular platform to support IMOStore. IMOS was designed to enable the development of additional modules and the integration of various applications. Figure 5.9 shows the final IMOS desktop environment, developed as a NodeJS Electron parent application, featuring in-built applications (IMOSlink, IMOStore, IMOShub, Settings) and use-case applications (RoboflowApp, RecyclingApp, SafetyPPEApp, DataVisionApp) as child components.

The presented interface is designed to integrate all existing modules in a unified space to enhance user interaction. This integration is achieved through the communication established by IMOSlink with Docker, which identifies locally installed and IMOS-compatible applications, as will be further detailed.

IMOSlink, a NodeJS Electron child of IMOS, serves as a bridge between IMOS and Docker. This module enables containerized application administration and orchestration, integrating smoothly with Docker for application deployment and execution. Figure 5.10 depicts the IMOSlink interface, which provides a centralized platform for users to manage other tasks. The `docker.js` middleware file is required for this feature, as it contains critical methods for container execution and dynamic configuration. It includes a full collection of functions for collecting Docker metadata and running Docker commands locally, ensuring effective container management in the IMOS environment.

Figure 5.9: IMOS desktop environment, with various locally installed apps.

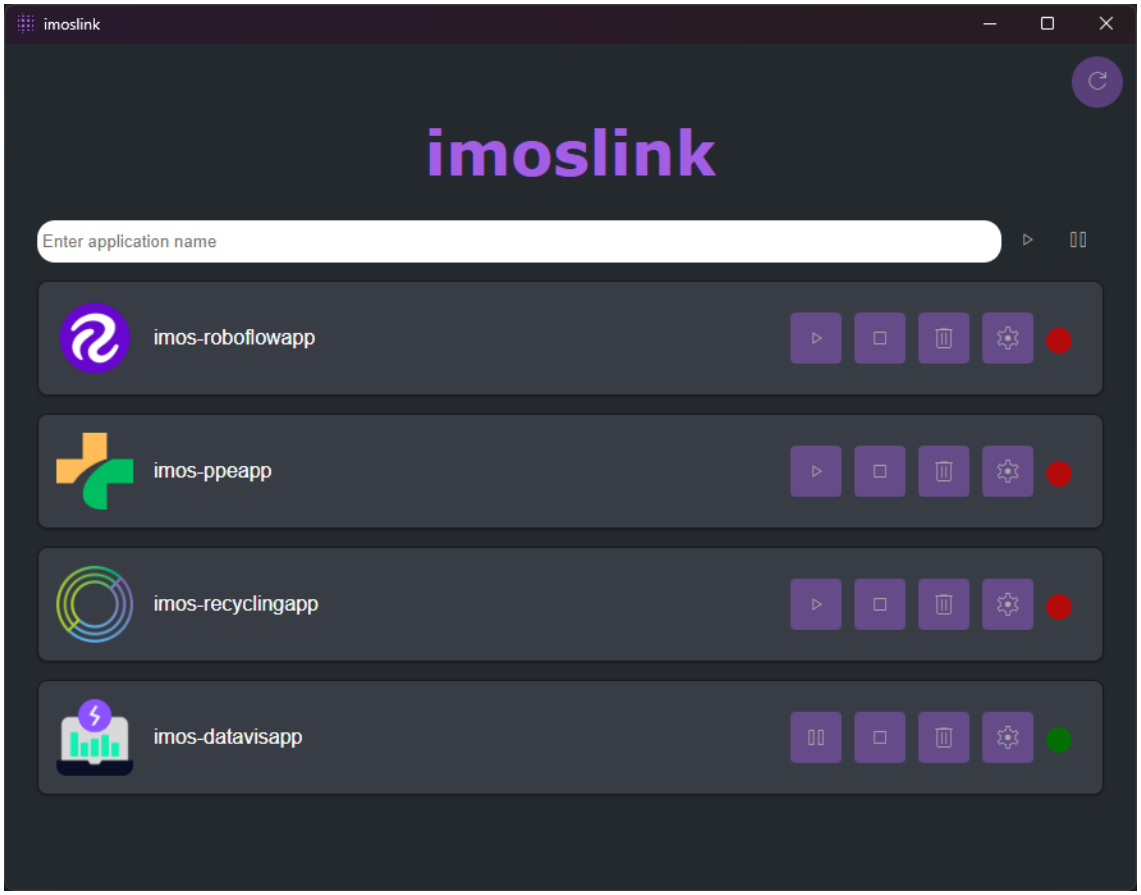


Figure 5.10: IMOSlink environment, with local applications ready for execution.

Table 5.2 provides an overview and explanation of the most relevant processes implemented for the integration of containerized solutions with Docker. These processes include functions for retrieving information from Docker, such as inferring about existing images, checking the execution status of containers, and obtaining labels and metadata from configurable new applications. Additionally, action functions are responsible for creating and configuring new Docker containers, starting and stopping working applications, and resetting or deleting specific modules. For more detailed information, the `docker.js` file is available on the project's GitHub repository [99].

Table 5.2: Insights on the most relevant integration functions from the docker.js file.

Function	Explanation
`getInstalledApps()`	This asynchronous function retrieves a list of installed Docker applications using the `dockerode` library. It filters the Docker images to identify those related to IMOS applications, distinguishing between single images and multi-container projects based on specific labels. The result is a dictionary of these applications categorized by type.
`doesContainerExist()`	This function checks if a Docker container with a given name exists. It uses `spawnSync` to execute the `docker ps -format '.Names'` command, which lists all container names. Similarly, `doesMultiContainerExist()` determines if a multi-container environment exists. It runs the `docker ps -format '.Labels'` command using `spawnSync`, parsing the output to check for the presence of a label indicating a multi-container project with the specified name.
`getImageMetadata()`	This function fetches metadata for a Docker image by executing a Docker command to inspect the image's labels. It processes labels indicating available and required configurations, constructing a metadata object that maps configurations to their required status. `getMultiImageMetadata()` gets all Docker images associated with the multi-container project and processes their labels to build a metadata object, similar to the single image metadata function.
`createDockerProcess()`	This function creates and starts a Docker container based on provided configuration data. It constructs and executes a `docker run` command with specified environment variables and port mappings. If the interface parameter is set to 1, it opens a web browser to the container's public ports. Similarly the `createMultiDockerProcess()` creates and starts a multi-container environment using `docker compose`.
`startDockerProcess()`	This function starts an existing Docker container or multi-container environment. It checks if the application is already running and starts it if not, either executing `docker run` for a single container, or `docker compose up` for multi-containers. If the interface parameter is set to 1, it opens a web browser to the public ports.
`stopDockerProcess()`	This function stops a running Docker container or multi-container environment. It executes appropriate Docker commands to stop each app-related container and waits for confirmation to ensure the application has indeed been paused.
`deleteDockerProcess()`	This function deletes a Docker container or multi-container environment. For single containers, it removes the container directly. For multi-container projects, it retrieves all associated images and deletes each one.

A critical function, detailed in Algorithm 1, is the `getInstalledApps()` function. This function is responsible for identifying and returning every Docker image or container that matches IMOS's integration requirements. It is executed during the IMOS boot-up process to determine which applications are locally available for use and to display their respective icons in the desktop environment. This ensures a seamless user experience by providing immediate access to all compatible applications and also enables user's to create their own IMOS-compatible applications.

Pseudocode 1 Get Installed Apps

Output: Dictionary mapping application names to their types (`dict`)

```
1:  function getInstalledApps
2:      Initialize a Docker object
3:      try
4:          List all Docker images using Docker object
5:          Initialize empty list for imosImages
6:          Initialize empty list for imosMultiImages
7:      for each image in Docker images do
8:          if image has RepoTags and any tag contains 'imos' then
9:              Add image name and type 'image' to imosImages list
10:         end if
11:     end for
12:     for each image in Docker images do
13:         Get image labels
14:         Get projectName from labels with key 'com.main.multicontainer'
15:         if projectName starts with 'imos' and not already in imosMultiImages then
16:             Add projectName and type 'multicontainer' to imosMultiImages list
17:         end if
18:     end for
19:     Combine imosImages and imosMultiImages into installedApps list
20:     Initialize empty dictionary appDictionary
21:     for each app in installedApps do
22:         Add app name as key and app type as value to appDictionary
23:     end for
24:     return appDictionary
25: end function
```

In order to scale the IMOS platform to be integration-ready with different types and sizes of applications, several application cases were developed to demonstrate the versatility and compatibility of the IMOS platform with a wide range of technologies, as we will see just ahead on Chapter 6. Proving that anything that can be containerized with Docker can run on IMOS.

The installation `.exe` files and packaging process for the apps were carried out using NSIS. The implementation of the build `.ns` files and Dockerfiles for each application require a careful understanding to ensure compliance with the integration needs of the platform:

1. **Creating Dockerfiles**. Each application should have its specific Dockerfile(s) depending if the application is a single container or a multicontainer app. It is recommended that the app's source files are saved inside a folder with the name `imos-<appname>` and Dockerfiles should be named `Dockerfile.<imagename>`;

2. **Setting `Dockerfile` labels**. For every Dockerfile the developer should set the image's required labels. This includes for example:

   a) `com.available.configs=WEBCAM_IP, MODEL_DETECTOR_PORT` where the user sets the available configuration settings for the application launcher;

   b) `com.required.configs=WEBCAM_IP` where the user sets from the available configurations which are required for the user to fill, the ones not required are auto filled with the `docker-compose` file default values;

   c) `com.user.display=False` where the user selects if the present image associated port is intended for display, or if it has a user interface;

   d) `com.main.multicontainer=imos-<appname>` is used in case the application is a multi-container, where the user sets the application name.

3. **Updating docker-compose**. The app's `docker-compose` file should be updated to comply with previously set labels and their default values if applicable. Here is an example: `MODEL_DETECTOR_PORT:$MODEL_DETECTOR_PORT:-5001:5001`. It is recommended to name multi-container images as `<appname>-<imagename>`;

4. **Saving app files to local directory**. Application source files are saved to IMOS' app repository directory `C:\imos\apps\AppName`. from the NSIS build file on Figure 5.11;

5. **Launching license and installation**. It is required for every application package to ask for agreement on certain terms before executing the `build_images.bat` file.

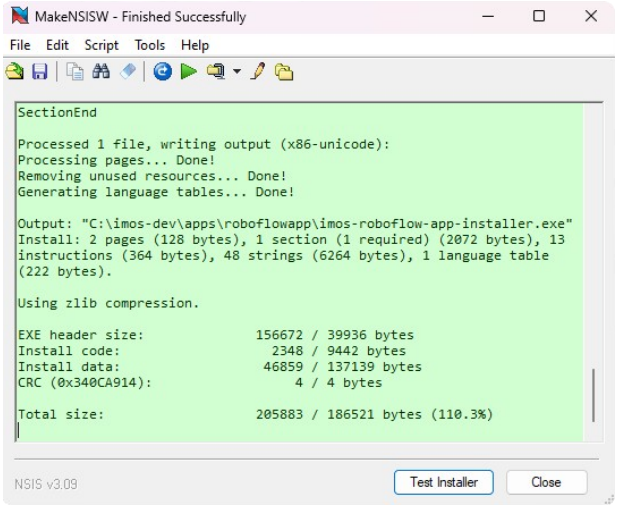


Figure 5.11: NSIS packaging and generating executable file for app use case.

Upon downloading a newly acquired application, an NSIS executable file becomes locally available for execution. Running this installer saves all of the application's source files to the correct IMOS directory and executes the necessary Docker commands to set up the required images for the application. If everything is correctly configured and the application's labels are properly assigned, the module will be installed and ready for use.

Post-installation, the `getInstalledApps()` function recognizes the new app as an IMOS application. If it is the first time the application is being used, opening it will trigger a configuration panel generated by the `getImageMetadata()` function, as shown in Algorithm 2. Here, users must fill in the required labels to set up the container or multi-container according to their preferences. Once the app settings are submitted, the application is created using `createDockerProcess()` or `createMultiDockerProcess()`.

Pseudocode 2 Get Image Metadata

Input: `imageName` - The name of the Docker image (`str`)
Output: Metadata dictionary with images' labels and requirements

```
1:  function getImageMetadata(imageName)
2:      Initialize empty metadata dictionary
3:      Run Docker command to inspect image labels
4:      if command succeeds then
5:          Get available_configs label from parsed labels
6:          Get required_configs label from parsed labels
7:          if available_configs label exists then
8:              Split available_configs into a list and trim spaces
9:              if required_configs label exists then
10:                 Split required_configs into a list and trim spaces
11:             end if
12:             for each config in available_configs do
13:                 if config is in required_configs then
14:                     Set metadata[config] to true
15:                 else
16:                     Set metadata[config] to false
17:                 end if
18:             end for
19:         end if
20:         return metadata
21:     else
22:         Log error message
23:         return null
24:     end if
25: end function
```

Algorithm 3 illustrates the `startDockerProcess()` function, which initiates a container or multi-container application that has been previously set up. This function takes the app's name and type as inputs and can optionally open the graphical user interface if provided. After configuring an application, the user can start it and access its interface.

Pseudocode 3 Start Docker Process

Input: `appName` - The name of the Docker container or multi-container (`str`)
Input: `type` - The type of Docker process 'image' or 'multicontainer' (`str`)
Input: `interface` - Open interface, default value is 1 (`int`)
Output: Logging of docker start command

```
1: function startDockerProcess(appName, type, interface = 1)
2:     if type is 'image' then
3:         if container is not running then
4:             Start the container using Docker command:
5:             docker run -d -name appName imageName
6:             if there is an error starting the container then
7:                 Log error message
8:             else
9:                 Log success message
10:            end if
11:        end if
12:        if interface is 1 then
13:            Get the port of the container
14:            if port retrieval is successful then
15:                Log the port
16:                Open browser at http://localhost:port
17:            else
18:                Log error
19:            end if
20:        end if
21:    else if type is 'multicontainer' then
22:        if multi-container environment is not running then
23:            Start Docker Compose for the multi-container environment:
24:            Change directory to C:\imos\apps\AppName
25:            docker-compose -f docker-compose.yml up -d
26:            if there is an error starting Docker Compose then
27:                Log error message
28:            end if
29:        end if
30:        if interface is 1 then
31:            Get the ports of the multi-container environment
32:            if port retrieval is successful then
33:                Log the ports
34:                for each port do
35:                    Open browser at http://localhost:port
36:                end for
37:            else
38:                Log error
39:            end if
40:        end if
41:    end if
42: end function
```

This stage of development was undoubtedly the most challenging, even without considering the work required for the application use cases. Overall, the IMOS environment was successfully developed as a NodeJS Electron application, encompassing the IMOStore and IMOSlink modules, integrating Docker as IMOS OCP, and refining the ecosystem through the integration and development of use-case applications presented in greater detail on Chapter 6.

5.3 Application Execution on IMOScloud

This section presents the final implementation stage related to the development of IMOScloud, a cloud computing option for running IMOS' apps in the cloud. This solution offers an alternative for executing containerized applications, especially for SMEs that may lack the hardware infrastructure to support numerous modules or handle complex computations.

The goal was to leverage a supposedly robust backend infrastructure of the IMOS servers to support on-demand application execution, providing a scalable and efficient solution for users with varying computational needs. As such, Figure 5.12 showcases an UML component diagram, providing a comprehensive introduction to this section.

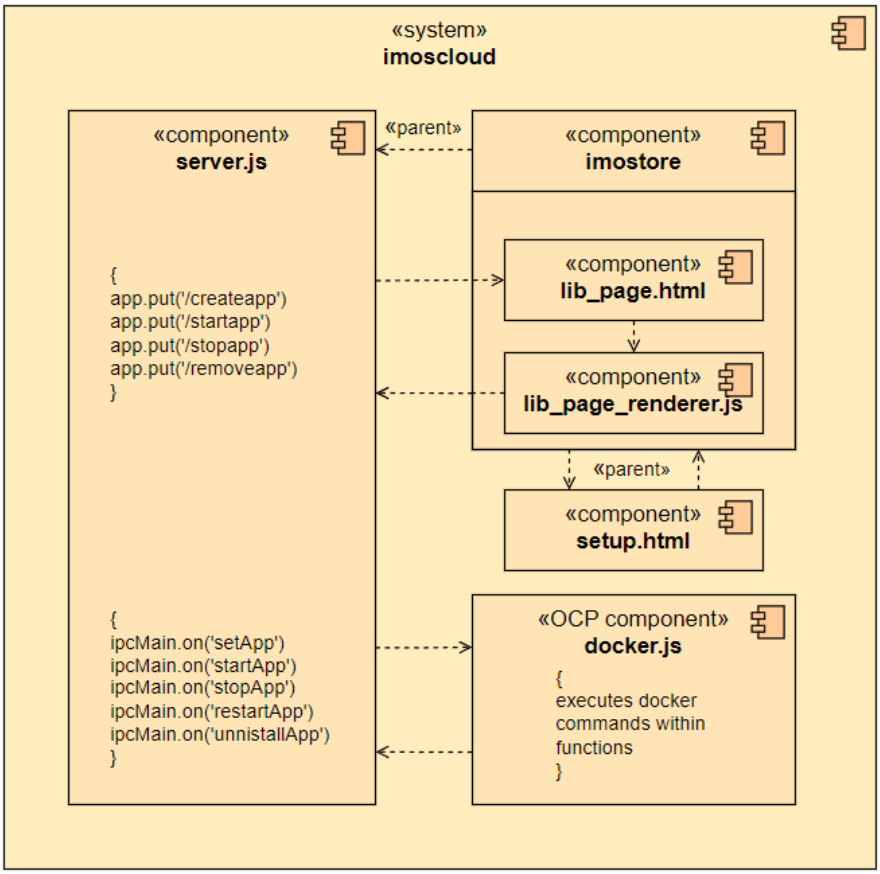


Figure 5.12: IMOScloud component diagram describing project's files and components.

This component diagram is very close to the one introducing IMOSlink. IMOScloud is not an Electron application, and for that reason it has no interface, instead it utilizes IMOStore's library page to display its functionalities for user-app management, as represented on the diagram. IMOScloud's main parent component is set in the server API itself, described in the `server.js` file and communicating with the Docker Cloud OCP through the `serverDocker.js` middleware file. Similarly to IMOSlink, IMOScloud application management actions are communicated through IPC and executed on the Cloud OCP, which has access to all of IMOStore's application repository.

Figure 5.13 illustrates the interface and functionalities of IMOScloud. This solution is, in fact, an iteration of IMOSlink available in the IMOStore library and deployed on the IMOS servers. It allows users to access their acquired applications and perform actions such as downloading, configuring, starting, stopping, and deleting apps in the cloud. This interface aims to facilitate the management of cloud-based applications.

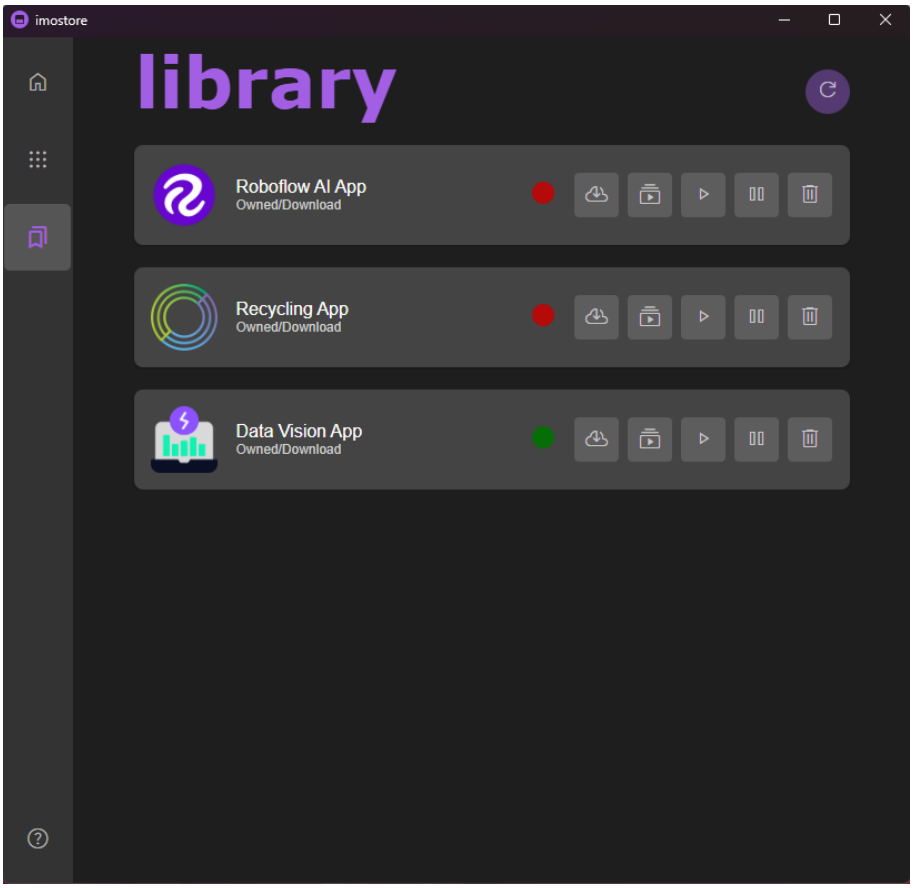


Figure 5.13: IMOScloud app execution and management on IMOStore library.

IMOScloud and the IMOS servers were deployed on an Ubuntu Server system to maximize resources and focus computational power on handling requests and executing users' apps. This deployment, highly beneficial for the project's proof of concept, expanded IMOS's compatibility with a Linux-based environment, as depicted in Figure 5.14. Leveraging Ubuntu Server enabled more efficient workload management, and demonstrated

the platform's potential applicability across diverse operating systems.

An important consideration is that, for safety concerns and cloud applications trust, only published apps on IMOStore are available to run on IMOScloud. When a new application is published, its files and images are also instantiated on the Ubuntu Server. Consequently, when a user decides to execute a specific application, the required containers are created and started in the `docker-compose` file with a name such as `container_name: $USER-appname-imagename`, where `USER` represents the username, required to differentiate which containers belong to which users.

```
[2024-07-18T11:13:17.775Z] Received creatapp PUT request
[Object: null prototype] {
  user: 'alts',
  app: 'Data Vision App',
  configs: '{"appName":"imos-datavisapp","type":"multicontainer","GR
AFANA_PORT":"","KSQLDBSERVER_PORT":"","SCHEMAREG_PORT":"","CONTROLCE
NTER_PORT":"","KAFKACNCT_PORT":"","BROKER_PORT1":"","BROKER_PORT2":"
","ZOOKEEPER_PORT":"","RESTPROXY_PORT":"","CONNECT_PORT":""}'
}
Multicontainer created and started successfully.
[2024-07-18T11:13:47.843Z] GET cloudapps of user request
{}
[2024-07-18T11:14:05.850Z] Received stopapp PUT request
[Object: null prototype] { user: 'alts', app: 'Data Vision App' }
Multicontainer stoped successfully.
[2024-07-18T11:14:39.960Z] Received startapp PUT request
[Object: null prototype] { user: 'alts', app: 'Data Vision App' }
Multicontainer started successfully.
[2024-07-18T11:14:59.810Z] GET cloudapps of user request
{}
[2024-07-18T11:15:00.574Z] Received stopapp PUT request
[Object: null prototype] { user: 'alts', app: 'Data Vision App' }
Multicontainer stoped successfully.
[2024-07-18T11:15:28.846Z] GET cloudapps of user request
{}
[2024-07-18T11:15:33.772Z] Received removeapp DELETE request
[Object: null prototype] { user: 'alts', app: 'Data Vision App' }
Multicontainer alts-imos-datavisapp deleted with all sub-containers.
```

Figure 5.14: IMOScloud requests logging from the sever side.

This final implementation stage successfully adapted the IMOSlink code and structure for the Ubuntu Server, coordinating server request handling with the execution of docker commands and application runtimes. While IMOScloud currently serves as a proof of concept on a centralized Ubuntu system where any user can deploy applications, future iterations should allocate separate server partitions for each user to ensure greater scalability and security.

5.4 Implementation Conclusions and Limitations

The technology stack selected was primarily oriented towards developing a proof-of-concept for the IMOS platform, focusing on easier execution, more common frameworks, and, consequently, better results with the limited time period set for implementation. While alternative technologies might be preferred for a final IMOS version, the current version effectively demonstrates the platform's capabilities. It supports app distribution, integration, and orchestration, both locally and in the cloud, while laying the groundwork for growing a community.

As expected from a project of this scale, more time was needed to fully realize the potential of IMOS. The process of building the platform revealed numerous areas for improvement. Although the current platform functions well as a proof of concept, the ideal IMOS would be a Linux-based operating system with an integrated OCP, rather than a NodeJS application built on Docker. Future considerations should also include alternative technologies for the OCP, as Docker's recent shift towards proprietary control has raised concerns within the developer community about its impact on open-source accessibility and innovation. A solid, and perhaps even more scalable option would be Helm Kubernetes [100], an open-source package manager that automates the deployment of software for Kubernetes.

Significant challenges were encountered in dynamically configuring each application and managing the parallel runtimes of different applications, which still impact platform performance. Additionally, while working on the NSIS build files for the app use cases, the author faced difficulties in packaging and generating executable and integration-ready installations for IMOS, necessitating some troubleshooting and optimization efforts.

Overall the showcased results in this chapter are highly positive, laying a promising foundation for future developments. The implementation goals were not only met but surpassed, with the final outcomes already providing a functional and optimized prototype [101] that adds significant value to the topic at hand.

6

Application Cases

In this chapter, the author presents various application examples created and tested to illustrate the IMOS platform's capabilities. These solutions employ a range of technologies for both back-end and front-end development, each with distinct purposes and executing authors. This diversity highlights the versatility of the IMOS platform, showcasing its compatibility with different application sizes, technologies, and functionalities.

The following sections present four applications developed, integrated, and published on IMOS. These include two specially built applications featuring interfaces and AI capabilities: the Recycling App and the Safety PPE App. Additionally, two externally sourced applications are showcased: the Data Vision App, a previous work by UNINOVA-CTS researchers [102], tested in a real-world industrial scenario, and the Roboflow App, based on the widely-used Roboflow open-source API. These examples demonstrate the platform's capabilities in software distribution and integration.

It is important to acknowledge that the development of the following applications and the results presented were not the primary objectives of this project. Some features may not be fully optimized, but they effectively demonstrate the integration and usability of these applications within the IMOS platform as proof of concept.

6.1 Recycling App

The Recycling App was one of the initial applications developed as a use case example for IMOS. This app employs a pre-trained AI model to detect various classes of residues, including paper, plastic, glass, cardboard, metal, organic, and critical residues.

To integrate this application, it was first necessary to identify the user-configurable settings required for proper application setup. Following the established documentation on containerization, labeling, and NSIS packaging of the application source files, the executable installation package was made available on the app's page on IMOStore. Post-installation, the application is available for configuration and execution. Figure 6.1 depicts the Recycling App's configuration setup screen, where the user is prompted to give the necessary configurations indicated by the developer, including a camera's IP stream.

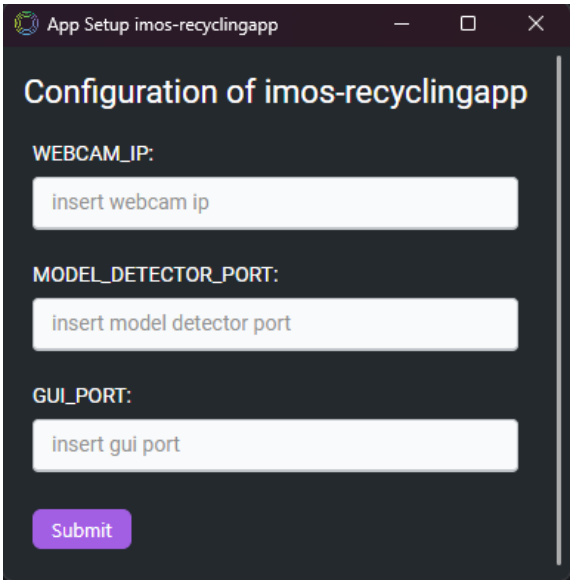
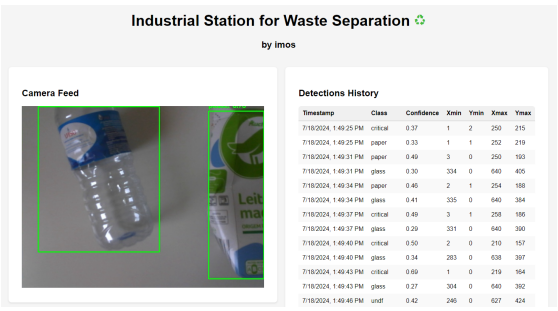


Figure 6.1: Configuration of the Recycling App on the setup console.

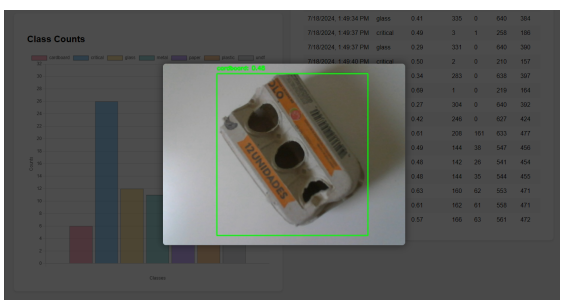
For the "appification" of the presented solution, the author developed the application consisting of two Docker containers: an inference server - `recyclingapp-model-detector`; and a graphical user interface client container - `recyclingapp-gui`.

The `recyclingapp-model-detector` consists of a Flask server, with several endpoints, managing HTPP requests, and servicing them with the PyTorch Hub Inference API [103], and OpenCV tools for drawing bounding boxes on images. The AI model was trained with the YOLOv5 model by Ultralytics [104], using a dataset consisting of more than 20,000 images, built from web scraping, real-data acquisition, and pre-processing, and each image labeled using the Roboflow labeling tool [105]. The presented recycling model displays very good results, since it is part of the author's entrepreneurship project and has a lot of previously developed work.

The results from the model inference are displayed on a NodeJS-based client, running within the `recyclingapp-gui` container, as shown in Figure 6.2. his interface provides the user with a console displaying the live camera feed and real-time model detections, along with an inference history table and graph.



(a) Live camera stream and model results.



(b) Display saved image results on-click.

Figure 6.2: Recycling App interface and demonstrative results.

A script named `webcam_to_ip.py` was created to simulate a functioning camera in an industrial site. The application connects to an active and available camera's IP address, does model detection, and returns findings to several endpoints. These results are used in the application's interface visualization and can be included into smart recycling processes via the given service-oriented endpoint, presented in Figure 6.3 and tested on Postman.

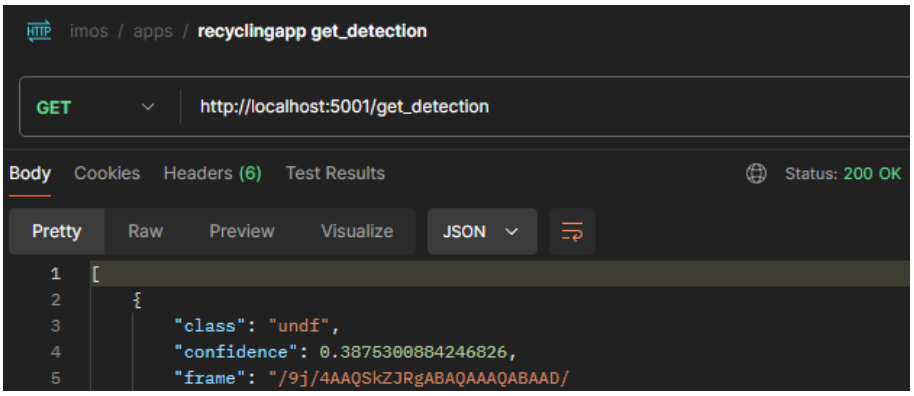
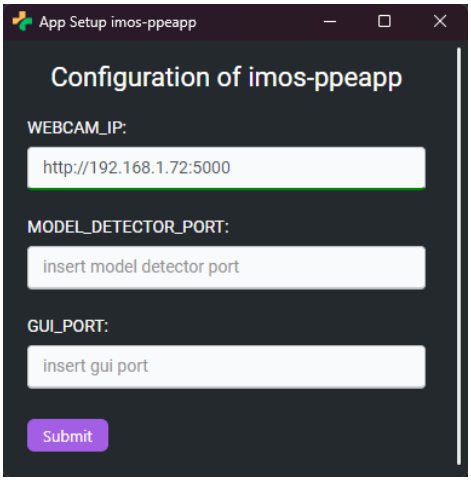


Figure 6.3: Get model detection service results, on Postman.

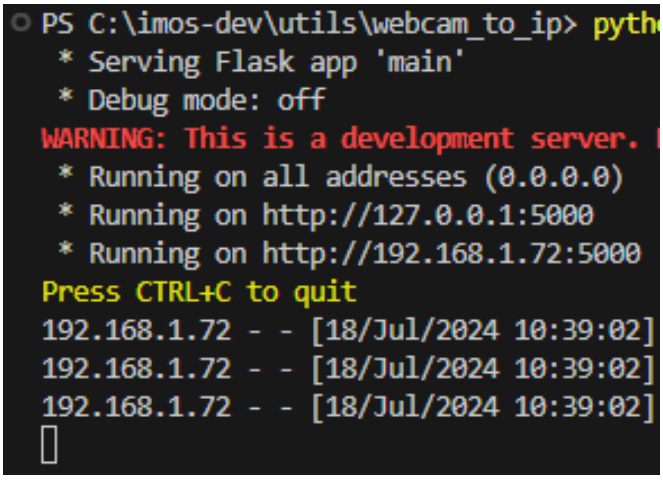
6.2 Safety PPE App

This application extends the previously developed interface and model inference framework to deliver a PPE detection solution. The AI model, trained with a smaller, web-scraped dataset and pre-processed data, is capable of identifying six classes: gloves, person, glasses, boots, vest, and helmet. The app is designed to integrate through its endpoints to enhance workplace safety by preventing accidents and ensuring compliance with safety protocols.

As previously outlined, the Safety PPE App necessitates a configuration process similar to that of the Recycling App, and its setup console is presented in Figure 6.4.



(a) PPE App setup console.

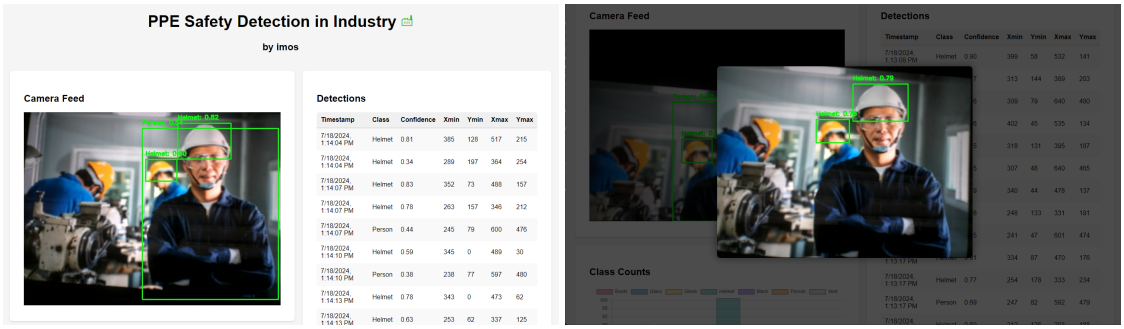


(b) Logs of the `webcam_to_ip.py` script execution.

Figure 6.4: Setup and configuration of the Recycling App.

Users must specify the IP address of the webcam to connect the app and designate the ports for running its two containers. It is essential to emphasize the importance of accurately configuring these settings to prevent conflicts, such as port or IP address collisions, which can disrupt application functionality. For testing, the author developed a script that streams a camera image to a random IP, simulating a webcam connection.

Similarly, the Safety PPE App features both a user interface, which displays live image results and inference history (as shown in Figure 6.5), and a detection endpoint for integration with on-site automation systems. A practical application of this setup would be to link the detection service with an interpreter and alert systems. For example, if the app identifies the class '`person`' missing the required protective equipment classes, the system could trigger a warning to ensure compliance with safety protocols.



(a) Live camera stream and model results. (b) Display saved image results on-click.

Figure 6.5: Safety PPE App interface and demonstrative results.

6.3 Data Vision App

Another application example, and perhaps the most notable use case presented, is the Data Vision App, introduced by an external work [102], and integrated with IMOS, following the established guidelines for containerization, labeling, and packaging. The authors developed a cloud-based ML application designed to predict energy consumption, which was tested in an industrial setting.

The Data Vision App utilizes a comprehensive technology stack, including a Node-RED API that hosts all trained ML models, Apache Kafka for messaging, PostgreSQL for data management, and a Grafana interface for user-friendly data visualization. This solution was implemented in "*real-world scenarios with robotic cells that meet Volkswagen and Ford standards. The results are promising, as models can accurately predict the expected consumption from the cells and allow managers to infer problems or optimize schedule decisions based on the energy consumption*" [102].

The demonstration of the authors' solution took place in the robotic assembly cells at Introsys SA's facility in Quinta do Anjo, Portugal, as depicted in Figure 6.6 where on the left we have the Volkswagen station, and on the right the Ford station.

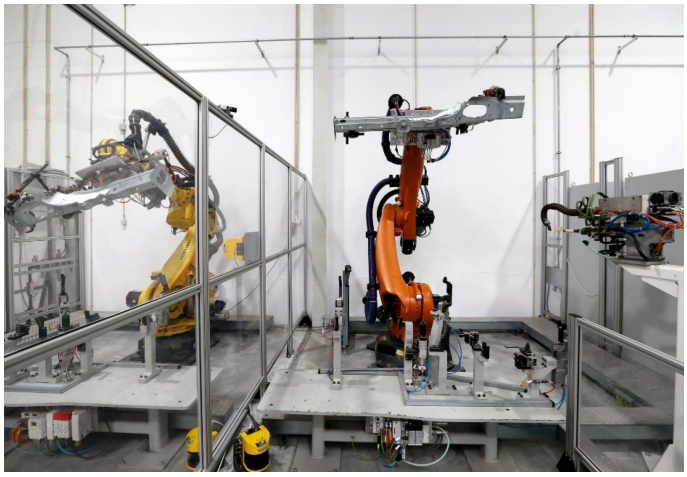


Figure 6.6: Introsys welding industrial robotic cells, by [102].

The workstations include numerous spot-welding procedures that meet the specifications of two major vehicle manufacturers. Each station has a 6-DoF robotic arm with a gripper, a fixture for the operator to handle the product, a stationary welding gun, and a variety of sensors to monitor the energy consumption of all these components. Figure 6.7 illustrates the comparison between real measured data and the predictions made by the application for energy consumption and duration across different recipes. The graph displays energy usage for cell and robot, highlighting the accuracy of the predictions.

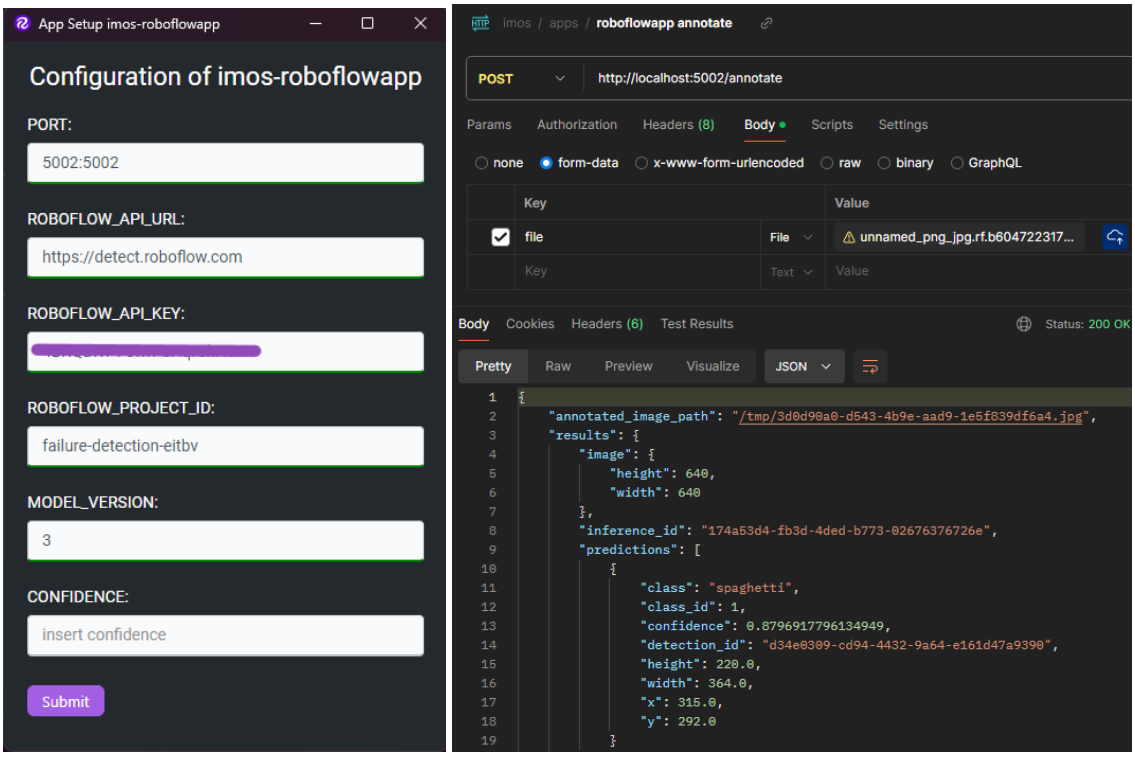
Figure 6.7: Real and predicted values of energy consumption, and duration, by [102].

6.4 Roboflow AI App

The Roboflow App leverages the Roboflow Inference API [106] to execute open-source AI models. By specifying the project ID and model version, users can effortlessly deploy pre-trained models. The application simplifies this process by allowing users to authenticate with their API key and select the model that best fits their needs.

To accomplish this, the Roboflow App was built using the recommended inference SDK engine. It processes pre-trained models according to user requests and returns results in JSON format through its Flask server. Additionally, the application stores annotated media on the user's host volume, typically located in the IMOS application directory (`C:\imos\apps\AppName\Volume`), as configured in IMOSlink. Its design supports running multiple instances with different models simultaneously, providing versatility for diverse use cases.

To utilize the Roboflow App, users must configure essential details at launch, including the Roboflow `API_URL`, `API_KEY`, `PROJECT_ID`, `MODEL_VERSION`, and desired `CONFIDENCE` level (defaulting to 0.5 if unspecified), as shown in Figure 6.8a. Once configured, the app operates based on these settings and provides an endpoint for annotation requests. This service supports automation of various processes and offers extensive flexibility depending on the selected model, including various options, from quality control inspection and fault detection, to safety monitoring and product sorting.



(a) Roboflow App setup console. (b) Display saved image results on-click.

Figure 6.8: Safety PPE App interface and demonstrative endpoint.

To demonstrate the application's functionality, a 3D-printing fault detection model was selected and configured for execution. Using Postman to request annotations for an image of a defective 3D-printed box, the app provided inference results as shown in Figure 6.8b. Additionally, the results were saved in the application's volume directory, depicted in Figure 6.9. This successfully validated the model's accuracy and confirmed the application's operational capability.

Figure 6.9: Roboflow App results of 3D-printing fault detection model.

Overall, the application use case development stage was particularly insightful, not only for the implementation of modular, industry-oriented applications, and the training of AI models, but especially for assessing IMOS's readiness to support these applications. While not everything integrated seamlessly at first, and certain challenges arose, the experience of working around diverse applications, from different developers, using various technologies and structures, greatly benefited IMOS. This process significantly enhanced the platform's adaptability and compatibility, preparing it for broader integration and more extensive use in various contexts.

7

Conclusions and Future Work

As this dissertation document comes to an end, the developed work has proven promising, building towards a platform to address both I4.0 challenges - including software integration, distribution, and modularity -, and I5.0 goals - focusing on sustainability, collaboration, and software symbiosis. The author hypothesized that designing a marketplace platform could support the creation of an ecosystem where stakeholders share resources and expertise, developing new solutions for SMEs. Continuous work towards the development of IMOS could set the groundwork for such an ecosystem, promoting collaboration and innovation.

Achieving an ideal collaborative environment and managing an industry-oriented marketplace is inherently challenging. This complexity likely explains why such a solution has yet to be widely implemented. The security risks, coupled with the extensive verification and quality assurance required for accepting users, developers, and their publications on the platform, are significant. Beyond the platform structure itself, there is a pressing need for a robust back-office to manage the solution.

Developing and ensuring solid standards for each application, module, asset, or action within the ecosystem would address these challenges. By automating compliance with platform standards, based on frameworks like RAMI 4.0 and AAS, the management process would be greatly simplified.

Overall, the project's goals were exceeded, yielding highly satisfactory results. The author conducted thorough research on the state of the art in software distribution and marketplace ecosystems within industrial contexts, analyzing initiatives such as Siemens' Insights Hub, vf-OS, and the ZDMP Platform. Additionally, the study explored best practices in modularization and containerization for software solutions and identified trending technologies shaping the future of manufacturing software applications.

Beyond the initial objectives established during the preparation phase - which focused on the ideation and implementation of IMOS, including the IMOStore and IMOSlink modules - the author developed two additional modules: IMOShub and IMOScloud. The latter was successfully implemented and integrated into the platform, with the results being showcased in both IMOS short demo [101] and extended [107] videos and the IMOS

GitHub repository [99].

Looking ahead from the current state and limitations of the IMOS platform, the roadmap includes initiating the development of IMOS as a Linux fork, before any further implementations. This shift is motivated by several factors that support industry suitability, since Linux offers a stable and secure environment. By transitioning to a Linux-based system, IMOS can leverage advantages regarding open-source development, greater flexibility, customization, and community support. This approach will also reduce dependency on high-code solutions like the proof of concept NodeJS and Electron-based application, optimizing overall system performance and scalability.

Following the proposed IMOS concept, it is evident that developing IMOShub with resource-sharing capabilities and collaborative workspaces is crucial for fostering the creation of CNs and building an industry-oriented community. While IMOS has made significant progress, continuous improvement and adaptation are essential to fully realize a secure, collaborative, and innovative industrial ecosystem.

Additionally, and as previously stated, utilizing Helm Kubernetes as the platform OCP, although challenging, represents a significant opportunity to scale the IMOS platform. It offers advanced capabilities for managing complex containerized applications. Transitioning to this technology could address the limitations posed by Docker's shift towards proprietary control, ensuring continued open-source accessibility and innovation.

Due to the complexity of the system and its collaborative nature, it would be interesting to have a standard description for each of the assets offered by IMOS and the assets made available by developers and companies. In this way, using a reference architecture such as RAMI 4.0, it is possible to use the AAS concept to create templates that developers and companies should follow to create their assets. This can facilitate the generation and distribution of assets and guarantees interoperability between them and the IMOS ecosystem.

Another key feature on the project's agenda addresses the emerging topic of Cloud Computing. The author presented a simplified solution of IMOScloud embedded in IMOStore, but it would be beneficial to conduct further research to scale the server infrastructure and identify the most suitable architectural solution for this goal. This could involve offering computational partitions as dedicated Virtual Machines for each user, or a Shared Cloud Spaces for users to run applications. Such structure would provide users with greater flexibility and reduce hardware dependency.

Ultimately, it is intended to provide comprehensive documentation, allowing users to fully leverage IMOS. Whether developing their own compatible applications, contributing to the platform's evolution, or adapting the platform's main distribution to their needs, users are provided with the tools to maximize IMOS's potential. By releasing this platform as open-source, the author seeks to foster a collaborative ecosystem, bringing the platform closer to its beneficiaries and ensuring that IMOS is shaped by the industry, for the industry.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] H. Lasi et al. "Industry 4.0". In: *Business Information Systems Engineering* 6 (4 2014-08), pp. 239–242. ISSN: 1867-0202. DOI: 10.1007/s12599-014-0334-4 (cit. on pp. 1, 4).

[3] E. Commission et al. *Industry 5.0 – Towards a sustainable, human-centric and resilient European industry*. Publications Office of the European Union, 2021. DOI: doi/10.2777/308407 (cit. on pp. 1, 5).

[4] D. Mourtzis, J. Angelopoulos, and N. Panopoulos. *A Literature Review of the Challenges and Opportunities of the Transition from Industry 4.0 to Society 5.0*. 2022-09. DOI: 10.3390/en15176276 (cit. on pp. 1, 6).

[5] C. Gröger et al. "Leveraging Apps in Manufacturing. A Framework for App Technology in the Enterprise". In: *Procedia CIRP* 7 (2013), pp. 664–669. ISSN: 22128271. DOI: 10.1016/j.procir.2013.06.050 (cit. on pp. 1, 7).

[6] J. M. Müller, O. Buliga, and K.-I. Voigt. "Fortune favors the prepared: How SMEs approach business model innovations in Industry 4.0". In: *Technological Forecasting and Social Change* 132 (2018-07), pp. 2–17. ISSN: 00401625. DOI: 10.1016/j.techfore.2017.12.019 (cit. on pp. 2, 5).

[7] K.-D. Thoben, S. Wiesner, and T. Wuest. ""Industrie 4.0" and Smart Manufacturing – A Review of Research Issues and Application Examples". In: *International Journal of Automation Technology* 11 (1 2017-01), pp. 4–16. ISSN: 1883-8022. DOI: 10.20965/ijat.2017.p0004 (cit. on pp. 2, 5).

[8] P. K. R. Maddikunta et al. *Industry 5.0: A survey on enabling technologies and potential applications*. 2022-03. DOI: 10.1016/j.jii.2021.100257 (cit. on pp. 2, 6).

[9] Y. Lu. "Industry 4.0: A survey on technologies, applications and open research issues". In: *Journal of Industrial Information Integration* 6 (2017-06), pp. 1–10. ISSN: 2452414X. DOI: 10.1016/j.jii.2017.04.005 (cit. on p. 5).

[10] L. Wang, M. Törngren, and M. Onori. "Current status and advancement of cyber-physical systems in manufacturing". In: *Journal of Manufacturing Systems* 37 (2015-10), pp. 517–527. ISSN: 02786125. DOI: 10.1016/j.jmsy.2015.04.008 (cit. on p. 5).

[11] H. Boyes et al. "The industrial internet of things (IIoT): An analysis framework". In: *Computers in Industry* 101 (2018-10), pp. 1–12. ISSN: 01663615. DOI: 10.1016/j.compind.2018.04.015 (cit. on p. 5).

[12] M. Rada. *Industry 5.0 definition*. 2020-05. URL: https://michael-rada.medium.com/industry-5-0-definition-6a2f9922dc48 (cit. on p. 5).

[13] S. Nahavandi. "Industry 5.0 - A human-centric solution". In: *Sustainability (Switzerland)* 11 (16 2019-08). ISSN: 20711050. DOI: 10.3390/su11164371 (cit. on p. 5).

[14] D. Mourtzis, J. Angelopoulos, and N. Panopoulos. "Closed-Loop Robotic Arm Manipulation Based on Mixed Reality". In: *Applied Sciences* 12 (6 2022-03), p. 2972. ISSN: 2076-3417. DOI: 10.3390/app12062972 (cit. on p. 6).

[15] D. Mourtzis, V. Siatras, and J. Angelopoulos. "Real-Time Remote Maintenance Support Based on Augmented Reality (AR)". In: *Applied Sciences* 10 (5 2020-03), p. 1855. ISSN: 2076-3417. DOI: 10.3390/app10051855 (cit. on p. 6).

[16] D. Mourtzis, J. Angelopoulos, and N. Panopoulos. "Operator 5.0: A Survey on Enabling Technologies and a Framework for Digital Manufacturing Based on Extended Reality". In: *Journal of Machine Engineering* 22 (1 2022-03), pp. 43–69. ISSN: 1895-7595. DOI: 10.36897/jme/147160 (cit. on p. 6).

[17] D. Mourtzis, J. Angelopoulos, and N. Panopoulos. "A Teaching Factory Paradigm for Personalized Perception of Education based on Extended Reality (XR)". In: *SSRN Electronic Journal* (2022). ISSN: 1556-5068. DOI: 10.2139/ssrn.4071876 (cit. on p. 6).

[18] K. Manikas and K. M. Hansen. "Software ecosystems – A systematic literature review". In: *Journal of Systems and Software* 86 (5 2013-05), pp. 1294–1306. ISSN: 01641212. DOI: 10.1016/j.jss.2012.12.026 (cit. on p. 9).

[19] J. Bosch and P. Bosch-Sijtsema. "From integration to composition: On the impact of software product lines, global development and ecosystems". In: *Journal of Systems and Software* 83 (1 2010-01), pp. 67–76. ISSN: 01641212. DOI: 10.1016/j.jss.2009.06.051 (cit. on p. 9).

[20] U. Eklund and J. Bosch. "Architecture for embedded open software ecosystems". In: *Journal of Systems and Software* 92 (2014-06), pp. 128–142. ISSN: 01641212. DOI: 10.1016/j.jss.2014.01.009 (cit. on p. 9).

[21] S. Jansen et al. "Shades of gray: Opening up a software producing organization with the open software enterprise model". In: *Journal of Systems and Software* 85 (7 2012-07), pp. 1495–1510. ISSN: 01641212. DOI: 10.1016/j.jss.2011.12.007 (cit. on p. 9).

[22] E. Papatheocharous, J. Axelsson, and J. Andrersson. "Issues and challenges in ecosystems for federated embedded systems". In: ACM, 2013-07, pp. 21–24. ISBN: 9781450320481. DOI: 10.1145/2489850.2489854 (cit. on pp. 9, 10).

[23] A. Yadav and S. Jayswal. "Modelling of flexible manufacturing system: a review". In: *International Journal of Production Research* 56 (7 2018-04), pp. 2464–2487. ISSN: 0020-7543. DOI: 10.1080/00207543.2017.1387302 (cit. on p. 10).

[24] L. Xu et al. "Reshaping the Landscape of the Future: Software-Defined Manufacturing". In: *Computer* 54 (7 2021-07), pp. 27–36. ISSN: 0018-9162. DOI: 10.1109/MC.2021.3074851 (cit. on p. 11).

[25] A. R. Akula et al. "Advanced Manufacturing Collaboration in a Cloud-based App Marketplace". In: ACM, 2017-05, pp. 146–155. ISBN: 9781450344876. DOI: 10.1145/3075564.3077547 (cit. on p. 11).

[26] *Siemens Insights Hub*. Accessed: 2024-05-03. URL: https://plm.sw.siemens.com/en-US/insights-hub/ (cit. on p. 12).

[27] V. K. Annanth, M. Abinash, and L. B. Rao. "Intelligent manufacturing in the context of industry 4.0: A case study of siemens industry". In: *Journal of Physics: Conference Series* 1969 (1 2021-07), p. 012019. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1969/1/012019 (cit. on p. 12).

[28] cordis.europa.eu CORDIS. *Virtual factory open operating system: VF-OS project: Fact sheet: H2020: Cordis: European Commission*. Accessed: 2023-11-06. 2016-09. URL: https://cordis.europa.eu/project/id/723710 (cit. on p. 13).

[29] V. Anaya et al. "Towards IoT Analytics. A vf-OS Approach". In: IEEE, 2018-09, pp. 570–575. ISBN: 978-1-5386-7097-2. DOI: 10.1109/IS.2018.8710476 (cit. on p. 13).

[30] D. Pape et al. "vf-OS Architecture". In: Wiley, 2018-10, pp. 77–82. DOI: 10.1002/9781119564034.ch9 (cit. on p. 14).

[31] L. M. D. Cunha, L. Stellingwerff, and A. Stam. "A Novel Approach to Software Development in the MicroserviceEnvironment of vf-OS". In: Wiley, 2018-10, pp. 115–119. DOI: 10.1002/9781119564034.ch14 (cit. on p. 14).

[32] F. Fraile et al. "Trustworthy Industrial IoT Gateways for Interoperability Platforms and Ecosystems". In: *IEEE Internet of Things Journal* 5 (6 2018-12), pp. 4506–4514. ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2832041 (cit. on p. 14).

[33] cordis.europa.eu CORDIS. *Zero defect manufacturing platform: ZDMP project: Fact sheet: H2020: Cordis: European Commission*. Accessed: 2023-11-06. 2019-02. URL: https://cordis.europa.eu/project/id/825631 (cit. on p. 15).

[34] F. Psarommatis, F. Fraile, and F. Ameri. "Zero Defect Manufacturing ontology: A preliminary version based on standardized terms". In: *Computers in Industry* 145 (2023-02), p. 103832. ISSN: 01663615. DOI: 10.1016/j.compind.2022.103832 (cit. on p. 15).

[35] A. A. Nazarenko et al. "Analysis of relevant standards for industrial systems to support zero defects manufacturing process". In: *Journal of Industrial Information Integration* 23 (2021-09), p. 100214. ISSN: 2452414X. DOI: 10.1016/j.jii.2021.100214 (cit. on p. 15).

[36] Fraile et al. "Reference Models for Digital Manufacturing Platforms". In: *Applied Sciences* 9 (20 2019-10), p. 4433. ISSN: 2076-3417. DOI: 10.3390/app9204433 (cit. on pp. 15, 20).

[37] J. Giao et al. "A Framework for Service-Oriented Architecture (SOA)-Based IoT Application Development". In: *Processes* 10 (9 2022-09), p. 1782. ISSN: 2227-9717. DOI: 10.3390/pr10091782 (cit. on p. 16).

[38] M. Abubakr et al. "Sustainable and Smart Manufacturing: An Integrated Approach". In: *Sustainability* 12 (6 2020-03), p. 2280. ISSN: 2071-1050. DOI: 10.3390/su12062280 (cit. on p. 17).

[39] B. Baldassarre et al. "Industrial Symbiosis: towards a design process for eco-industrial clusters by integrating Circular Economy and Industrial Ecology perspectives". In: *Journal of Cleaner Production* 216 (2019-04), pp. 446–460. ISSN: 09596526. DOI: 10.1016/j.jclepro.2019.01.091 (cit. on p. 17).

[40] H. Zhang et al. "Making the business case for sustainable manufacturing in small and medium-sized manufacturing enterprises: A systems decision making approach". In: *Journal of Cleaner Production* 287 (2021-03), p. 125038. ISSN: 09596526. DOI: 10.1016/j.jclepro.2020.125038 (cit. on pp. 17, 18).

[41] L. M. Camarinha-Matos, A. D. Rocha, and P. Graça. "Collaborative approaches in sustainable and resilient manufacturing". In: *Journal of Intelligent Manufacturing* 35 (2 2024-02), pp. 499–519. ISSN: 0956-5515. DOI: 10.1007/s10845-022-02060-6 (cit. on pp. 18, 36).

[42] P. Li and P. Jiang. "Enhanced agents in shared factory: Enabling high-efficiency self-organization and sustainability of the shared manufacturing resources". In: *Journal of Cleaner Production* 292 (2021-04), p. 126020. ISSN: 09596526. DOI: 10.1016/j.jclepro.2021.126020 (cit. on p. 18).

[43] A. Zeid et al. "Interoperability in Smart Manufacturing: Research Challenges". In: *Machines* 7 (2 2019-04), p. 21. ISSN: 2075-1702. DOI: 10.3390/machines7020021 (cit. on p. 19).

[44] S.-W. Lin et al. *Architecture Alignment and Interoperability An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper Contributors 1*. Accessed: 2023-10-25. URL: https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf (cit. on pp. 20, 21).

[45] X. Ye and S. H. Hong. "Toward Industry 4.0 Components: Insights Into and Implementation of Asset Administration Shells". In: *IEEE Industrial Electronics Magazine* 13 (1 2019-03), pp. 13–25. ISSN: 1932-4529. DOI: 10.1109/MIE.2019.2893397 (cit. on p. 20).

[46] IBM. *Industry 4.0 architecture for manufacturing*. Accessed: 2023-10-25. URL: https://www.ibm.com/cloud/architecture/architectures/industry-40/reference-architecture (cit. on p. 21).

[47] Y. Lu, F. Riddick, and N. Ivezic. "The Paradigm Shift in Smart Manufacturing System Architecture". In: 2016, pp. 767–776. DOI: 10.1007/978-3-319-51133-7_90 (cit. on p. 21).

[48] O. Boiko et al. "MES/ERP Integration Aspects of the Manufacturing Automation". In: 2020, pp. 15–24. DOI: 10.1007/978-3-030-40724-7_2 (cit. on p. 22).

[49] S. Gruner, J. Pfrommer, and F. Palm. "RESTful Industrial Communication With OPC UA". In: *IEEE Transactions on Industrial Informatics* 12 (5 2016-10), pp. 1832–1841. ISSN: 1551-3203. DOI: 10.1109/TII.2016.2530404 (cit. on p. 22).

[50] T. Coito et al. "A Middleware Platform for Intelligent Automation: An Industrial Prototype Implementation". In: *Computers in Industry* 123 (2020-12), p. 103329. ISSN: 01663615. DOI: 10.1016/j.compind.2020.103329 (cit. on p. 22).

[51] J. Sreemathy et al. "Overview of ETL Tools and Talend-Data Integration". In: IEEE, 2021-03, pp. 1650–1654. ISBN: 978-1-6654-0520-1. DOI: 10.1109/ICACCS51430.2021.9441984 (cit. on p. 22).

[52] S. Vinoski. "Integration with Web services". In: *IEEE Internet Computing* 7 (6 2003-11), pp. 75–77. ISSN: 1089-7801. DOI: 10.1109/MIC.2003.1250587 (cit. on p. 22).

[53] A. Torayev et al. "Towards Modular and Plug-and-Produce Manufacturing Apps". In: *Procedia CIRP* 107 (2022), pp. 1257–1262. ISSN: 22128271. DOI: 10.1016/j.procir.2022.05.141 (cit. on p. 23).

[54] T. Lyu, U. D. Atmojo, and V. Vyatkin. "Towards cloud-based virtual commissioning of distributed automation applications with IEC 61499 and containerization technology". In: IEEE, 2021-10, pp. 1–7. ISBN: 978-1-6654-3554-3. DOI: 10.1109/IECON48115.2021.9589945 (cit. on p. 24).

[55] T. Goldschmidt et al. "Container-based architecture for flexible industrial control applications". In: *Journal of Systems Architecture* 84 (2018-03), pp. 28–36. ISSN: 13837621. DOI: 10.1016/j.sysarc.2018.03.002 (cit. on pp. 24, 25).

[56] C. Perera et al. "A Survey on Internet of Things From Industrial Market Perspective". In: *IEEE Access* 2 (2014), pp. 1660–1679. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2389854 (cit. on p. 26).

[57] J. Arents et al. "Human–Robot Collaboration Trends and Safety Aspects: A Systematic Review". In: *Journal of Sensor and Actuator Networks* 10 (3 2021-07), p. 48. ISSN: 2224-2708. DOI: 10.3390/jsan10030048 (cit. on p. 26).

[58] L. D. Evjemo et al. "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories". In: *Current Robotics Reports* 1 (2 2020-06), pp. 35–41. ISSN: 2662-4087. DOI: 10.1007/s43154-020-00006-5 (cit. on p. 27).

[59] S. Robla-Gomez et al. "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments". In: *IEEE Access* 5 (2017), pp. 26754–26773. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2773127 (cit. on p. 27).

[60] A. R. Sadik, B. Urban, and O. Adel. "Using Hand Gestures to Interact with an Industrial Robot in a Cooperative Flexible Manufacturing Scenario". In: ACM, 2017-02, pp. 11–16. ISBN: 9781450352802. DOI: 10.1145/3068796.3068801 (cit. on p. 27).

[61] M. R. Pedersen et al. "Robot skills for manufacturing: From concept to industrial deployment". In: *Robotics and Computer-Integrated Manufacturing* 37 (2016-02), pp. 282–291. ISSN: 07365845. DOI: 10.1016/j.rcim.2015.04.002 (cit. on p. 27).

[62] J. Wang et al. "Deep learning for smart manufacturing: Methods and applications". In: *Journal of Manufacturing Systems* 48 (2018-07), pp. 144–156. ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.01.003 (cit. on p. 27).

[63] M. Sharp, R. Ak, and T. Hedberg. "A survey of the advancing use and development of machine learning in smart manufacturing". In: *Journal of Manufacturing Systems* 48 (2018-07), pp. 170–179. ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.02.004 (cit. on p. 27).

[64] R. Rai et al. "Machine learning in manufacturing and industry 4.0 applications". In: *International Journal of Production Research* 59 (16 2021-08), pp. 4773–4778. ISSN: 0020-7543. DOI: 10.1080/00207543.2021.1956675 (cit. on pp. 27, 28).

[65] S. Deng and T.-H. Yeh. "Using least squares support vector machines for the airframe structures manufacturing cost estimation". In: *International Journal of Production Economics* 131 (2 2011-06), pp. 701–708. ISSN: 09255273. DOI: 10.1016/j.ijpe.2011.02.019 (cit. on p. 28).

[66] J. Woodward and N. Gindy. "A hyper-heuristic multi-criteria decision support system for eco-efficient product life cycle". In: IET, 2010, pp. 201–205. ISBN: 978-1-84919-199-9. DOI: 10.1049/cp.2010.0436 (cit. on p. 28).

[67] U. K. Yusof, R. Budiarto, and S. Deris. "Harmony search algorithm for flexible manufacturing system (FMS) machine loading problem". In: IEEE, 2011-06, pp. 26–31. ISBN: 978-1-61284-211-0. DOI: 10.1109/DMO.2011.5976500 (cit. on p. 28).

[68] S.-j. Wu et al. "A Neural Network Integrated Decision Support System for Condition-Based Optimal Predictive Maintenance Policy". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 37 (2 2007-03), pp. 226–236. ISSN: 1083-4427. DOI: 10.1109/TSMCA.2006.886368 (cit. on p. 28).

[69] J. Chen, Z. Zhang, and F. Wu. "A data-driven method for enhancing the image-based automatic inspection of IC wire bonding defects". In: *International Journal of Production Research* 59 (16 2021-08), pp. 4779–4793. ISSN: 0020-7543. DOI: 10.1080/00207543.2020.1821928 (cit. on p. 28).

[70] A. Glaeser et al. "Applications of deep learning for fault detection in industrial cold forging". In: *International Journal of Production Research* 59 (16 2021-08), pp. 4826–4835. ISSN: 0020-7543. DOI: 10.1080/00207543.2021.1891318 (cit. on p. 28).

[71] F. Zangaro, S. Minner, and D. Battini. "A supervised machine learning approach for the optimisation of the assembly line feeding mode selection". In: *International Journal of Production Research* 59 (16 2021-08), pp. 4881–4902. ISSN: 0020-7543. DOI: 10.1080/00207543.2020.1851793 (cit. on p. 28).

[72] L. Fang et al. "Design and Development of the AI-assisted Safety System for Hazardous Plant". In: IEEE, 2020-10, pp. 60–64. ISBN: 978-0-7381-0545-1. DOI: 10.1109/CISP-BMEI51763.2020.9263603 (cit. on p. 28).

[73] L. Fang et al. "Design and Development of Industrial Safety APPs". In: IEEE, 2022-05, pp. 526–530. ISBN: 978-1-7281-8115-8. DOI: 10.1109/ICETCI55101.2022.9832162 (cit. on p. 28).

[74] Z. Zhang et al. "Safety Helmet and Mask Detection at Construction Site Based on Deep Learning". In: IEEE, 2023-05, pp. 990–995. ISBN: 978-1-6654-9079-5. DOI: 10.1109/ICIBA56860.2023.10165396 (cit. on p. 28).

[75] M. M. Daud, H. M. Saad, and M. T. Ijab. "Conceptual Design of Human Detection via Deep Learning for Industrial Safety Enforcement in Manufacturing Site". In: IEEE, 2021-06, pp. 369–373. ISBN: 978-1-6654-0343-6. DOI: 10.1109/I2CACIS52118.2021.9495856 (cit. on p. 28).

[76] Y. Xiao et al. "Real-time Object Detection for Substation Security Early-warning with Deep Neural Network based on YOLO-V5". In: IEEE, 2022-05, pp. 45–50. ISBN: 978-1-6654-4357-9. DOI: 10.1109/GlobConET53749.2022.9872338 (cit. on p. 29).

[77] M. H. F. A. Hazza, E. Y. T. Adesta, and A. H. Taha. "Simulation of real time tracking system using RFID technology to enhance quality activities in flexible manufacturing system". In: IEEE, 2017-08, pp. 121–124. ISBN: 978-1-5386-1771-7. DOI: 10.1109/ISCBI.2017.8053557 (cit. on p. 29).

[78] E. Svertoka et al. "Wearables for Industrial Work Safety: A Survey". In: *Sensors* 21 (11 2021-06), p. 3844. ISSN: 1424-8220. DOI: 10.3390/s21113844 (cit. on p. 29).

[79] L. M. Camarinha-Matos and H. Afsarmanesh. "The Evolution Path to Collaborative Networks 4.0". In: 2021, pp. 170–193. DOI: 10.1007/978-3-030-81701-5_7 (cit. on p. 35).

[80] U. Nations. *Transforming Our World: The 2030 Agenda for Sustainable Development*. Accessed: 2024-04-29. 2015. URL: https://sdgs.un.org/2030agenda (cit. on p. 35).

[81] L. M. Camarinha-Matos, R. Fornasiero, and H. Afsarmanesh. "Collaborative Networks as a Core Enabler of Industry 4.0". In: vol. 506. 2017, pp. 3–17. DOI: 10.1007/978-3-319-65151-4_1 (cit. on p. 35).

[82] I. Khan et al. "Collaborative Innovation, Collaborative Capabilities and Value Co-creation in an Industry 4.0 Context: An Empirical Evidence". In: *ArXiv* abs/2103.16279 (2021). URL: https://api.semanticscholar.org/CorpusID:232417059 (cit. on p. 35).

[83] S. E. D. Falco et al. "Open collaborative innovation and digital platforms". In: *Production Planning Control* 28 (16 2017-12), pp. 1344–1353. ISSN: 0953-7287. DOI: 10.1080/09537287.2017.1375143 (cit. on p. 36).

[84] N. Smorodinskaya et al. "Innovation Ecosystems vs. Innovation Systems in Terms of Collaboration and Co-creation of Value". In: 2017. DOI: 10.24251/HICSS.2017.636 (cit. on p. 36).

[85] A. Neves et al. "A comprehensive review of industrial symbiosis". In: *Journal of Cleaner Production* 247 (2020-02), p. 119113. ISSN: 09596526. DOI: 10.1016/j.jclepro.2019.119113 (cit. on p. 36).

[86] L. Camarinha-Matos and H. Afsarmanesh. "Classes of Collaborative Networks". In: IGI Global, 2008, pp. 193–198. DOI: 10.4018/9781605667706.ch021 (cit. on pp. 36, 37).

[87] *Node.js v20.x Documentation*. Accessed: 2024-02-02. URL: https://nodejs.org/docs/latest-v20.x/api/index.html (cit. on p. 55).

[88] I. K. Chaniotis, K.-I. D. Kyriakou, and N. D. Tselikas. "Is Node.js a viable option for building modern web applications? A performance evaluation study". In: *Computing* 97 (10 2015-10), pp. 1023–1044. ISSN: 0010-485X. DOI: 10.1007/s00607-014-0394-9 (cit. on p. 55).

[89] *Electron Documentation*. Accessed: 2024-02-02. URL: https://www.electronjs.org/docs/latest/ (cit. on p. 55).

[90] K. Kredpattanakul and Y. Limpiyakorn. "Transforming JavaScript-Based Web Application to Cross-Platform Desktop with Electron". In: 2019, pp. 571–579. DOI: 10.1007/978-981-13-1056-0_56 (cit. on p. 55).

[91] *Docker Documentation*. Accessed: 2024-03-17. URL: https://docs.docker.com/ (cit. on p. 55).

[92] D. Merkel. "Docker: lightweight Linux containers for consistent development and deployment". In: *Linux Journal* 2014 (2014-03) (cit. on p. 55).

[93] *Express Guide*. Accessed: 2024-02-24. URL: https://expressjs.com/en/guide/routing.html (cit. on p. 55).

[94] M. Wicha and B. Pańczyk. "Performance analysis of REST API technologies using Spring and Express.js examples". In: *Journal of Computer Sciences Institute* 29 (2023-12), pp. 352–359. ISSN: 2544-0764. DOI: 10.35784/jcsi.3796 (cit. on p. 55).

[95] *MongoDB Atlas Documentation*. Accessed: 2024-02-24. URL: https://www.mongodb.com/docs/atlas/ (cit. on p. 55).

[96] S. H. Aboutorabi et al. "Performance evaluation of SQL and MongoDB databases for big e-commerce data". In: IEEE, 2015-08, pp. 1–7. ISBN: 978-1-4673-9181-8. DOI: 10.1109/CSICSSE.2015.7369245 (cit. on p. 55).

[97] *NSIS User Manual*. Accessed: 2024-03-29. URL: https://nsis.sourceforge.io/Docs/ (cit. on p. 55).

[98] J. B. Tyndall. "Building an effective software deployment process". In: ACM, 2012-10, pp. 109–114. ISBN: 9781450314947. DOI: 10.1145/2382456.2382482 (cit. on p. 55).

[99] *IMOS Project GitHub Repository*. Accessed: 2024-07-30. URL: https://github.com/altsmpegado/imos (cit. on pp. 56, 60, 63, 81).

[100] *Helm Kubernetes*. Accessed: 2024-06-27. URL: https://helm.sh/docs/ (cit. on p. 72).

[101] A. M. Pegado. *IMOS Short Demo Video*. YouTube video. Accessed: 2024-08-29. 2024. URL: https://youtu.be/gMBulW-aJ8Y (cit. on pp. 72, 80).

[102] N. Freitas et al. "Cloud-Based Machine Learning Application for Predicting Energy Consumption in Automotive Spot Welding". In: *Processes* 11 (1 2023-01), p. 284. ISSN: 2227-9717. DOI: 10.3390/pr11010284 (cit. on pp. 73, 76, 77).

[103] *PyTorch Inference API*. Accessed: 2024-04-13. URL: https://pytorch.org/serve/inference_api.html (cit. on p. 74).

[104] *Ultralytics YOLOv5 GitHub*. Accessed: 2024-04-02. URL: https://github.com/ultralytics/yolov5 (cit. on p. 74).

[105] *Roboflow*. Accessed: 2024-03-19. URL: https://roboflow.com/ (cit. on p. 74).

[106] *Run a Model - Predict on an Image Over HTTP. In: Roboflow Inference Documentation*. Accessed: 2024-04-24. URL: https://inference.roboflow.com/quickstart/run_model_on_image/ (cit. on p. 78).

[107] A. M. Pegado. *IMOS Platform Overview and Project Insights*. YouTube video. Accessed: 2024-08-29. 2024. URL: https://youtu.be/HzBn3Vpyl_c (cit. on p. 80).