



HAL
open science

Digital Twin for the Dynamic Orchestration of Autonomous and Embedded Systems

Yining Huang

► **To cite this version:**

Yining Huang. Digital Twin for the Dynamic Orchestration of Autonomous and Embedded Systems. Modeling and Simulation. Sorbonne Université, 2024. English. NNT: 2024SORUS020 . tel-04679161

HAL Id: tel-04679161

<https://theses.hal.science/tel-04679161v1>

Submitted on 27 Aug 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sorbonne Université

Ecole Doctorale Informatique, Télécommunications et
Electronique

Laboratoire d'informatique de Sorbonne Université

Thesis defended by **Yining HUANG**

In order to become Doctor from Sorbonne Université

Defended on 08/02/2024

**DIGITAL TWIN FOR THE DYNAMIC ORCHESTRATION OF
AUTONOMOUS AND EMBEDDED SYSTEMS**

*Jumeau numérique pour l'orchestration dynamique de
systèmes autonomes et embarqués*

Thesis supervised by

| | | |
|-------------------|----------|---------------|
| Jacques MALENFANT | LIP6 | Supervisor |
| Saadia DHOUB | CEA List | Co-supervisor |

Reviewers

| | | | |
|-------------------|-------|-----------|----------|
| Jean-Michel BRUEL | IRIT | Professor | Reviewer |
| Marianne HUCHARD | LIRMM | Professor | Reviewer |

Committee members

| | | | |
|--------------------|-------------|-------------------|------------------|
| Jean-Michel BRUEL | IRIT | Professor | Reviewer |
| Marija JANKOVIC | LGI EA 2606 | Professor | Examiner |
| Emmanuel CHAILLOUX | LIP6 | Professor | President |
| Saadia DHOUB | CEA List | Research Engineer | Supervisor |
| Jacques MALENFANT | LIP6 | Professor | Director |

Acknowledgments

First of all, I would like to thank my supervisors. Dr. Saadia Dhouib, who gave me this valuable opportunity and has been guiding me for many years. Also Prof. Jacques Malenfant, who has not only supported me academically, but also given me great encouragement and support in my life.

I would also like to thank my thesis committee members and my follow-up committee members who were involved in the validation survey for this research project: Prof. Jean-Michel Bruel, Prof. Marianne Huchard, Prof. Emmanuel Chailloux, Prof. Marija Jankovic and Prof. Jérémie Guiochet. I really appreciate their passionate participation and their valuable comments on my thesis.

I would also like to sincerely thank my colleagues. Dr. Chokri Mraidha, as the head of our lab, has given me a lot of support and help in my work, Dr. Quang-Duy Ngyuen who is very experienced and talented always gives me pertinent advice and guidance, and Dr. Nesrine Ben, has always been like a sister to me in my work and life. The same goes to all the LSEA colleagues and people I have collaborated with, I can't list them all here because there are too many people I want to thank. I am grateful to CEA for accepting me and giving me a platform for my research.

I would also like to thank Dr. Henri Sohier, who introduced me to the path of academia, starting with my first internship. He gave me a lot of honest advice when I was young and still unsure of my future direction. The same thanks goes to all my colleagues during my time at IRT SystemX for making me feel relaxed and good about the academic atmosphere.

I also want to thank my friends for being there for me during my PhD. I would like to thank Yifan Yang for staying up late with me, Zehui Xuan for spending many enjoyable weekends with me, all my friends lived in the Impasse Cerisaie house for supporting me, and all my friends in China for giving me remote encouragement. And a special thanks to Mimi, Heizai and Mika, the three kittens who accompanied me through these long days.

Finally, I must express my very profound gratitude to my parents and my grandparents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Jumeau numérique pour l'orchestration dynamique de systèmes autonomes et embarqués

Résumé

À mesure que les technologies de contrôle et d'information progressent, les systèmes de production modernes évoluent vers une automatisation et une intelligence accrues. Le concept de jumeau numérique prend de plus en plus d'importance dans divers secteurs, offrant une méthode claire pour gérer des systèmes complexes par le biais de contreparties numériques. Toutefois, ce domaine émergent présente plusieurs défis et questions de recherche critiques, notamment en ce qui concerne l'application des systèmes de jumeaux numériques à l'industrie. Au-delà des défis posés par la supervision de systèmes cyber-physiques complexes, il existe un besoin pressant d'une représentation complète et interopérable des produits, des processus de production et des ressources de l'usine, qui puisse s'intégrer de manière transparente à divers équipements. Cette thèse étudie trois questions principales : l'interopérabilité, l'adaptabilité et la robustesse des systèmes de production dans le contexte de l'industrie 4.0. Elle propose une architecture de production auto-adaptative basée sur la représentation des capacités de production (CBSAM), qui utilise des approches d'ingénierie logicielle telles que l'ingénierie basée sur les capacités, l'ingénierie dirigée par les modèles (IDM) et l'approche MAPE-K (surveiller, analyser, planifier, exécuter et connaissance) pour améliorer l'adaptabilité et l'efficacité des systèmes de production. L'architecture proposée intègre une boucle de rétroaction pour permettre l'auto-adaptabilité du système de jumeau numérique, qui assure la continuité du processus de production sur ses performances et s'adapte aux changements du système physique. Cette recherche va au-delà de l'architecture conceptuel pour développer des outils logiciels pratiques, en intégrant la technologie jumeau numérique dans un écosystème existant d'ingénierie des systèmes basée sur les modèles (ISBM) pour compléter l'architecture CBSAM. Un cas d'usage académique développé dans le laboratoire présente en détail la mise en œuvre de la méthodologie CBSAM. Enfin, cette thèse se termine par une généralisation à l'architecture et à l'achèvement de la mise en œuvre comme perspective pour l'avenir.

Mots clés : jumeaux numériques, production auto-adaptative, ingénierie dirigée par les modèles, ontologie, interopérabilité, informatique sémantique

Digital twin for the dynamic orchestration of autonomous and embedded systems

Abstract

As control and information technology advance, modern manufacturing systems are evolving towards increased automation and intelligence. The concept of the digital twin is becoming increasingly prominent across various sectors, offering a clear method for managing complex systems via digital counterparts. However, this emerging field presents several critical challenges and research questions, particularly in applying digital twin systems to industry. Beyond the established challenges in supervising complex cyber-physical systems, there is a pressing need for a comprehensive, interoperable representation of products, production processes, and plant resources that can integrate seamlessly with diverse equipment. This thesis investigates three primary issues: interoperability, adaptability, and robustness of manufacturing systems in the context of Industry 4.0. A capability-based self-adaptive manufacturing architecture (CBSAM) has been proposed in this thesis, utilizing software engineering approaches like capability-based engineering, Model-Driven Engineering (MDE), and the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) framework to enhance the adaptability and efficiency of production systems. The proposed architecture incorporates a feedback loop to enable the self-adaptivity of the digital twin system, which ensures the continuity of the production process on its performance and adapts to changes in the physical system. This research extends beyond conceptual methods to develop practical software tools, integrating Digital Twin (DT) technology into an existing Model-Based Systems Engineering (MBSE) ecosystem to complete the CBSAM architecture. An academic testbed developed in the laboratory showcases the CBSAM methodology's implementation in detail. Finally, this thesis concludes with a generalization to the architecture and the completion of the implementation as a perspective for the future.

Keywords: digital twins, self-adaptive manufacturing, model-driven engineering, ontology, interoperability, semantic computing

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | General context | 1 |
| 1.2 | Research Problems | 2 |
| 1.2.1 | Interoperability | 2 |
| 1.2.2 | Adaptability | 3 |
| 1.2.3 | Robustness | 3 |
| 1.3 | Contribution Overview | 3 |
| 1.4 | Manuscript Plan | 6 |
| 2 | Related Work Studies | 7 |
| 2.1 | Industrial Digital Twin & MDE | 7 |
| 2.1.1 | DSLs for Industrial Digital Twin Models | 8 |
| 2.1.2 | AAS Implementations | 11 |
| 2.1.3 | Asset Connection and Orchestration | 13 |
| 2.2 | Capability-Based Engineering | 14 |
| 2.3 | Semantic Digital Twin | 15 |
| 2.3.1 | Manufacturing Resource Capability Ontology (MaRCO) | 17 |
| 2.3.2 | OML Adapter | 17 |
| 2.4 | Self-Adaptive CPS | 18 |
| 2.4.1 | Property Value Statement | 18 |
| 2.4.2 | RDF Graph Stream Processing | 20 |
| 2.4.3 | Fault detection methods | 21 |
| 2.5 | Conclusion | 22 |
| 3 | Capability-Based Self-Adaptive Manufacturing Architecture | 25 |
| 3.1 | Architecture Overview | 25 |
| 3.2 | Specification | 29 |
| 3.3 | Design | 30 |
| 3.3.1 | AAS Modeling environment | 30 |
| 3.3.2 | Capability Submodel Design | 33 |
| 3.3.3 | Operational Data Submodel Modeling | 35 |
| 3.3.4 | Monitoring Submodel Modeling | 36 |
| 3.4 | Engineering & Deployment | 41 |
| 3.4.1 | Capability Checking | 41 |
| 3.4.2 | Feasibility Checking | 44 |
| 3.4.3 | Skill Execution | 44 |
| 3.5 | Operations & Maintenance | 45 |
| 3.5.1 | Monitoring | 45 |
| 3.5.2 | Analysis | 45 |
| 3.5.3 | Planning | 46 |
| 3.5.4 | Execution | 47 |

| | | |
|----------|--|------------|
| 3.5.5 | Knowledge Integration | 47 |
| 3.6 | Conclusion | 48 |
| 4 | Implementation | 49 |
| 4.1 | Overview | 49 |
| 4.2 | Modeling Environment Development | 50 |
| 4.2.1 | AAS Model Design Diagrams | 50 |
| 4.3 | Capability Checking @ Design Time Development | 53 |
| 4.3.1 | Model Transformation Module | 54 |
| 4.3.2 | Capability Matchmaker Module | 57 |
| 4.3.3 | User Interface Module | 58 |
| 4.4 | Skill Execution & Orchestration Development | 59 |
| 4.4.1 | Deployable AAS Server Architecture | 59 |
| 4.4.2 | Node-RED Package Development | 60 |
| 4.5 | Monitoring & Diagnosis Development | 62 |
| 4.5.1 | Local Monitor Knowledge Generation | 62 |
| 4.5.2 | Data Acquisition and Real-Time Stream generation | 63 |
| 4.5.3 | Runtime Analysis | 65 |
| 4.6 | Conclusion | 66 |
| 5 | LocalSEA Testbed | 69 |
| 5.1 | Overview | 69 |
| 5.2 | Testbed Requirement | 70 |
| 5.3 | LocalSEA Testbed Composition | 71 |
| 5.3.1 | Hardware Architecture | 71 |
| 5.3.2 | Software Architecture | 73 |
| 5.3.3 | Communication Architecture | 74 |
| 5.4 | CBSAM Architecture Implementation | 77 |
| 5.4.1 | Specification | 77 |
| 5.4.2 | AAS Model Design | 86 |
| 5.4.3 | Engineering & Deployment | 95 |
| 5.4.4 | Operation & Maintenance | 98 |
| 5.5 | Conclusion | 99 |
| 6 | Conclusion & Perspectives | 101 |
| 6.1 | Contributions | 101 |
| 6.1.1 | Proposed Architecture | 101 |
| 6.1.2 | Software Implementations | 102 |
| 6.1.3 | Testbed Validation | 102 |
| 6.1.4 | Publications | 103 |
| 6.2 | Future Perspectives | 104 |
| 6.2.1 | Short-Term Perspectives | 104 |
| 6.2.2 | Long-Term Perspectives | 104 |
| 6.2.3 | Domain Perspectives | 105 |
| | Figures | 106 |
| | Tables | 109 |
| | Bibliography | 109 |

| | | |
|----------|---------------------------------------|------------|
| A | Résumé en Français | 119 |
| A.1 | Contexte Général | 119 |
| A.2 | Problématiques de Recherche | 120 |
| A.2.1 | Interopérabilité | 121 |
| A.2.2 | Adaptabilité | 121 |
| A.2.3 | Robustesse | 122 |
| A.3 | Contributions Principales | 122 |
| A.4 | L'architecture CBSAM | 125 |
| A.5 | Contenu du manuscrit | 128 |
| A.6 | Conclusion et perspectives | 129 |
| B | Capability Checking Module | 131 |
| B.1 | Download extensions | 131 |
| B.2 | Example | 132 |
| C | AAS BaSyx Node-RED Package | 137 |
| C.1 | Prerequisite | 137 |
| C.2 | Install Steps | 137 |
| C.3 | Workflow for List Days | 137 |
| C.3.1 | Flow Deployment | 137 |

Chapter 1

Introduction

Contents

| | | |
|------------|------------------------------|----------|
| 1.1 | General context | 1 |
| 1.2 | Research Problems | 2 |
| 1.2.1 | Interoperability | 2 |
| 1.2.2 | Adaptability | 3 |
| 1.2.3 | Robustness | 3 |
| 1.3 | Contribution Overview | 3 |
| 1.4 | Manuscript Plan | 6 |

1.1 General context

The future industry will be dominated by highly autonomous and adaptive intelligent manufacturing systems. Lot-size-one systems, as well as plug-and-produce concepts, imply producing an increased variety of products in a highly flexible and timely manner and making commissioning and maintenance more efficient. The smaller the production lot, the greater the need for frequent reconfiguration, which requires a reduction in the interval between lots hence the automation of the reconfiguration process. The speed with which manufacturers, in particular SMEs, can reconfigure the production to a new run and thus respond to clients and avoid costly machine downtime is critical to maintaining commercial success and profit margins. These systems should possess a high degree of autonomy to deal with the reconfiguration of production lines and to cope with a wide variety of unforeseen situations.

The Industry 4.0 (I4.0) [1] paradigm appears to integrate information and communication technologies into novel manufacturing systems to elevate the efficiency and quality of the production processes. Under this concept, self-control of complex systems is the basis for modern industrial production. In order to establish such a self-adaptable system, the ability to autonomously orchestrate and maintain the complex system is essential. The flexible manufacturing and predictive maintenance of the production line management in the vision of Industry 4.0 will primarily rely on digital twin [2] technology.

The digital twin paradigm [3] aims to achieve real-time feedback from the physical system to its digital counterpart (Cyber part in a Cyber-Physical System). Model-driven engineering (MDE)[4] is an approach that first arose in the software development domain, which emphasizes the use of models to design and build systems. In the context of industrial digital twins, MDE technologies can be used to create and maintain the digital twin model. It provides a basis for building digital twin models that simplify the design and development of complex systems. By using accurate MDE methods, manufacturers can

implement in a rigorous way the control, simulate, and optimize the production process under recognized software engineering principles and processes. Besides, the concept of models@run.time [5] in MDE leverages the benefits of models, such as abstraction and modularity, through the entire lifecycle of a system. The idea of processable and executable models meets the definition of the digital twin when the models remain in active connection with the physic asset during runtime. The abstraction allows designers and developers to focus on high-level concepts that enhance human comprehension. The modularity ensures the flexibility and scalability of the system management. Models@run.time reflects the current state of the system, which provides an framework for self-adaptation. The self-adaptation loop consists of four phases: Monitoring, Analysis, Planning, and Execution. This feedback loop is pivotal, which allows dynamic adaptation to changes and ensures the robustness of the system.

1.2 Research Problems

However, several key challenges and research questions have emerged in the realm of digital twin systems and their applications to the industry. Besides well-known challenges in the supervisory control of complex cyber-physical systems, the need to provide a comprehensive representation of products, production processes, and plant resources in an interoperable way among heterogeneous equipment represents significant challenges. Three central problems investigated in this thesis are *interoperability*, *adaptability* and *robustness* of manufacturing systems in the Industry 4.0 era.

1.2.1 Interoperability

Interoperability [6] refers to the ability of different systems and organizations to work together seamlessly. It can be subdivided into syntactic interoperability and semantic interoperability. Syntactic interoperability denotes that systems have the ability to communicate and exchange data, which includes specified data formats, communication protocols, interface descriptions, etc. The standards provide rules, guidance, templates, etc., for the consistency and conformity of a specific domain, which regulates the system construction in a consistent and transparent manner. However, in emerging fields such as Industry 4.0 or digital twins, it is not easy to find suitable standards and to implement them. This difficulty often arises due to reliance on specific vendors and technologies. The challenge is to standardize all the participants in an Industry 4.0 digital twin system and provide an efficient framework to establish the digital twin models, thereby making them vendor-independent and technology-independent.

RQ1 How can we identify and leverage the current existing standards and technologies that we can use to design a digital twin model for the production plans and the plant resources in a both machine-interpretable and user-friendly way?

With syntactic interoperability, the same physical entity can be represented differently by distinct stakeholders. Even more, two different entities could be represented in the same way. When dealing with a large number of physical entities, it is crucial to be able to identify the entities that have the same or equivalent capabilities, in the deepest sense. Semantic interoperability considers more than just efficient data exchange between systems. It also includes the fully understood and processed by all parties. Many research units and groups have realized this problem and studied this topic. However, the issue of semantic interoperability is too domain-specific, and the semantic expressions in different domains might differ greatly. Therefore, according to the boundary of our project, we mainly summarize the following two questions.

RQ2 How can we semantically define the models in order to determine the most appropriate selection of available plant resources that fulfill the production plan requirements?

RQ3 How to semantically process the monitoring data obtained from the digital twin during execution and diagnose the risks in the manufacturing system?

1.2.2 Adaptability

System adaptability refers to a system that is able to timely react to changes. A manufacturing system is the integration of all the existing resources, desired products, and production processes. Therefore, an adaptive manufacturing system should be able to respond to changes in all parts. As market demands evolve or internal goals change, production lines may need adjustments. Therefore, a vital aspect of adaptability is to automatically realize the re-planning of the production line layout when the process workflow changes. Another important aspect is the ability for rapid configuration/reconfiguration. The system should be designed to be easily reconfigurable without significant downtime [7]. These challenges raise the following research questions:

RQ4 What methodologies can be developed to facilitate dynamic re-planning of production lines in a flexible and automated manner?

RQ5 What strategy can be employed to ensure rapid and efficient system reconfiguration in response to varying operational needs?

1.2.3 Robustness

Furthermore, robustness, as defined in [8], is a key feature of a manufacturing system. It refers to the system or component that can function correctly in the presence of invalid inputs or stressful environmental conditions. It requires the system's reactivity, fault tolerance, and self-adaptability. The difference between an adaptive system and a self-adaptive system reflects whether the system includes a capability to decide by itself when and how to be adapted. One well-known approach to implementing such a capability is a feedback loop organized around the four steps proposed by IBM [9]: monitoring, analysis, planning, and (adaptation) execution. The latter is capable of adjusting its own runtime behavior in order to achieve system objectives. Monitoring and diagnosis over dynamic digital twin models are crucial to realizing self-adaptive systems. Monitoring updates the digital twin with real-world data. The diagnostic module examines the data and uses the information carried by the digital twin to make decisions. Then the digital twin should interact with the real world and perform decisions. These characteristics can make it possible for the manufacturing systems to not only keep operating at their optimum but also to provide recovery plans in case of anomalies. Therefore, the system's self-adaptability process can be theoretically completed by relying on digital twin technologies. However, the implementation in practice of combining these theories in an appropriate and low-cost way remains an open challenge.

RQ6 How can real-time monitoring and diagnosis modules be effectively integrated into existing systems to ensure continuous operation during execution phases?

1.3 Contribution Overview

The main purpose of this doctoral thesis is to find solutions to the above problems. It aims to implement a flexible production management method prototype based on digital twin technology in an MDE environment and to validate the approaches on an academic

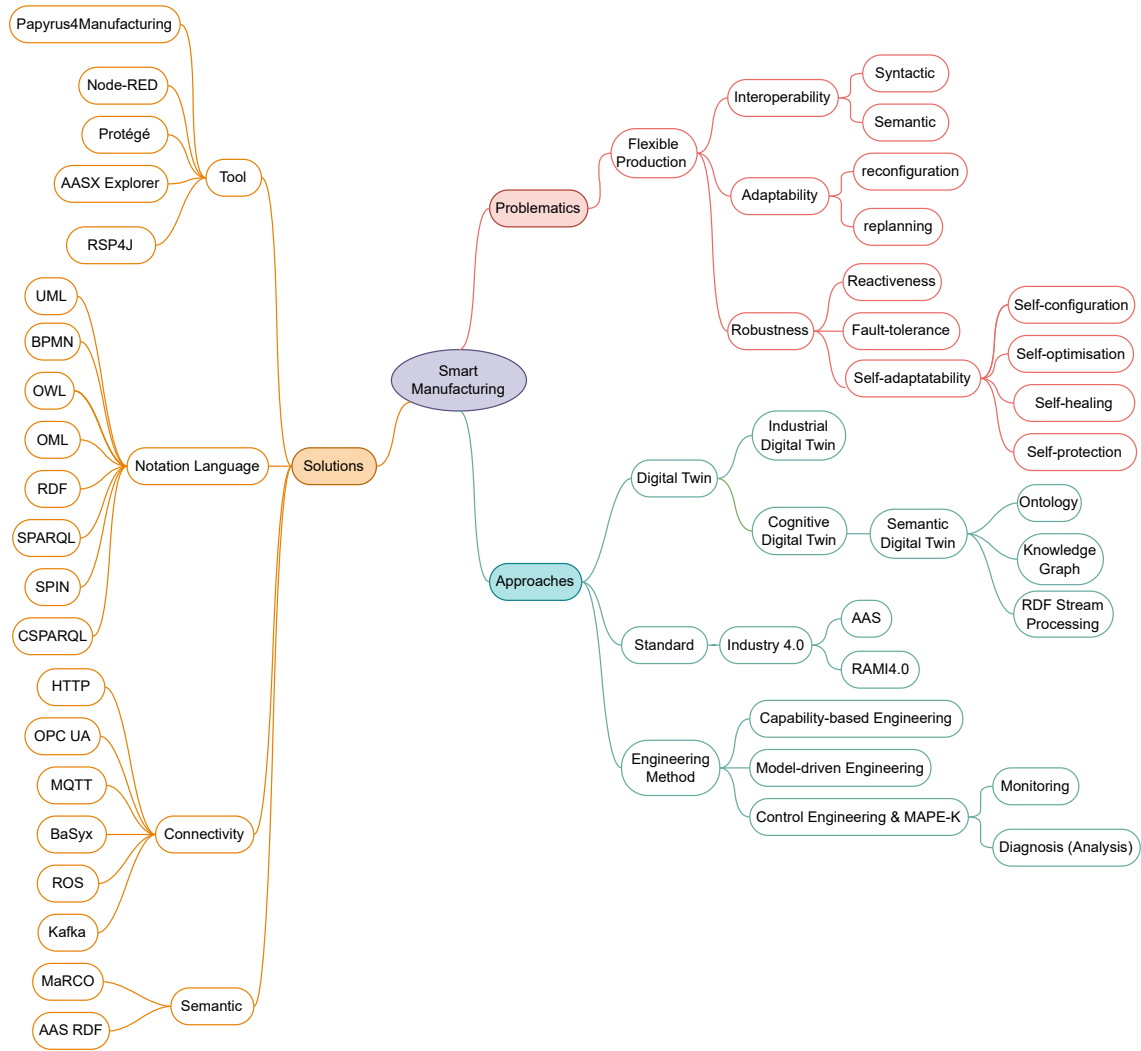


Figure 1.1: Mindmap for concepts appear in the thesis

demonstrator. Figure 1.1 illustrates the problematic analyzed in the previous section, the approaches adopted to shape the methodology, and the solutions incorporated in the implementation.

- The primary contribution is to create a modeling environment designed for industrial digital twin development. This environment should address the issue of syntactic interoperability by using MDE methodologies. The Asset Administration Shell (AAS) [10] is a strong candidate for standardizing the industrial digital twin models, proposed by Plattform Industrie 4.0. The Papyrus4Manufacturing (P4M) toolset is developed to provide an MDE approach to the AAS implementation. This toolset integrates user-friendly editors for standardizing the model creation of production participants. P4M provides not only the creation of digital twin models of production systems but also an automatic deployment functionality based on model transformations and code generation. The development of a specific toolset tailored for this specification language is a key aspect of this contribution. (1) The AAS digital twin modeling environment provides a common interface to describe all kinds of production participants. (2) The modeling of production processes requires a standardized language for process design. The Business Process Model and Notation (BPMN) [11] is a set of graphical representations that describe business processes. The contribution includes the integration of BPMN modeling plugin into the digital

twin modeling environment.

- The AAS standard provides a syntactic interoperability interface for all assets involved in smart factories. However, there is still a need to fill the gap regarding semantic interoperability, in order to allow efficient understanding between Industry 4.0 components. Ontologies define semantic models of data combined with relevant domain knowledge and formulate inference strategies. Therefore, it is a highly relevant approach to bringing semantic interoperability. The second contribution consists of proposing an ontology-based AAS modeling method that enables the semantic interoperability of AAS digital twin models. This approach bridges the barriers between ontology representation (specifically OWL) and AAS models in P4M (UML models). This contribution is about annotating AAS digital twin models with semantic meaning extracted from ontology knowledge. This implementation mainly contains two different ontologies (1) MaRCO (Manufacturing Resource Capability Ontology) [12] to provide semantic descriptions for manufacturing capabilities, and (2) to complement model-driven engineering tools with automated reasoning.
- The capability-based engineering (CBE) [13] approach addressing the adaptability challenges of Industry 4.0 flexible product lines management. CBE proposes to automatically transform a series of abstract production workflows (the production lots) exhibiting their required capabilities into production plans that select, configure and operate the resources offering matching capabilities. This engineering method responds to the adaptability research challenges mentioned in the previous section. The limitation of current syntactic-only resource matching algorithms has been overcome by implementing semantic interoperability based on ontologies *i.e.*, by transforming AAS-based plant models into MaRCO instances. The main contribution of this part is (1) to refine the CBE architecture from the model design to the execution phase, (2) to implement the automatic capability checking functionality in P4M, and (3) to enable the remote process execution.
- To enhance the robustness, the MAPE-K loop appears to be an accurate method to support a self-adaptive system. The implementation of this feedback loop incorporates semantic computing techniques during model execution. Semantic computing goes beyond traditional computing methods by giving data a more profound, contextual meaning, leading to smarter decisions by the system. Specifically, we utilize runtime semantic annotations to attach raw data with the ontology concepts dynamically in RDF format. RDF (Resource Description Framework), as defined by W3¹, is a standard model for data interchange on the Web, which has features that facilitate data merging even if the underlying schemas differ. Moreover, the annotated data act as the input for RDF stream processing, which allows the continuous query and manipulate of semantic data streams. This combination of semantic annotations and RDF stream processing improves the system's self-awareness of the performance in real-time.
- In order to leverage the previous contributions, a model-driven capability-based self-adaptive system architecture has been proposed. The architecture integrates the MAPE-K feedback loop to the previously mentioned CBE architecture. It provides a methodology to formulate such an AAS-based digital twin manufacturing system from the specification phase until the operation and maintenance phase.
- An academic demonstrator is built to showcase and validate the approaches proposed for the research problems. My contributions of developing this demonstrator consist

¹<https://www.w3.org/RDF/>

of (1) identifying an accurate production process scenario for the demonstrator, (2) designing functional digital twin models for the components and the process, (3) automatic deployment and orchestration of functional digital twin models to execute the real-world devices.

1.4 Manuscript Plan

The introduction about the general context and research problem has been mentioned early in this chapter. The rest of this dissertation is organized as follows.

As introduced in the previous section, in this thesis, we have integrated and implemented many technologies and concepts. Chapter 2 compares the different existing technologies and alternative solutions, including the MDE & industrial digital twin implementation, capability- and skill-based engineering, ontology and metamodel for semantic digital twins, and self-adaptive cyber-physical systems. This chapter investigates the state-of-the-art of the above topics and adjusts the reason for the methodology and technology selection for the implementation.

Chapter 3 presents an extension architecture of the capability-based engineering method with a closed feedback loop in order to construct an interoperable and self-adaptive manufacturing system. The architecture follows the formal engineering phases [14] from the specification to the maintenance. This proposed architecture provides a conceptual basis for the previous research questions.

Chapter 4 focuses on the architecture implementation, which involves putting the conceptual architecture discussed in the previous chapter into practical applications. This chapter emphasizes the contributions made specifically in the realm of implementing the CBSAM architecture.

Chapter 5 introduces an academic testbed, LocalSEA, dedicated to the testing and validation of our research. This chapter presents the main components and the general scenarios to give insight into the implementation section. Then, an example based on the testbed showcases the full elaboration of the implementation details of the CBSAM methodology.

Chapter 6 discusses and concludes this dissertation.

Chapter 2

Related Work Studies

Contents

| | |
|--|-----------|
| 2.1 Industrial Digital Twin & MDE | 7 |
| 2.1.1 DSLs for Industrial Digital Twin Models | 8 |
| 2.1.2 AAS Implementations | 11 |
| 2.1.3 Asset Connection and Orchestration | 13 |
| 2.2 Capability-Based Engineering | 14 |
| 2.3 Semantic Digital Twin | 15 |
| 2.3.1 Manufacturing Resource Capability Ontology (MaRCO) | 17 |
| 2.3.2 OML Adapter | 17 |
| 2.4 Self-Adaptive CPS | 18 |
| 2.4.1 Property Value Statement | 18 |
| 2.4.2 RDF Graph Stream Processing | 20 |
| 2.4.3 Fault detection methods | 21 |
| 2.5 Conclusion | 22 |

2.1 Industrial Digital Twin & MDE

The convergence of information technology (IT) and operations technology (OT) in the modern factory is the basis of the Industry 4.0 [1] revolution. This convergence of the traditionally separate domains of IT (focusing on data, computing, and communications) and OT (focusing on controlling and monitoring physical processes) facilitates seamless data exchange and real-time analytics. The fact that machines and production systems can communicate directly with business-level IT systems, provides the basis for more flexible and customized production processes. As presented in [15], the digital twin technology introduces fresh possibilities for managing and controlling systems of growing complexity by establishing the technical groundwork for Industry 4.0. Grieve presented in [16] a brief history of the digital twin from a concept without a name to a widely spread paradigm.

However, digital twins are interpreted and understood differently by different individuals and research groups, who view and define digital twins in terms of the specific requirements and perspectives of their respective domains. In a thorough survey [17], the authors provide an extensive exploration of digital twins, compiling various definitions and essential characteristics from literature, along with the range of applications developed across different fields. This work serves as a vital reference for understanding the diverse aspects and practical implications of digital twins in technology and innovation. In this dissertation, it is important to select a definition of the digital twin that is closely

related to the area of focus. According to the definition of French AIF¹ (Industry of the Future Alliance), the definition of digital twins are:

- *A digital twin is an organized set of digital models representing a real-world entity designed to address specific issues and uses.*
- *The digital twin is updated in relation to reality, with a frequency and precision adapted to its issues and uses.*
- *The digital twin is equipped with advanced operating tools, including the ability to: understand, analyze, predict, and optimize.*

The use of industrial digital twins can help companies optimize their operations, reduce downtime, and improve overall efficiency. As the complexity of the system increases, the transformation of the model itself, as well as the integrability, connectivity, and scalability between models become crucial. The Model-Driven Engineering (MDE) paradigm [18] is an approach first raised in software development that emphasizes the use of models to design and build systems. Under this premise, MDE, which revolves around abstract models and focuses on alignments and transformations between models, brings new potential to digital twins. Therefore, MDE is now imposing itself as an essential direction in the field of digital twins. The unique characteristics of connectivity and extensibility between models ease the design and maintenance of digital twin systems. With the rise of the internet of things (IoT) and sensor network technologies, real-time data from physical devices and business information have been incorporated into the scope of the model. At the same time, the modeling of the asset's entire life cycle at the core of the digital twin also brings self-adaptation and autonomy from design to operation to model-based systems engineering (MBSE) [4].

Models@run.time [5] refer to the execution of models at runtime, which means that the models themselves are used to guide the behavior of a running system. This concept is first addressed by the MDE community to extend the use of software models. With models@run.time, models remain active and executable during runtime, allowing for dynamic adaptation and reconfiguration of the system's behavior. Models@run.time enable various capabilities, including dynamic reconfiguration, self-healing, self-optimization, and context-aware adaptation. These capabilities are particularly valuable in dynamic and complex systems, such as cyber-physical systems, IoT applications, and distributed systems. [19] explore the use of MDE in developing smart Cyber-Physical Systems (CPS), particularly sustainability systems like smart grids and cities. They emphasize the integration of engineering and scientific models to address the complex challenges in these systems, which involve balancing social, environmental, and economic factors. In the context of industrial digital twins, MDE can be used to create and maintain the digital twin model throughout the asset's lifecycle. Bordeleau et al. have analyzed the application of MDE technologies to digital twins in [20]. This approach can help companies to reduce the time and cost of developing and maintaining digital twins, while also improving their accuracy and reliability.

2.1.1 DSLs for Industrial Digital Twin Models

The domain-specific modeling language (DSL) is always an important topic in the MDE domain as mentioned by Wortmann *et al.* [21]. In MDE, DSLs help to create models that are more expressive and easier to understand by both domain experts and machines, thus increasing the interoperability of the system. The drive towards Industry 4.0 and the novel concept of the Asset Administration Shell have invited professionals across industries

¹<http://www.industrie-dufutur.org/>

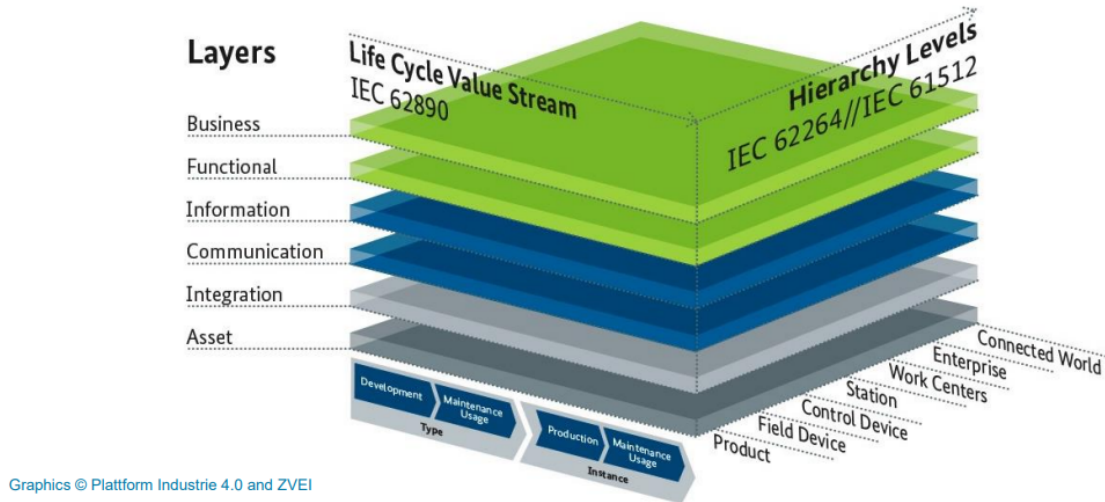


Figure 2.1: RAMI model (reproduced from: Standardization Council Industrie 4.0)

and the scientific community to put efforts towards harmonizing digital representations in the domain. These efforts have led to standardization activities like - the development of the RAMI (Reference Architecture Model Industrie 4.0) [22] and the Details of the Asset Administration Shell (AAS) [10], which helps in providing the cornerstones while developing the AASs for representing any considered ecosystem of Cyber-Physical Systems (CPS).

The Reference Architecture Model for Industry 4.0 (RAMI 4.0 see Figure 2.1) [22] is the first reference architecture model to accurately describe the Industry 4.0 (I4.0) components. It enables I4.0 stakeholders (architects, developers, business decision-makers, etc.) to adopt a common perspective and a common understanding of I4.0 systems, resources, and assets. RAMI 4.0 encompasses the most important aspects of I4.0 components, making them a worldwide identifiable participant able to communicate, through its virtual representation and an asset. The former reflects the asset in the five upper layers of RAMI (integration, communication, information, function and business processes).

The Asset Administration Shell (AAS) [10] is defined as *a standardized digital representation of the asset*. It identifies the Administration Shell and the assets represented by it, holds digital models of various aspects (submodels) and describes technical functionality exposed by the Administration Shell or respective assets. Part of the AAS metamodel is illustrated in Figure 2.2, and Table 2.1 gives definitions of some important concepts of AAS that are involved in this thesis. The Submodels are composed of Submodel Elements. These elements can be product properties, process variables and parameters, events for observing properties, references to external data sources or files, capabilities, operations and entities of the composite I4.0 component. The relationships among AAS components are defined in [23], including the composition of entities in a bill of material submodel and the property connections established between entities. It defines the architecture for a digital twin, including the structure of information models and the communication interfaces used to connect different components. Many organizations' efforts on the AAS implementation make it more competitive to dominate the domain. The related works about the implementation of AAS will be presented in 2.1.2. By using AAS, companies can create a standardized and syntactic interoperable way to represent and communicate information about their assets. This helps to facilitate the development and integration of digital twin technologies in an industrial context.

The AASs can be placed at any part of the RAMI 4.0 model cube in Figure 2.1.

| AAS | Definition |
|---------------------------|---|
| Asset | Physical or logical object owned by or under the custodial duties of an organization, having either a perceived or actual value to the organization |
| AssetAdministration Shell | Standardized digital representation of the asset, the cornerstone of the interoperability between the applications managing the manufacturing systems. It identifies the Administration Shell and the assets represented by it, holds digital models of various aspects (submodels) and describes technical functionality exposed by the Administration Shell or respective assets. |
| Submodel | Models which are technically separated from each other and which are included in the asset administration shell. |
| SubmodelElement | Submodel elements include data properties as well as operations, events and other elements needed to describe a model for an asset. |
| SubmodelElementCollection | A submodel element collection is a set or list of submodel elements. |
| ConceptDescription | The definition of a concept |
| Property | Defined characteristic suitable for the description and differentiation of products or components. |
| Operation | An operation is a submodel element with input and output variables. |
| Capability | Implementation-independent potential of an Industrie 4.0 component to achieve an effect within a domain. |
| Entity | An entity is used for modeling composite AASs. An entity references an asset and has relationships with other entities. |

Table 2.1: Excerpt of AAS metamodel V3RC2 definition from [10]

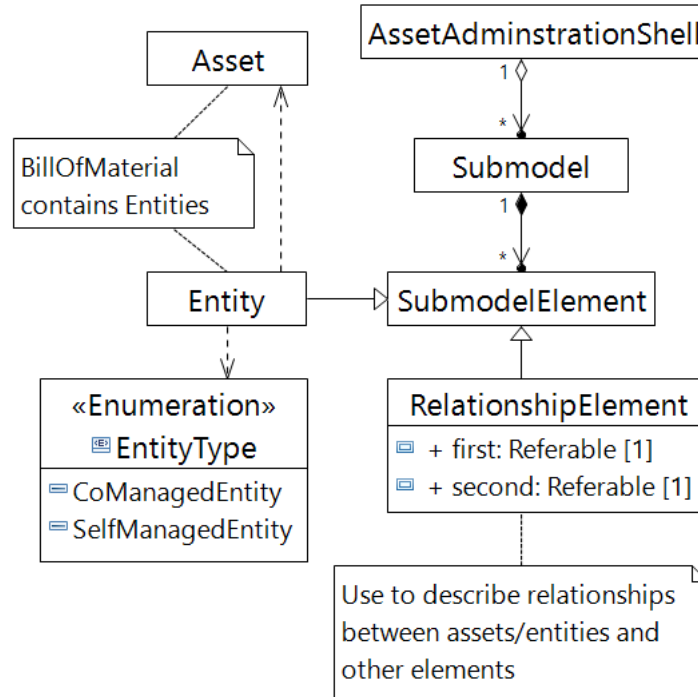


Figure 2.2: AAS composite metamodel

Considering the perspective of the vertical axis (layer), by definition, the assets correspond to the asset layer, and the asset administration shells realize the integration of assets and can extend upward to include the information of each layer. Then from the perspective of the left horizontal axis (Life Cycle & Value Stream), an asset can be defined as a type or an instance, which also conforms to the RAMI 4.0 model. Finally, from the perspective of the right horizontal axis (Hierarchy Levels), since each I4.0 component can have its own asset administration shell and the composable characteristics of AAS, the AAS model can represent any level of “entities” in the axis.

2.1.2 AAS Implementations

A detailed overview of RAMI 4.0 and AAS is given in [24]. And [25] provides an in-depth explanation of the concepts defined in AAS and gives suggestions to Industry 4.0 stakeholders. Open source AAS implementations (i.e. BaSyx [?], NovAAS [26]) provided by different organizations are discussed in [27]. NovAAS contributes towards a web-based reference implementation of the concept asset administration shell. Now researchers tend to integrate OPC UA Information Models to the AAS skills descriptions; [28] shows a mapping between AAS concepts and OPC UA information models. In survey [29], the authors listed all the open-source implementations of the AAS standard such as the modeling frameworks and middlewares, which provide the ability to connect AAS models with their physical assets.

The advancements are not just limited to the reference architectures and standards but are also seen in the various implementations of the concept of AAS. These projects include not only the development of Software Development Kits (SDKs) such as BaSyx [?], but also some implementation efforts like *Fraunhofer Advanced AAS Tools (FA³ST)* [30], *AAS Package Explorer* [31], *SAP I4.0 AAS* [32] and *NovAAS* [26].

Eclipse BaSyx provides an execution infrastructure for AAS models, but it doesn’t support a ready-to-use HMI (Human-Machine Interface) tool for non-tech-savy users. In-

stead, it provides software development kits in Java, .Net Core, and C++ [33]. The SDKs act as the basis for creating applications where information is modeled and transferred using the standards of the AAS.

FA³ST [30] is a Java-coded service-oriented tool. It includes a predefined implementation for HTTP and OPC UA-based protocols endpoint, JSON serializer and deserializer, file and database-backed persistence manager, as well as MQTT and OPC UA-based asset connections. A FA³ST service can be deployed either as a Java JAR file or as a Docker container. Compared to BaSyx, FA³ST provides more features, such as the integration with Apache StreamPipes (a toolbox for Industrial IoT with a focus on stream processing) as well as with the international data spaces (IDS).

The AASX Package Explorer (AASX PE) is a user-friendly application with a GUI which provides working tools and components for the creation of AAS models based on the specification [10] [31]. As AASX PE is a tool most used today for the specification of AAS models, many middlewares offer the possibility of being configured with .aasx files for execution.

Both the FA³ST service and AASX Package Explorer offer an HTTP and an OPC UA-based service endpoint; however, in the AASX Package Explorer, they are not synchronized, meaning that changes to the DT via one type of endpoint are not reflected in the other. The AASX PE provides some functionality beyond the specification and is not implemented by FA³ST Service, such as HTTPS/SSL MQTT endpoint and a graphical user interface. However, connecting the DT to existing assets is limited to OPC UA.

On the other hand, the SAP I4.0 AAS, implemented in JavaScript, TypeScript, and Go, offers a GUI for describing an asset as per the standard of AAS. NovAAS [26] is a Node-RED-based implementation of the AAS specification [33]. It has a strong focus on JSON, HTTP, MQTT, and usability, e.g., it provides user management and a dashboard to visualize live data. Still, it does not address essential parts of the specification, such as different data formats or OPC UA. Moreover, NovAAS is realized using only Node-RED so it is less capable to integrate with other systems.

An aspect that lacks in all of the above-mentioned tools for the creation of the AAS is the model-driven approach. The CEA has initiated the development of an open source model-driven toolset for the AAS specification. This toolset Papyrus4Manufacturing² (P4M), extends the UML modeling tool Papyrus [34] to meet the AAS specification. In addition to the AAS models creation, P4M facilitates the deployment of these models by automatic code generation. The generated executable code from the AAS models permits the establishment of communication between the digital twin and its physical counterpart. The above features are detailed in [35]. In light of the benefits that model-driven development approaches bring, it is apparent that a model-driven development approach to the AAS is also necessary. The tool Papyrus4Manufacturing [36] provides exactly the needed MDE approach to AAS. P4M was developed on top of the UML modeling tool, which provides features restricted to the standard - Details of the Asset Administration Shell, in order to provide a graphical modeling environment that helps to model an AAS.

Moreover, while Roth and Rumpe (2015) [37] mention that code generators are an integral part of any model-driven development process, Höllder et al. [38] say that constructive generation or synthesis of code from the models needs to be among the first steps of a model-based development process implying the necessity of code generation. Thus, during the development of P4M, the feature of code generation from the models developed using the UML-based modeling tool was taken into account. The BaSyx Java SDK [39], an open-source middleware for the implementation of AAS, was used as the code generation target. BaSyx stands out for its suitability as a starting point due to its executable

²<https://eclipse.dev/papyrus/components/manufacturing/documentation.html>

features and compatibility with the AAS standard. More description about the BaSyx project will be given in Section 2.1.3. Moreover, it is important to note that the selection of BaSyx does not limit the generality of the MDE approach, which means that although BaSyx is the initial tool of choice, our approach remains compatible with other tools.

Table 2.2: Tools implementing AAS standard

| Tool | HMI | MBSE | Asset Connection | AAS Execution |
|--------------------|-----------------|------|------------------------------------|---|
| AASX PE | Yes | No | OPC UA | Yes (with different middlewares) |
| NOVAAS | Yes (web based) | No | OPC UA/ HTTP/ MQTT | Yes |
| SAP | Yes (web based) | No | No | Yes |
| FA ³ ST | No | No | OPC UA/ HTTP/ MQTT | Yes |
| P4M | Yes | Yes | OPC UA/ HTTP/ MQTT/ WebSocket/ ROS | Yes (automatic code generator to BaSyx) |

Table 2.2 illustrates a comparison of the previous related works, i.e. tools implementing the AAS Standards. We can see that all the tools are implementing the AAS standard and propose an execution of the AAS model by manually creating the software code. However, only P4M toolset is (1) providing the automatic generation of the executable code from the AAS models and (2) ensuring the synchronization between the AAS models and the executable code. One of the contributions of the thesis is proposing a model-driven toolset for the AAS specification in the context of Papyrus4Manufacturing. The work extends Papyrus4Manufacturing to enable (1) modeling composite AASs (2) modeling and executing production processes.

2.1.3 Asset Connection and Orchestration

Eclipse BaSyx [?] is one of the first implementations as an execution infrastructure of digital twins using AAS. Choosing Eclipse BaSyx as a middleware for Industrie 4.0 applications is driven by its comprehensive support for its alignment with open standards. As an open-source platform, BaSyx embodies the principles of the fourth industrial revolution by facilitating inclusive participation from businesses of all sizes, research institutions, and academia. Its implementation of the AAS as a standardized digital twin, exemplifying its commitment to industry standards. Moreover, its versatile connectivity options, including MQTT³ (Message Queuing Telemetry Transport), OPC UA⁴ (OPC Unified Architecture), S3 cloud technology⁵, and PLC4X⁶ (under development), enable seamless integration with various entities ranging from edge devices to PLCs to cloud-based systems and more. By offering reusable Industrie 4.0 components and SDKs in multiple programming languages, BaSyx significantly streamlines the development of Industrie 4.0 solutions, enhancing efficiency, minimizing downtime, and reducing costs. This makes BaSyx an ideal middleware choice for future development.

The article [40] focuses on evaluating the performance and scalability of AAS when integrated into the BaSyx environment through experiments.

³<https://mqtt.org/>

⁴<https://opcfoundation.org/about/opc-technologies/opc-ua/>

⁵<https://aws.amazon.com/fr/s3/>

⁶<https://plc4x.apache.org/>

BPMN (Business Process Model and Notation) [11] is used in process orchestration design due to its standardized, widely understood notation, which facilitates clear and consistent documentation of business processes. Its ability to visually represent complex workflows, makes it a valuable tool for simplifying and understanding complex processes. Despite the criticisms for its complexity and visual semantic ambiguity, BPMN is still generally considered as a representation enhancing the communication between different stakeholders, bridging the gap between business analysts, process designers and IT professionals, ensuring business and IT alignment. BPMN's flexibility enables it to model a wide range of business processes and adapt to changing needs, while its compatibility with business process management systems simplifies process automation and execution.

However, BPMN diagrams themselves are not executable. However, being a digital twin of a production process requires for orchestrating the digital twin at the application level. For the implementation in this thesis, Node-RED will serve as an execution engine for workflows expressed in BPMN. Node-RED⁷ provides a visual programming interface that simplifies the creation and management of complex workflows. This process-based development tool is particularly user-friendly, allowing programmers and non-programmers alike to easily configure and link different data sources and services. Its broad support for various integrations makes it highly adaptable to different IoT ecosystems, a key aspect of digital twin technology that requires seamless interaction with a large number of sensors, devices, and systems. The open source nature of Node-RED provides rich resources and ongoing enhancements, supported by a strong community, enhancing its reliability and scalability. The authors of [41] propose a novel workflow manager built on Node-RED. This proposed solution enables the Node-RED to load and execute business processes using BPMN recipes.

2.2 Capability-Based Engineering

Under the concept of AAS, capability-based engineering (CBE) is a method to realize flexible production. The idea is proposed by Plattform Industrie 4.0 in [13], where they stated three fundamental elements in a production practice are resource, product, and process, therefore it is important to build accurate AAS models for them. Processes use resources to build products and in the I4.0 vision processes express the capabilities required to build the products while resources express the capabilities they are offering. The main goal of CBE is to design, implement and operate the system according to the functionalities required in each step of the production process, rather than explicitly specifying the actual production resources. The aim is to abstract and decouple the production process and its requirements with the resources. Given accurate representations of processes and resources, CBE aims at automatically finding available resources for a process by matching their offered capabilities with the required ones.

First and foremost, it is essential to define some key concepts, as the understanding of terms like “capability” and “skill” can vary across different fields. For instance, in the referenced work [42], the author highlights that there is often confusion among many individuals regarding definitions, such as mixing up “skill” with “task”. The author's definition is completely different from our understanding, which confirms that there is a bit of arbitrariness in the definition of terminology. Therefore, before continuing, we need to emphasize again the definitions of the terms to be used. In this paper, we will adhere to the naming conventions and definitions as outlined in [13] and [43].

- **Capability:** The implementation-independent description of the function of a resource to achieve a certain effect in the physical or virtual world.

⁷<https://nodered.org/>

- Skill: The asset-dependent implementation of the function of a resource to achieve a certain effect in the physical or virtual world
- Service: A service specifies the means of provision of one or more capabilities offered by a service provider to a service requester and extends its description with commercial aspects.

As described in [13], a *Capability_Submodel* can be used to describe the abstract functionality of a resource. The SubmodelElement *Capability* is used to define the abstract functionality that the digital twin possesses or the production processes requirement. The *Skill* describe the concrete implementation. The CBE is divided into three main phases.

- Capability Checking: this phase aims to automatically match the resource with the designed production process by their semantic function description without explicitly assigning the resource. Ontology is a method to construct semantic descriptions for domain concepts and attributes. Ontology concepts can be connected and combined using semantic rules, so reasoners will infer possible production plans for required/provided capabilities matching.
- Feasibility Checking: a procedure to bring in environmental factors and constraints to validate the production plan. The feasibility checking is supposed to take into account operational, environmental or more global constraints which are not directly linked to the resources but which can lead to the preference of one resource. This will further determine the feasible parameters that meet the conditions or avoid possible faults (i.e. collision detection, trajectory calculation) before the deployment on the real production line.
- Skill Execution: puts the resources into operation according to the validated production work plan. The production plan will be executed according to the process orchestration design.

In article [44], the authors aim to provide a structured overview of skills-based manufacturing research, conducts a literature survey of ETFA (IEEE Conference on Emerging Technologies and Factory Automation) contributions over the past decade, and analyzes 34 relevant papers to explore the current status, consensus and research gaps in capabilities and skills. A significant increase in related research articles after 2019 is stated, which shows the interest of researchers on this topic. Among them, the ontology widely discussed to express and create holistic models.

In conclusion, we want to use appropriate capability description ontology and technologies through this thesis to develop a framework that supports capability engineering to solve the adaptability challenge mentioned in the introduction.

2.3 Semantic Digital Twin

Knowledge and ontology are closely related concepts in the field of knowledge management, where ontology refers to the study of the nature of existence and the relationship between different entities, while knowledge more generally refers to human understanding of the world. Ontologies provide a framework for organizing and classifying knowledge for accurate consistency and semantics. At the same time, knowledge can make ontologies better relevant to real-world problems. The Knowledge Graph serves as a powerful driver for the adoption of Semantic Web standards and all the semantic technologies that implement them. The Knowledge Graph improves data management and content management to new levels of efficiency by introducing semantic metadata, and breaks down silos, allowing them to interoperate with various forms of data and knowledge.

As stated in [45], *A language ideal to represent phenomena in a given domain if the metamodel of this language is isomorphic to the ideal ontology of that domain, and the language only has as valid specifications those whose logical models are exactly the logical models of the ideal ontology.* Ontology and metamodel provide a framework for understanding and constructing complex systems. A metamodel can be considered an instantiation of an ontology, providing a concrete representation of the concepts and relationships within a specific domain. In this way, ontology and metamodeling are complementary approaches to understanding complex systems. The work of [46] describes combining knowledge and model-driven methods to complement each other.

Applying semantic knowledge to represent and reason about the data associated with an asset allows for more advanced data analytics, which can lead to insights and optimizations that might not be possible with traditional digital twin technologies. By combining ontology technologies to model construction, we can bring digital twin systems a formal description of the concepts and relationships, which can help to facilitate the sharing and integration of data across different systems and organizations.

Semantic interoperability has long been recognized as a major concern in the field of industrial digital twin systems. This subsection introduces this problem, and then leads to two related works that will be reused in our solution.

The Digital Twin Consortium published a whitepaper [47] on the digital twin system interoperability framework. It introduces seven interoperability concepts that frame the design considerations necessary to make systems interoperate at scale. The article [48] introduces the definition of semantic interoperability in the context of Industry 4.0 and Smart Manufacturing as follows: “Semantic interoperability enables systems to interpret meaning from structured data in a contextual manner. Semantic interoperability relies on ontology-based “contextual metadata” supplementing “data” to form “information” exchanged among connected systems. This ontology must account for metadata exchanged between disparate systems and environments. It represents the highest level of interoperability between connected systems - beyond syntactic interoperability”.

[49] provides an overview of various articles and applications of data analysis, expert knowledge, and knowledge-based system drivers in production systems. On top of that, it describes how to use “data analysis” in a production system to create knowledge-based digital twin systems. [50] articulates new concepts of value creation through the use of digital twin decision support services in industrial service ecosystems, and discusses mixed semantic modeling and model-based systems engineering for their implementation. A customizable conversion system for converting ABB Ability™ digital twins to Asset Administration Shell format is presented in [51], showing a real-world example for interoperability in industrial environments. A. Perzylo et al. [52], introduces concepts developed by the BaSys 4.0 initiative dealing with the semantics of manufacturing skills, orchestrating higher-level skills from basic skills, and using them in a cognitive manufacturing framework.

Ontologies bring to systems engineers and researchers the high value of semantic interoperability and makes them aware of the importance of combining ontology vocabularies with system model design. [53] introduced an approach of a dynamic mapping of the ontology vocabularies into system models stereotyped by meta-classes defined in a profile. This approach enriches the semantic meanings of system modeling without affecting the definition of existing meta-models.

According to the above work and many other articles not mentioned, the use of ontologies to solve semantic interoperability appears as a common solution in the field. Our idea is to combine ontology-based knowledge representation with the AAS digital twins to achieve the semantic interoperability between digital twins. In our work, we overcome the limitation of current syntactic-only resource matching algorithms by implementing semantic interoperability based on ontologies *i.e.*, by transforming AAS-based plant mod-

els into ontology instances and then query the expanded ontology to find the needed resources. To achieve this, we rely on two former works described next: MaRCO [12] provides capability-related ontology for manufacturing systems, and the OML Adapter [54] provides a transformation basis from OWL ontologies to OML and UML models.

2.3.1 Manufacturing Resource Capability Ontology (MaRCO)

Ontologies are widely accepted for knowledge representation in specific domains. In [13], C4I ontology [55] is given as an example for the semantic representation of capabilities. C4I is designed with a broad focus, encompassing a wide spectrum of industrial capabilities, processes, resources, and products. This generality makes it applicable across various manufacturing scenarios. It plays a crucial role in enhancing interoperability among diverse systems in industry, facilitating effective communication and data exchange.

The OWL-based Manufacturing Resource Capability Ontology (MaRCO) [12] is used to describe the capabilities of manufacturing resources. MaRCO's strength lies in its precision and granularity in modeling the capabilities and limitations of manufacturing resources, which is critical for complex and advanced manufacturing processes. The expressive power of MaRCO supports the representation of simple resources but also their combination into collaborative resources, hence a good candidate for capability-based engineering. In addition, MaRCO is also provided as a complete capability matchmaking web service [56]. While its implementation language OWL has good knowledge representation features, pure OWL [57] is limited when it comes to querying. To effectively support semantic-based resource selection, SPARQL (SPARQL Protocol and RDF Query Language) [58] has been chosen to implement the capability matchmaking rules. More precisely, SPARQL allows to write queries that combines the capability parameters of several resources to select sets of compatible and covering resources for given requirements. Finally, the use of SPIN (*SPARQL Inference Notation*) allows to represent SPARQL queries as knowledge within the ontology and then the SPIN API allows to make inferences and generate new individuals within the ontology.

2.3.2 OML Adapter

The OML (Ontological Modelling Language) [54] is defined by the openCAESAR⁸ platform, which is also an ontology description language inspired by OWL and SWRL (Semantic Web Rule Language). OML is a modeling language designed for ontologies, which aims to close the gap between modeling and programming languages. It offers a structured framework for representing ontologies with complex relationships and hierarchies in a machine-readable format.

Implemented using the Eclipse Modeling Framework (EMF), OML benefits from a robust Java API, enhancing its accessibility and ease of integration within a variety of software development environments. This integration with EMF not only facilitates the creation and manipulation of ontology models but also ensures compatibility with a wide range of existing tools and libraries in the Java ecosystem.

Moreover, the utility of OML is further extended by tools such as the OML Adapter provided by the openCAESAR project. This adapter enables the round-trip transformation between OML and UML (Unified Modeling Language), allowing for greater flexibility and interoperability in modeling practices. This feature is especially beneficial for projects that require the integration of ontological models with MDE methodologies.

In the subsequent of this dissertation, more detailed insights into the process of converting between OML and UML are provided, illustrating the practical applications of this transformation in our platform. This discussion will include examples of how OML has

⁸<http://www.opencaesar.io/oml/>

been effectively utilized within our platform, demonstrating its utility in bridging the divide between abstract ontological concepts and their practical implementation in software solutions.

2.4 Self-Adaptive CPS

Self-adaptive systems [9] are characterized by their ability to autonomously adjust and optimize their behavior based on dynamic changes in environmental and internal conditions. Self-adaptivity is an important subject in complex systems, making it particularly relevant for cyber-physical systems (CPS), where maintaining the robustness is often crucial.

Research in self-adaptive manufacturing systems is paving the way for more flexible, efficient, and resilient manufacturing processes. In the realm of self-adaptive systems, one significant approach presented in [59] utilizes a multiagent framework. This approach is particularly geared towards addressing the challenges inherent in the realization of self-adaptive systems, where multiple agents collaborate and adapt to changing manufacturing conditions and requirements. Furthermore, the work presented in [60] introduces an innovative method for the design of knowledge structures in Cyber-Physical Production Systems (CPPS). This approach aims at minimizing or avoiding adverse impacts on cognitive activities, thus enhancing the system's ability to predict and mitigate disruptions before they escalate. Seiger et al., in their study [61], delve into the potential of a self-adaptive workflow framework within the context of Cyber-Physical Systems (CPS). Their framework is built upon the MAPE-K (Monitor, Analyze, Plan, Execute over a shared Knowledge) feedback loop, a well-known concept in the field of autonomic computing, which is instrumental in enabling systems to self-manage and adapt to changing conditions. Additionally, the research conducted by Ma et al. [62] brings a new perspective with an executable model-based approach. This approach is specifically tailored to test self-healing behaviors in CPS under a variety of environmental uncertainties. The model-based nature of this approach allows for a systematic and thorough examination of how systems can autonomously detect, diagnose, and repair faults, thereby ensuring continuous operation.

By the following subsections, the Property Value Statements (PVS) metamodel will be analyzed in order to facilitating effective and semantic data exchange between systems. Then an explore of semantic reasoning over data streams by the RDF stream processing will be presented. Follows with a study of existing fault diagnosis methods for ensuring the reliability and resilience of self-adaptive CPS.

2.4.1 Property Value Statement

The property value statement (PVS) proposed in DIN SPEC 92000 [63], standardized the exchange of values and statements on properties in Industry 4.0 applications. The metamodel of PVS is shown in Figure 2.3.

- **SubjectID:** points to the subject where this property belongs.
- **PropertyID:** refers to the definition of this property.
- **PredicateRelation:** relation between the comparative value in an expression and the effective value.
- **PredicateValue:** comparative value the property magnitude expression refers to.
- **ExpressionSemantic:** describes the context and purpose of the statement that defines the meaning of the logic expressed by the predicate. Important categories of

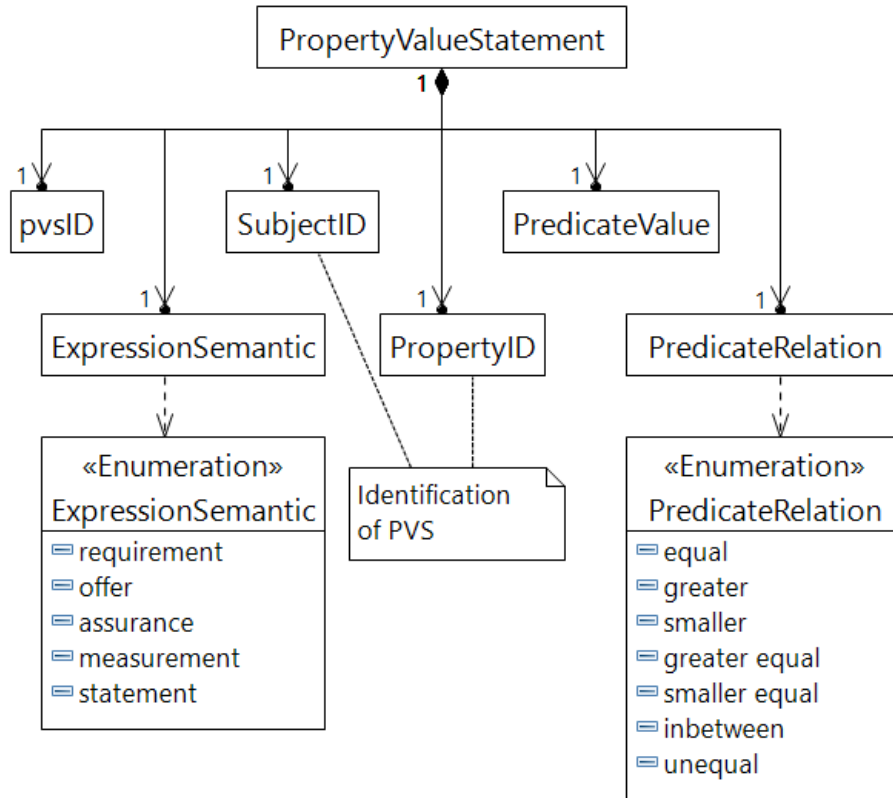


Figure 2.3: Property value statement metamodel

expressions are, for example, assurance (A), requirement (R), offer (O), statement (S), measurement (M).

- Assurance: statement that, in case of a realization, the actual value will be set in predicate relation to the predicate value.
- Requirement: statement that the effective value shall be set in predicate relation to the predicate value.
- Offer: statement that, during the realization, the actual value can be set in predicate relation to the predicate value.
- Statement: expression about the value of a property.
- Measurement: actual value measurement.

In [64], the concepts of property value statement are presented which fit to the standardized property definition models and build a platform for a metadata-based information handling in the digital factory.

More generally, capability-based engineering has roots into the concept of reflection [65]. Smith [66] introduced the seminal concept of reification as a causally connected representation of a program and its execution state *i.e.*, a digital twin used by the program to reason about and act upon itself. This concept so much permeated over the last 40 years that it is impossible to provide here a fair account but just point a few of the most relevant works. The primary goal of reflection is to dynamically adapt systems, especially to meet quality of service (QoS) objectives. In this very active area, PVS-like statements match required QoS of systems to offered QoS of their building blocks. The language QML [67], for example, builds QoS contracts constraining measures of QoS against requirements both individually and as probability distributions. This kind of approaches has been

developed in networks, service-oriented architectures (SOA), robotics, etc. Orchestrating services given their QoS in SOA has been developed, for example, by [68], and even as the selection, at call-time, of the service among alternatives with the best current QoS treated as a multicriteria decision problem [69]. In robotics, [70] develops a constraint-based composition logic for component-based architectures where PVS-like statements are attached to rich interfaces both to statically prove the correctness of assemblies and to get configuration settings respecting time and QoS constraints. The paper [71] focuses on three critical aspects: ensuring functionality through semantic matching, optimizing Quality-of-Service (QoS) measurements, and adhering to global QoS constraints. Their experiments, based on the Web Service Challenge 2009 dataset, demonstrate the approach's ability to efficiently identify optimal or near-optimal solutions while improving fitness and execution time. [72] proposes a resource-conscious framework to plan autonomous robotic missions that attacks capability-based engineering in a way very similar to RAMI 4.0, albeit not using this standard.

2.4.2 RDF Graph Stream Processing

Data streams are one of the main sources of information in the digital twin applications, and it is required to make these streams available in a machine interpretable manner. For this to become a reality, there is a need for the definition of standards and guidelines on how to produce and consume structured data streams. The Resource Description Framework (RDF) [73], a World Wide Web Consortium (W3C) recommendation, plays a significant role in structuring and representing data on the Web. RDF effectively organizes data in a triple format (subject-predicate-object), enabling a flexible, graph-based data structure that is ideal for linking and integrating diverse data sources.

However, the traditional RDF model is aligned with the persistent data paradigm and focuses primarily on managing a fixed set of data elements in a static knowledge base. The knowledge base remains relatively stable over time, allowing for efficient retrieval, querying, and maintenance. However, the static nature of RDF contrasts with the dynamic and fluid nature of data flows.

In digital twin applications, data elements are not static but flow continuously, creating an infinite and constantly evolving sequence of information. This continuous flow raises a great challenge for data management and processing, as traditional methods of storing and querying static data sets are insufficient. Addressing this gap requires innovation in data representation and processing models. In this context, the W3C RSP Community Group [74] has taken the task to explore the technical and theoretical proposals that incorporate streams to the RDF model and to its query language. Approaches such as streaming reasoning, which extends traditional reasoning techniques to handle dynamic data. This allows for the real-time processing of streaming data, enabling information to be interpreted and utilized immediately as it is generated. These advances are critical to realizing the full potential of digital twin technology, where timely, accurate data is key to mirroring and optimizing real-world processes and systems. A survey [75] analyses the growth and traces the result of the RSP community.

RSP4J (RDF Stream Processing for Java) [76] is an open source library⁹ designed for building RDF Stream Processing (RSP) Engines, aligning with the RSP-QL (a reference model for explaining the semantics of RSP dialects and the execution semantics RSP engines). Inspired by the OWL API and other Semantic Web research efforts, particularly in Stream Reasoning, RSP4J aims to foster the application of Semantic Web concepts through practical and user-friendly software tools. The use of existing engines with RSP4J involves integrating well-established RDF Stream Processing engines like CSPARQL [77]

⁹<https://github.com/streamreasoning/rsp4j>

and CQELS [78] into the RSP4J framework. This integration allows for leveraging the unique features and capabilities of these engines within the RSP4J environment. [79] highlights the challenges in stream reasoning (SR) due to a lack of standardization among SR engines. It focuses on how the RSP-QL model in the RSP4J framework aids in standardizing SR semantics. A survey comparing SR engines on performance and configurability shows RSP4J implementations outperforming CSPARQL. The need for further improvement in SR engines is also noted.

2.4.3 Fault detection methods

Fault detection determines the occurrence of fault in the monitored system. It consists of detection of faults in the processes, actuators and sensors by using dependencies between different measurable signals. Related tasks are fault isolation and fault identification. The task of fault diagnosis consists of the determination of the type of the fault, with as many details as possible such as the fault size, location and time of detection. Iqbal et al. [80] present a novel approach to fault detection and isolation (FDI) in automotive instrument cluster systems, focusing on computer-based manufacturing assembly lines. Traditionally limited to simple boundary checking, this paper introduces an automated FDI method based on deep learning. The method is capable of diagnosing and locating multiple classes of faults in real-time working conditions, demonstrating superiority over other established FDI methods by effectively modeling spatial/temporal patterns in data from manufacturing systems equipped with local and remote sensing devices.

Fault detection in high-cost and safety-critical processes has become increasingly important, as early detection can prevent the progression of abnormal events. Miljković's survey [81] provides a comprehensive overview of the major methods and the current state of research in this field. The paper categorizes fault detection methods into (1) data and signal model-based methods, (2) process model-based methods, and (3) knowledge-based methods, offering clear definitions of fault, failure, and malfunction. This classification aids in understanding the varied approaches used in fault detection and their respective applications. This article plays a great role in guiding us in the definition of metamodel later.

Knowledge-based methods are increasingly pertinent due to their ability to handle complex, data-intensive environments. Knowledge-based approaches utilize a more dynamic framework, often incorporating advanced technologies like artificial intelligence, machine learning, and semantic web technologies. Chi et al. [82] examine knowledge-based fault diagnosis in the Industrial Internet of Things (IIoT). They discuss the limitations of plain model-based and data-driven diagnosis approaches in the IIoT context, where complexity can increase exponentially due to the high connectivity among devices. Their work advocates for knowledge-based approaches, which use ontologies to improve interoperability and provide high-level reasoning and responses to nonexpert users. This approach is gaining preference over traditional methods in recent IIoT systems, underscoring the importance of constructing effective knowledge bases.

Búr et al. [83] propose a distributed graph query model for runtime monitoring of CPS, using attributed graphs for high-level knowledge representation. Their models are adaptable in continuously evolving environments, extends publish-subscribe middleware like DDS, enabling dynamic creation and deletion of graph nodes and scalable performance in real-time, resource-constrained environments. In the context of aircraft fault diagnosis, Tang et al. [84] explore the construction and application of knowledge graphs. They emphasize the efficiency of fault diagnosis using deep learning and heuristic rules to extract fault knowledge from data, significantly aiding maintenance engineers in accurately identifying faults. Xu et al. [85] focus on ontology-based fault diagnosis methods for loaders, overcoming complexities in fault diagnosis knowledge. Their method integrates ontology

with case-based reasoning (CBR) and rule-based reasoning (RBR), using Semantic Web Rule Language (SWRL) rules to cover shortages of the CBR method, especially when concerned cases are lacking.

In the realm of robotics, Hernández et al. [86] introduce a self-adaptation framework based on functional knowledge to enhance robot autonomy. They integrate a metacontroller on top of the robot control system, using a functional ontology to adapt the control architecture for failure recovery. Bozhinoski et al. [87] present MROS, a model-based framework for runtime adaptation of robot control architectures in ROS. MROS utilizes domain-specific languages and an ontology-based implementation of the MAPE-K and meta-control visions, showcasing benefits in mission execution quality and extensibility across robotic applications. Both [86] and [87] are focused on the self-adaptation of a robot, while the thesis focuses on the self-adaptation of a distributed system of robots constituting a production line.

2.5 Conclusion

By linking theoretical and conceptual insights with practical challenges in flexible production, this section will return to the six research questions listed in the introduction and underscore the necessity of a research approach that bridges this gap. It will highlight how the thesis intends to contribute to both theoretical understanding and practical application in the field. This is a crucial link between the literature review and the subsequent parts of the paper, where the original contributions and solutions are further developed.

RQ1 How can we identify and leverage the current existing standards and technologies that we can use to design a digital twin model for the production plans and the plant resources in a both machine-interpretable and user-friendly way?

Section 2.1 first analyzed the potential of combining Model-Driven Engineering (MDE) with digital twins. This analysis illuminated various advantages of this integration. By leveraging MDE's structured approach to model creation, deployment and maintenance, ensuring they remain accurate reflections of their physical counterparts.

Following this, the focus shifted to Asset Administration Shell (AAS) as the emerging standard for digital twins in the realm of Industry 4.0. AAS offers a structured meta-model for the industrial digital twin, which is an important basis for the syntactic interoperability. An examination of how various research groups are currently implementing AAS is included in this section. Through this analysis, it became evident that there is still a gap in the field - a need for a method that utilizes MDE in conjunction with the expressive meta-model provided by AAS. Such a method would bridge the theoretical aspects of MDE with the practical functionalities of AAS, leading to more efficient to industrial digital twin implementations. To this end, we have included UML class diagrams for the AAS model design, the BPMN process diagram for the process design also the UML structure diagram to show the internal structure of an asset.

The proposed method includes not only the graphical modeling environment for the model design, but also the development of automated deployment functions as a part of this method. These functions are critical for facilitating the creation and deployment of digital twin models, making the process more user-friendly and less time-consuming. This AAS-compliant MDE approach would not only enhance the scalability of digital twin technologies but also make them syntactic interoperable to a wider range of use cases and applications.

RQ2 How can we semantically define the models in order to determine the most appropriate selection of available plant resources that fulfill the production plan requirements?

In Section 2.3, we discussed the crucial need for semantic interoperability and effective communication in the Industry 4.0 context. And analyzed the significant advantages of choosing ontologies for semantic expression in a specific domain.

However, it is essential to select an appropriate ontology to accurately describe a specific domain, in our case, the representation of capabilities of resources in the manufacturing sector. MaRCO, as an ontology for describing manufacturing capabilities, stands out due to its comprehensive taxonomy and the inclusion of built-in SPIN rules for resource-process matchmaking. The joint use of models and ontologies makes it necessary to maintain automatic consistency between the two representations. Therefore, we have analyzed OML as a bridge between ontology and modeling. OML facilitates the integration of the rich semantic structures of ontologies with the practical implementation to the modeling framework.

Despite the existence of many current methodologies, we identified a gap that necessitates a method to integrate the AAS model with ontological semantics. Such integration would endow digital twin models with both syntactic and semantic interoperability. The proposed method aims to leverage the strengths of AAS in modeling the physical aspects of assets and the ontological approach for detailed semantic representation and rich inference features.

RQ3 How to semantically process the monitoring data obtained from the digital twin during execution and diagnose the risks in the manufacturing system?

In *RQ2*, we discussed the semantic expression of ontologies. However, knowledge imbued with semantics, when combined with appropriate rules, can exhibit powerful reasoning capabilities. Traditional knowledge-based reasoning methods are typically suited to static models, which do not align with the dynamic nature of digital twins. This necessitates a reasoning mechanism that can handle not just static data but also adapt to continuously changing data streams. Therefore, in Section 2.4.2, we explored the advantages of RSP and various implementations for the dynamic querying. RSP provides a solution in this regard. It allows systems to process data in a stream format, enabling real-time analysis and reasoning.

Our research aims to the needs of real-time monitoring and analysis for digital twin models in a semantic interoperable manner and also at the metadata level. Such models will not only reflect the current state of the system but also make decisions based on real-time data.

RQ4 What methodologies can be developed to facilitate dynamic re-planning of production lines in a flexible and automated manner?

In Section 2.2, we discussed a CBE method aimed at abstractly conceptualizing the AAS digital twin model of production resources, processes, and products within “capability” at a semantic level regardless the actual implementation. This method begins with automated capability checking through abstract semantic reasoning to align available resources with the designed process, followed by verification through simulation (feasibility checking) and the automated reconfiguration (skill execution). Conceptually, this method enhances the adaptability of the system and has garnered widespread attention and recognition in the field.

However, at the outset of my work, all efforts related to this method were purely conceptual, with no practical implementation of the method in place. We have developed the design time capability checking phase of this concept by integrating ontologies semantics and the reasoning feature within the AAS modeling framework.

RQ5 What strategy can be employed to ensure rapid and efficient system reconfiguration in response to varying operational needs?

Subsequent to the capability checking phase, there's a crucial need for the reconfiguration and orchestration of the digital twin models. In this part of the process, establishing connections from the model to the asset and facilitating communication between models within a system are of paramount importance. Addressing this need, BaSyx was chosen as a middleware solution. Its capabilities extend beyond merely providing connection methods with various industrial protocols to field devices. BaSyx also plays a vital role in deploying AAS models as REST servers, which significantly eases the accessibility of the model by various applications. This deployment strategy is instrumental in integrating digital twins into broader industrial systems, ensuring seamless data flow and interoperability.

On the execution and orchestration side, Node-RED emerges as a user-friendly and potent tool for managing device operations. However, the default nodes provided by Node-RED often require additional effort for the utilization, especially for users who are not specialists in the field. To address this challenge, we have designed a set of customized nodes that matches with the BPMN semantics. These nodes are designed to simplify the understanding and creation of process orchestrators.

RQ6 How can real-time monitoring and diagnosis modules be effectively integrated into existing systems to ensure continuous operation during execution phases?

In Section 2.4, we discussed the significance and feasibility of the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) method for realizing self-adaptive systems. Additionally, we explored various existing methods for system diagnosis.

We have developed a method for rule-based diagnosis that operates on dynamic data streams. The essence of this method lies in its ability to process and analyze data stream, as it flows continuously from various sources within the production system. By focusing on dynamic data streams, the method is capable of capturing the most current state of the system, offering timely insights that are crucial for effective diagnosis.

Another primary contribution is the integration of the MAPE-K approach with previously established methods to form a more comprehensive methodology. This integration culminates in a framework that leverages digital twins to achieve a synthesis of interoperability, adaptability, and robustness, ultimately enabling flexible production.

Chapter 3

Capability-Based Self-Adaptive Manufacturing Architecture

Contents

| | | |
|------------|-------------------------------------|-----------|
| 3.1 | Architecture Overview | 25 |
| 3.2 | Specification | 29 |
| 3.3 | Design | 30 |
| 3.3.1 | AAS Modeling environment | 30 |
| 3.3.2 | Capability Submodel Design | 33 |
| 3.3.3 | Operational Data Submodel Modeling | 35 |
| 3.3.4 | Monitoring Submodel Modeling | 36 |
| 3.4 | Engineering & Deployment | 41 |
| 3.4.1 | Capability Checking | 41 |
| 3.4.2 | Feasibility Checking | 44 |
| 3.4.3 | Skill Execution | 44 |
| 3.5 | Operations & Maintenance | 45 |
| 3.5.1 | Monitoring | 45 |
| 3.5.2 | Analysis | 45 |
| 3.5.3 | Planning | 46 |
| 3.5.4 | Execution | 47 |
| 3.5.5 | Knowledge Integration | 47 |
| 3.6 | Conclusion | 48 |

3.1 Architecture Overview

Whenever we think of future manufacturing, the images that always come to mind are of highly flexible production lines and fully autonomous systems. However, as we mentioned in the introduction, there are still many challenges in the realm of realizing this vision, including interoperability, adaptability, and robustness. This section provides a generic approach for the construction of such a flexible and self-adaptable manufacturing system. Figure A.3 and Figure A.4 illustrate the architectural overview of this capability-based self-adaptation manufacturing (CBSAM) approach. The above problems presented in Section 1.2 can be tackled through this architecture with the integration of the various advanced approaches and methods that have been chosen in the previous section.

Tailored for the manufacturing system under the Industry 4.0 context, the CBSAM architecture has been carefully designed to be easy to use and engage with by non-specialists,

bridging the gap between complex technical frameworks and practical, user-friendly applications. Our overall goal is to create an architecture that not only meets current Industry 4.0 trends and requirements but is also forward-thinking and facilitates the integration of emerging tools and technologies. By employing software engineering methodologies like MDE, we abstract the complexity of the digital twin system, making it easier to understand, manage, and control. This architecture is further complemented by the MAPE-K loop structuring, which forms a self-adaptive system essential for ensuring the production systems' runtime behavior and real-time decision-making to meet the business requirement. The use of ontology in systems facilitates semantic interoperability and enables semantic computing. It allows systems to understand and interpret the context and meaning of data, rather than just processing raw data. This understanding leads to more intelligent and efficient processing, analysis, and decision-making, as the systems can comprehend and work with the underlying semantics of the data they handle. The integration of the emerging standard AAS for digital twin modeling, is a strategic choice that not only adheres to industry standards to ensure the syntactic interoperability in accommodating diverse workflows and equipment types. This architecture significantly enhances our system's flexibility in processing commands and orchestrating production lines.

This CBSAM architecture supports the engineering phases as presented by G.Urgese et al. [14], with the intention of covering the production system lifecycle from the beginning of design to the production operation process.

- Specification phase:** This phase collects information related to the production system specification. In order to ensure the smooth progress of the following phases, some indispensable elements need to be identified at this very first phase. Three main aspects of achieving the CBSAM are capability, operational, and monitoring submodel information. The capability specification of each resource includes collecting and analyzing the resource data sheets. The process capability specification consists of the identification of the working scenario. The appropriate manufacturing capability semantic representation should be selected for the capability specification as well. The operational information refers to the information related to the asset connection, for instance, the OPC UA information model and server configurations can be a good basis. Specifications for monitoring and diagnosis necessitate the inclusion of establishing a relationship between monitoring-related semantic requirements and dynamic data that the digital twin model can monitor. It is essential to specify the potential events, like the device's malfunction, for the production system with the help of the domain experts.
- Design phase:** All AAS models (Resources, Processes, and Products) should be designed at this phase according to the specification information gathered in the previous phase. The structure of an AAS model and the existing creation methods are introduced in Section 2.1.1. An AAS model is composed of submodels, and submodels are composed of submodel elements. Different submodels are used to describe different aspects, so three different submodels are created to contain information related to "Capability", "Operational data", and "Monitoring". A "Capability_Submodel" describes the manufacturing capabilities from the abstract level by annotating the SubmodelElement *Capability* with ontology semantics. The "OperationalData_Submodel" details the operational properties and the executable operations. This is the submodel that exchanges data with physical assets, including the read/write of dynamic property values and the invocation of available operations. The "Monitoring_Submodel" can be designed for various assets, including different production resources and production processes. It concerns to design of the elements to be monitored for each asset at this stage, the events that may be triggered, and

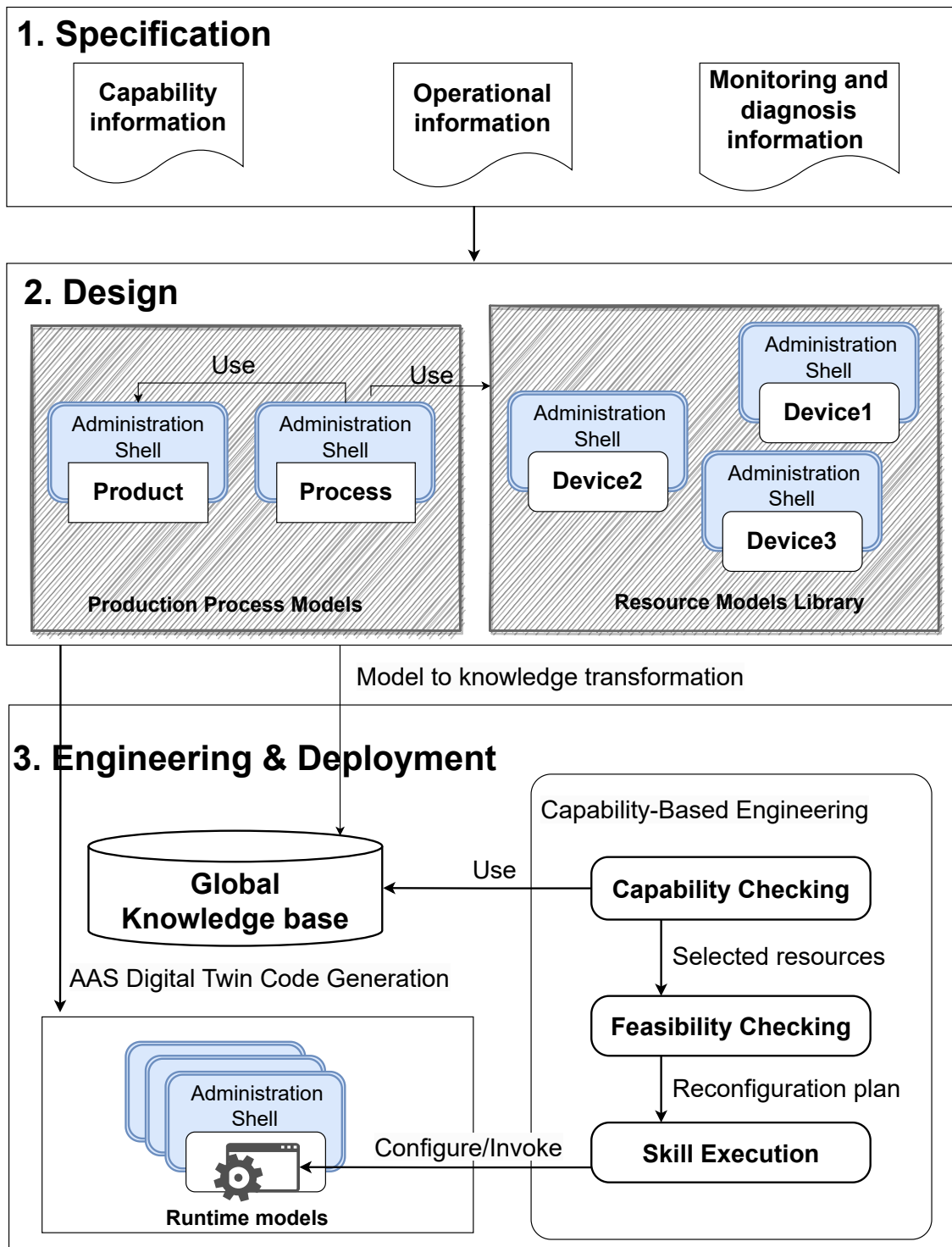


Figure 3.1: Overview of capability-based self-adaptation manufacturing architecture - Part 1

the trigger conditions of the events (the diagnosis rules).

- Engineering & Deployment phase:** A global knowledge base can be generated by a model to knowledge transformation at the engineering phase. This knowledge base contains information on AAS models specified and designed in the previous phases. Meanwhile, each AAS model can be generated to an AAS server for executable mode after the model design. These generated AAS servers enable the connection to the assets and the transparency and interoperability of data exchange during the assets' execution. Then, an initial CBE process should be realized for the first system configuration, where the production work plan can be found and validated as introduced in Section 2.2. The capability checking module uses the initial global knowledge base, which automates the production plan selection from the resource pool. The inferred knowledge results obtained in the capability checking step are conserved for future maintenance. The feasibility checking module not only validates the resource candidates inferred by the capability checking module but also finds the appropriate parameters for the system configuration. The resource parameters are reconfigured as the validated plan. At the skill execution step, the previously determined parameters are configured to the device. A production process orchestrator will interact with the executing AAS resource model according to the validated production plan. This orchestrator invokes the operations following the process design. A table 3.1 summarizes different transformations involved in the engineering and deployment phase.

| Name | Target | Input(Type) | Output(Type) |
|--------------|-----------|--------------------------------------|--------------------------------|
| AAS2MaRCO | Knowledge | Annotated capability submodels (UML) | MaRCO individuals (OWL) |
| AAS2Monitor | Knowledge | Annotated monitoring submodels (UML) | Monitor individuals (OWL) |
| AAS2CSPARQL | Knowledge | Annotated monitoring submodels (UML) | Event queries (OWL) |
| BPMN2NodeRED | Code | Business process (BPMN) | Executable workflow (Node-RED) |
| AAS2BaSyx | Code | AAS model (UML) | Deployable BaSyx code (Java) |

Table 3.1: Different transformations in CBSAM architecture

- Operations & Maintenance phase:** During the production operation, the whole process can be monitored by collecting data from the equipment on the production line. Continuous local knowledge can be extracted by transforming the real-time data into knowledge graphs. A knowledge-based fault diagnosis is required in this phase to analyze the real-time data stream coming from the production line. Faults can be detected by rules predefined in the knowledge base. According to the repair recommendation gathered from the analysis phase, a system replanning is required to ensure rapid management and adjustment methods in abnormal situations to achieve a stable and continuous production process. Depending on the repair advice, either a local effector will be triggered to reinvoke an operation or the central recovery effector will be activated. When the planning module decides to replace the resource from the production line, the execution module will update the local knowledge to the global knowledge base and invoke the central recovery effector. The central

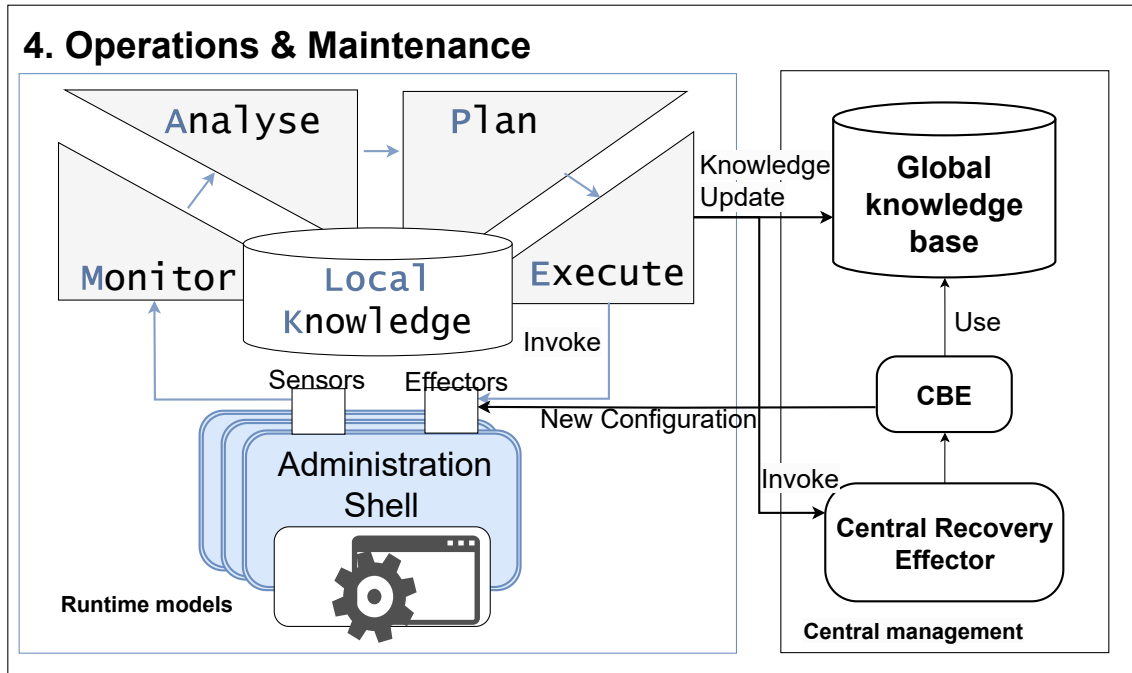


Figure 3.2: Overview of capability-based self-adaptation manufacturing architecture - Part 2

recovery effector will start a CBE query, and in this specific process, the capability checking will rely on the updated global knowledge base.

Through this architecture, it is possible to build a factory digital twin system from the ground up that supports both syntactic and semantic interoperability. The use of MDE technologies also facilitates swift deployment of executable models through automation. CBE permits the rapid replanning and reconfiguration of the production lines. And the MAPE-K loop empowers the manufacturing system to be able to make optimal decisions and respond to changes according to the runtime behavior.

3.2 Specification

The capability specification is foundational for the realization of CBSAM. In diverse multi-vendor production environments, a common resource capability information model is imperative for swift system design and adjustments. This model should enable resource providers to describe their product’s features in a way that’s easy to compare and help system planners align product needs with resource offered capabilities. We opted for MaRCO ontology [12] for the semantic representation of manufacturing resource capabilities in our study. Completed semantic selection shows that in order to fully utilize resource capabilities, all relevant resource data sheets must be systematically collected, whether from suppliers or manufacturers. Beyond just resource details, pinpointing product parameter necessities and recognizing specific operational scenarios are also vital. We must also parameterize the manufacturing processes needed to create the product and critical aspects like efficiency, accuracy, and other functional and non-functional manufacturing characteristics.

Operational information is essential for establishing a bidirectional connection between models and assets, and it is also a crucial component that makes a general information model meet the fundamental characteristics of digital twins. Through the establishment

of a digital twin model, the bottleneck issue of connecting information between the information technology (IT) layer and the operational technology (OT) layer can be resolved. Therefore, in this part of the specification, we can gather information related to asset connectivity. The primary focus is on identifying the communication protocols of devices, whether it's OPC UA, MQTT, or any other standard, ensuring compatibility in transparent data transmission. Within the OPC UA information model, data are constructed with a clear hierarchical structure that ensures consistent and direct access. Metadata, or data about data, is critical to understanding the context and lineage of operational data. The specification should clarify how metadata is collected, stored, and linked to actual operational data, ensuring users can trace the origin and transformation of any data point. Therefore, taking the OPC UA protocol as an example, in the phase of operational data specification, we need to collect the information model on the OPC UA server.

For the specification of monitoring and diagnostics, a clear mapping needs to be established between each semantic requirement and its corresponding data point in the digital twin. We need to specify whether the object is equipment, a production process, or a product, as well as all the dynamic information that needs to be monitored. This not only facilitates accurate monitoring of the dynamic properties of the asset but also ensures real-time feedback from the digital twin for subsequent intervention in abnormal situations. Defining precise alert thresholds for each potential event ensures timely notification. The event specifications should also detail the level of time granularity and time windows required to monitor data. The definition of time granularity significantly affects the real-time degree of monitoring and processing speed. This has important implications for event tracking in production systems.

3.3 Design

This design phase concerns the modeling of digital twin models within a manufacturing system, particularly the three fundamental elements (Resources, Processes, and Products) involved in the production practice. The use of the AAS standard shapes the industrial digital twin in a syntactic interoperable manner. Combining the AAS standard with MDE provides a comprehensive framework that not only supports the conceptualization of digital twins but also ensures compatibility and interoperability across different systems and platforms. Therefore the need to present this modeling environment is essential. However, while the MDE provides the necessary AAS modeling structure, it requires further refinement through the development of aspect-oriented model designs. These models are crucial for detailing specific characteristics and functionalities of the digital twins, in order to meet the requirements. The subsequent subsections introduce the AAS modeling environment and offer a detailed model design guide. This guide outlines various methodologies essential for the design of these digital twin models.

3.3.1 AAS Modeling environment

The AAS-based modeling environment was designed as an ISO/IEC/IEEE 42010 compliant architecture framework [88] (Fig.3.3). It provides several modeling editors in order to create multiple views for the description of AAS-based digital twins architectures. The modeling environment is embedding a domain-specific modeling language (DSML) that governs all the viewpoints of the architecture framework. The DSML is a UML profile that implements the AAS meta-model (version 3RC1). We have chosen the UML profile mechanism since the AAS-based modeling environment is implemented as an extension of Papyrus [34], which is an open-source model-driven workbench supporting the OMG modeling language standards: UML, SysML, and BPMN (Business Process Model No-

tation [11]). Extending an existing modeling environment is a well-suited approach for developing DSMLs in an iterative way, taking advantage of the already existing modeling diagrams and consequently avoiding developing the environment from scratch. Deploying AAS models using UML profiles enables building and describing digital twins in a standardized way, thereby promoting common understanding, interoperability, and data sharing between different applications and systems.

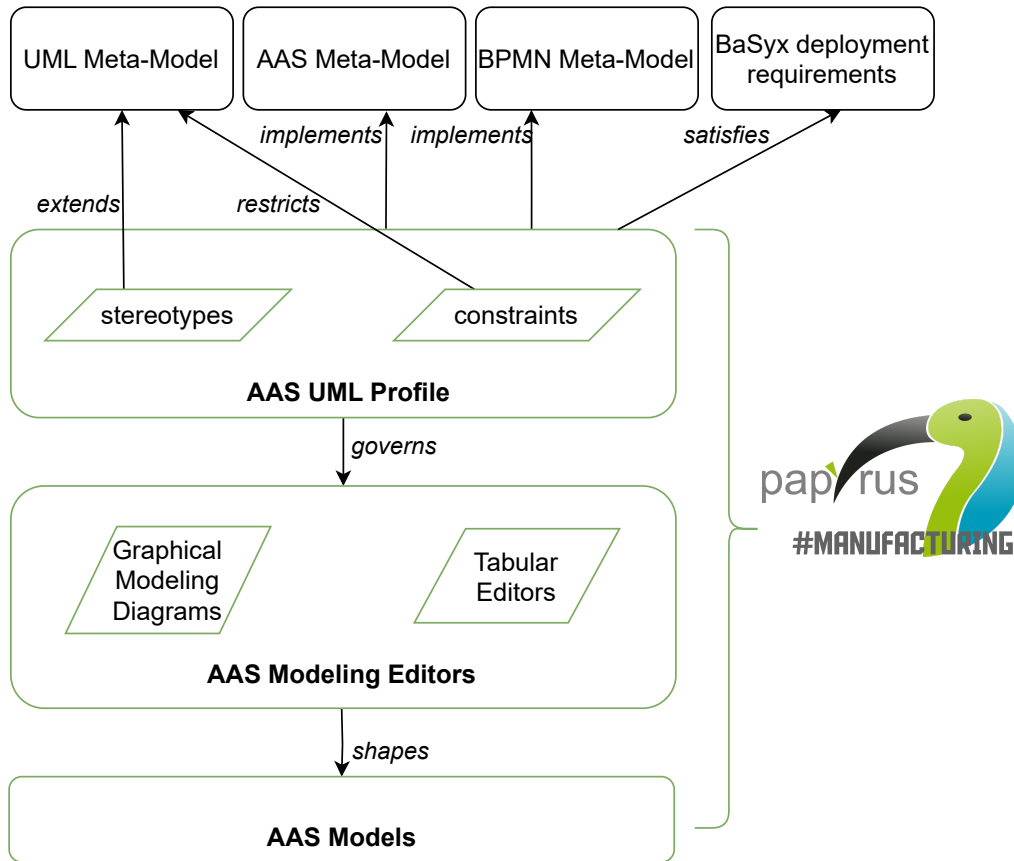


Figure 3.3: Papyrus4Manufacturing modeling environment

UML profiles are a straightforward mechanism for extending the UML meta-model with concepts specific to a particular domain. The primary extension construct in a profile is the “Stereotype”. We established a correspondence (Table 3.2) between the AAS meta-model and the UML meta-model, ensuring that the mapping adheres to the inherent semantics of each meta-model construct for compliance.

| UML Meta-model | AAS Meta-model |
|----------------|--|
| Class | Asset, AssetAdministrationShell, Submodel, SubmodelCollection, Reference, ConceptDescription |
| Property | Capability, Entity, DataElement (Property, File, ReferenceElement, etc.), Event |
| Operation | Operation |
| DataType | AssetInformation |

Table 3.2: AAS Meta-model and UML Meta-model Mapping

- AssetAdministrationShell, Asset, Submodel, SubmodelElementCollection, Reference, ConceptDescription stereotypes extend the UML meta-class Class since each of the semantics of these AAS concepts are compliant with the semantics of Class: “The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects” [89]. In order to restrict the semantics of the UML meta-class Class, we define constraints attached to each stereotype. For example, an Asset must not contain attributes, operations, and behaviors.
- Each SubmodelElement extends a specific meta-class depending on its semantics. For example, the “Operation” stereotype extends the UML meta-class Operation with the restriction: the return parameter of the UML operation is not considered since AAS operations support only in, out, and inout parameters.
- Capabilities and Skills are specializations of SubmodelElements. Capabilities are represented using the “Capability” stereotype that extends the Property meta-class from UML with the constraint that the Capability does not have a type. Skills are represented using the “Operation” stereotype that extends the Operation meta-class from UML. In capability-based engineering captured from [13] and [43], the Capability concept (a type of AAS SubmodelElement) is an abstract description of the functionality of a production resource while the Skill concept is the asset-dependent implementation to achieve a certain effect. Different resources may have the same capability but implemented by different skills. The main goal of capability-based engineering is to design, implement and then dynamically operate the system according to the functions required in each step of the production process, rather than explicitly specifying the actual production resources.

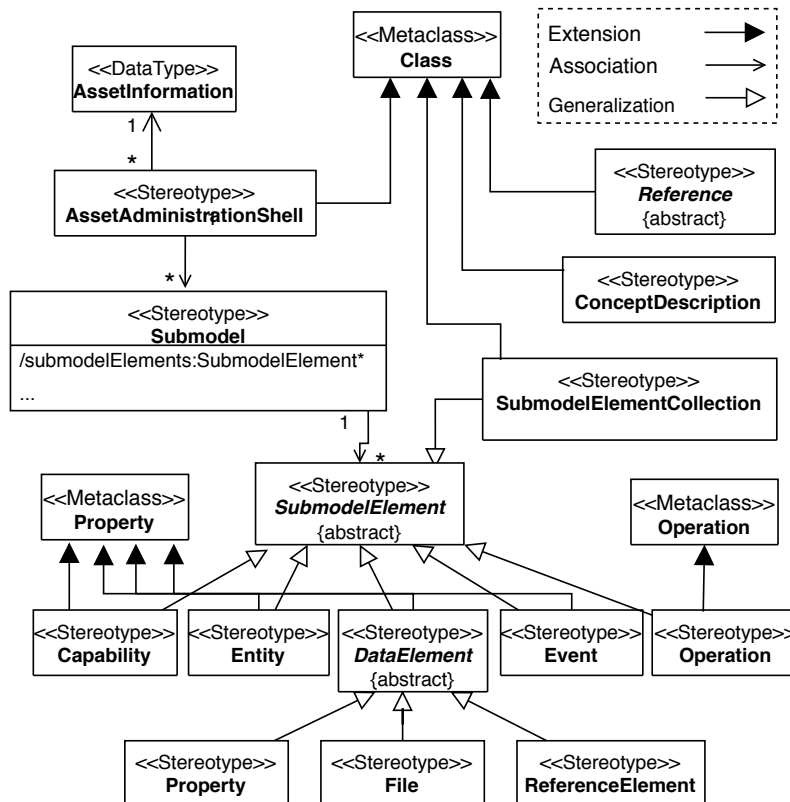


Figure 3.4: Excerpt from the AAS UML profile

3.3.2 Capability Submodel Design

The objective of capability submodel design is to be able to select resources according to their “capability” at a semantic level. Our platform relies on the alignment of digital twins AAS models (using UML profiles) and ontologies (in MaRCO). This alignment requires a comprehensive definition of mappings between concepts in the different languages used to express our AAS models and ontologies.

UML, as a graphical language, allows the creation of visual models of digital twins, thereby simplifying the analysis and design of digital twin systems. Additionally, the use of specific profiles or extensions to UML allows the model to be tailored to the specific needs and semantics of the digital twin, thus ensuring an accurate semantic representation of the model. Ontology serves as a structural framework that systematically classifies and defines relationships between various concepts within a specific domain or across multiple domains. Ontology provides a powerful way to define the semantics and relationships between different entities and concepts, thereby ensuring coherent understanding and interpretation among the platform’s various applications and users.

However, a method to guarantee consistency between the AAS model and the ontology is required. This requires a comprehensive and detailed mapping framework as a transformation layer that ensures that the concepts, properties, and relationships defined in the UML-based AAS model are semantically and syntactically consistent with those expressed in the MaRCO-based ontology. This mapping also establishes a foundation that can leverage consistent data and ontology reasoning rules to seamlessly integrate and execute rule-based analytics and various AI-driven applications. This cannot be achieved through a primitive model-driven engineering framework alone.

General Concepts Mapping

Originally the study focused on the mapping between OWL concepts and UML metaclasses for the general concepts mapping. However, in order to achieve the transformations between AAS and OWL models, a tool OML adapter has been used for the implementation in Papyrus. We chose to use the OML adapter because it allows automatic conversion of the ontology concepts into a UML profile and its extraction from UML profile-compliant instance models back to OWL. OML is a language to describe ontologies, where adapters for transformations from OML to UML, and UML to OWL are provided. In this context, OML can be seen as an intermediate language to enable the conversions.

| OWL | OML | UML Metaclass | UML Profile |
|---|-----------------------|------------------------|-------------------------------|
| Class | Aspect / Concept | Abstract Class / Class | Stereotype |
| Individual | Instance | Instance specification | Stereotype applicable element |
| Object Property | Relation Entity | Association / Property | Stereotype attribute |
| Data Property | Property | Property | Stereotype attribute |
| Cardinality, MinCardinality, MaxCardinality | exactly min max | Multiplicity | Multiplicity |

Table 3.3: General concept mapping

To pave the way to model alignment, a mapping between OWL, OML, UML general concepts and the AAS-UML profile concepts has been designed (see Table 3.3). The

classes in OWL are represented as aspects and concepts in OML. The aspect refers to the abstract class, while the concept refers to the class. They are all transformed to stereotypes of a UML profile. An individual in OWL refers to an instance in OML and instance specification in UML, this represents a UML element to which a stereotype is applied to. The OWL object properties are represented in OML as relation entities and refer to associations or properties in UML. The OWL data properties refer to properties both in OML and UML. The object properties and data properties are transformed to the attributes of a stereotype.

In OWL, the concept of “class” is equivalent to “aspect” or “concept” in OML, “abstract class” or “class” in UML metaclass, and is mirrored by “stereotype” in UML profile. Furthermore, the concept of “individuals” in OWL is consistent with “instances” in OML, equivalent with “instance specifications” in UML metaclasses, and is referred to “elements” that apply to “stereotypes” in UML. At the same time, the “object attributes” of OWL are parallel to the “relationship entities” in OML, matched with the “association” or “attribute” in the UML metaclass, and the “stereotype attributes” in the UML profile terminology. Similarly, “data attributes” in OWL are consistent with “properties” in OML and UML metaclasses, and map to another form of “Stereotype attributes” in UML Profile. For the cardinalities, OWL classifies expressions into “Cardinality”, “MinCardinality” and “MaxCardinality”, while OML simplifies them to “exactly”, “min” and “max”. Instead, both UML Metaclasses and UML Profiles leverage “multiplicity” to convey these constraints.

AAS & MaRCO Vocabularies Mapping

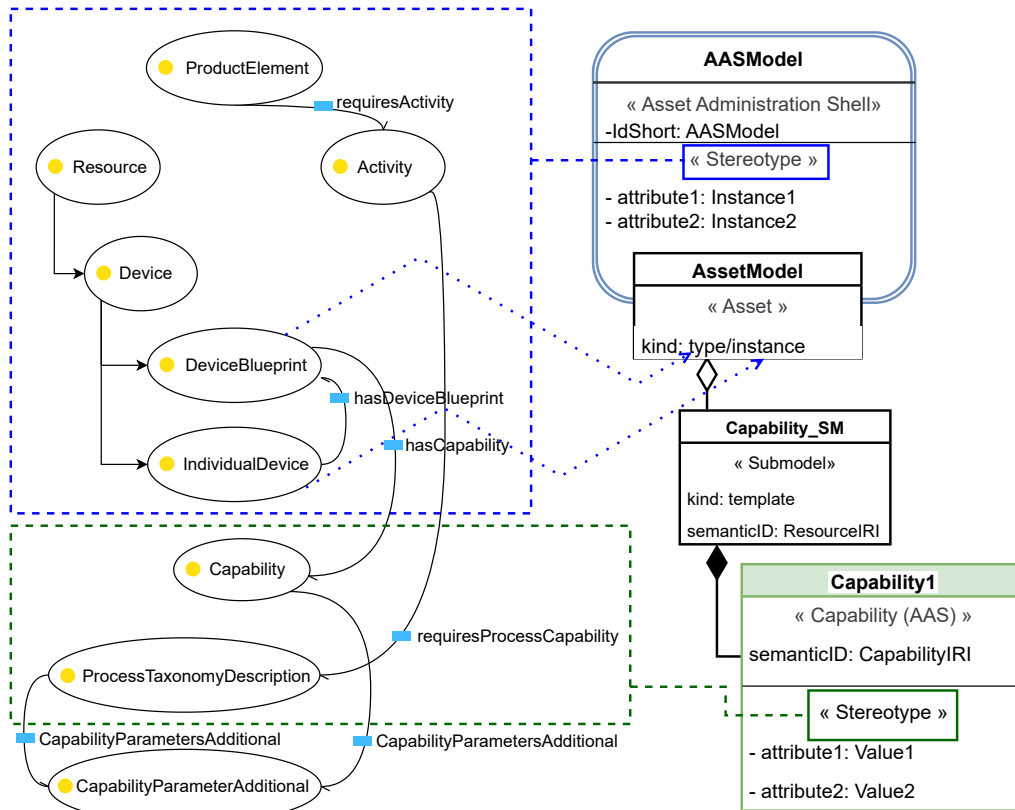


Figure 3.5: AAS Marco vocabularies mapping

In order to use OML for our MaRCO-specific semantics, the mappings between general concepts are not enough. Hence, we defined transformation rules between the vocabularies

of MaRCO and AAS metamodel in Figure 3.5. MaRCO concepts are on the left, while AAS concepts are in the middle. As presented in Table 3.3, the OWL classes are transformed to stereotypes in a UML Profile. A subset of MaRCO concepts have been chosen, such as *Resource*, *ProductElement* and *Activity*, which are transformed to stereotypes that can be applied to the “Asset Administration Shell” models of resources, products and processes. It is worth mentioning that the *DeviceBlueprint* ontology class needs to correspond to the AAS model with the asset kind as “type”, while the *IndividualDevice* can only match the AAS model with the asset type as “instance”. The *Capabilities* or *ProcessTaxonomyDescriptions* stereotypes should be applied to AAS “Capability” models. Then the object properties *requiresProcessCapability* and *hasCapability* refer to the attributes of these stereotypes. And the value type of these attributes should be *Capabilities* or *ProcessTaxonomyDescriptions*. Each stereotype may contain two different types of attributes, one is the scalar properties such as weight or depth which refers to the data property parameter, and the other is object properties that point to other stereotyped elements in the model package. In the ontology representation, IRI (Internationalized Resource Identifier) serves as a fundamental component for uniquely identifying classes, properties, and instances (individuals). The use of IRI refers to the distinction between entities in the ontology. In the AAS standard specification, an identifier *semanticID* is defined to facilitate the interoperability between different systems and platforms. By using this identifier, different systems can more easily recognize and process shared data. By applying stereotype to an AAS model, the *semanticId* of the AAS model will be associated with the vocabulary’s IRI in MaRCO ontology.

Once the AAS models are annotated with stereotypes from the MaRCO ontology, these AAS models become instance models that comply with MaRCO semantics. So we can convert these stereotype applicable elements to MaRCO compliant OWL individuals. Semantic connections between model elements are also transformed according to the previously mentioned mapping rules. So an AAS class is transformed to an OWL individual, and its type is either a *Resource*, a *ProductElement*, or an *Activity*. It may contain object properties *hasCapability* or *requiresProcessCapability* with the value of individual capabilities as defined in the AAS model.

3.3.3 Operational Data Submodel Modeling

The design of the operational data submodel aims to arrange the various dynamic properties and operations to facilitate data acquisition and remote orchestration. It involves the insertion of server information, which is acquired during the preceding specification stage, into the AAS submodel in a structured and coherent manner. It’s imperative that the connection information is consistent and accurate, to ensure the models are valid and reliable, thereby guaranteeing the integrity of future deployments and applications of this AAS digital twin.

In this operational data submodel, the concrete skill can be designed. As defined in [43], skill represents the implementation of abstract capability at the device level, and also represents the behavior that will be actually performed during operation. Skills are usually implemented by the orchestration of a set of atomic actions. In the model design stage, we chose to use “AAS Operation” to model skills and actions. Various processing languages can be used to describe the composition of behaviors, such as BPMN, FSM, BehaviorTree, etc.

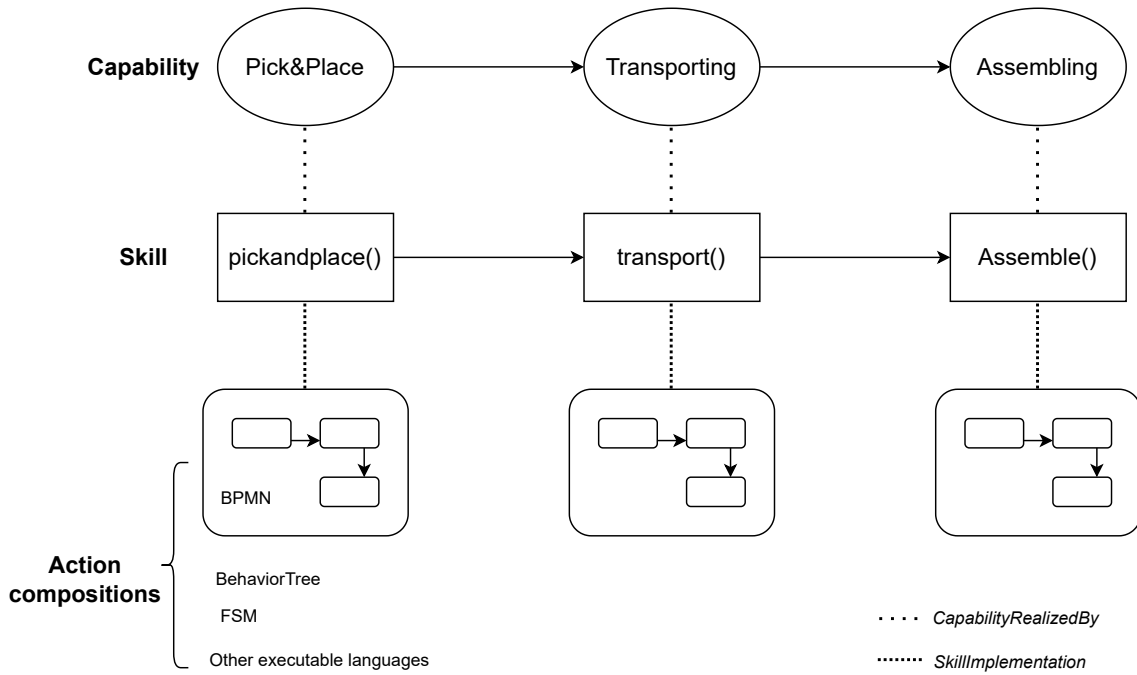


Figure 3.6: Relations between capability and skill

3.3.4 Monitoring Submodel Modeling

This section presents the proposed conceptual metamodels for self-adaptation, which includes concepts related to the MAPE-K loop of CBSAM. As we discuss the modeling based on AAS models, the metamodel extends the AAS metamodel. As knowledge is an important part of the MAPE-K loop, this section also includes mapping the introduced metamodel and the ontology concepts for the implementation. This essentially means examining how our system responds to changes in runtime behavior and adapting its operations accordingly to maintain optimal performance. When confronted with deviations or anomalies, we will shed light on how the self-adaptive system can invoke self-repair protocols, ensuring resilience and operation continuity. This consistency is critical not only to maintain consistency between the digital twin and its physical counterpart, but also to ensure seamless interoperability and communication within the integrated digital platform. The system's self-adaptability is essential to ensure production on time with the required product quality from the business level.

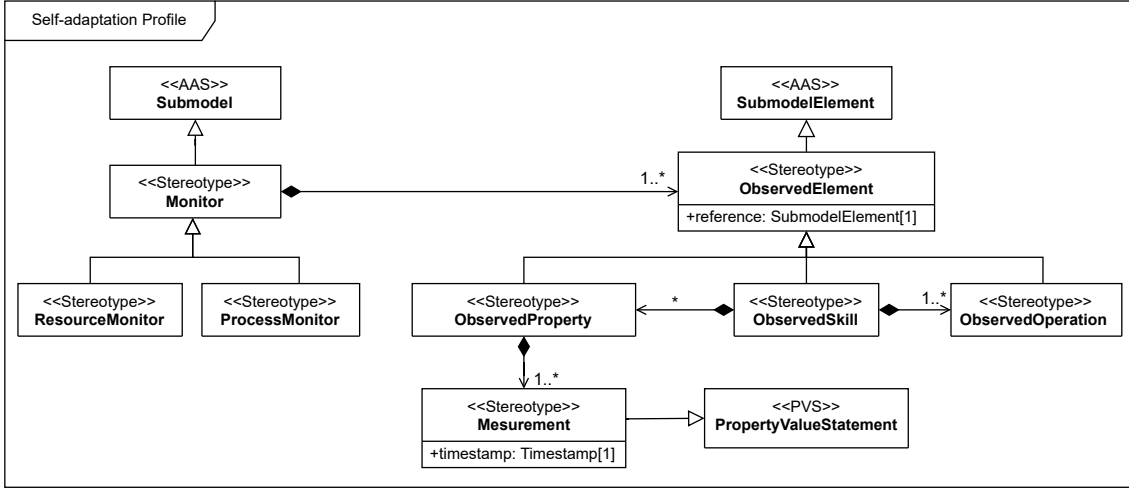


Figure 3.7: Monitoring metamodel

Monitoring Metamodel

Like the previous method of capability interoperability, we also implement this monitoring metamodel through the UML profile. The monitoring metamodel is shown in Figure 3.7. There are two types of *Monitors*. *ResourceMonitor* is used to observe and manage manufacturing resources to ensure optimal functionality and performance. *ProcessMonitor* is to monitor whether the entire production line is running smoothly according to the given process work plan. *Monitor* inherits the characteristics of *Submodel* from AAS profile and can have its own observation objects *ObservedElements*.

We regard *ObservedElement* as a *SubmodelElement*. Each *ObservedElement* has a special attribute *reference*, which points to its original observing *SubmodelElement*. According to different observation types, we divide *ObservedElement* into three categories, namely *ObservedProperty*, *ObservedOperation*, and *ObservedSkill*. Skill is an implementation of a function specified through a capability that is deployed on a specific resource, which provides an interface to be invoked by other systems by encapsulating the internal complex functions. So to observe a skill, one needs to observe also the properties and operations wrapped into this interface. The data collected in real-time is the measurement of *ObservedProperty*. This *Measurement* stereotype will inherit from *PropertyValueStatement* and has the special attribute *timestamp* to record the measurement time. It is worth mentioning that the value of the *ExpressionSemantic* attribute of *Measurement* should always be “measurement”.

Diagnosis Metamodel

The construction of this diagnosis metamodel Figure 3.9 is inspired by the work of [90] and [91], where [90] introduces a conceptual model of a self-healing CPS system, and [91] gives definitions to different fault types. The analysis phase includes the fault diagnosis and provides a recovery method. Our *Diagnosis* is supposed to be realized by domain knowledge so that we focus more on the type of *RuleBasedDiagnosis*. A stereotype *DiagnosisRule* is created to describe the rules for performing diagnosis reasoning. In the implementation phase, the *DiagnosisRules* models will be converted into RSP-QL queries for runtime reasoning. To achieve this function, some necessary attributes must be included in *DiagnosisRule* models. *Threshold* inherits also from *PropertyValueStatement* to define constraints. An RSP-QL query verifies the real-time data (*Measurements*) conforms to the constraint values (*Thresholds*) within a time window. The *window_range* defines the

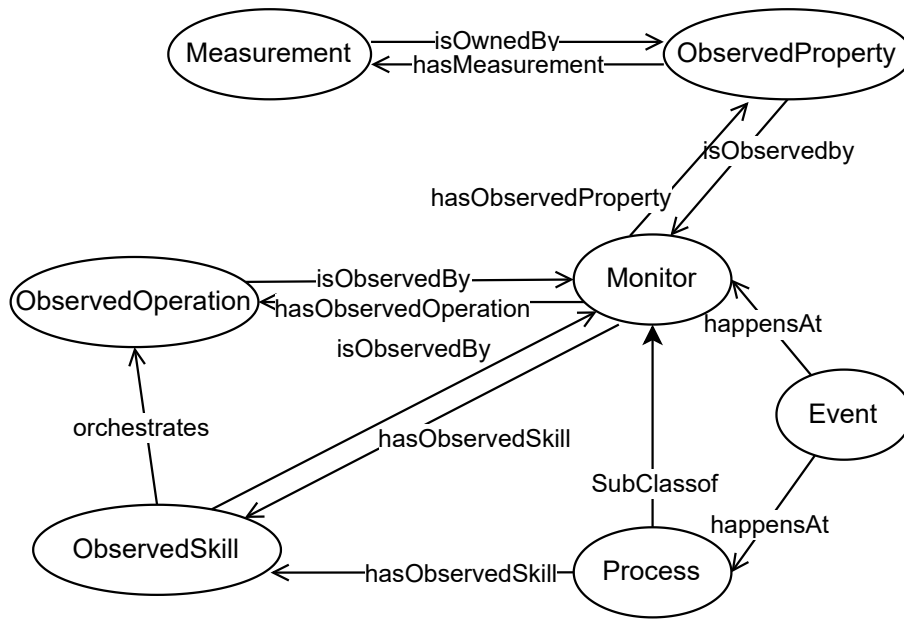


Figure 3.8: Monitoring ontology object properties

Table 3.4: Definitions of concepts for monitoring

| Concept | Definition |
|-------------------|--|
| Monitor | An entity that is responsible for observing and recording activities or changes. |
| ResourceMonitor | A specialized monitor designed to observe and manage manufacturing resources to ensure optimal functionality and performance. |
| ProcessMonitor | A specialized monitor observing the status and performance of a process. |
| ObservedElement | An entity, function, or variable within a system that is subject to observation and analysis, by the diagnosis tool or system. |
| ObservedProperty | A particular characteristic or attribute of an element being monitored, which can be measured or evaluated. |
| ObservedOperation | A function or method being monitored for performance, success, failure, or other measurable metrics within a system. |
| ObservedSkill | A skill being monitored for the process required capability implementation. |
| Measurement | Quantifiable data or metrics obtained through observation, pertaining to an observed property, or observed operation. |

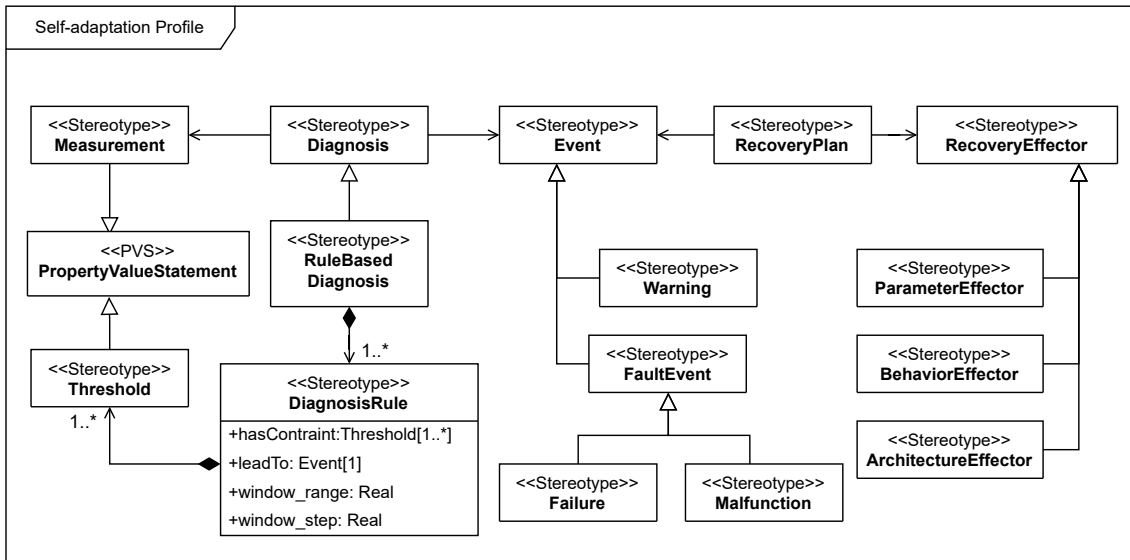


Figure 3.9: Diagnosis metamodel

window time frame, and the *window_step* indicates the query execution frequency. Finally, the query results will lead to different *Events*.

This analysis module should not only point out the *Faults* but also provide the ability to issue *Warnings*. Therefore, they are all classified as *Events*. To clarify the terminology, *Warning* is the potential problems of the system, while *FaultEvent* is the detection of an unacceptable deviation of the system compared with the standard conditions. *Failure* means a system cannot perform the demanding operation. *Malfunction* refers to the operation is not executed in the desired way.

After the *Diagnosis*, the *RecoveryEffector* provides the recovery method to heal the system. As defined in [90], three kinds of *RecoveryEffectors* are classified by the nature of the modification. *ParameterEffector* modifies the property of the system component. *BehaviorEffector* makes changes to the behavior. *ArchitectureEffector* refers to adding, replacing, or removing a component from the system.

Table 3.5: Definitions of concepts for diagnosis

| Concept | Definition |
|----------------------|---|
| Threshold | A specified limit or boundary that must be maintained in order to keep the system or process running smoothly. |
| Diagnosis | The process of identifying the root cause or nature of a problem or deviation within a system through systematic investigation and analysis. |
| RuleBasedDiagnosis | A diagnostic approach that employs a set of predefined rules or criteria to identify issues or anomalies within a system. |
| DiagnosisRule | A specific guideline or criterion used within a rule-based diagnostic framework to detect and categorize issues. |
| Event | A significant occurrence or change of state within a system, typically triggering specific behaviors or responses. |
| Warning | A preemptive alert or notification that signals a potential problem or risk, typically allowing for preventive action to be taken. |
| FaultEvent | An event characterized by a malfunction or failure within a system, typically requiring diagnostic and corrective actions. |
| Failure | A state in which a system or component is unable to perform its intended function due to an issue or fault. |
| Malfunction | An aberrant state in which a system or component is not operating according to its intended or expected manner. |
| RecoveryEffector | A component or process designed to restore a system or application to a normal or safe state following a failure or deviation. |
| ParameterEffector | A module or function that influences or adjusts parameters within a system. |
| BehaviorEffector | An element that manages or manipulates to change the behavior. |
| ArchitectureEffector | A function or component that modifies the architectural elements of a system, potentially altering its structure or behavior. |
| AdaptationAction | A deliberate modification or adjustment enacted within a system to accommodate changes or to optimize its performance under varying conditions. |

3.4 Engineering & Deployment

Digital twins play a key role in the engineering phase. Automatically generated digital twins (AAS2BaSyx) are not only deployable, but can also accurately reflect real-world entities and processes. It plays an important role in building knowledge bases (AAS2MaRCO, AAS2Monitor, AAS2CSPARQL) during this engineering phase. These knowledge bases record the initial state of the system in detail, providing reference points and basic data layers for future diagnosis, modifications, and system enhancements. The process of finding and validating the resources that meet the work plan requirement, the capability-based engineering method, enables the system's rapid replanning.

The deployment phase involves thorough configuration and integration of the process, ensuring a seamless start-up workflow through the deployed digital twin. The production process designed earlier will be transformed into an orchestrator (BPMN2NodeRed). Synchronization of virtual and physical systems is addressed here, ensuring consistency and optimal functionality between them.

Figure 3.10 shows the whole process of the capability-based engineering. In a model-based Digital Twin production system, each resource (or asset) has its own representative AAS provided by different stakeholders (product and process designers, equipment suppliers, integrator, etc.). The AAS contains the technical descriptions (nameplate), the simulation models, the operational data, or other business information. The resource pool of a plant contains all the resources as well as the system layout design. During the design phase, the system architect specifies the products and their manufacturing processes. The rounded rectangles in the figure represent different levels in the automation pyramid from ISA95 [92]. From top to bottom, they represent the manufacturing operation management (level3), the monitoring and automated control (level2), and the manipulation of production processes (level1). In the latter level, the "AASs" (or digital twins) are continuously updated to represent the assets real time status.

The AAS of a resource describes the provided capabilities and related features, without knowing the process to complete and the product to produce. The AAS of a process describes the capabilities required by the work plan and some environmental constraints. The AAS of a product describes the product from different aspects. Flexible production systems must automatically match resources, processes and products in order to achieve continuous capacity-based engineering. Capability checking takes the AAS capability sub-models of process, products, and resources as input and computes the possible resource combinations that may currently achieve production. During the feasibility checking step, these combinations and environmental contexts will be simulated to validate the selected resource combinations against their current constraints. Then the next step automatically supervises the skill execution of the selected models. The supervisor deploys the selected resource pool models according to the reconfiguration plan obtained through the capability-based reconfiguration phase. During the whole process execution, the supervisor monitors the status of all asset models and will re-plan the production process in a timely manner when abnormalities are detected.

3.4.1 Capability Checking

Design Time Capability Checking

Capability-based engineering aims to deploy resources dynamically rather than directly specifying the actual production participants. By defining the capabilities required for the product's production process and letting the automated production line management system find the resources and implement the process to achieve the reliability of the digital twin production system. The capability checking module design, as shown in Figure 3.11,

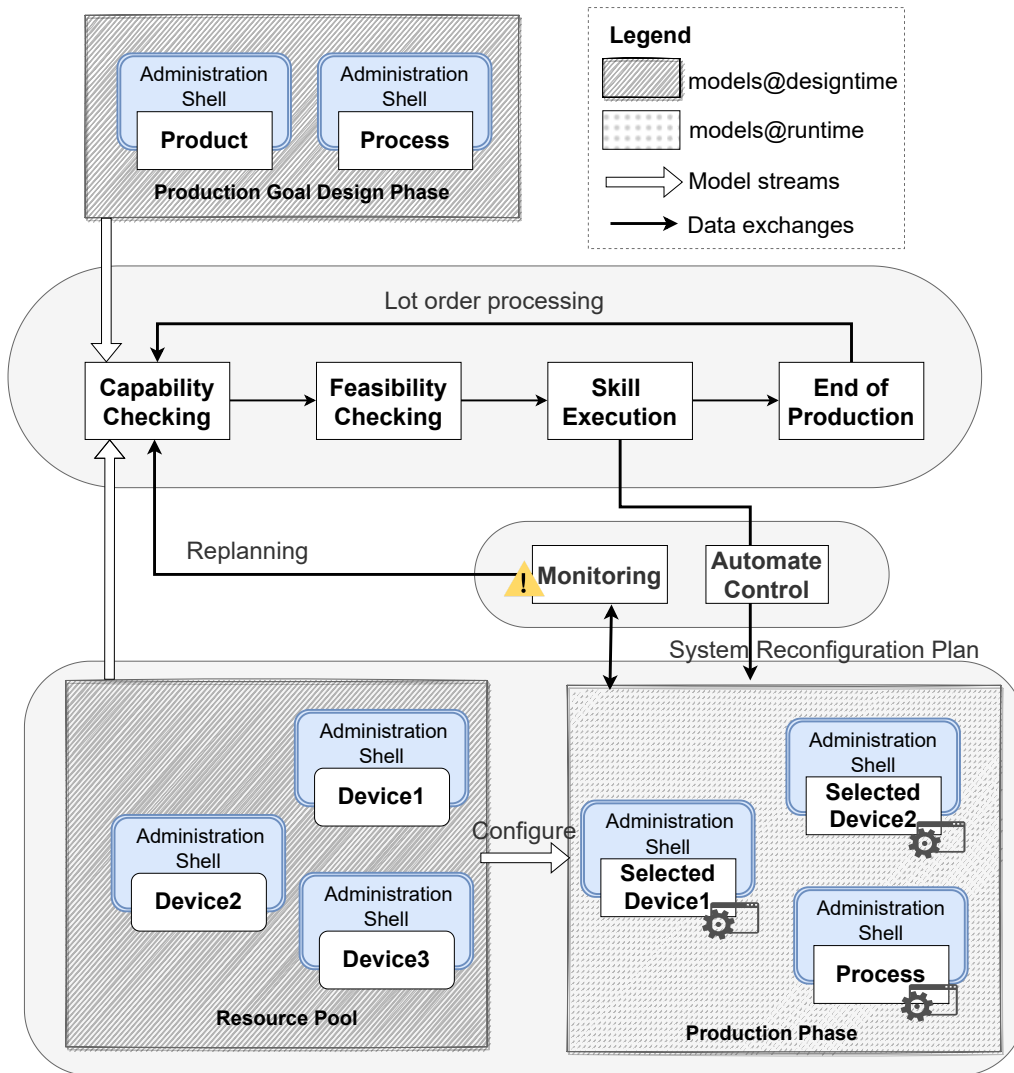


Figure 3.10: Capability-based reconfiguration approach

interacts with AAS models to set/get their semantics and then to trigger the capability matchmaking reasoner in order to compute the best resources matching the requirements of each production process. The four stages depicted in Figure 3.11 are:

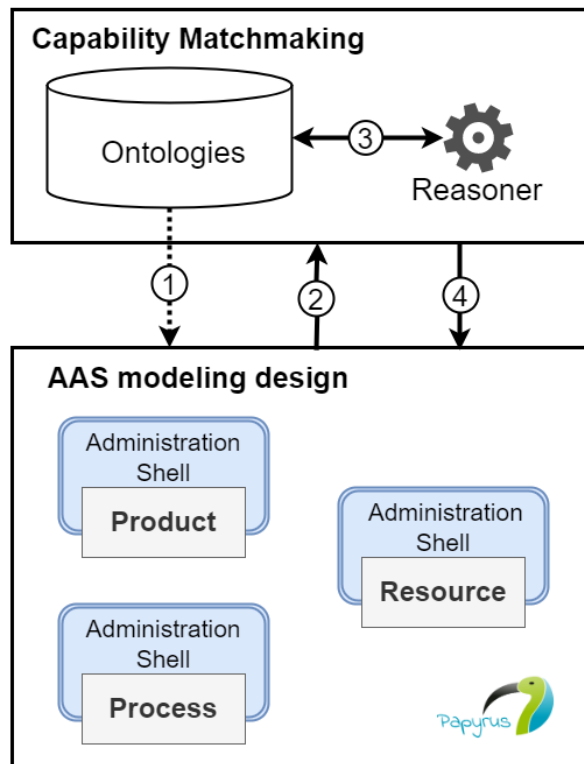


Figure 3.11: Capability checking architecture

1. The designer annotate the AAS models with semantic definitions (semanticIds) from the ontology.
2. The designer triggers the automatically transformation of the AAS models (Product, Process, Resources) into ontology compliant individuals.
3. With the input individuals, the automated reasoning engine matches the capabilities required by the process with the capabilities provided by the resources.
4. Finally the capability checking module returns the matchmaking result to the designer.

Since there was a comprehensive expert investment in its design, the ontology will not frequently change over time but to add new resources and drop decommissioned ones. Consequently, the first stage (ontology to UML profile conversion part) only needs to be performed once, as long as the ontology concepts do not change. The second, third, and fourth stages will be repeated, whenever a PPR model update occurs. All the actions represented by the arrows shown in Figure 3.11 are automated, system architects only need to define and select the required production models. A concrete example to describe this capability checking process shows how to select a device that can provide transporting capability from the alternative resources when an object needs to be moved in the production process.

Runtime Capability Checking

The major difference between the capability checking at design time and the capability checking at runtime concerns the availability and status of the resource. As illustrated in Figure 3.10, by monitoring the models@run.time, runtime capability checking can be triggered by the recovery adapter, which will be presented in Section 5.4.1. With the knowledge integration from runtime data (See Section 3.5.5), the actual status of the system will be updated to the global knowledge base that is considered by the runtime capability matchmaking mechanism.

3.4.2 Feasibility Checking

Feasibility Checking is a procedure that brings in environmental factors and constraints. Simulations for feasibility checks are conducted during the feasibility checking phase, which enables the engineers and developers to test and validate the systems in a virtual environment before physically implementing them in the real world. This will further filter out the skills that meet the conditions and will be deployed on the production line. This enables the engineers and developers to optimize, and validate systems in a virtual environment before physically building and implementing them in the real world.

3.4.3 Skill Execution

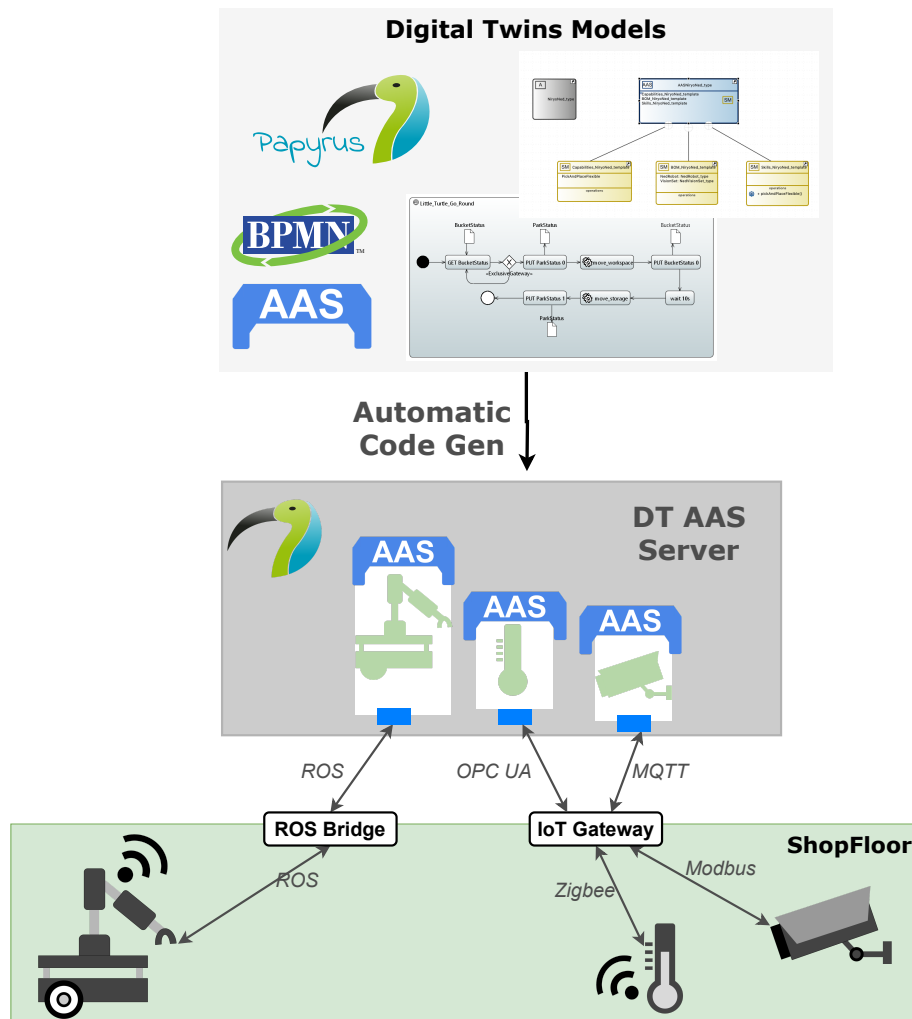


Figure 3.12: Skill execution architecture

The diagram 3.12 is an illustration of the top-down skill execution architecture, highlighting the key stages in which the field equipment operates according to the business process designed in the process digital twin model. This representation emphasizes the importance of seamless integration between conceptual design and actual execution, bridging the gap between virtual planning and physical execution. In this phase, the middleware plays an integral role in enabling the connection between the digital twin model and the physical shop floor equipment.

As introduced previously, BaSyx [39] is an appropriate candidate for the skill execution. Digital twin models and business processes are transformed from models into executable code. The BaSyx AAS digital twin deployed as a REST server adds another layer of flexibility and accessibility to the system. This feature enables the external applications that support the HTTP protocol to interact effortlessly with the digital twin server. These applications can perform a range of functions, from configuration and execution to orchestrating digital twins.

3.5 Operations & Maintenance

A Cyber-Physical System (CPS) incorporating a MAPE-K (Monitor, Analyze, Plan, Execute over a shared Knowledge base) architecture represents a strategic approach that primarily facilitates intelligent and adaptive system behaviors for the operations and maintenance phase. The effort of integrating MAPE-K into CPS can be comprehensively elucidated through AAS digital twin. Figure 3.13 shows the structure of an individual self-adaptive loop for a single AAS. The description of each module will be precise in the following subsections.

3.5.1 Monitoring

In the context of CPS, the Monitor component perpetually observes the system's operational status, considering both cyber and physical aspects, ensuring that any deviations or anomalies are promptly identified. It continuously collects data regarding the system's state and performance through various sensors and log files.

Based on the designed framework, the monitor already has a good understanding of the specific elements within the system that require to be closely observed. This prior knowledge allows for a more targeted and efficient monitoring process. One component of the monitoring mechanism is the "Stream Generator". This component performs the task of processing real-time data collected from the system. It continuously generates raw data into runtime RDF graphs that provide a dynamic and semantic representation of the ongoing performance and state of the system. This generator not only facilitates immediate understanding and analysis but also ensures that real-time data is seamlessly integrated into the local knowledge base.

3.5.2 Analysis

The analysis component methodically examines the data acquired by the monitoring phase, identifying patterns, detecting anomalies, and scrutinizing system performance against predefined benchmarks or objectives. This analysis module can utilize various data processing and machine learning techniques to derive insights and ascertain the system's condition and performance.

In the designed framework, the analysis module is integrated in the form of a "Continuous Query Engine". This engine enables the dynamic execution of queries, which are generated from the AAS2CSPARQL procedure of each AAS model. It enables intelligent parsing through the real-time knowledge graph to discern patterns that indicate potential

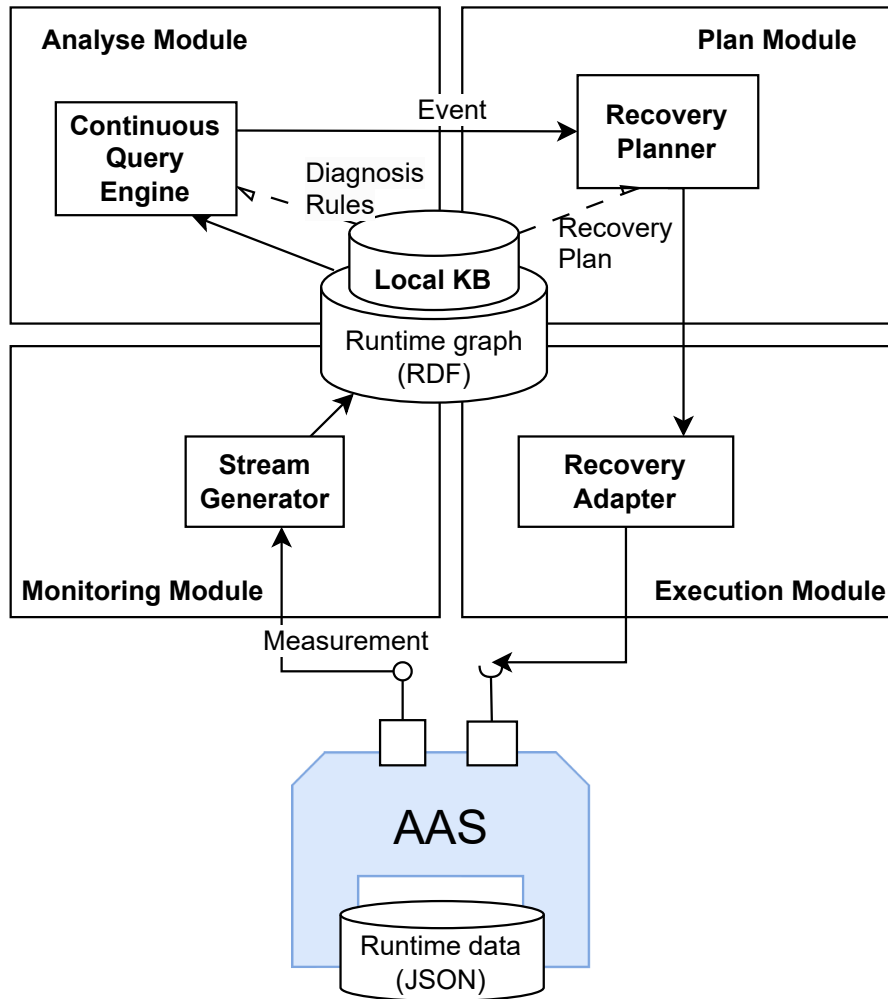


Figure 3.13: MAPE-K loop for a single AAS model

system anomalies or dysfunctions. This is a rule-based analysis method. However, the implementation of this part is not limited to this method. The main mechanism is analysis based on dynamic data.

3.5.3 Planning

Plan seeks to devise strategies or create actionable plans based on the insights obtained during the analysis phase. This involves crafting potential responses, corrections, or adaptive strategies to cater to the system’s present and future operational requirements. The planning component can leverage optimization algorithms and planning techniques to construct efficient and feasible plans.

The “recovery planner” is an integral part of this phase and is tasked with making decisions after an event or anomaly is detected. It assesses whether the detected event is a behavior malfunction, a major failure, or a system warning. Based on this assessment, the recovery planner determines the appropriate course of action. This may include minor adjustments or corrections to mitigate minor malfunction, or it may include a comprehensive recovery strategy in the event of a major failure. Additionally, the planning component also includes proactive measures to preemptive actions based on predictive analysis to prevent potential warnings.

3.5.4 Execution

Execution implements the strategies formulated in the planning phase. This component interacts directly with the system, enacting control commands, and deploying adaptive actions to align system behavior with the formulated plans. The execution phase might involve activating actuators, modifying system parameters, or altering the configuration of the cyber or physical components to implement the strategies. This requires a high level of precision and responsiveness, as the system needs to quickly adapt to the dynamic operating environment.

The execution phase is the phase in which the strategic plan is realized, translating the strategies elaborated during the planning phase into concrete actions. The addition of different types of effectors (parameters, behaviors, and architectures) adds a layer of complexity to this phase, allowing for a more nuanced and efficient implementation of operational strategies.

3.5.5 Knowledge Integration

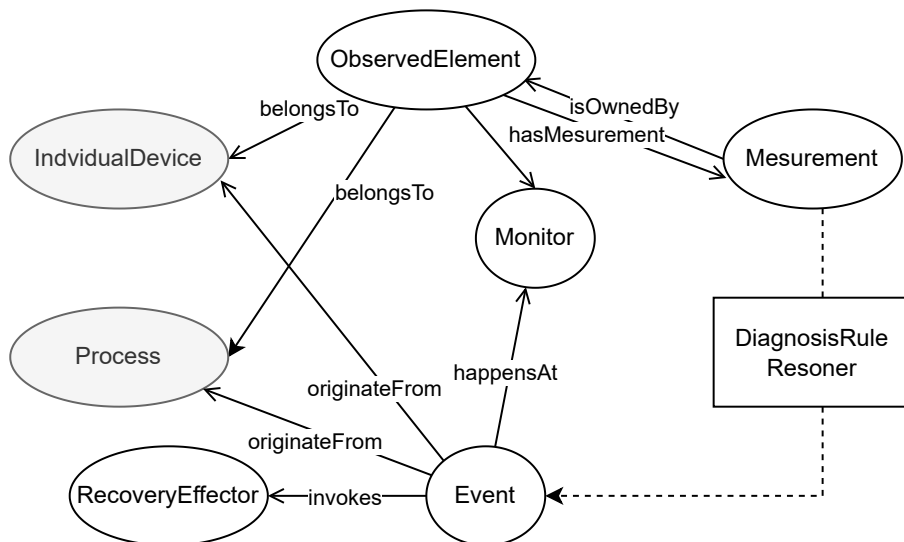


Figure 3.14: Ontology relation

However, in order to aggregate all the digital twin models together, it is necessary to establish a common knowledge base. After constructing our self-adaptation metamodel, we need to connect it with the existing knowledge base, that is, the knowledge results obtained in the capability checking stage. Because we chose MaRCO Ontology as capability checking, we also combined the main concepts we built with MaRCO to form a new knowledge graph. Figure 3.14 shows the relations.

The relationship between the *ObservedElement* and its potential parent entities, *IndividualDevice* and *Process*, is foundational to our metamodel. This relationship ensures that any observed element can be accurately traced back to its source, whether a specific device or a process. Similarly, the origination of an *Event* from either an *IndividualDevice* or a *Process* provides insights into the original cause of the event. Such traceability provided by the relationships is crucial for the context awareness of the system, which updates the change to the production cell for the new planning phase.

Integrating our self-adaptation concepts with the MaRCO Ontology allows us to combine newly acquired knowledge with existing knowledge and create new relationships. It combines distributed knowledge information gathered from different monitors into a

centralized knowledge base. Then enters the new planning phase and provides a new production plan for the self-adaptive manufacturing system.

3.6 Conclusion

This chapter provides a generic approach to building a flexible, adaptive manufacturing system, CBSAM, an architecture that selects and integrates state-of-the-art methodologies for ease of use and participation by non-specialists. The CBSAM architecture is designed for manufacturing systems in the context of Industry 4.0, finding a balance between complex technical frameworks and practical, user-friendly applications. The complexity of the digital twin is abstracted and made easier to understand, manage and control by adopting software engineering methods such as MDE, which is further complemented by the MAPE-K loop architecture to form an adaptive system. The use of ontologies in the system promotes semantic interoperability and enables semantic computing. It enables the system to understand and interpret the context and meaning of data, allowing the system to analyze, predict and plan more wisely.

Chapter 4

Implementation

Contents

| | |
|--|-----------|
| 4.1 Overview | 49 |
| 4.2 Modeling Environment Development | 50 |
| 4.2.1 AAS Model Design Diagrams | 50 |
| 4.3 Capability Checking @ Design Time Development | 53 |
| 4.3.1 Model Transformation Module | 54 |
| 4.3.2 Capability Matchmaker Module | 57 |
| 4.3.3 User Interface Module | 58 |
| 4.4 Skill Execution & Orchestration Development | 59 |
| 4.4.1 Deployable AAS Server Architecture | 59 |
| 4.4.2 Node-RED Package Development | 60 |
| 4.5 Monitoring & Diagnosis Development | 62 |
| 4.5.1 Local Monitor Knowledge Generation | 62 |
| 4.5.2 Data Acquisition and Real-Time Stream generation | 63 |
| 4.5.3 Runtime Analysis | 65 |
| 4.6 Conclusion | 66 |

4.1 Overview

Implementation marks the process of bringing the methods explored in the previous section to practical realization. This phase is the concrete application of the concepts, strategies, and techniques discussed in the previous chapters. It bridges the gap between conceptual exploration and realistic results, emphasizing the practical application of conceptual insights. This phase is not only a test of the feasibility of the proposed methods, but also an opportunity to refine them by hands-on experience and feedback. To ensure consistency with the research questions and contributions presented in the Introduction, the structure of this chapter revisits the initial assumptions and objectives mentioned previously. The outline of this chapter is as follows.

Section 4.2 presents the Papyrus4Manufacturing (P4M) toolset development. The presentation includes the integration of user-friendly editors for standardizing the AAS-compliant digital twins creation, and the automatic deployment functionality based on model transformations and code generation methods.

The following two sections present the partial implementation of the CBE operation. Section 4.3 explains how an MDE approach can aggregate around digital twin modeling tools both I4.0 technologies and AI (Knowledge Representation and Reasoning) tools.

The implemented platform aligns modeling and ontological elements by AAS2MaRCO transformation, in order to get both the manipulable models and an ontology on which we can make semantic queries. This module not only provides semantic descriptions for digital twin models, but also complements model-driven engineering tools with automated reasoning. Section 4.4 describes a tool developed in order to ease the orchestration of process digital twin. The implementation brings on the transformation from a static BPMN process into an executable Node-RED flow.

Section 4.5 implements the monitoring and diagnosis of dynamic AAS models with semantic computing technologies. These features are essential to realizing the self-adaptive CBE, which provides the ability to detect or predict the failure or potential issues of the system to reduce production line downtime.

The central objective of this thesis is to demonstrate the practicality of applying the proposed architectural design in an educational and experimental framework. It should be emphasized that while certain aspects are advantageous, they are not indispensable for fulfilling the main goal of the study. Specifically, the deployment of applications in real-time is a fundamental aspect yet its treatment can be postponed to future work, given that the primary focus of our case study is not centered on real-time operations. Moreover, the selection of tools for this project, though efficient, results in unpredictable delays. However, this factor does not undermine the principal results of the thesis, which is to ascertain the practical application of the architecture within a well-defined, experimental environment and can be expanded to cover real-time issue later on.

4.2 Modeling Environment Development

4.2.1 AAS Model Design Diagrams

P4M provides a graphical modeling environment, and different types of diagrams are implemented to describe different perspectives. AAS design diagrams are able to visually express the “who contains what” relationship to achieve clarity at varying degrees of precision. BOM (Bill Of Material) diagrams focus on the relationships between entities at the same level, emphasizing the interactions and connections between them. BPMN process diagrams provide a comprehensive way for process modeling to all business stakeholders. I participated as a member of a comprehensive series of tutorial videos¹ production. This resource is an excellent supplement for in-depth understanding and practical guidance of the P4M toolset.

AAS Design Diagram

The AAS design diagram (e.g. Figure 4.1) is an extension of UML class diagram which defines the detailed information of an asset. Within this diagram, one can list and display all information related to an asset. In order to better organize and give a clear structure, different aspects of information are grouped into different submodels.

We can add submodel elements as additional layers of information for each submodel block. The IDTA² (Industrial Digital Twin Association) provides a list of submodel templates to provide a uniform for the submodels. This helps to improve the interoperability of the AASs defined by different users. For example, the submodel Nameplate help to shape the asset nameplate information in a unified manner. The P4M provides the functionality of importing those .aasx template files.

¹<https://www.youtube.com/playlist?list=PL9nkS1KDTMm7IH0ucDZ7Yj1JyZnwSxTk9>

²<https://industrialdigitaltwin.org/en/>

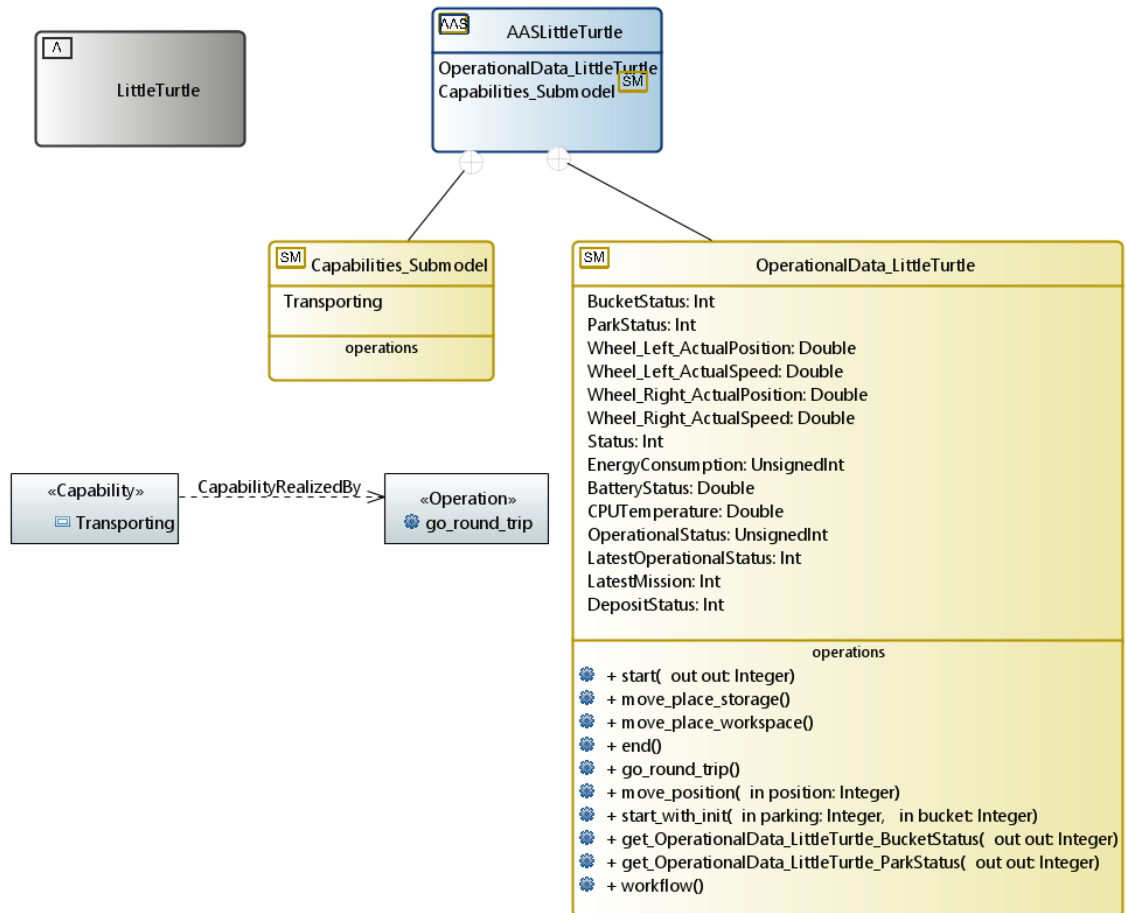


Figure 4.1: AAS design diagram

Composite AAS Modeling

I4.0 components represented by an AAS can be combined into a new I4.0 component. Then, certainly a composite I4.0 component should provide a composite method to show the sub-components it contains. By defining a bill of material (BOM) in a submodel, all the sub-components can be listed here as “Entities”. These entities can be classified into two types, co-managed or self-managed. Co-managed entity refers to an entity that needs to be managed by a higher-level entity, because it does not have its own asset administration shell. On the contrary, a self-managed entity has an AAS attached to itself. The connections between two entities are realized by their joint properties. As defined in [10, 23], if two entities are connected with each other, it means that at least one property of one of the assets is set into relation with at least one matching property of the other asset. By a property of an asset, we mean a property defined in one of the submodels of this asset, for instance, the number of objects remaining in stock.

In order to visualize the composite AAS and the connections between sub-components, it is possible to create a BOM diagram for the BOM Submodel of the composite AAS, which is a customized UML composite structure diagram. Figure 4.2 shows the customized BOM diagram view in P4M, where the example diagram is the composition of a robotic cell. It shows the interactions of the five assets in a closed environment. The exchange data between them are property values related to a part. In detail, the property item number of the storage holds the value that counts the total parts in the storage; the property load of Niryo Ned, UR3e, the TurtleBot3 WP, and the conveyor belt hold a value corresponding to whether a device is loading a part; the properties counter 1 and 2 of a working scenario

hold an accumulated number of parts passing each transporter.

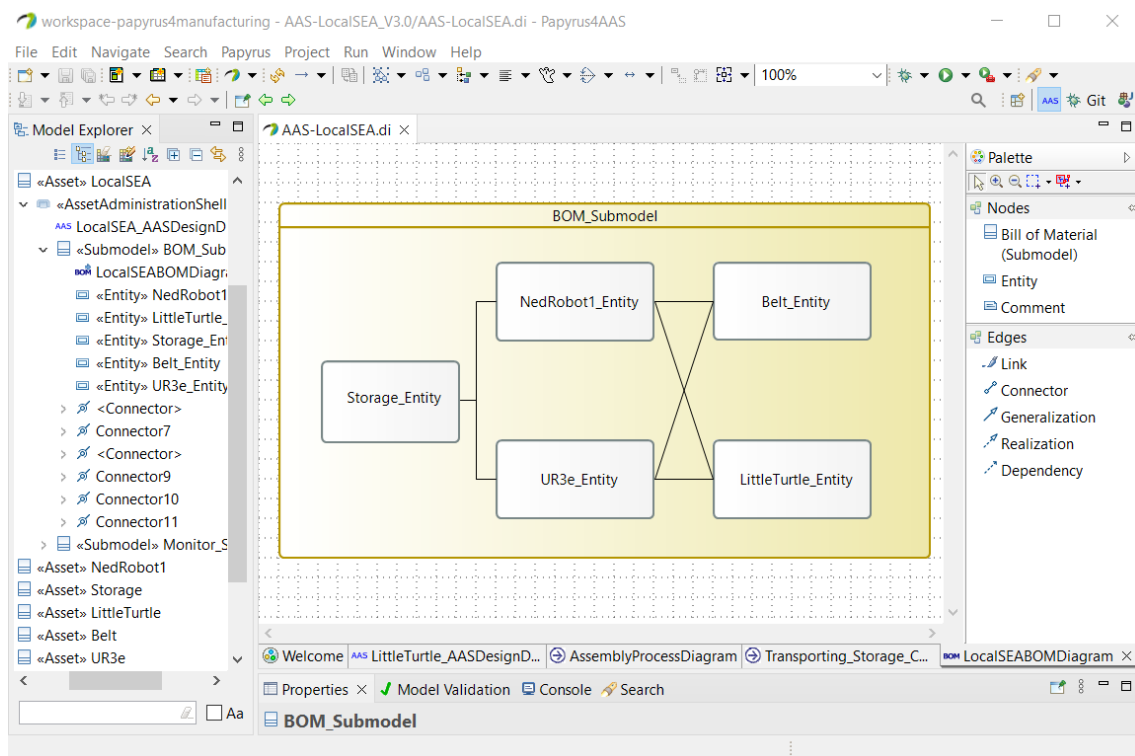


Figure 4.2: Bill of material composition diagram

Process Modeling

The production process is also represented in an Asset Administration Shell with a submodel “Work Plan” where we define the process steps and requirements. BPMN allows the creation of end-to-end business processes designed to convey a variety of information to a wide audience. We choose to attach the process model to the submodel element “operation” as shown in Fig. 4.3, from this submodel element, a BPMN Process diagram is created to describe the tasks and possibly execute them. This BPMN Process diagram is a customized UML activity diagram composed of a subset of BPMN concepts as mentioned in 3.3.1. The use of this standardized diagram can allow non-specialists to define production processes.

- The “Lane” is reused to represent different production process steps to distinguish work on different production equipment.
- “Tasks” are atomic activities in the process. Different types of tasks are defined in BPMN to distinguish the types of inherent behaviors that tasks can express.
 - “Task” that is not further specified is called “abstract task”.
 - A “Service Task” is a “Task” that uses some sort of service, which calls an operation during execution.
- The “Gateways” are used to control the convergence and divergence of sequence flows in the process, which is a gating mechanism that allows or prohibits passing through the gateway. Gateway is not necessary, if there is no need for control flow in the production process.

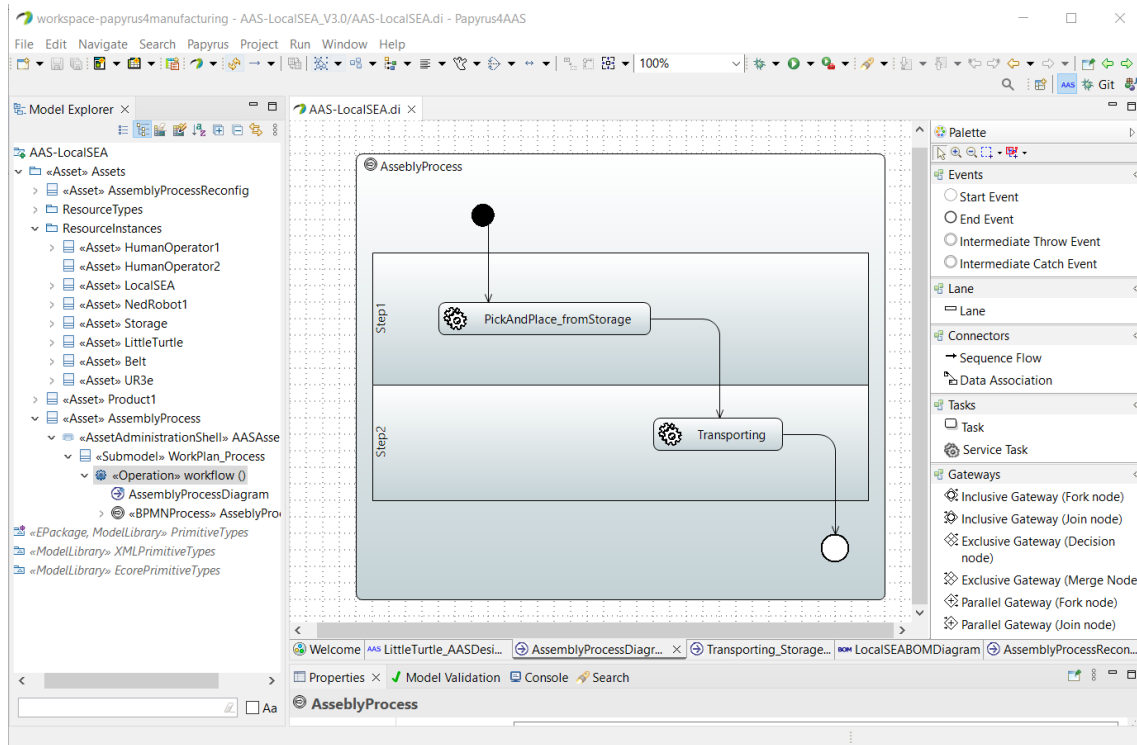


Figure 4.3: Workflow BPMN process diagram

4.3 Capability Checking @ Design Time Development

When we implement the capability checking architecture, the numbers used in Figure 4.4 represent the implementation process of their tagged stage in Figure 3.11. This entire capability checking feature is developed as an Eclipse plug-in bundled with P4M [35].

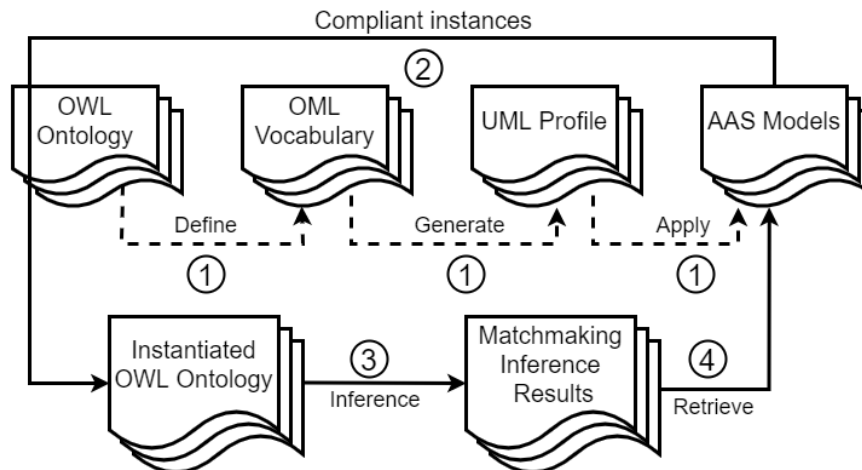


Figure 4.4: Capability checking implementation workflow

1. Semantic annotation.
 - (a) Define. The original MaRCO is defined in OWL format, while the OML Adapter provides only the conversion from OML to UML profile. In order to reuse the adapter, the first step is to define the selected part of MaRCO

concepts into OML vocabulary. As can be seen from, some attributes but not all of them are chosen from the original ontology. Although some information is ignored in this step, these vocabularies can still correspond to the ontology concept with the same name in the final conversion process, because we have defined the same URL as the original ontology in the OML file.

- (b) Generate. This converting process is realized by the OML Adapter, which generates UML profile from the OML files defined in the previous step.
 - (c) Apply. The generated MaRCO UML profile file will be applied to the AAS model as stereotypes. The model designer will refine the AAS models based on the properties of the equipments and capabilities provided in the profile.
2. Compliant instances. This step concerns to regenerate the specified AAS models to MaRCO compliant instances in an OWL file. All AAS models and the information stored in the stereotypes that come from the MaRCO profile will be converted as OWL individuals that conform to the MaRCO ontology.
 3. Inference. In this phase, the instances to be selected will be inferred from the newly generated individuals and the production process required capabilities, thanks to the capability matchmaking SPIN rules embedded in MaRCO ontology.
 4. Retrieve. The related inference results can be processed and selected by a SPARQL query (Listing 4.1). The results will point to the originally designed AAS models with the matching between OWL instances and AAS models in P4M.

Our capability checking implementation involves three different modules: (A) the model transformation module for the ontology concept conversions between different file natures, (B) the capability matchmaker module for inferences, and (C) user interface module for launching capability checking requests and displaying the reasoning results in Papyrus4Manufacturing. To implement the above functions, we have selected two well-established jobs. One is OML Adapter, which is used to convert OWL to UML profile. The other is MaRCO ontology, on the one hand, because the description of capabilities in manufacturing perfectly suits our needs. On the other hand, it also provides complete inference rules for capability matchmaking.

4.3.1 Model Transformation Module

The model transformation module provides a round-way transformation between OWL ontologies and UML models. The three dotted steps in Figure 4.4 correspond to the first stage introduced in Section 3.4.1, which enriches AAS models with semantic annotations in the manufacturing capability domain. As mentioned earlier, once generated from the ontology, this UML profile can be reused for all forthcoming actions.

The OML adapter takes care of the transformation from OML vocabularies to UML profiles. However, the MaRCO ontology was initially described in OWL format. To better use the existing works, we need to first define OML vocabularies referencing the original OWL ontology. Figure 4.5 shows an example of OML syntax of semantic concept expression that keeps aligned with the example of OWL concept structure of MaRCO (Figure 4.6).

After confirming the definition of OML vocabularies and the corresponding relationship between these concepts and UML meta-model, we can obtain the MaRCO UML Profile (Figure 4.7) through OML adapter. Here we will briefly introduce some concepts from the MaRCO ontology involved in this capability matchmaking process. The MaRCO ontology is composed of several distributed ontologies. By using the OML Adapter, a subset of MaRCO vocabularies is transformed into a UML profile that can be applied to AAS models

```

////////////////////////////////////CLASSES////////////////////////////////////
/**
 * Resource
 */
concept Resource
concept Device :> Resource
concept DeviceCombination :> Resource [
  restricts relation hasIndividualDeviceOrDeviceCombination to min 2
  restricts some relation hasIndividualDeviceOrDeviceCombination to IndividualDevice
  restricts some relation hasIndividualDeviceOrDeviceCombination to RealDeviceCombination
]
concept FactoryUnit :> Resource
//Device
concept DeviceBlueprint :> Device
concept IndividualDevice :> Device [
  restricts relation hasDeviceBlueprint to exactly 1
]
//DeviceBlueprint
concept GraspingDevice :> DeviceBlueprint
concept MovingDevice :> DeviceBlueprint
concept ModifyingDevice :> DeviceBlueprint
//IndividualDevice
concept IndividualGraspingDevice :> IndividualDevice
concept IndividualMovingDevice :> IndividualDevice
concept IndividualModifyingDevice :> IndividualDevice
//DeviceCombination
concept RealDeviceCombination :> DeviceCombination
concept TestDeviceCombination :> DeviceCombination
//FactoryUnit
concept Cell :> FactoryUnit

```

Figure 4.5: Exemplary OML syntax

as stereotypes, including different sub-classes of the concepts. The capabilities are separated into simple capabilities like *Moving* and combined capabilities like *PickAndPlace*, and these capabilities have parameters to describe their characteristics. The combined capabilities are compositions of simple or other combined capabilities, these information are defined in the Capability Model ontology. The resource model stereotypes define different resource types, including atomic resources (*DeviceBlueprint* and *IndividualDevice*) as well as different resource combination types including *DeviceCombination*, and the combination at the *FactoryUnit* level. The concepts of *Product*, *Process* and a selection of *ProcessTaxonomyDescription* have been included in the UML profile as well.

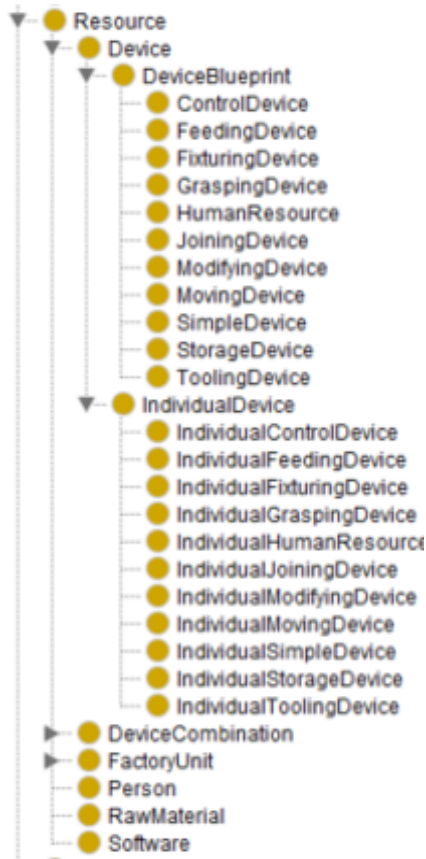


Figure 4.6: MaRCO screenshot in Protégé

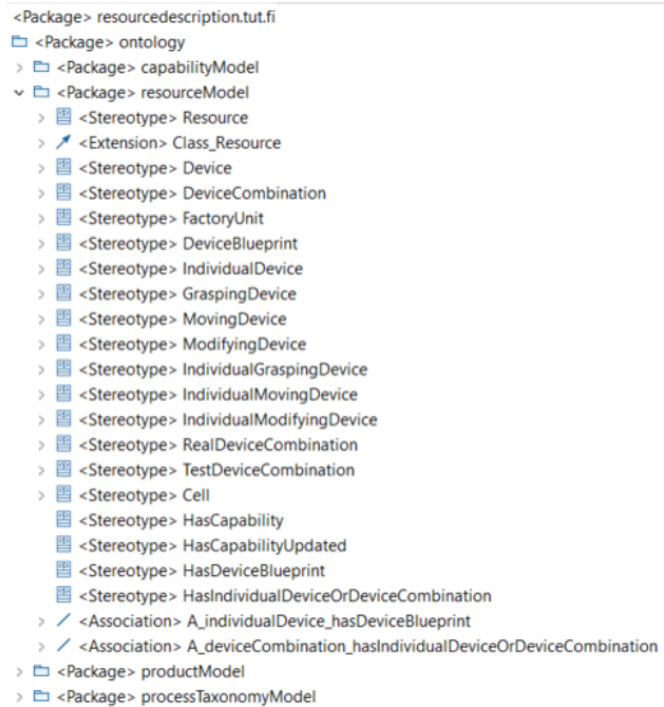


Figure 4.7: MaRCO UML profile

The MaRCO concepts in the generated UML profile are applied as stereotypes to the AAS models. The concrete mapping rules are described in Section 3.4.1. The designer improves the AAS model based on the device properties and capabilities provided in the configuration file. The system designer should refine the stereotyped AAS models based on the properties of the equipment and capabilities. The semanticID concept is designed to refer the semantic meaning of the submodel or submodel element of an AAS model. So when we assign a MaRCO concept to an AAS element, the semanticID should refer to the IRI of this concept in the ontology.

The second step refers to the second stage in the capability checking architecture (Figure 3.11), which generates the MaRCO concept instances from the AAS system model for further inferences. Based on the APIs provided by the org.eclipse.uml2.uml and org.semanticweb.owlapi packages, we developed a converter. This plugin serves for the transformation from stereotyped UML models to OWL individuals. All AAS models and the information stored in the stereotypes that come from the MaRCO profile will be converted as OWL individuals that conform to the MaRCO ontology (AAS2MaRCO transformation).

Three important variables and data structures are integral to the conversion process.

- OWLDataFactory *df*: to handle the creation of OWL entities.
- Map <Element, OWLNamedIndividual> *individualMap*: to map the UML elements with OWL entities.
- List<Triple<OWLObjectProperty, Element, OWLNamedIndividual>> *opTriple*: to save the triple between OWL object properties, UML elements, and OWL named

individuals.

The conversion process is managed through methods like `createOWL(Resource resource, OntologyManager manager)`, which converts an entire EMF resource or UML model into an OWL ontology. The `convertElement(Element current, OntologyManager manager)` method handles the conversion of individual UML elements to `OWLNamedIndividuals`, considering their `semanticID` for correspondence mapping. Property handling is a critical aspect of this class. The `addProperties(...)` method adds object and data properties to an OWL individual, reflecting the properties of the UML model. Finally, the generated ontology is saved in RDF XML format. For example, in Figure 4.8 shows the generated MaRCO individual of a Process model “AASProcess1” that belongs to the Process class definition and has two object properties.

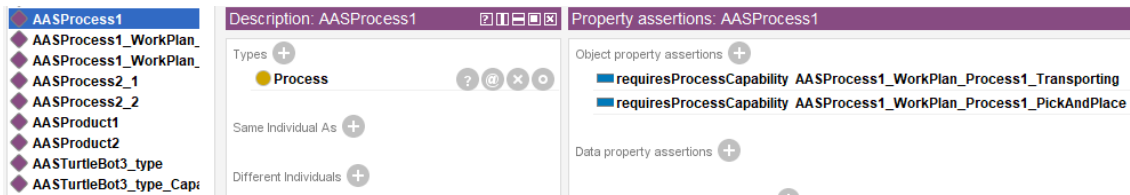


Figure 4.8: Exemplary OWL instance

4.3.2 Capability Matchmaker Module

The capability matchmaker is responsible for resource combination and combined capability computation, as well as the matchmaking reasoner which aligns the corresponding capabilities between production processes and resources. The implementation of this module reuses as much as possible other existing open-source projects. The MaRCO ontology and the associated SPARQL queries and SPIN rules are open-source [93].

By leveraging all the existing packages, we have implemented a matchmaking plugin that performs the automatic capability matchmaking. This plugin includes the following components:

1. Initialization: Initializing a client for matchmaking and setting up locations for input and output data. This includes loading and managing ontological models using the Apache Jena framework, a popular Java framework for working with RDF and OWL ontologies.
2. Matchmaker: It reads in ontological models, sets up matchmaking conditions, and performs the matchmaking process. The process involves finding required capabilities, generating combination possibilities, and executing matchmaking rules. The pre-defined SPARQL queries update the capabilities for the individual devices and compute combined capabilities for the device combinations.
3. Rule Executor: The SPIN rules integrated in the Parameter Rule ontology are executed in order to infer these novel capabilities' parameters. The SpinAPI (provided by TopBraid) is used for the reasoning process. And the SPARQL queries can be executed by Openllet reasoner.
4. Result Processor: After the matchmaking process, results are processed and formatted, in order to be used in further steps of the planning or, more precisely, the simulation process in Papyrus4Manufacturing.

The matchmaking reasoner deals with the matching between capabilities required by the process and capabilities provided by the newly updated resource system. During this process, not only are the capabilities matched at the name level *has capability match*, but also the adaptations of the parameters *can be implemented with* are computed. These reasoned relationships and inferred elements are saved in a separate file (match.ttl).

4.3.3 User Interface Module

This user interface ties the above two modules together and establishes a relationship with the model in the modeling environment. The usage scenario we envisage is shown in Figure 4.9. The module is mainly composed of 3 parts:

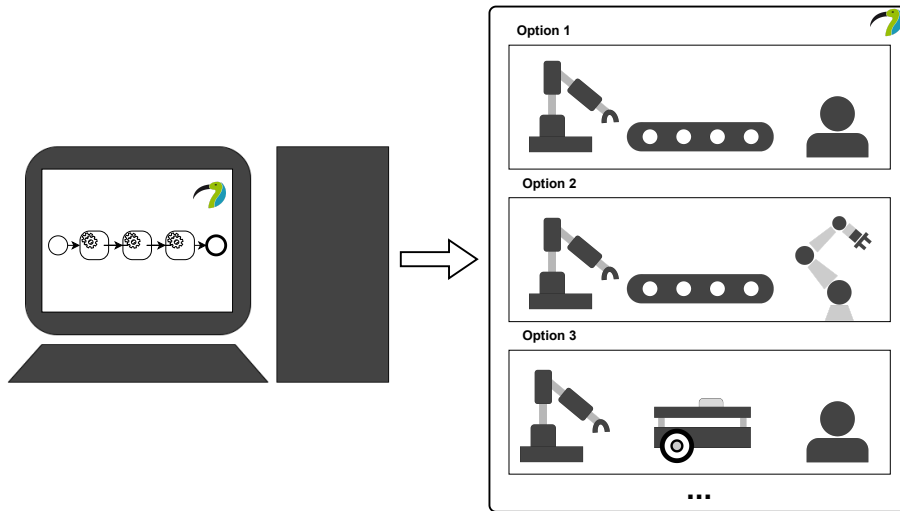


Figure 4.9: User interaction scenario

1. **Simplicity in Selection:** Designed for users without specialized technical expertise, this tool should feature a user-friendly interface for the production process selection. It incorporates straightforward elements like right-click menus and pop-up windows with checkbox. These components simplify the organization and selection of models, aiding users in easily choosing the required products and processes to perform matchmaking.
2. **Sequential Invocation of Components:** The user's action triggers a series of backend processes starting with the AAS2MaRCO converter.
3. **Result Presentation:** The result retrieval aims to integrate and extract the results of ontology inferences, return them to the user, and save them for later use. We defined a SPARQL query (Listing 4.4) to automatically extract information from newly reasoned relationships. We want to select the equipment (either an individual or a combined device) that can provide the capabilities required for the production process through the query (Line 10). Via Lines 13-15, it is possible to select all processes participating in the capability checking. Lines 17 select the capabilities required for the aforementioned production processes. The eighth line finds the devices capable of implementing the required capabilities. The capability matchmaking results show the *DeviceBlueprints* or *DeviceCombinations* that can realize the capability. However, in our application, the production process is realized by the device instances (*IndividualDevices*). So when the result is a *DeviceBlueprint*, we will find all available *IndividualDevices* belonging to it (Line 22-25). The results are

sorted out via a popup window for the users to choose from. And the inferred information is again connected to the AAS digital twin models. The selected information can then be included as input for a feasibility checking or device deployment step coming next.

```

1 # created for aas models use only
2 PREFIX mmo: <https://resourcedescription.rd.tuni.fi/ontology/MatchmakingOntology#>
3 PREFIX pm: <https://resourcedescription.rd.tuni.fi/ontology/productModel#>
4 PREFIX cm: <https://resourcedescription.rd.tuni.fi/ontology/capabilityModel#>
5 PREFIX rm: <https://resourcedescription.rd.tuni.fi/ontology/resourceModel#>
6 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
7 PREFIX owl: <http://www.w3.org/2002/07/owl#>
8 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
9
10 SELECT distinct ?process ?requirement ?required ?match ?deviceBlueprint ?
    deviceCombination ?individualDevice
11 WHERE {
12     # get all activity instances
13     ?activityCls rdfs:subClassOf+ pm:Activity .
14     ?process rdf:type ?activityCls .
15     filter not exists { ?process mmo:ignoreProcess true } .
16     # get process capability if any
17     ?process pm:requiresProcessCapability ?requirement .
18     ?requirement pm:matchmakingRequired ?required .
19     optional {
20         ?requirement mmo:hasCapabilityMatch ?match .
21         optional {
22             ?deviceBlueprint rm:hasCapability ?match .
23             ?individualDevice rm:hasDeviceBlueprint ?deviceBlueprint .
24         }
25         optional {
26             ?deviceCombination rm:hasCalculatedCapability ?match .
27         }
28     }
29     # get duration from performance i.e. how long it takes to execute step with
    this match
30     optional { ?performance pm:duration ?time }
31 }
32 order by ?requirement

```

Listing 4.1: SPARQL query for result extraction

4.4 Skill Execution & Orchestration Development

4.4.1 Deployable AAS Server Architecture

The code generation mechanism in P4M simplifies implementation over the BaSyx middleware, on the one hand it neatly handles the deployment to the AAS HTTP server. By implementing this widely used and open mechanism, AAS digital twins can be easily accessed via the REST API while active on the AAS server. As a result, a range of IT-level applications, such as a range of data analytics applications and dashboards, can seamlessly connect to the AAS server, allowing them to obtain data and orchestrate directly from the AAS digital twin.

On the other hand, by utilizing BaSyx as middleware, critical communication links with physical assets are established. The communication protocol adapters supported by P4M have been mentioned above. This function has been implemented in P4M, and my task is to solve the problem of the dynamic orchestration of the digital twin models at the application level.

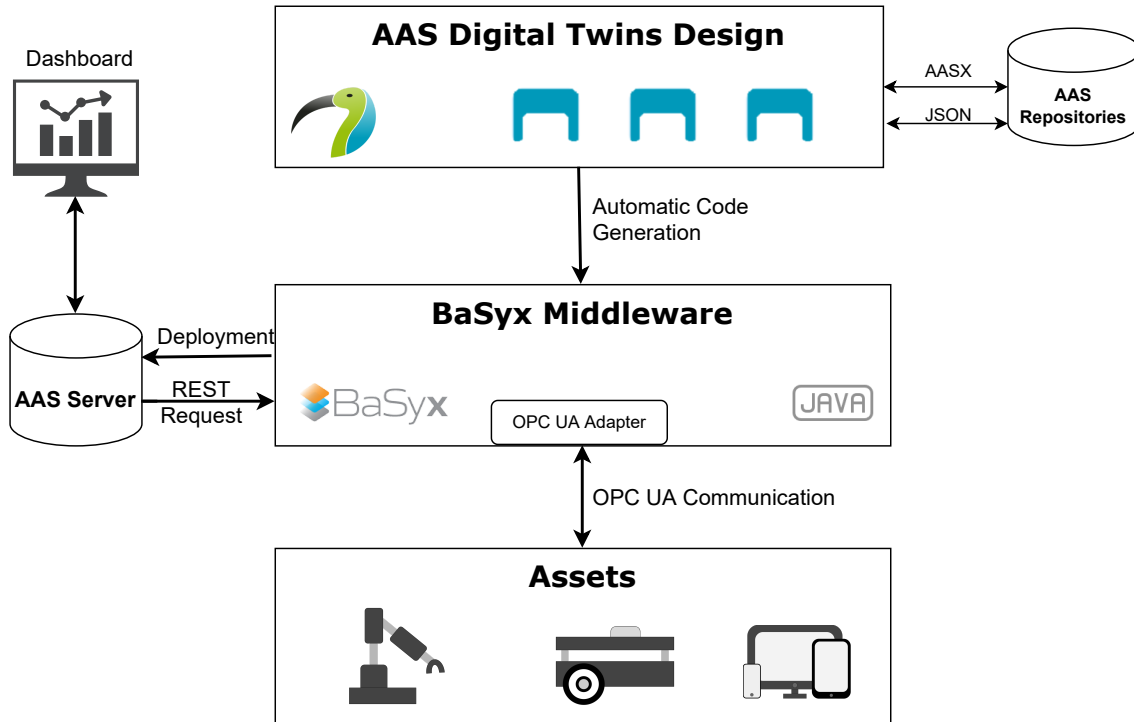


Figure 4.10: Papyrus4Manufacturing architecture

4.4.2 Node-RED Package Development

In order to bring the production process to real-world orchestration, we leveraged Node-RED, an innovative visual programming environment that enables seamless integration between physical devices and online services through APIs. This tool is particularly adept at transforming BPMN diagrams into executable workflows, exemplified in P4M where the BPMN process is effortlessly converted into a Node-RED flow (as illustrated in Figure 4.11).

The creation of the two specialized Node-RED nodes aims at facilitating the development effort. The idea behind developing these modules was to enable the execution of BPMN diagrams designed in P4M, thus orchestrating the AAS digital twin models. These nodes facilitate the orchestration of processes through REST APIs provided by BaSyx server. This offers a high degree of flexibility and control in process management. Firstly, the “AAS Operation” node allows for the direct calling of AAS Operations with just the necessary URL endpoint and input parameters. Secondly, the “AAS Property” node provides a straightforward way to manage the values associated with a URL endpoint.

Table 4.1 presents a mapping between BPMN concepts and their corresponding Node-RED nodes, providing a clear guide for translating process diagrams into an executable workflow. The *Start Event* in BPMN is associated with the *HTTP IN* node from the network package provided in Node-RED by defaults. Conversely, the *End Event* is linked to the *HTTP RESPONSE* node, signaling the process’s completion. For tasks denoted as *ServiceTask* in BPMN, the *AAS Operation* node from the *aas* package is used, while *ServiceTask* with Dataflow is translated into the *AAS Property* node, both facilitating interaction with the AAS digital twins. Timing control, represented by the *Timer intermediate* in BPMN, is executed using the *delay* function in Node-RED. Lastly, decision-making in the process, indicated by the *Exclusive Gateway* in BPMN, is implemented through the *switch* node from the function package in Node-RED.

The underlying technology stack for the customized Node-RED nodes combines the programming language JavaScript for scripting functions, with HTML for defining the user

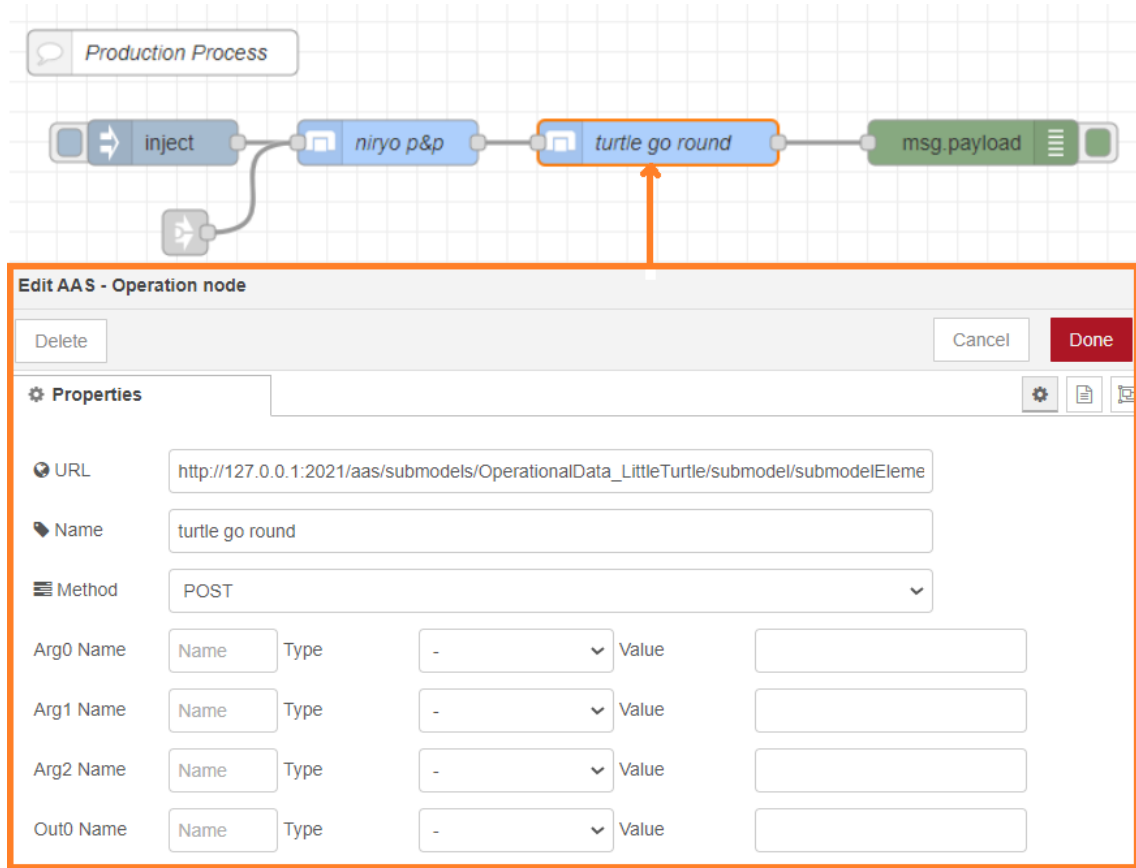


Figure 4.11: Node-RED flow representation of Process1

interface. JavaScript’s ubiquity and flexibility make it a choice for creating the functional aspects of the nodes, while HTML’s standardization across the web allows for a familiar and widely supported method of structuring the nodes’ visual elements. This combination ensures that the nodes are not only powerful in terms of functionality but are also user-friendly and accessible to those who may not have extensive programming experience.

These custom nodes enhance the automation and efficiency of production processes. This integration represents a significant step forward, as it bridges the gap between design and execution, allowing for sophisticated process management through digital twins.

| BPMN Concept | Node-RED Node |
|----------------------|---------------------------------|
| Start Event | HTTP IN (network package) |
| End Event | HTTP RESPONSE (network package) |
| ServiceTask | AAS Operation (aas package) |
| ServiceTask Dataflow | AAS Property (aas package) |
| Timer intermediate | delay (function package) |
| Exclusive gateway | switch (function package) |

Table 4.1: BPMN and Node-RED nodes mapping

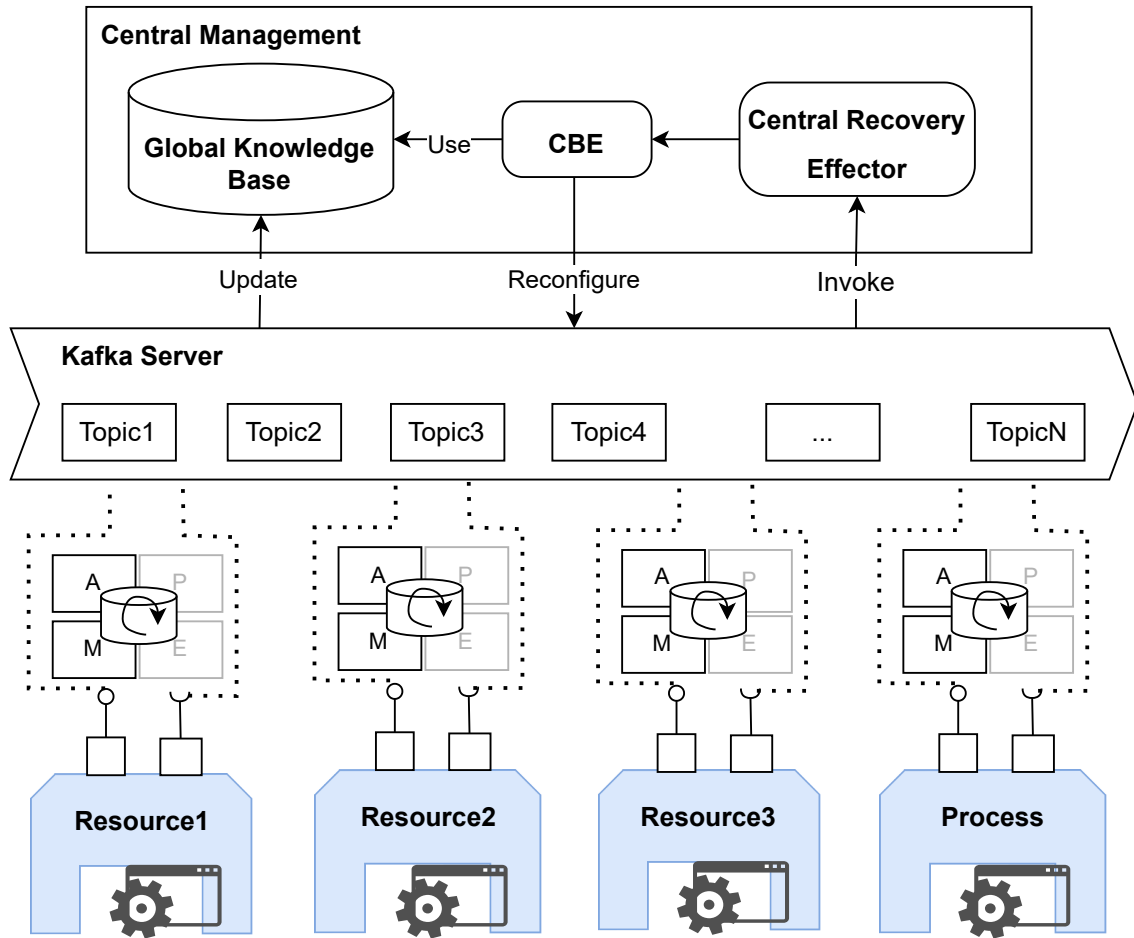


Figure 4.12: Overall run-time monitoring & diagnosis implementation

4.5 Monitoring & Diagnosis Development

There are several important modules to realize the self-adaptation loop, which includes the monitor, analyze, plan, and execute modules as seen in Figure 4.12 and Figure 3.13 presented in Section 3.5, also including the integration with the knowledge base. The right part shows the plan and execution modules will not be detailed in the following sections because they are out of the scope of this thesis. A Kafka server is integrated for the global message transfer.

4.5.1 Local Monitor Knowledge Generation

Subsequent to the detailed model design, an automatic generation mechanism facilitates the transformation of UML AAS models into ontology individuals. This procedure results in creating an OWL file for each monitor submodel. This file includes all ontology model instances related to monitoring, inclusive of the submodel itself.

Among them, this submodel will be converted into a Monitor. If its observation object is an *ObservedProperty* or *ObservedOperation* type, it will simply produce the corresponding element. If the observation object is an *ObservedSkill*, it will traverse the properties and operations that it participates in the sub-process and convert them into *ObservedElements*. This ontology will be generated along with the dynamic model described in the next section. This generated ontology imports the monitoring ontology presented in Section 3.5.5.

The generation of *DiagnosisRules* into CSPARQL Queries leverages the attributes de-

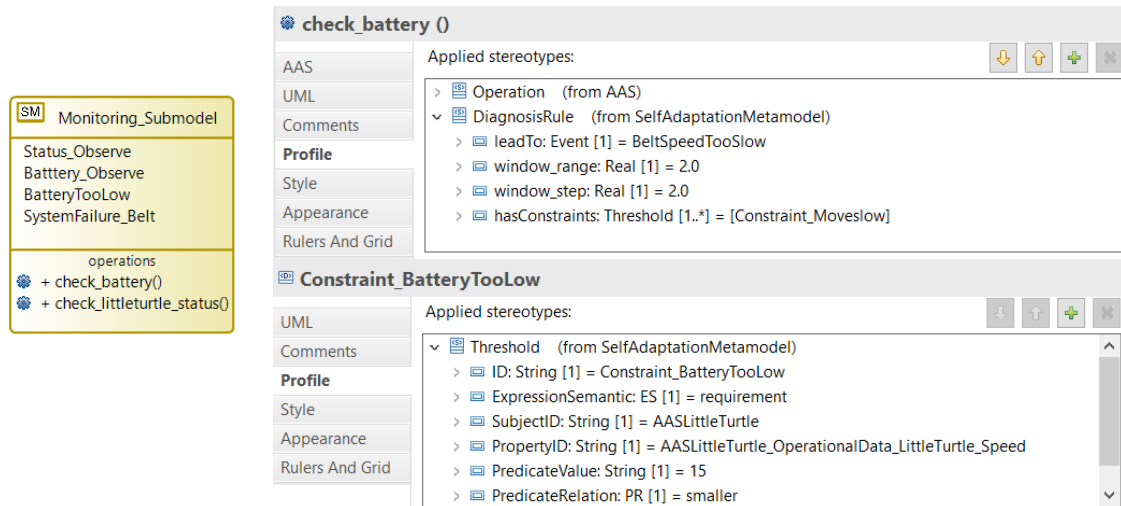


Figure 4.13: Diagnosis rule modeling in P4M

scribed within the metamodel. Since the metamodel inherently contains the necessary information, it facilitates a seamless conversion, ensuring that the rules are comprehensive and aligned with the system’s data structure (From Figure 4.13 to Listing 4.4). This translation into CSPARQL, a query language for streaming RDF data, enables real-time processing and diagnosis of data streams, which is crucial for dynamic environments where conditions change rapidly. The capability to generate CSPARQL queries from *Diagnosis-Rules* underscores the system’s adaptability and intelligence. By utilizing the predefined attributes in the AAS model, the system can automatically formulate queries that are tailored to the specific diagnostics required. The automated generation of queries minimizes the likelihood of human error and reduces the need for manual intervention, resulting in a more robust and reliable diagnostic process. As these queries are executed, they can detect anomalies, predict potential issues, and suggest corrective actions, thereby supporting proactive maintenance and decision-making.

4.5.2 Data Acquisition and Real-Time Stream generation

Real-time data plays an indispensable role in the digital twin system, which is also the basis of realizing the self-adaptive system. The AAS servers offer connectivity to devices and make real-time operational data accessible in json format. The monitor module is supposed to interact only with the AAS servers but not OPC UA server level. To have a more stable and manageable message distributing system, we have implemented Kafka server for the data sharing. However, the data need to be pre-processed by adding semantic value to raw data. A stream generator offers the transformation functionality from JSON stream data to RDF stream.

A Kafka producer generates the information of an AAS server to different Kafka topics. When an AAS is registered to the AAS register, the producer continuously fetches data from this AAS server. It sends the data to a Kafka topic and then fetches and sends the *SubmodelElements* with a given frequency. The code (Listing 4.2) sets up the Kafka producer configuration. It sets producer acknowledgments to “all” to ensure that record writes are fully acknowledged. The retries configuration is set to 0, indicating that it won’t retry sending a message if an error occurs. Buffer size and linger time are configured to control the batching behavior of the producer. The *buffermemory* setting allocates memory for the producer’s buffering, which holds records that haven’t been transmitted to the server yet. The producer uses string serializers for both the key and the value, which means

that the producer will convert keys and values to strings before sending them. An instance of `KafkaProducer` named `producer_turtle` is created with the configured properties. This producer is meant to send string keys and values. Within a try block, a `KafkaAASProducer` object named `ksp_turtle` is instantiated with the URL `http://127.0.0.1:2021/aas/`. The method “`publishAAS()`” would then be responsible for the actual data publishing.

```

1 //Assign topicName to string variable
2 String topicName = "TestAASSubmodelProducer";
3
4 // create instance for properties to access producer configs
5 Properties props = new Properties();
6
7 //Assign localhost id
8 //It specifies the Kafka server location with localhost:9092, indicating that Kafka
   is running on the local machine at port 9092.
9 props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
10
11 //Set acknowledgements for producer requests.
12 props.put("acks", "all");
13
14 //If the request fails, the producer can automatically retry,
15 props.put("retries", 0);
16
17 //Specify buffer size in config
18 props.put("batch.size", 16384);
19
20 //Reduce the no of requests less than 0
21 props.put("linger.ms", 1);
22
23 //The buffer.memory controls the total amount of memory available to the producer
   for buffering.
24 props.put("buffer.memory", 33554432);
25
26 props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer"
   );
27 props.put("value.serializer", "org.apache.kafka.common.serialization.
   StringSerializer");
28 Producer<String, String> producer_ned = new KafkaProducer<String, String>(props);
29 try {
30     KafkaAASProducer ksp_trutle = new KafkaAASProducer(producer_ned, "http
   ://127.0.0.1:2021/aas/");
31     ksp_trutle.publishAAS();
32 } finally {
33     producer_ned.close();
34 }
35 }

```

Listing 4.2: Kafka producer example

Each monitor module has a Kafka consumer stream generator that interfaces with a Kafka cluster, consumes data from monitor-relevant topics, and processes it. It is notable that the consumer stream generator is designed for RSP (RDF Stream Processing) reasoning use. It is initialized using a stream URI, Kafka topics, and Kafka consumer properties. The class continuously polls monitor related Kafka topics for new records, with each record being transformed into a JSON object that’s appended with a timestamp. This processed data is then added to a data stream.

Once the JSON stream is formed, in order to perform subsequent RSP reasoning, we need to convert the JSON stream into RDF triples. RML is used to describe rules for transforming structured data into RDF datasets, and we need to define the RML mapping rule in a ttl file. This mapping is applied using the `CARMLJSONMapper` class, and the resulting RDF triples are added to a `DataStream`. The data in this stream is expected to be in JSON format, as indicated by the `rml:referenceFormulation ql:JSONPath` directive (Listing 4.3). A triples map, “`MesurementMapping`” uses the previously defined logical source. For each iteration over the source: A subject is created based on the values

```

1
2 # definition of a logical source
3 <source> rml:source [
4   a carml:Stream ;
5   carml:streamName "MYSTREAM" ;
6 ] ;
7 rml:referenceFormulation ql:JSONPath ;
8 rml:iterator "$" ;
9
10 # definition of a triple map
11 <#MeasurementMapping> a rr:TriplesMap ;
12 rml:logicalSource <source> ;
13 rr:subjectMap [
14   rr:template "MYSTREAM
15   #{idShort}_measurement_{timestamp}" ;
16   rr:class monitor:Measurement ;
17 ] ;
18
19 rr:predicateObjectMap [
20   rr:predicate monitor:isOwnedBy ;
21   rr:objectMap [
22     rr:template "MYSTREAM#{idShort}" ;
23   ] ;
24
25   rr:predicate
26     monitor:PredicateValue ;
27   rr:objectMap [
28     rml:reference "value" ;
29   ] ;
30
31   rr:predicate monitor:Timestamp ;
32   rr:objectMap [
33     rml:reference "timestamp" ;
34   ] ;
35 ] ;
36 }

```

Listing 4.3: RML mapping

extracted from the source (namely, `idShort` and `timestamp`).

4.5.3 Runtime Analysis

The analyzer first loads the monitoring ontology and then runs CSPARQL queries on the data stream generated by the monitor using the CSPARQL engine. CSPARQL queries must be registered with the CSPARQL engine to process the RDF data stream. These queries check for specific conditions in the operational data coming from the monitor stream. Different *Events* will be constructed according to the diagnosis rules. Listing 4.4 is an example of CSPARQL query to verify the battery status of LittleTurtle.

The runtime diagnosis implementation consists of different steps.

- Initialization of Continuous Query Execution:

The `JenaContinuousQueryExecution` registers a new continuous query with the CSPARQL engine. The query and configuration are passed as parameters.

- Query Type Check:

The step checks the type of the CSPARQL query (whether it is a CONSTRUCT or SELECT query). CONSTRUCT queries typically create new RDF triples based on the query pattern. SELECT queries select results from existing triples.

- Continuous Query Execution:

```

1 PREFIX mo: <http://cea.list.papyrus4manufacturing/monitoring#>
2 PREFIX turtle: <http://cea.list.papyrus4manufacturing/monitoring/turtle#>
3 PREFIX : <https://www.geldt.org/stream#>
4 REGISTER RSTREAM <http://out_speed/stream> AS
5 CONSTRUCT {turtle:BatteryTooLow a mo:Warning}
6 FROM NAMED WINDOW <w> ON <http://example.org/test/rdf> [RANGE PT2S STEP PT2S]
7 WHERE {
8   WINDOW ?w {
9     ?measurement a mo:Measurement ;
10      mo:PredicateValue ?value ;
11      mo:isOwnedBy ?prop .
12   FILTER(?prop = turtle:BatteryStatus && ?value < \"15\")
13   }
14 };

```

Listing 4.4: CSPARQL query example

Continuous query execution implies that the query is evaluated continuously as the data changes, rather than just once. The configuration of the CSPARQL engine can be seen in Listing 4.5. This is particularly useful in dynamic environments where the data is frequently updated. Upon the type of resulting events, different effectors will be invoked.

```

1 # This sets the engine to use event time for processing, meaning that it relies on
   the timestamps of the events in the data stream itself.
2 rsp_engine.time=EventTime
3 rsp_engine.base_uri=http://streamreasoning.org/csparql/
4 rsp_engine.stream.item.class=org.streamreasoning.rsp4j.csparql2.stream.
   GraphStreamSchema
5
6 # Sets the format for responses from the engine to JSON-LD (JSON for Linked Data)
7 rsp_engine.response_format=JSON-LD
8
9 # Query results should occur when a window closes.
10 rsp_engine.on_window_close=true
11
12 # The engine should only consider windows with non-empty content for processing.
13 rsp_engine.non_empty_content=true
14
15 # The processing or querying is not periodic.
16 rsp_engine.periodic=false
17
18 # The engine does not trigger actions or computations on every change in content.
19 rsp_engine.on_content_change=false
20
21 # Sets the engine's tick, or processing interval, to be driven by time.
22 rsp_engine.tick=TIME-DRIVEN

```

Listing 4.5: CSPARQL engine configuration

- Output Stream Consumer:

This step adds a consumer to the output stream of the continuous query execution. The consumer seems to be performing some form of adaptation action on the output. The exact nature of this action would depend on the implementation of adaptation action to be provided by AAS.

4.6 Conclusion

This chapter provides a holistic implementation of the CBSAM architecture mentioned in the previous section. P4M, a toolkit for the model design, the development, and deployment. The workflow development of the orchestration between different digital twins.

And a concrete implementation of the concept of capability-based engineering. Due to the time constraints, the development of the self-adaptive system has been limited to the monitoring and analyzing phase, and only some initial explorations and attempts have been made.

Chapter 5

LocalSEA Testbed

Contents

| | | |
|------------|--|-----------|
| 5.1 | Overview | 69 |
| 5.2 | Testbed Requirement | 70 |
| 5.3 | LocalSEA Testbed Composition | 71 |
| 5.3.1 | Hardware Architecture | 71 |
| 5.3.2 | Software Architecture | 73 |
| 5.3.3 | Communication Architecture | 74 |
| 5.4 | CBSAM Architecture Implementation | 77 |
| 5.4.1 | Specification | 77 |
| 5.4.2 | AAS Model Design | 86 |
| 5.4.3 | Engineering & Deployment | 95 |
| 5.4.4 | Operation & Maintenance | 98 |
| 5.5 | Conclusion | 99 |

5.1 Overview

In essence, the digital twin system is a cyber-physical construct that encompasses real world entities and digital entities. Thus, beyond offering a conceptual architecture, the ability to implement and validate it in reality becomes a critical step. To furnish a platform for validating research theories, our researchers have devised a testbed. One of the key goals is to establish a robust and flexible experimental base that can adapt to varying research needs and innovation trajectories. The digital layer, that is, the AAS integration part, is an important aspect of testbed to complete the development of digital twins. Furthermore, the utilization of an Industry 4.0 test bed promotes an in-depth understanding of the practical challenges and opportunities that may be encountered when transitioning from conceptual to practical application.

Developing a testbed is a complex task that entails constructing a specialized environment where both software and hardware can undergo rigorous testing. This environment is meticulously designed to replicate real-world conditions, ensuring that the tests conducted provide accurate and reliable results. The primary goal of such a testbed is to simulate the actual use cases and challenges that the system or application will face in the real world. These use cases are important for validating the research results and identifying potential issues in its intended environment. In Section 5.2, a set of requirements and limitations have been considered.

Section 5.3 presents an academic testbed developed in the laboratory for versatile utilization. This section provides extensive details to help readers comprehend the system

as a cyber-physical entity, encompassing more than just its virtual perspective that has been mainly analyzed since the beginning of this manuscript.

Section 5.4 allows for a full elaboration of the implementation details of the CBSAM methodology introduced in chapter 3. This section shows how to establish a flexible manufacturing system that is interoperable across entities, easily adapts to changes in demand, and has strong stability to ensure that the production process runs smoothly.

5.2 Testbed Requirement

The specific requirements for setting up a testbed can greatly vary, depending on the nature and complexity of the application or system being tested. This variation in requirements demands a high level of customization and careful planning to ensure that the testbed effectively addresses the unique aspects of each application. The development of a testbed must also consider future scalability and adaptability, allowing for system updates or new features for future applications. The consideration of real life limitations is also important for the realization of a feasible and reasonable testbed. Here are some requirements that have been considered when developing this testbed:

1. **Context: Identifying the context.**

Our testbed is designed to serve multiple objectives, primarily centered on the realm of Industry 4.0. In terms of context, the testbed will be implemented in scenarios that are inherently aligned with the paradigms of Industry 4.0 and it should follow the RAMI4.0 model. One of the main objectives of this testbed is to test and validate the realization of CBSAM architecture presented in Chapter 3.

2. **Methodology: Selecting an appropriate methodology model.**

Using a methodology model for a testbed development provides a systematic and organized way to ensure that every aspect is considered, thereby promoting consistency across different projects. This systematic approach provides clear direction, ensuring the understands of the steps and goals. The consistency provided by a methodology also supports scalability for different projects. The methodology to develop the testbed is mini-waterfall with triggers. Mini-waterfall inspired by the well-known waterfall methodology also contains fives phases: (1) specification, (2) design, (3) implementation, (4) testing, and (5) maintenance. More details about the development process and evolution of this testbed are presented in work [94].

3. **Environment: Considering real-life limitations.**

The testbed should closely mimic the actual production system in the real industry. Therefore, the selection of the configuration of each part such as hardware, software, and network needs to be close to reality, such as selecting a common industrial network protocol (like OPC UA). It is also necessary to consider some limitations that exist in real life, such as some devices that do not have intelligent control capabilities. Therefore, in addition to smart devices with powerful processing units, the components selected to make up the testbed also need to include devices with low processing capabilities and processor-less parts.

However developing a testbed, particularly in an academic setting, must navigate various real-life constraints and considerations, among which budget and test boundaries are prominent. The choice of robots for education use is, therefore, a decision influenced by balancing budget constraints with the need for sufficient technical complexity to ensure meaningful testing and research outcomes. Furthermore, there is

an absence of a wireless network with guaranteed service within this testbed’s design and operation.

4. **Composition: Determining the composition of testbed.**

According to [95], a general outline of the construction of an instrumental I4.0 testbed is provided. It needs to be composed of different building blocks, including digital entity, physical entity, interface, communication, sensor, data conversion, data storage, software, intelligence, etc.

5. **Scenarios: Defining test scenarios.**

There’s also a need to set the scenario for the testbed based on the content of the test. This scope might differ from testing a single module to the integration testing of multiple components or the entire system. Planning the testbed scenario helps ensure that the testing platform can offer a realistic and efficient testing environment.

5.3 LocalSEA Testbed Composition

A small-scale testbed has been developed in the laboratory named LocalSEA 5.1. The testbed provides an experimental environment to deploy technologies, demonstrations, and use cases addressing Industry 4.0. LocalSEA conforms to the RAMI4.0 model so that the data and information can be easily shared over different layers. This section is dedicated to present the LocalSEA testbed composition from several aspects, including hardware architecture, software architecture, and communication architecture. The hardware architecture subsection focus on the descriptions of different physical devices in the testbed. The software architecture discusses the frameworks, applications, and software tools utilized. The communication architecture aspect examines the networking and data exchange protocols in the testbed.

5.3.1 Hardware Architecture

This subsection introduces the overall physical elements of LocalSEA, which contains several components for production, communication, and control. It includes two robotic arms, an AGV, two buttons with a controller, a conveyor belt, storage and working areas, a multi server, end devices and human operators.

- **Niryō Ned¹**: NedRobot

A 6-axis collaborative robot designed for education and light industry tasks. Built with open-source software and hardware, it offers a user-friendly experience for robotics and automation learning purpose.

- **UR3e²**: UR

A flexible collaborative 6-axis robot arm designed for assembly tasks and automated workbench scenarios. Developed by Universal Robots, it offers precision control, easy programming, and safety features for human-robot collaboration.

- **TurtleBot3 Waffle Pi³**: LittleTurtle

A modular and customizable mobile robot platform designed for robotics research and education.

¹<https://docs.niryō.com/product/ned>

²<https://universal-robots.com/products/ur3-robot>

³<https://robotis.us/turtlebot-3-waffle-pi>

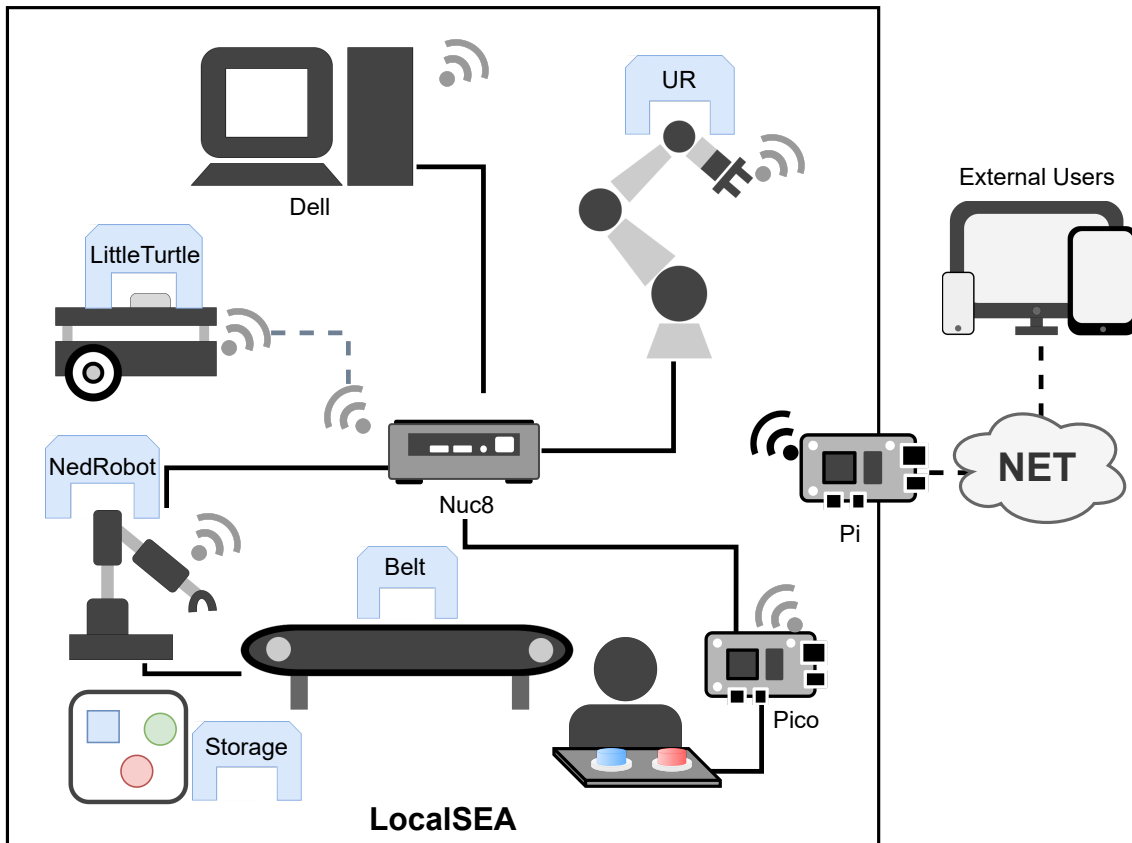


Figure 5.1: LocalSEA cell

- **Conveyor belt:** Belt

The Niryo Conveyor Belt is an accessory designed for automation and educational purposes, compatible with Niryo robots. And it is also possible to be further configured to work with other robots. It's equipped with two DC motors to manage both speed and direction and can be easily integrated with other devices through its user-friendly interface. Ideal for simulating industrial assembly line processes.

- **Storage zone:** Storage

A specific place for storing.

- **Workspace area:** WS

A dedicated space designed for specific tasks.

- **Raspberry Pico:** Pico

A low-cost, high-performance microcontroller board. It offers a powerful solution for embedded systems projects and DIY electronics.

- **Raspberry Pi 3 Model B+:** Pi

A credit-card-sized computer offers improved performance and connectivity. In LocalSEA, it is an I4.0 gateway that provides a connection to the internet.

- **Nuc8:** Nuc8

A compact and powerful mini PC, which acts as a multi-server in the testbed.

- **End devices:**

Hardware components that serve as communication endpoints in a network. These devices can range from personal computers, smartphones, and tablets for data or service requests.

- **Human operators:**

In the LocalSEA testbed, human operators primarily serve to perform activities that are too complex to be performed by a machine.

- **Button:**

A simple device that can send a specific signal or command when pressed. Using the button as a sign for users to indicate the completion of a task is a practical application in a testing environment.

5.3.2 Software Architecture

Software components can be programs and middlewares. Some of the physical entities have computational units, and some of them do not. For instance, the conveyor belt is controlled by Niryo Ned, which does not have a separate processor itself. Same for the buttons that Raspberry Pico controls. Other production devices such as the Niryo Ned robotic arm, UR3e robotic arm, and TurtleBot3 mobile robot have their own calculation unit. The software components are installed to these smart devices for different usages. The following list will detail the software components included in each intelligent physical entity.

- Nuc8:
 - OPC UA server: provides operational data and device operational access through the OPC UA information model that belongs to communication and information.
 - Docker Nned controller: a motion planner of Niryo Ned.
 - Node-RED server: for streamlined device communication.
 - Prosgresql server: storage and management of historical data.
 - Mosquitto broker: enables MQTT communication.
- NedRobot:
 - ROS 1
 - OPC UA-ROS Bridge
- UR3:
 - ROS 1
 - OPC UA-ROS Bridge
- LittleTurtle:
 - ROS 2
 - OPC UA-ROS Bridge
- Pico:
 - Button control

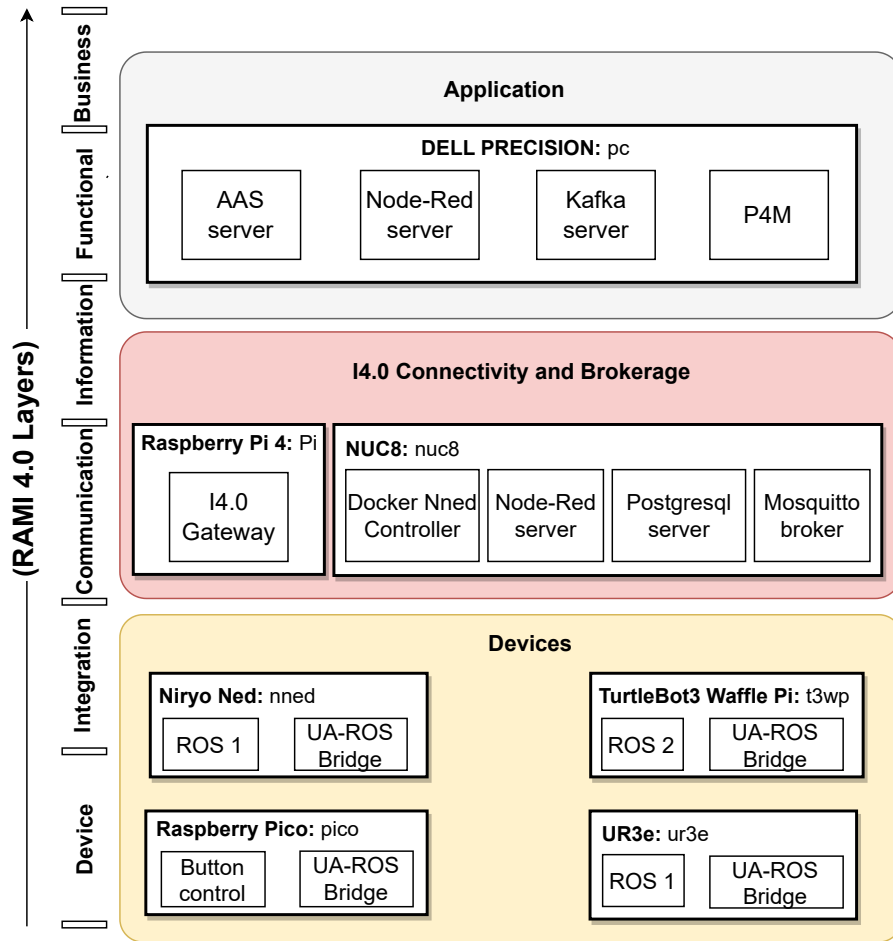


Figure 5.2: LocalSEA description

- OPC UA-ROS Bridge
- Pi:
 - I4.0 gateway: provides a connection to the internet.
- Application server:
 - P4M: Executing the implementation of these software models completes the CBSAM architecture.
 - AAS server
 - Node-RED server
 - Kafka server: for information distribution.

5.3.3 Communication Architecture

Communication is an essential part of digital twin systems, so it is important to understand the testbed's network architecture. However, enabling communication connections is not the focus of this thesis. Therefore, in the following content, some simple technical information will be provided about the LocalSEA communication architecture. To facilitate a common understanding, the presentation will follow the TCP/IP model layering [96]. It is worth mentioning that this part is not the contribution of this thesis. The reason to include this subsection aims to provide a better understanding of the entire testbed.

- **Network Interface Layer:**

Figure 5.1 illustrates the network interfaces provided by various physical entities within the system. All system components are equipped with a Wi-Fi module, which allows for the creation of a wireless network architecture. However, due to its mobility and nature as an AGV, the Turtlebot3 relies on Wi-Fi connectivity as it cannot be tethered with a wired connection. Meanwhile, the other robots and components within the testbed are relatively stationary, allowing them to establish connections through RJ45 Ethernet cables. This decision to employ wired connections for the stationary devices helps maintain low-latency communication and ensures a stable network environment, which is crucial for the reliable operation of LocalSEA. In factories, fixed equipment is also often connected via a wired network. In contrast, the Turtlebot3's wireless connection is optimized to support its mobility and functionality within the system.

- **Internet Layer:**

All devices within LocalSEA testbed communicate at the Internet Layer using the IP protocol. Wired networks are less susceptible to interference and can ensure higher service quality than wireless networks. IP protocol allows for the devices to be uniquely identified and located within the network system, facilitating the efficient routing of data packets. Furthermore, the IP-based communication system enables access from external networks and remote locations, which enables the remote control of a digital twin system.

- **Transport Layer**

The communication at the transport layer is described in article [97]. In order to enrich the scenario, the communication architecture is rather complex in this testbed, which is composed of different middlewares (ROS1 & ROS2) and protocols (mainly OPC UA). Regarding robotic implementation, ROS 1 and ROS 2 are two widely-used middleware. ROS stands for Robot Operating System. They are open-source and receive great support from the robotic community. In detail, ROS 1 uses XMLRPC combined with TCP/IP-based and UDP-based message transport, and ROS 2 relies on Data Distributed Service (DDS).

Open Platform Communication Unified Architecture (OPC UA) standard is selected to be the core communication protocol of LocalSEA since it is a potential candidate to overcome the Information Technology (IT) and Operational Technology (OT) convergence challenge [98]. With more detail of the OT data layer, OPC UA provides two PubSub communication modes: broker-based and broker-less. Broker-based means there is a broker in the middle that manages topics. A topic is an association between a data source and the information required to create links between the publisher and subscriber sides. Broker-less relies on the multicast mechanism of the UDP/IP stack. In detail, a publisher publishes a message to a multicast address. All subscribers who subscribe to the multicast address can receive the message. The IT layer communication in our testbed, is communication between AAS servers and OPC UA server. This connection is established by BaSyx connectors, and the communication protocol is TCP.

The above methods are widely adopted by robotics and industry domains. However, different robots using different middlewares can only communicate by additionally using one of the bridge solutions available in the robotic community. And it is even more challenging when deploying these robots as part of an OPC UA-based industrial testbed. In [97], the authors have investigated this problem and proposed an approach (5.3) that uses UA bridges to bridge the ROS 1 and ROS 2 spaces to

the testbed’s OPC UA PubSub network. This UA bridge works as a logical portal with two sides: one side is its ROS space, and the other side is the OPC UA PubSub network. This approach enables robots with the same middleware to collaborate in their space with their default communication method and can still have a route to communicate with other devices through a UA bridge.

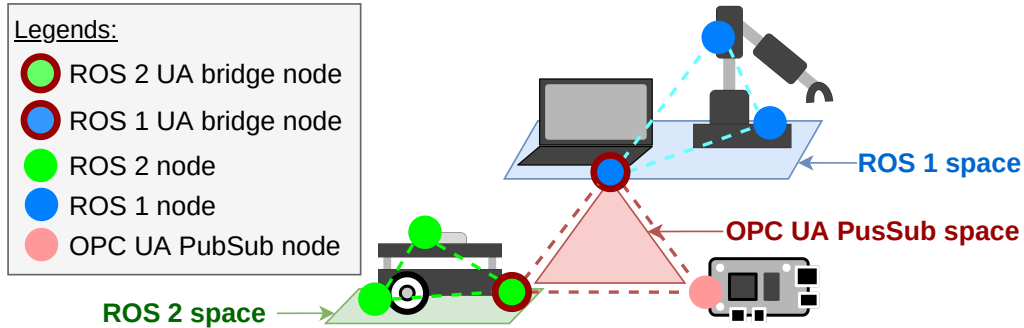


Figure 5.3: The architecture of ROS 1 and ROS 2 with UA bridges

- **Application Layer:**

The application layer is the top layer of the TCP/IP network model that interacts directly with end users. It provides communication between software and lower layers of the network model. As presented in Section 4.4.1, the AAS servers enable the HTTP connection from client side to the operational data and digital twin related information. The use of AAS takes care of the data exchange between user applications and the field devices. Instead of controlling and monitoring field device directly, the users can access all the information through the digital twin. However, one potential drawback of the AAS server is the added latency in data communication. Introducing an additional layer of abstraction like the AAS can lead to delays in data retrieval and transmission, which might be critical in real-time applications where instantaneous response is essential. This latency might affect the industrial application, especially scenarios requiring real-time critical at high frequency.

To the best of our knowledge, there is only one work [40] conducted on the latency caused by BaSyx-based AAS. However, there are too many influencing factors in the process that can lead to inaccurate conclusions. So with this testbed, the focus of the study is not to conclude what kind of latency the AAS layer produces specifically during the connection process. In order to approximately characterize the type of applications achievable with the current implementation, we have estimated the average latencies and that this can serve as a guide to decide whether a certain monitoring/diagnosis is feasible or not. Below is a table of a network connection evaluation in the absence of AAS (see Table 5.1).

Table 5.1: A summary of simple network communication evaluation for the testbed

| | TurtleBot3 WP | Niryo Ned | UR3e | End Device |
|-------------------------|----------------|----------------|----------------|---------------|
| Network Type | Wifi 802.11n | RJ45 cable | RJ45 cable | RJ45 cable |
| Transportation Protocol | UDP | UDP | UDP | TCP |
| Throughput | 1.05 Mbits/sec | 1.05 Mbits/sec | 1.05 Mbits/sec | 932 Mbits/sec |
| Jitter | 1.462 ms | 0.022 ms | 0.022 ms | |
| Loss | 0% | 0% | 0% | 0% |
| Latency | 1.396 ms | 0.317 ms | 0.317 ms | 0.383 ms |

A disparity in throughput is evident, with the TurtleBot3 WP, Niryo Ned, and UR3e all at 1.05 Mbits/sec, significantly lower than the End Device’s 932 Mbits/sec, suggesting a higher data handling capacity for the latter. Jitter, an indicator of network stability, is notably higher in the wireless TurtleBot3 WP at 1.462 ms compared to the near-identical and lower jitter in the wired Niryo Ned and UR3e. This implies the instability in the wireless connection. The latency as a factor for the network responsiveness, also varies, with the TurtleBot3 WP experiencing the highest latency at 1.396 ms, indicating faster response times in the wired connections. All devices in this testbed are connected to a local network, and the number of the connected devices are limited. Therefore they are all reporting zero packet loss, which reflects overall stable connections in both wireless and wired setups in this condition. This table can serve as an indicator for the frequency setting of the MAPE-K loop implementation.

5.4 CBSAM Architecture Implementation

It is first necessary to briefly review the structure of the CBSAM architecture as well as the overall goals of the implementation. This provides the necessary basis of understanding for reading this implementation phase section.

As presented in Chapter 3, the CBSAM architecture covers four phases of a production system lifecycle: specification, design, engineering & deployment, and operation & maintenance. Therefore, this section follows this structure in describing how these phases are implemented in the LocalSEA academic use case from scratch. This section carefully outlines the implementation process for each phase and how it could be practically applied on the LocalSEA testbed.

5.4.1 Specification

Requirement Diagram

The specification phase can be implemented in a variety of ways. According to the AAS engineering methodology proposed in a structured way is to employ requirement diagrams. The requirement diagrams provide a visual representation and system layout of the necessary details and interconnections. By doing so, stakeholders can more easily identify all the requirements from different aspects, which leads to more informed strategic decisions and robust system design. Figure 5.4 presents the functional requirements for the AAS digital twin management in order to realize the CBSAM architecture. The requirements are formalized using SysML v1.6 language.

The three primary requirements of the resource AAS digital twins development are:

1. The requirement of exposing manufacturing capability-related information of the asset.
2. The necessity of accessing operational data from the asset, and exposing this information to external applications underscores the interoperability nature of AAS digital twins.
3. The predefinition of all potential events and the submodel elements to monitor.

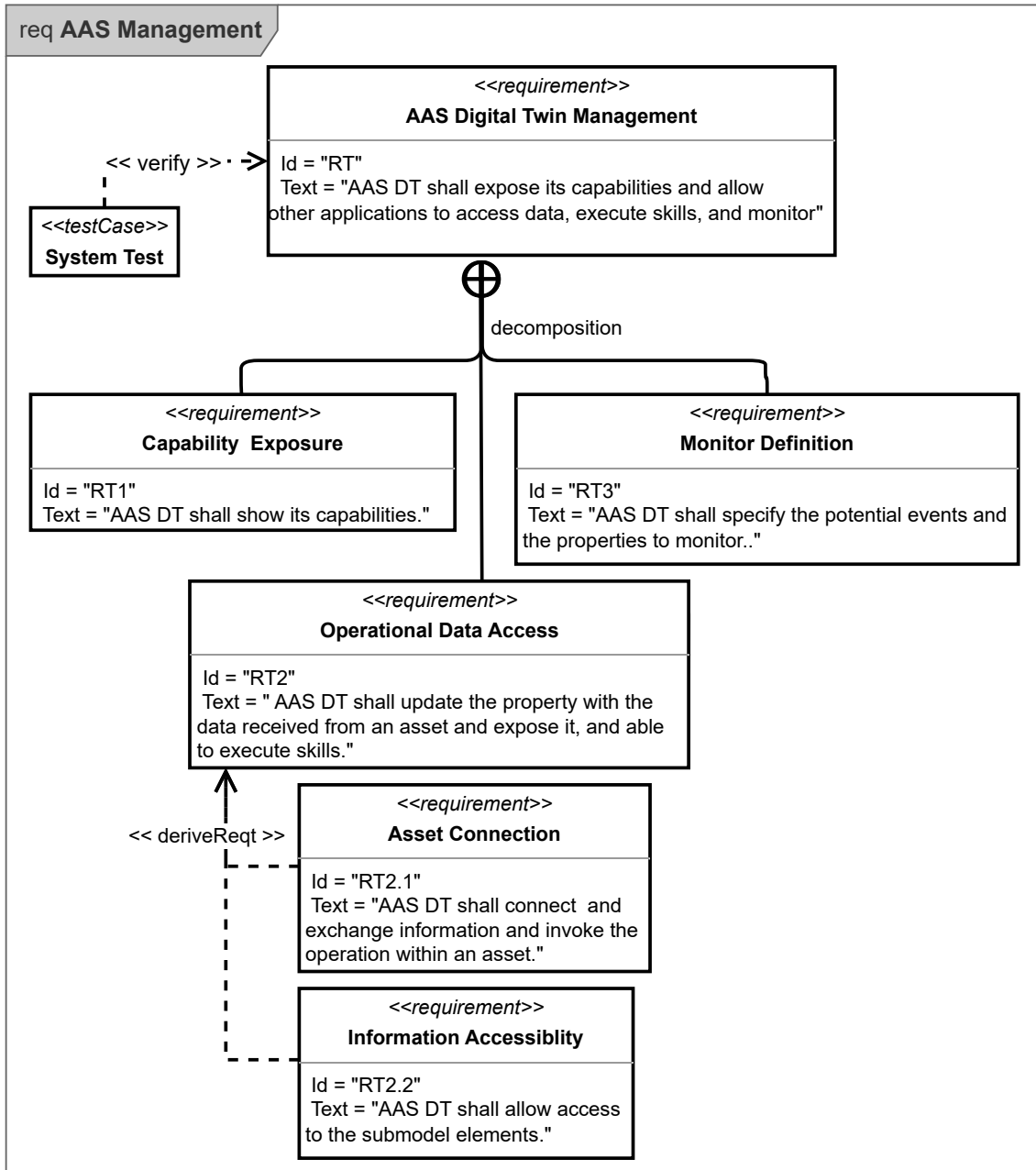


Figure 5.4: REQs related to the AAS digital twin management

Required Scenario

By constructing a comprehensive, detailed, and criticable relevant scenario in a testbed environment, the practical application of the conceptual concepts and methods outlined

previously will be explored in practice. The purpose of creating this testbed scenario is not only to affirm the validity and utility of our proposed methods, but also to concretely illustrate how they can be coordinated to address the research questions posed in the introduction.

In this scenario, a new product has been designed and the system architect wants to automatically configure a robust production line. The use case considered in this thesis starts from this potential scenario and tackles partially the transportation process with the help of Papyrus4Manufacturing toolset. By doing so, we aim to identify the capabilities and constraints required by the process, the behavior anomalies within the process execution, and the adaptation advice. When one of the participants is not working properly, the system should be aware of the fact and able to fix or find an ultimate solution according to the diagnosis rules given by the domain expert.

Therefore, a simple production process is described as follows:

- Detect and grasp the required pieces from the storage unit and place them on the transporting device.
- Transport the required parts to the assembly area.

Capability Information

The implementation of the capability checking module is fundamentally dependent on a well-defined ontology of capability specifications. This process requires a comprehensive definition and integration of information related to capability specifications. Figure 5.5 aims to show the structure and name abbreviations of MaRCO ontology. The matching rules are provided by MaRCO ontology in the mmo ontology through embedding SPIN rules, which serves to align the capability offered by the resource (defined by cm) with the capability required by the production process (defined by pt). Figure 5.6 and 5.7 respectively illustrate examples of the capability model and the process taxonomy model in the original MaRCO ontology. The specification of capability information draws from both the ontology properties and the technical data of the testbed.

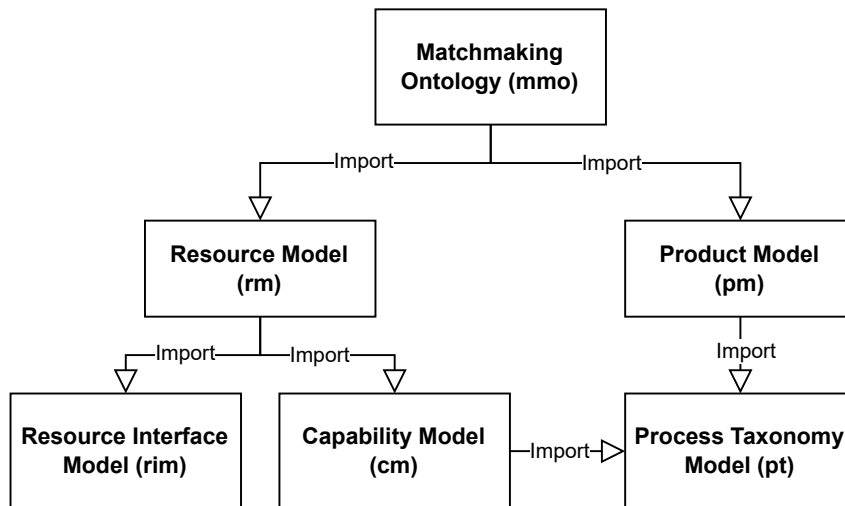


Figure 5.5: MaRCO structure

- **Resource Specification:**

As presented in Section 5.3 NedRobot1, LittleTurtle, Belt, and UR3e are the resource assets existing in the LocalSEA testbed. The MaRCO ontology and the robot technical documents are studied to build accurate capability submodels. Both LittleTurtle

Transporting — <http://resourcdescription.tut.fi/ontology/capabilityModel#Transporting>

Annotations Usage

Annotations: Transporting

Annotations +

rdfs:comment [type: xsd:string] @ x o

Capability to transport items from one place to another.
For example robot attached with a gripper, or conveyor alone have this capability.

Description: Transporting

Equivalent To +

SubClass Of +

- acceleration_x_max max 1 rdfs:Literal
- acceleration_y_max max 1 rdfs:Literal
- acceleration_z_max max 1 rdfs:Literal
- accuracy exactly 1 rdfs:Literal
- CombinedCapability
- dof max 6 rdfs:Literal
- dof min 1 rdfs:Literal
- hasAllowedItemShapeAndSize_max min 0 owl:Thing
- hasAllowedItemShapeAndSize_min min 0 owl:Thing
- hasInputCapability some
(ExpansionGrasping or FingerGrasping or HoldingByGravity or MagneticGrasping or VacuumGrasping)
- hasInputCapability some Moving
- hasItemSize_max exactly 1 owl:Thing
- hasItemSize_min exactly 1 owl:Thing
- hasWorkspaceTypeAndDimensions exactly 1 owl:Thing
- hasWorkspaceTypeAndDimensions only Workspace
- payload exactly 1 rdfs:Literal
- repeatability exactly 1 rdfs:Literal
- speed_x_max max 1 rdfs:Literal
- speed_y_max max 1 rdfs:Literal
- speed_z_max max 1 rdfs:Literal
- Transporting

Figure 5.6: MaRCO transporting capability definition in Protégé

and Belt specialize in transportation functions. NedRobot1 excels primarily in pick-and-place tasks. UR3e is more advanced that can achieve pick-and-place with a vision detection function. (It is worth mentioning that the Niryo Ned itself is capable of visual detection. In this particular scenario, to enhance the functionality of the capability checking module, it is imperative that the NedRobot is designated as a device equipped only with the standard PickAndPlace capability, thereby augmenting the complexity of the scene.)

Here, the LittleTurtle is taken as an example to explain the capability information extraction. The capability transporting is defined in MaRCO ontology⁴ as shown in Figure 5.6.

(1) In ontology representation, object properties and data properties are used to define the characteristics of classes. The “subclass of” relationship seen in this context is a way of stating that instances of a class must have certain properties, not that the properties are themselves subclasses. From the concept definition, we can easily found the properties related to Transporting capability such as *acceleration*, *speed*, *accuracy*, *degree of freedom*, etc.

(2) The term *Transporting* displayed at the bottom of the figure is a reference to the Transporting class within the pt ontology (see Figure 5.5). This semantic relationship between ontology concepts is the basis for the automatic capability matchmaking feature.

LittleTurtle is an instance of TurtleBot3 Waffle Pi, and the capability-related hardware specifications can be found in their official manual⁵ as follows:

- Accuracy: 10 *mm*
- Maximum payload: 30 *Kg*
- Degree of freedom: *translational movement to all directions*
- Repeatability: 0.5 *mm*
- Maximum translational velocity: 0.26 *m/s*
- Maximum rotational velocity: 1.82 *rad/s* (104.27 *deg/s*)
- Acceleration range: [0, 4457932] *rev/min*² \approx 40.86 *m/s*²
- Size (L * W * H): 281 *mm* * 306 *mm* * 141 *mm*

Although the information that can be found in the specification may not cover all fields demanded in MaRCO, it is not very important. This is simply an illustration of the possibilities of our CBSAM.

• Product & Process Specification:

The raw material of the product we designed here is some plastic blocks. They will be passed between different devices according to the scenario. Therefore, the capabilities required by this partial manufacturing process are: *PickAndPlace* and *Transporting*. According to MaRCO’s definition of product, a product needs to identify its mass and *requiredActivity*, i.e. the process required to produce it. And the *Process* needs to identify the *requiredCapabilities* defined in pt ontology.

Figure 5.7 shows a concrete example of the definition of Transporting in pt ontology. Accordingly, an example will be used below to illustrate how to specify the required capability of a process. The properties related to the matchmaking are *requiredAccuracy*, *requiredDOF*, and *requiredSpeed*.

⁴<https://resourcdescription.rd.tuni.fi/ontology/capabilityModel>

⁵<https://manual.robotis.com/docs/en/platform/turtlebot3/features/>

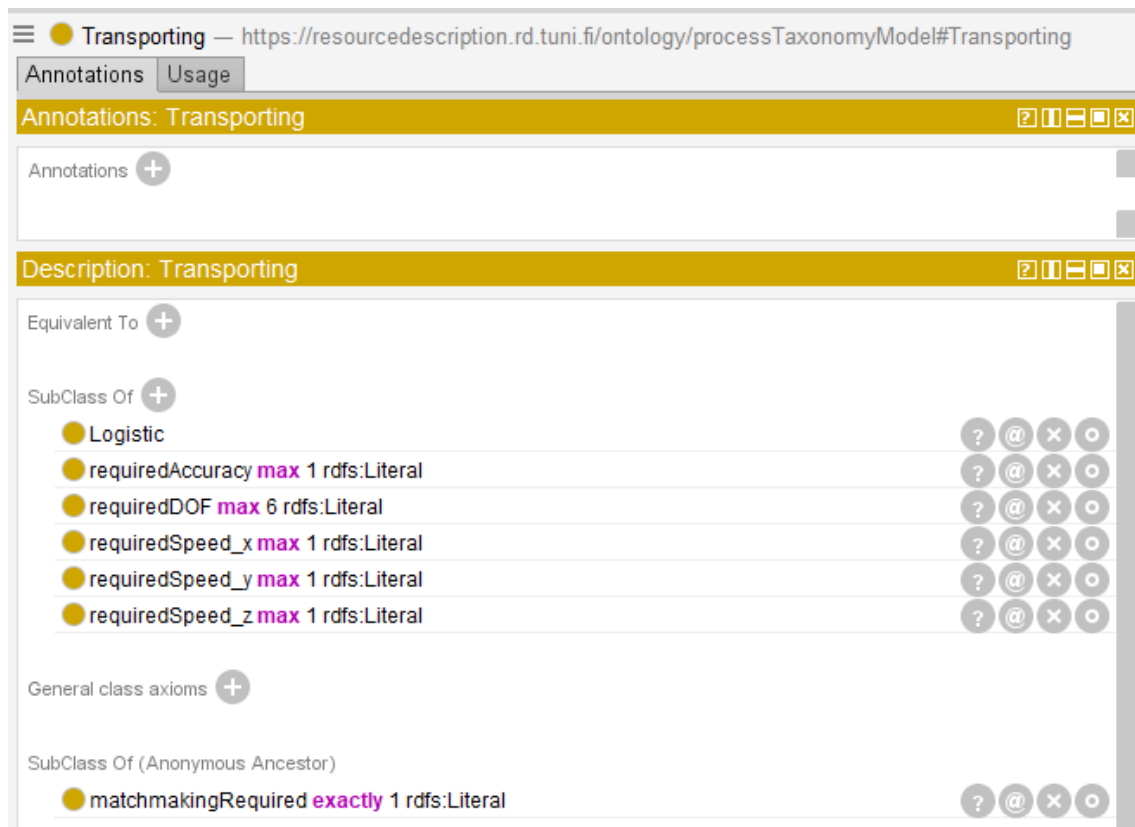


Figure 5.7: MaRCO required transporting capability definition in Protégé

- requiredAccuracy: 10 *mm*
- requiredDOF: *translate_x*
- requiredSpeed_x: 0.02 *m/s*

Based on the provided examples, it's clear that LittleTurtle offers the precision needed for the process, which demands a 10mm accuracy. Additionally, LittleTurtle is capable of supplying the process with a translational speed along the x-axis of 0.02m/s. These qualities facilitate the realization of semantic matchmaking between the two capabilities.

Operational Data Information

• Asset Connection:

The specification of operational data access in the AAS digital twin system provides a flexibility in the deployment, which allows the system to be configured in a parameterized manner rather than through pre-established connections. The AAS Digital Twin should be configured to receive data from within the asset. Upon receiving this data, the AAS digital twin updates the appropriate attributes to reflect the current state of the asset. This update mechanism allows the digital twin to become a sufficiently accurate representation of the physical asset for the applications to be performed correctly. The AAS digital twin should also be able to expose the updated data to external systems and stakeholders. In the integration between the Asset Administration Shell (AAS) Digital Twin and the assets, the OPC UA standard is fundamental for internal connectivity. Consequently, the specification of operational information entails gathering the information model of OPC UA server. The OPC UA server configuration of LocalSEA can be found in Figure 5.8.

The LittleTurtle is utilized here as an illustrative example. Tables 5.2 and 5.3 delineate the OPC UA server information specific to LittleTurtle. The first table illustrates dynamic properties that reflect the status of the system, while the second details the various methods available for executing skills. These methods vary in complexity, encompassing simple actions such as moving forward and backward, as well as more sophisticated tasks like navigating to a pre-defined position in the cartography or following a set trajectory.

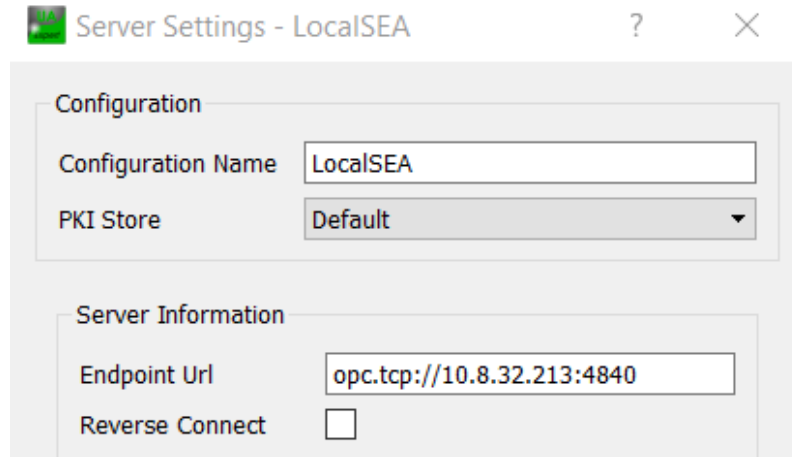


Figure 5.8: LocalSEA OPC UA server configuration

Table 5.2: Dynamic properties of Little Turtle in OPC UA server

| Data | Type | Rate | Node Id | Unit |
|----------------------------|--------|-------|---------------|-------|
| OperationalStatus | UInt16 | Event | ns=6, i=5181 | - |
| LatestMission | UInt16 | Event | ns=6, i=5182 | - |
| LatestOperationalStatus | UInt16 | Event | ns=6, i=5183 | - |
| Wheel Left ActualPosition | Double | 5Hz | ns=5, i=6123 | Rad |
| Wheel Left ActualSpeed | Double | 5Hz | ns=5, i=6143 | Rad/s |
| Wheel Right ActualPosition | Double | 5Hz | ns=5, i=6126 | Rad |
| Wheel Right ActualSpeed | Double | 5Hz | ns=5, i=6146 | Rad/s |
| T3wp Status | Int32 | 5Hz | ns=6, i=65411 | - |
| T3wp EnergyConsumption | UInt32 | 0.1Hz | ns=6, i=65412 | kJ |
| T3wp BatteryStatus | Double | 5Hz | ns=6, i=65413 | - |
| CPU Temperature | Double | 0.1Hz | ns=5, i=6180 | °C |
| Bucket Status | Int32 | Event | ns=6, i=6031 | - |
| Park Status | Int32 | Event | ns=6, i=6021 | - |

The Table 5.2 indicates that property data updating occurs at regular intervals, which represents a discrete sampling of the reported values. This results in latency, creating a discrepancy between the real-world situation and its digital twin representation. Additionally, all these measurements are prone to errors stemming from the instrumentation that affect the accuracy with which the digital twin mirrors.

Table 5.3: OPC UA methods of LittleTurtle

| Method | Node Id | Input |
|-------------------------------|--------------|--------|
| t3wp_act_start | ns=6;i=61001 | - |
| t3wp_act_move_place_workspace | ns=6;i=61002 | - |
| t3wp_act_move_place_storage | ns=6;i=61003 | - |
| t3wp_act_move_place | ns=6;i=61004 | - |
| t3wp_act_start_with_init | ns=6;i=61011 | UInt32 |
| t3wp_act_end | ns=6;i=61099 | - |
| t3wp_behave_go_round_trip | ns=6;i=61100 | - |
| t3wp_restart | ns=6;i=61998 | - |
| t3wp_shutdown | ns=6;i=61999 | - |

- **Information Accessibility:**

This functionality is realized by BaSyx. Since the specification required by BaSyx is the AAS model, the description will be expanded in the design phase.

Monitoring & Diagnosis Information

- **Resource Operating Events:**

The specification phase of the system’s operation is indispensable. This involves analyzing each component’s role and impact on the overall system performance. The elements selected for monitoring are those whose performance metrics are essential to the smooth functioning of the entire system.

For each identified AAS model, it is crucial to specify the possible events that need monitoring. This includes defining the nature and type of each event. Events can range from routine operational status to critical system alerts. The RSP4J will be used as a framework for runtime stream processing. The CSPARQL is chosen as the inference query language, which verifies the constraint values within a time window. Specifically, in the definition of CSPARQL syntax, the term *window_range* (W_R) refers to the time frame of the window, while *window_step* (W_S) indicates the frequency at which queries are executed.

The constraints and performance metrics for each event are detailed in Table 5.4. This table provides an essential overview of the events associated with the resource performance, delineating both the event types and their corresponding constraints. These limits are related to the performance of the device, so no matter how the process is defined, the above constraints should always be met to ensure that the system is functioning properly. For example, to ensure optimal performance of the conveyor belt, a constraint is set where the belt speed should exceed 20 mm/s. These performance metrics are pivotal in evaluating the efficiency and effectiveness of the AAS models. A critical aspect of monitoring is the detection of system failures. In our case study, a failure is indicated when the “Status” value of any resource drops to 0.

Table 5.4: Exemplary events of LocalSEA resources

| Monitor | EventType | Event | Constraints | W_R | W_S |
|--------------|-----------|---------------|---------------------------------|-------|-------|
| Belt | Warning | SpeedTooLow | Speed \leq 20 | 1 | 1 |
| | Failure | SystemOFF | Belt.Status = 0 | 2 | 2 |
| NedRobot1 | Failure | SystemOFF | NedRobot1.Status = 0 | 2 | 2 |
| | Warning | CPUTooHot | NedRobot1.Temperature \geq 60 | 2 | 2 |
| Storage | Warning | NumberLow | Number \leq 1 | 2 | 2 |
| | Warning | NoObjects | Number = 0 | 2 | 2 |
| LittleTurtle | Warning | BatteryTooLow | BatteryStatus \leq 15 | 10 | 10 |
| LittleTurtle | Failure | SystemOFF | LittleTurtle.Status = 0 | 2 | 2 |
| UR | Failure | SystemOFF | UR.Status = 0 | 2 | 2 |
| | Warning | CPUTooHot | UR.Temperature \geq 60 | 2 | 2 |

- **Process Related Events:**

Event specifications on processes need to be analyzed for different scenarios. This is because the constraints encountered in different scenarios will be different depending on the desired product and even product orders for the same product. For instance, as shown in Table 5.5 events are categorized into types such as 'Failure', 'Malfunction', and 'Warning'. A 'Failure' event, like 'ResourceSystemOFF', occurs when any resource status equals zero and has specific implications in scenarios where uninterrupted resource availability is critical. It is assigned a weight of 2 in both W_R (time-based window) and W_S (frequency) categories.

Similarly, 'Malfunction', characterized by 'DefectiveOperation', is triggered when a resource's operation status is at 3. This type of event may have different repercussions in scenarios where precision and operational efficiency are paramount. The 'Warning' category includes events such as 'NoObjectsInStorage', indicating zero storage, and 'NoObjectArrive', occurring when all infrared sensor statuses are zero. These warnings are essential in scenarios where inventory management and timely arrival of objects are critical factors. Thus, the table not only catalogs these events but also implicitly guides the analysis of how different constraints in varied scenarios can influence the process flow and the quality of the end product.

Table 5.5: Exemplary events of related to scenario process

| EventType | Event | Constraints | W_R | W_S |
|-------------|--------------------|--|-------|-------|
| Failure | ResourceSystemOFF | \exists Resource.Status = 0 | 2 | 2 |
| Malfunction | DefectiveOperation | \exists Resource.OperationStatus = 3 | 2 | 2 |
| Warning | NoObjectsInStorage | Storage.Number = 0 | 2 | 2 |
| Warning | NoObjectArrive | \forall Infrared_Sensor_Status = 0 | 20 | 1 |

5.4.2 AAS Model Design

This section introduces the design of AAS models leveraging the AAS graphical modeling diagrams within the context of the P4M framework. It delves into the methodology of constructing precise AAS models, utilizing these diagrams and incorporating the information amassed during the preceding specification phase. In that phase, data is methodically collected and categorized into various aspects such as capability specification, operational data specification, and monitoring specification.

To effectively construct digital twin models that encapsulate all relevant real-world information within a singular model for each asset, it is imperative to integrate the collected data as comprehensively as possible. The integration process is designed to ensure that the digital twin sufficiently reflects the multifaceted nature of the physical asset. Owing to the diverse and complex nature of these models, the presentation of this process will be segmented, allowing for a more detailed and clearer explanation of the design and implementation strategies. This approach facilitates a deeper understanding of how each aspect of the collected information contributes to the representation of the assets in the digital twin models.

The Table 5.6 categorizes these components into four main types: `Resource_Type`, `Resource_Instance`, `Process`, and `Product`. Each type is evaluated against three main aspects: `Capability`, `Operational`, and `Monitor`. For `Resource_Type`, the table indicates a direct relevance to the `Capability` aspect, highlighting its role in defining the potential abilities of a resource, but it shows no direct correlation with `Operational` and `Monitor` aspects. On the other hand, `Resource_Instance` is associated with all three aspects. It's not only related to its type (`Capability`) but also plays a significant role in the `Operational` and `Monitor` phases, underlining its importance in the functioning and oversight of the actual resources. The `Process` category is marked as integral to all three aspects: `Capability`, `Operational`, and `Monitor`. This indicates the all-encompassing nature of processes within the AAS models. Lastly, the `Product` category is linked with the `Capability` aspect.

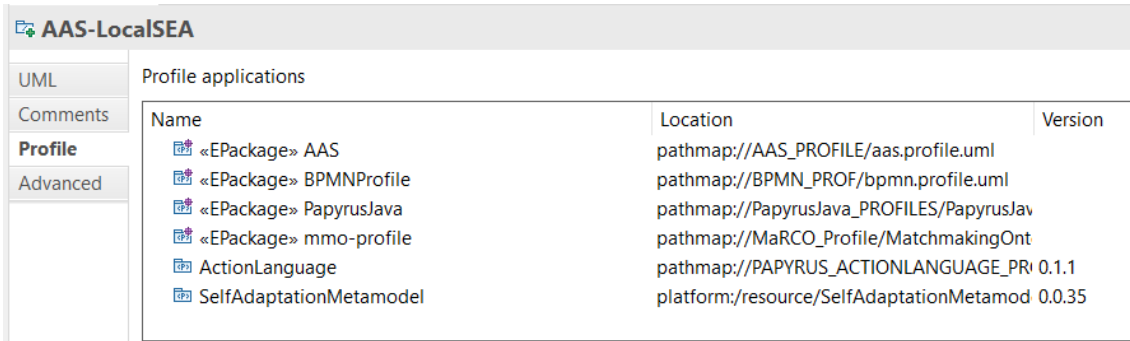
| | Capability | Operational | Monitor |
|--------------------------------|---------------------|-------------|---------|
| <code>Resource_Type</code> | ✓ | | |
| <code>Resource_Instance</code> | related to its type | ✓ | ✓ |
| <code>Process</code> | ✓ | ✓ | ✓ |
| <code>Product</code> | ✓ | | |

Table 5.6: Overview of AAS model design

In the previous chapters, we emphasized the significance of using a UML profile for the implementation of a metamodel in an MDE environment. We also discussed several critical metamodels, such as AAS, MaRCO, and the self-adaptive metamodel. The implementation step is the application of these metamodels to our current model.

This application is essential to guarantee that our model is fully compatible with and can effectively utilize all the previously mentioned DSLs. Incorporating the AAS metamodel, for instance, will allow our model to better manage and represent digital twins of physical assets. By applying MaRCO profile, our model will be able to represent manufacturing resources and their capabilities. The self-adaptive metamodel will be particularly beneficial when designing the monitoring and diagnosis conditions and requirements.

Figure 5.9 shows different UML profiles applied to the AAS model package.



| Name | Location | Version |
|-------------------------|--|---------|
| «EPackage» AAS | pathmap://AAS_PROFILE/aas.profile.uml | |
| «EPackage» BPMNProfile | pathmap://BPMN_PROF/bpmn.profile.uml | |
| «EPackage» PapyrusJava | pathmap://PapyrusJava_PROFILES/PapyrusJava | |
| «EPackage» mmo-profile | pathmap://MaRCO_Profile/MatchmakingOnt | |
| ActionLanguage | pathmap://PAPYRUS_ACTIONLANGUAGE_PRI | 0.1.1 |
| SelfAdaptationMetamodel | platform:/resource/SelfAdaptationMetamod | 0.0.35 |

Figure 5.9: Profile application to AAS model package

Resource Type AAS Model Design

- **AAS:**

In the AAS model (see Figure 5.10), the component designated as the resource type is primarily focused on capability information, which is effectively implemented through the *DeviceBlueprint* stereotype, a feature originating from the MaRCO profile. This aspect of the model is critical in defining the functional capabilities of the device or resource in question. However, the scope of the AAS model for the resource type extends beyond just capabilities. It is also designed to integrate a range of additional informational elements that are vital for a comprehensive understanding of the resource. These elements include “Nameplate”, which provides basic identification details; “Technical Specification”, offering detailed descriptions of the resource’s technical attributes; “Mechanical Interface”, outlining the mechanical connectivity aspects; and “Electrical Interface”, detailing the electrical connection specifications. Together, these components provide a holistic view of the resource, encompassing both its functional potential and its technical and physical characteristics, thereby enhancing the model’s utility and applicability in various contexts.

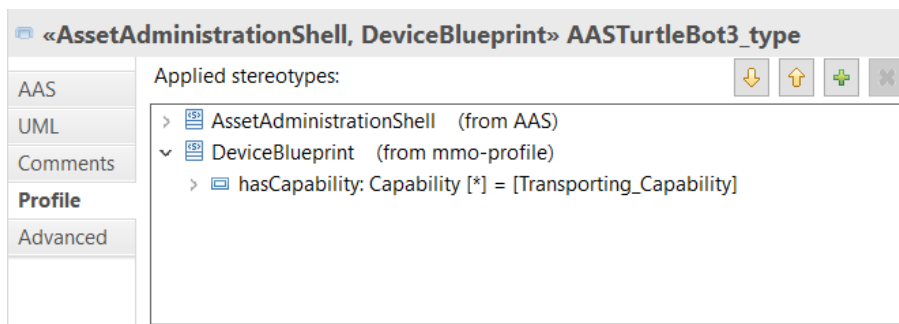


Figure 5.10: TurtleBot3_Type AAS model design

- **Capability Submodel:**

In the capability submodel (see Figure 5.11), it is required to populate the capability element in accordance with the information related to capabilities as outlined in the Section 5.4.1. This submodel involves a meticulous alignment of the submodel’s capability element with the predefined specifications, ensuring consistency and accuracy. As demonstrated in the accompanying figure, the procedure for integrating these capability-related details is methodically illustrated, guiding the accurate representation of capabilities within the submodel. This step is crucial for maintaining the integrity of the model and ensuring that it accurately reflects the specified ca-

pabilities, thereby providing a reliable and effective tool for analysis and application in relevant scenarios.

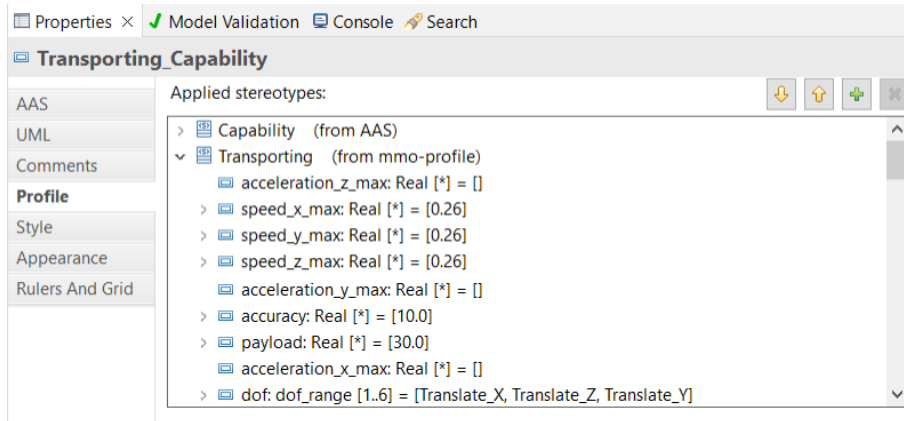


Figure 5.11: TurtleBot3_Type capability design

Resource Instance Model

- **AAS:**

To effectively establish digital twin models that are accurate and deployment-ready, certain attributes of the AAS model require identification. For an instance model, the attribute *derivedFrom* must reference its corresponding type model. For instance, AASLittleTurtle (see Figure 5.12) should refer to AASTurtleBot3_Type. As listed earlier, this AAS model should encompass at least three distinct submodels: the “Capability_Submodel”, “OperationalData_Submodel”, and “Monitoring_Submodel”.

For the deployment of the AAS model as an executable BaSyx server, it is imperative to specify endpoint information during this phase. For instance, to deploy the server locally, the address should be set to “localhost”. Additionally, each AAS server must be assigned a unique port to prevent any connection conflicts. Furthermore, each AAS model requires a unique identifier to ensure its distinctiveness.

In the context of the AAS resource instance model, it should also be annotated with the *IndividualDevice* stereotype. This annotation signifies the model’s alignment with the *DeviceBlueprint* model, which is essential for accurately defining the model within the AAS framework. This step is critical for ensuring that each resource instance is properly represented and identifiable within the overall digital twin system. And it is also crucial for the further engineering phase.

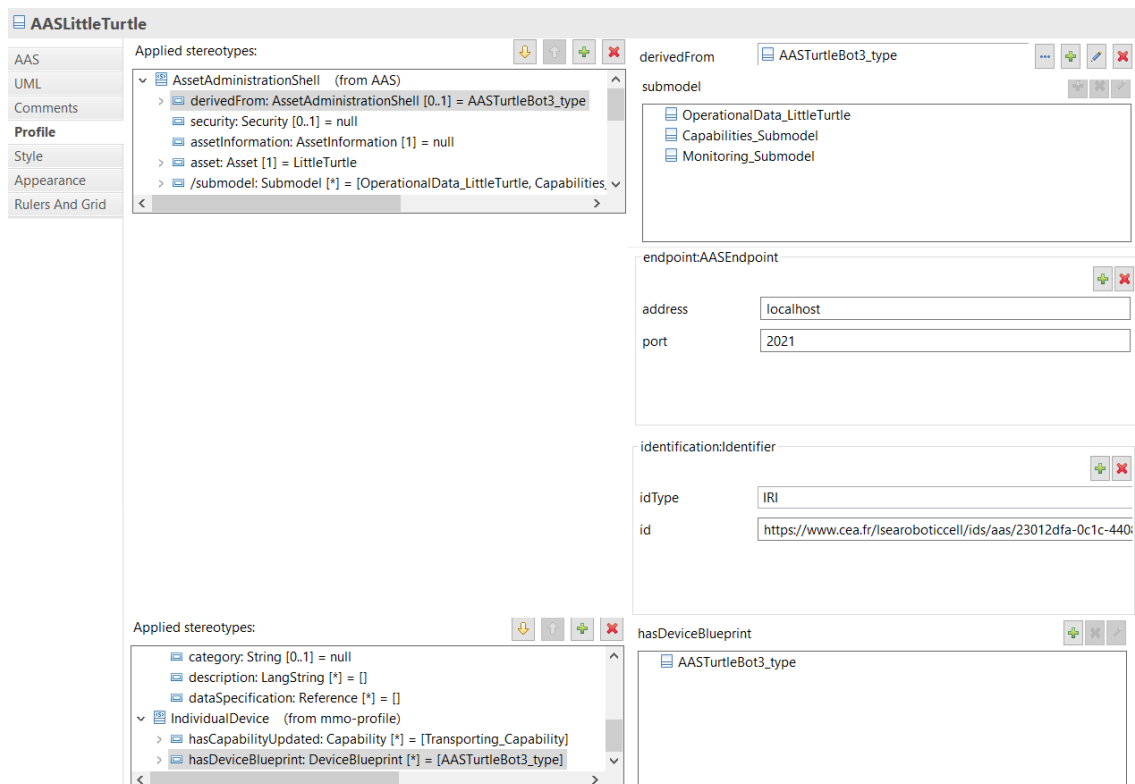


Figure 5.12: LittleTurtle AAS model design

- **OperationalData Submodel:**

Figure 5.13 details the configuration of a Property of the “OperationalData Submodel” named “BucketStatus”. The “BucketStatus” property is marked as *isDynamic*, and is associated with an OPC UA protocol endpoint, as shown by its address (`opc.tcp://10.8.32.213:4840`) and protocol attributes. This endpoint is named “Wlan0”, which refers to a wireless network interface and is linked to a NodeId characterized by a namespace index and an identifier. The above information are already gathered from the OPC UA server.

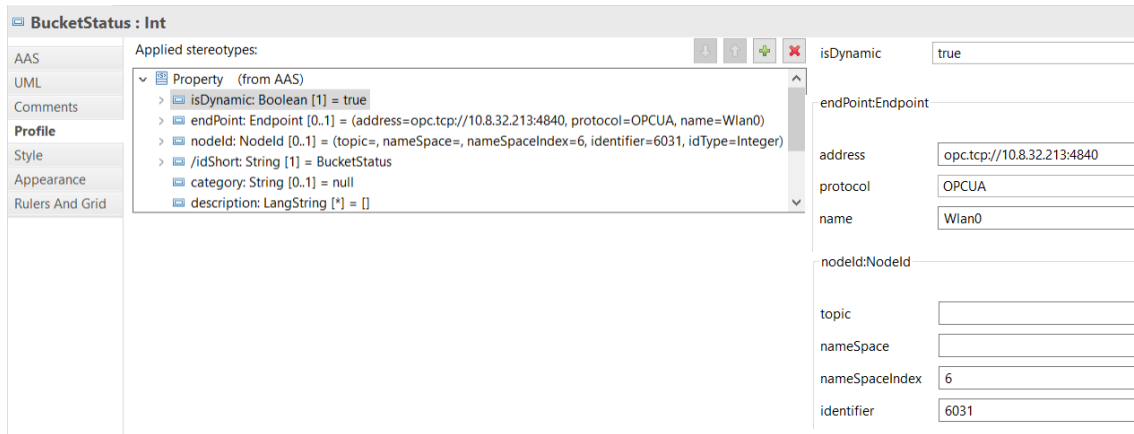


Figure 5.13: Little Turtle property

In the “OperationalData Submodel” delineated herein, additional to the dynamic properties and operations as specified in Section 5.4.1, there exists the requisite to architect the skill as an operation. Skills embody the tangible executions of capabilities. For example, the skill employed by LittleTurtle to actualize the capability *Transporting* is denominated as “go_round_trip”. The conduct of this skill is depictable through a BPMN (Business Process Model and Notation) process diagram, the representation of which is illustrated in Figure 5.20.

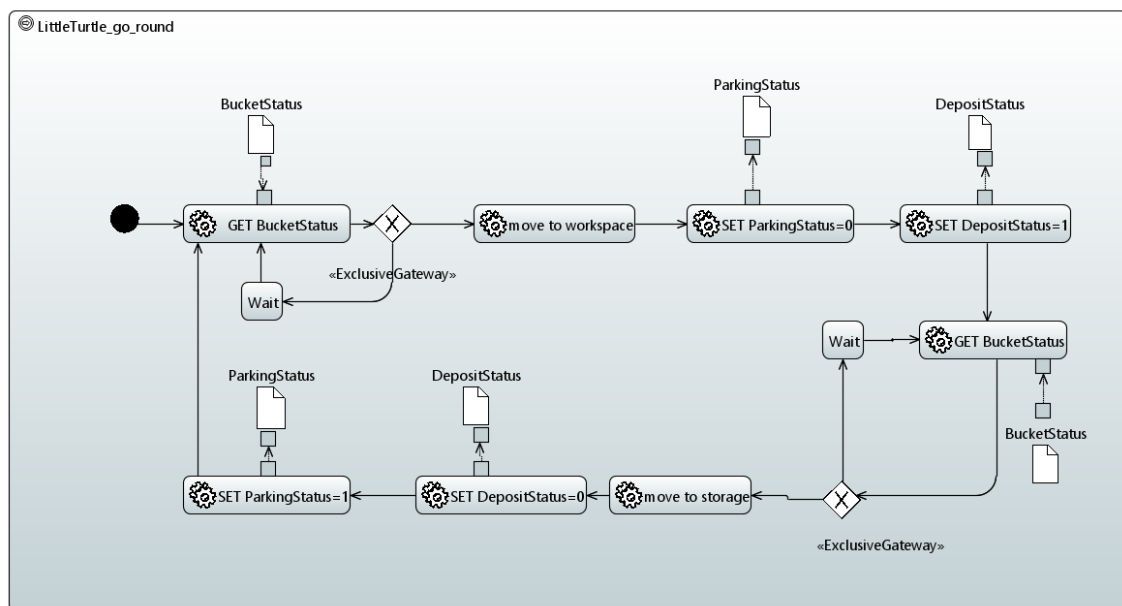


Figure 5.14: Little Turtle behavior BPMN representation

• **Monitoring Submodel:**

As an AAS submodel, the monitoring model can be designed for various assets, including different production resources and production processes. According to the previous introduction, we need to design the elements to be monitored for each asset at this stage, the events that may be triggered, and the trigger conditions of the events (the diagnosis rules).

For example, consider the scenario illustrated in Figure 5.15. In this instance, the belt monitor is responsible for observing its own “Status” and “Battery” properties. The diagnosis rules are applied through certain operations, such as “check_battery”, which regularly verifies the battery status every two seconds. Two kinds of events are predefined in the diagnosis. The first event is a warning when the battery is lower than 15%, which means the battery of little turtle is too low to ensure a smooth production process. The second event is “SystemOFF”, which indicates the system is in OFF mode. Either the robot is not started, or the robot is broken and can not be started.

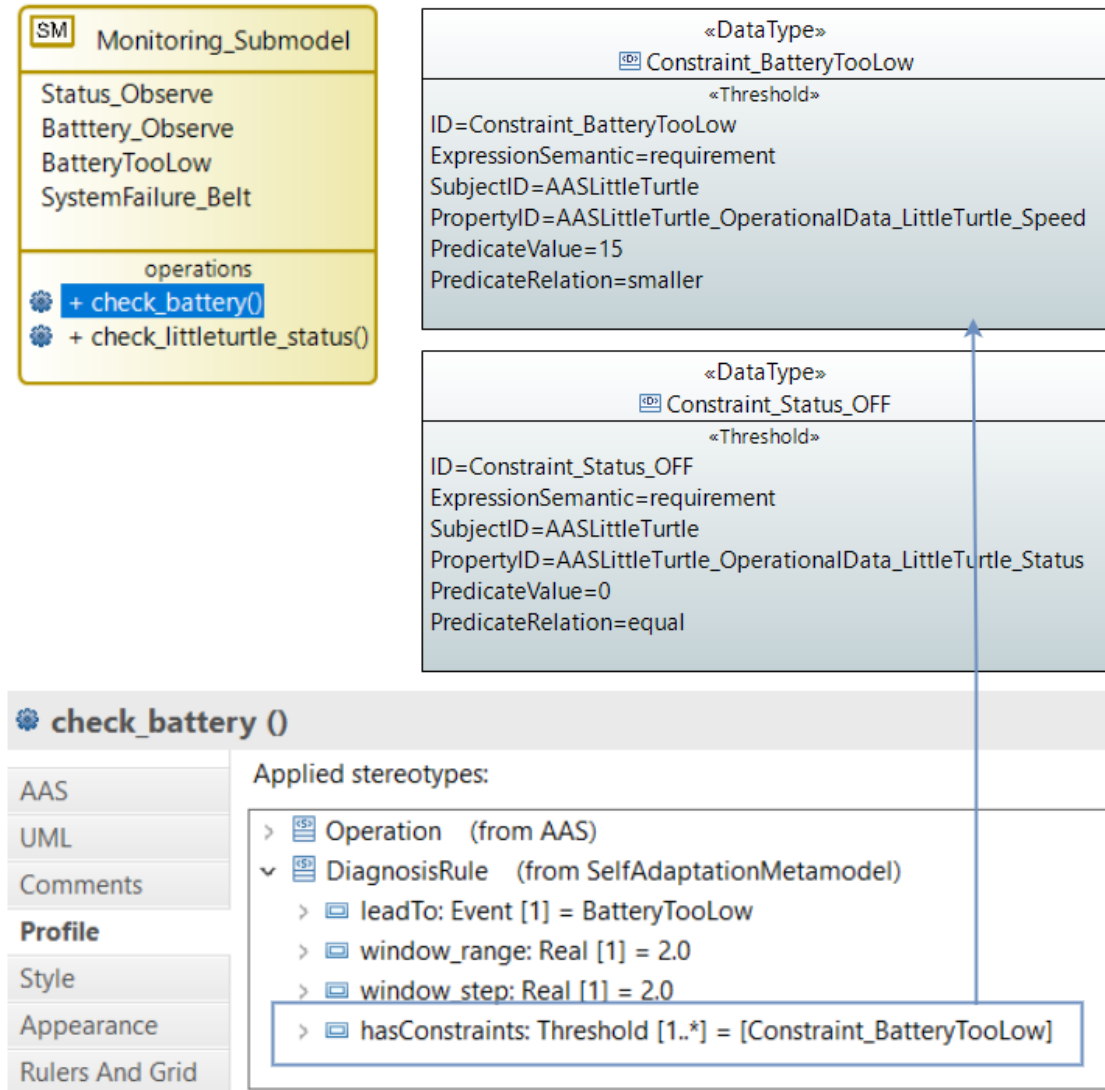


Figure 5.15: Little Turtle monitor

Process AAS Model Design

- **AAS:**

The AAS model of a process should encompass at least three distinct submodels: the “Capability_Submodel”, “OperationalData_Submodel”, and “Monitoring_Submodel”. At the same time, the *Process* (from mmo-profile) stereotype should be applied to the AAS model (see Figure 5.16), which indicates this object is semantically recognized as a process within the model. The presence of *requiresProcessCapability* suggests that this AAS component is designed to perform or manage specific functions or tasks within an automated or industrial process. The capabilities “PickAndPlace” and “Transporting” imply actions typically involved in manufacturing or logistics processes.

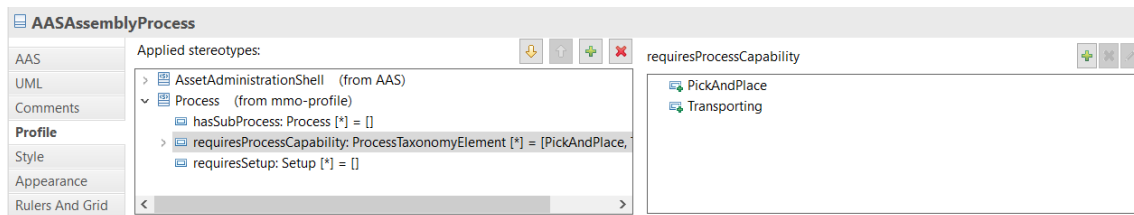


Figure 5.16: Process AAS annotation design

- **Capability Submodel:**

The Capability_Submodel should include the capabilities required by the given process. Each capability should be annotated with concepts coming from the pt (Process Taxonomy Ontology) and well-refined with specified data. As shown in Figure 5.17, the attribute fields are supposed to be filled in with the information given in Section 5.4.1.

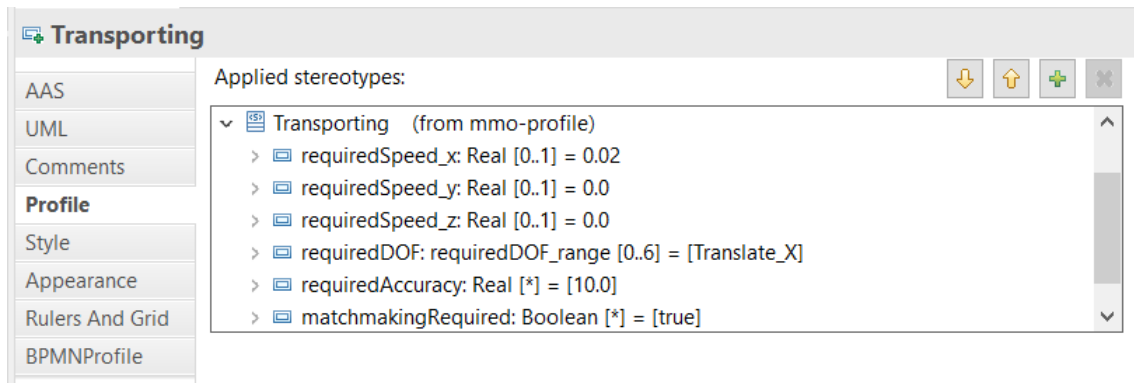


Figure 5.17: Required capability (Transporting)

- **OperationalData Submodel:**

The OperationalData_Submodel should include the process workflow that is the BPMN Process diagram. The process diagram of “AASAssemblyProcess” is showcased in Figure 5.18. As the project progresses into the engineering and deployment stages, each service task within this diagram will be allocated a specific skill, a decision that is determined based on the requirements and outcomes identified during the engineering phase. This allocation is essential for ensuring that each task is executed efficiently and effectively, aligning with the overall project objectives.

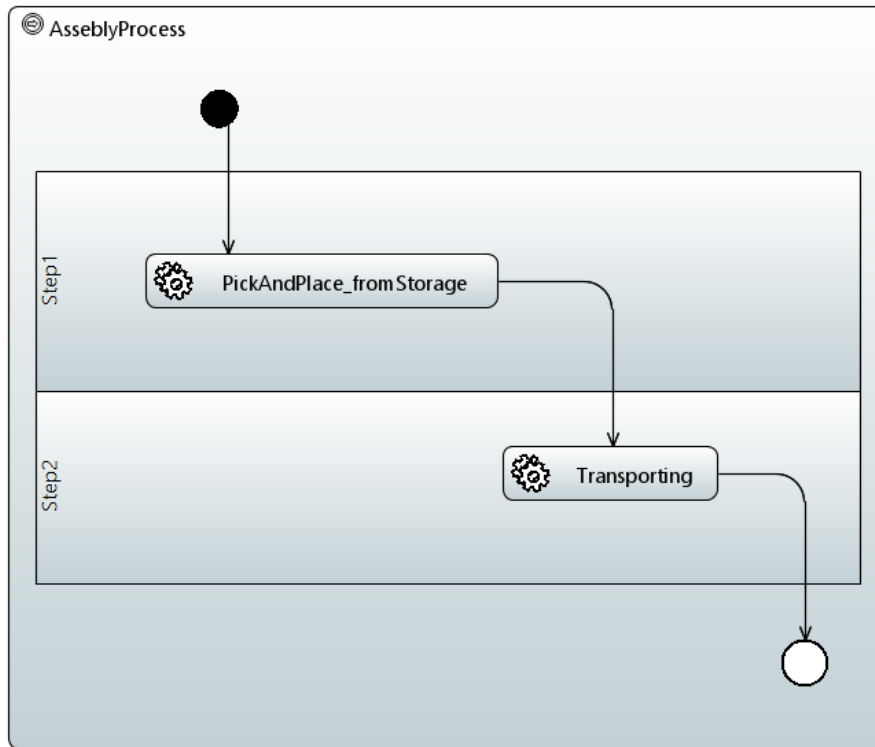


Figure 5.18: Assembly process BPMN diagram

- **Monitoring Submodel:**

At the core of this submodel are observed properties, which include status observations for entities named Ned and LittleTurtle, suggesting these are critical components or agents within the system whose operational statuses are continuously tracked. Other observed properties such as “Ned_pap_Observe”, “LittleTurtle_transporting_Observe”, “Storage_Number_Observe”, and “Infrared_Sensor_Observe” indicate a detailed surveillance setup designed to monitor an array of operational metrics, from transport activities to storage logistics and sensor data.

The submodel also encapsulates a set of basic events that serve as triggers or alerts within the system. These include notifications for malfunctions tagged as “DefectiveOperation”, and warnings for inventory issues such as “NoObjectsInStorage” and “NoObjectArrive”. These events are critical for maintaining system integrity, as they likely initiate corrective actions or flag issues for immediate attention.

Lastly, the operations listed like “checkOperationStatus()”, “checkSkills()”, “checkStorageNumberStatus()”, and “checkSensorStatus()” are indicative of diagnostic or analytical functions. These operations are purposed to evaluate the current state of various system elements, such as the efficacy of performed skills, the accuracy of storage counts, and the functionality of sensors.



Figure 5.19: Process monitor

Product AAS Model Design

- **AAS:**

the “ProductElement” stereotype from the mmo-profile shows that “AASProduct1” is recognized as a product component within the model, with specific attributes associated with it. The attributes listed include ‘mass’, which is of type Real and currently set to 0.1, likely indicating the weight of the product or part in some unit. The *entityID* attribute is of type String and appears to be empty, suggesting that a unique identifier for the product element has yet to be assigned or is not displayed in the snapshot.

Additionally, the *requiresActivity* attribute refers to an Activity array, which is linked to “AASAssemblyProcess” AAS model. This indicates that “AASProduct1” is associated with an assembly process, suggesting that its existence or function is contingent upon the activities defined within the AASAssemblyProcess, likely detailing the steps or operations required to produce or manage the product.

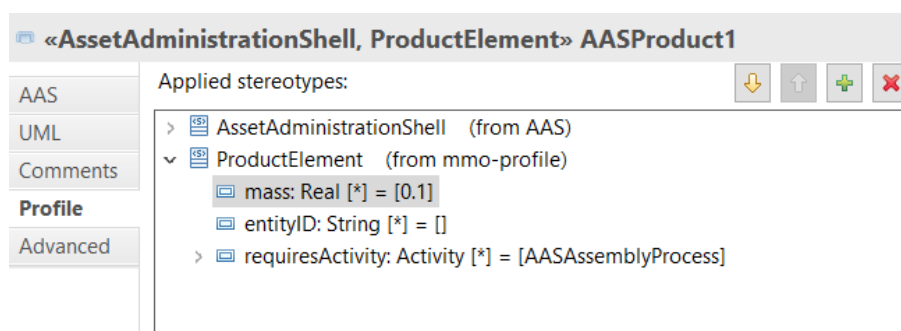


Figure 5.20: Little Turtle behavior BPMN representation

5.4.3 Engineering & Deployment

For the implementation of the Engineering & Deployment phase, the focus is on the combination of conceptual design and emerging technologies used to implement functional systems. According to Chapter 3, this phase consists mainly of the automated generation and deployment of AAS digital twins, the initial CBE process, and the construction of the initial knowledge base.

BaSyx Server Deployment

The functionality of transforming AAS models into BaSyx code facilitates the deployment of an AAS model into an HTTP server. The asset’s dynamic information model can be accessed and modified through the HTTP requests sent to the dynamic AAS server. Additionally, the API for invoking AAS operations is available. To orchestrate the production process, all the resources’ AAS servers involved in the process plan should be started and continuously accessible. The highlighted part of Figure 5.21 indicates that the “AASServer” of LittleTurtle is actively listening at `http://localhost:2021/aas` as designed in the previous section.

```

AASServer [Java Application] [pid: 19184]
15:26:45.332 [main] INFO c.f.a.p.c.AasProperties - Settings loaded from 'application.properties'.
15:26:47.454 [milo-nonce-util-secure-random] INFO o.e.m.o.s.c.u.NonceUtil - SecureRandom seeded in 0ms.
15:26:47.477 [main] INFO o.e.m.o.s.c.OpcUaClient - Eclipse Milo OPC UA Stack version: 0.6.1
15:26:47.477 [main] INFO o.e.m.o.s.c.OpcUaClient - Eclipse Milo OPC UA Client SDK version: 0.6.1
15:26:48.204 [Thread-0] INFO o.a.c.h.Http1NioProtocol - Initializing ProtocolHandler ["http-nio-2021"]
15:26:48.223 [Thread-0] INFO o.a.t.u.n.NioSelectorPool - Using a shared selector for servlet write/read
15:26:48.231 [Thread-0] INFO o.a.c.c.StandardService - Starting service [Tomcat]
15:26:48.232 [Thread-0] INFO o.a.c.c.StandardEngine - Starting Servlet Engine: Apache Tomcat/8.5.41
15:26:48.428 [Thread-0] INFO o.a.c.h.Http1NioProtocol - Starting ProtocolHandler ["http-nio-2021"]
15:26:48.448 [main] INFO a.m.AASServer - AAS 'AASLittleTurtle' is listening at: http://localhost:2021/aas
15:26:48.448 [main] INFO a.m.AASServer - You can find the API documentation at https://app.smogger.com/op12/BaSyx/basyx_submodel_repository_http_rest_api/v1#

```

Figure 5.21: AAS BaSyx server deployment example

Process Reconfiguration Capability Checking

During the design phase, the MaRCO Ontology profile is applied to the LocalSEA models. Also, the AASs have applied stereotypes corresponding to the different types of *Resources* existing in MaRCO. The stereotype *DeviceBlueprint* is applied to “AAS TurtleBot3_type” contains the information about a Turtlebot3 robot in LocalSEA. The capability *Transporting* mentioned above are attached to AAS capabilities owned by the “AAS TurtleBot3_type” as stereotypes. “LittleTurtle”, an instance of Turtlebot3, is defined as an *IndividualDevice*, so the attribute *hasDeviceBlueprint* is set to “AAS TurtleBot3_type”.

Once the user selects the product to produce, the rest of capability checking process is fully automated and triggered by a right-click command. Firstly, the OML Adapter is called to transform the AAS models into MaRCO instances. The resulting *AASs.owl* file contains all the AAS model capability-related information. Then the capability matchmaker takes the resources and product descriptions as input to infer the matchmaking results. These inference results are generated in the same folder as *AASs.owl* under the name of *matches.ttl*. Figure 5.22 shows the changing status of the required capability *Transporting* in the LocalSEA production process at different stages of capability checking. In the modeling environment, the corresponding stereotypes are applied to “AASProcess1” model.

As shown at the upper part of the figure, the attribute of *matchmakingRequired* is set to true to trigger the matchmaking inference. All the information defined in the process model is written to *AASs.owl*, as shown in the middle screenshot. The lower part of the figure shows the inferred information stored in *matches.ttl* after inference by the capability matchmaker. The figure shows the transporting process *has capability match* with the human operator typed devices, ConveyorBelt typed devices, or a human operator.

But it can only be implemented with TurtleBot3, because the parameters of the conveyor belt in this case doesn't fit the requirement.

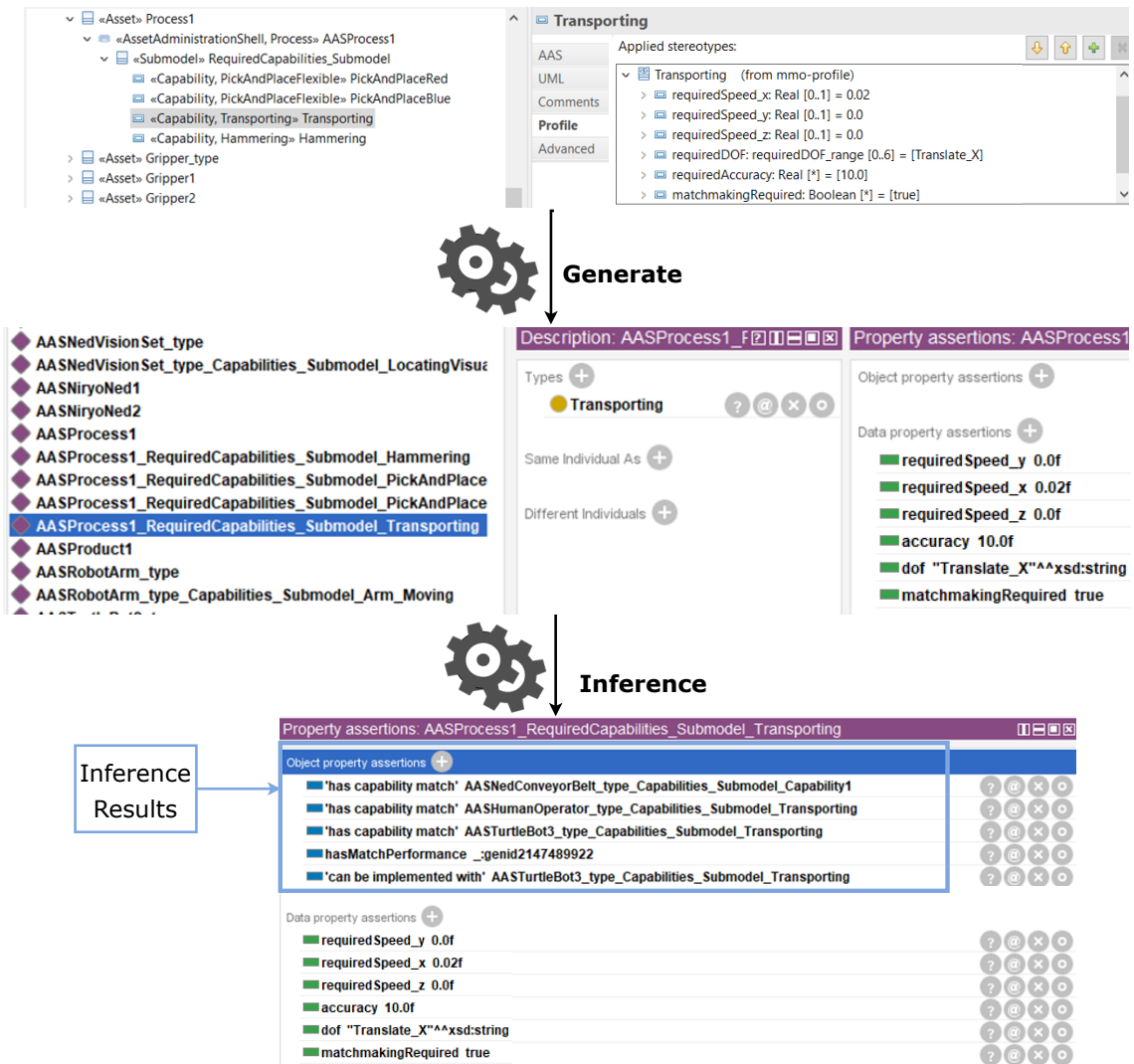


Figure 5.22: An AAS2MaRCO generation of a required capability

The capability checking results of the “AASProduct1” are grouped in a pop-up window shown in Figure 5.23. According to these results, *PickAndPlaceFlexible* can be implemented by Niryos, *Transporting* can be done by TurtleBot or conveyor belt, and human operators can realize all the capabilities required in this process, which just matches our previous definition of LocalSEA devices. Through this result list, the user can select the production line combination to be further checked in the feasibility checking module.

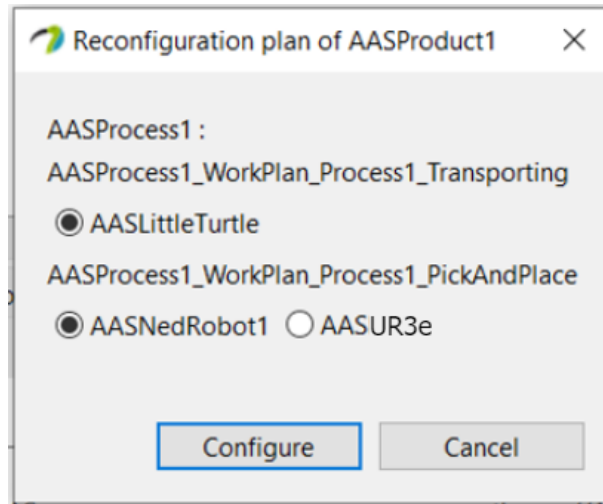


Figure 5.23: Capability checking result window

Process Orchestration

The functionality of transforming AAS models into BaSyx code facilitates the deployment of an AAS model into an HTTP server. The asset's dynamic information model can be accessed and modified through the HTTP requests sent to the dynamic AAS server. Additionally, the API for invoking AAS operations is available. To orchestrate the production process, all the resources' AAS servers involved in the process plan should be started and continuously accessible. To visualize the result of the process execution, we published a video⁶ in the documentation of Papyrus4Manufacturing. Figure 5.24 is an example flow that represents the transporting behavior of Little Turtle, which is defined in a BPMN diagram (see Figure 5.20).

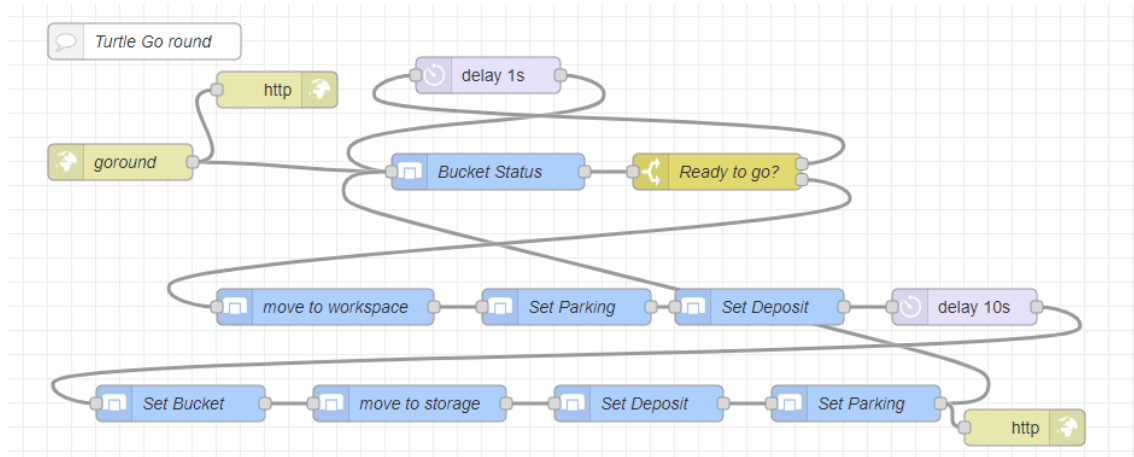


Figure 5.24: Node-RED orchestration flow of Little Turtle

Global Capability Knowledge Generation

Following the completion of the capability checking procedure, a 'match.ttl' file is produced, which serves as a repository for all the AAS capability information along with inferred knowledge. This file emerges as a foundational component for constructing a global knowledge base. Its significance lies in the encapsulation of comprehensive global

⁶https://youtu.be/G5Hfinm_guE

information which is pertinent to the system’s capabilities and the relational knowledge inferred during the capability checking. It also provides a structured framework that can facilitate the integration and synchronization of knowledge across the platform. As such, the `match.ttl` file not only reflects the current state of AAS capabilities but also paves the way for the enhancement of data cohesion and interoperability within the entire manufacturing system. Moreover, by serving as a central point of reference, it can greatly improve the efficiency of querying and updating system capabilities.

5.4.4 Operation & Maintenance

The Figure 5.25 shows the project structure of the process’s monitor. *ConsumerStreamGenerator.java* generates a stream for Kafka consumer stream for RSP reasoning use. This class subscribes to the topics that are generated from the AAS monitoring submodel. The class *ProcessKafkaConsumer.java* consumes Kafka messages, which involve reading data from a Kafka topic and processing it with the CSPARQL queries. *aasmap.ttl* contains the mapping definitions from JSON raw data to AAS RDF graph. *monitor_process1.ttl* contains the mapping definitions from JSON raw data to self-adaptation ontology-compliant triples. *Process1Monitor.ttl* is the RDF data model generated from the AAS monitoring submodel. *monitoring.owl* is the self-adaptation ontology representation presented in Section 3.3.4. Each of these files is essential from setting up the data streaming and consumption with Kafka to defining the data models and configurations for how the monitoring should be structured and understood semantically.

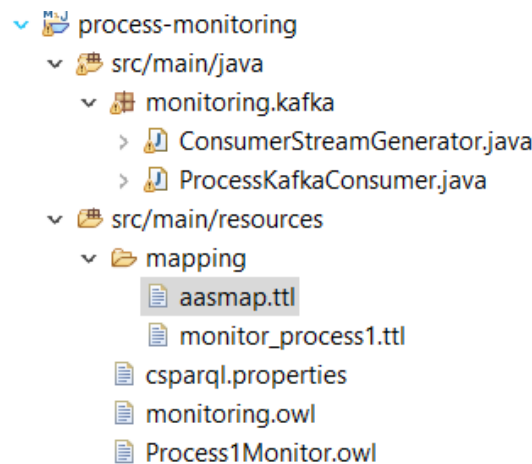


Figure 5.25: Project structure of process monitor

Here we refer to the scenario described in section 4.5.3, which discusses a continuous query evaluates whether the battery level of the LittleTurtle is sufficiently maintained to persist in its assigned task. The following listing (Listing 5.1) provides a snapshot of the query result at a given time. This excerpt illustrates how the system monitors and responds to the battery status, showcasing the practical application of the query in a real-world scenario. Based on the experimental data we have gathered, it has been observed that the disparity between the event time and the processing time for this particular query typically ranges from 7 to 15 milliseconds. This time difference reflects the efficiency and responsiveness of the system in handling real-time data. The event time refers to the actual moment when the data or event occurs, while the processing time is when the system processes this information. A narrower gap between these two times indicates a more efficient system, capable of quickly responding to changes in the LittleTurtle’s battery level and ensuring timely decision-making for its continued operation.

In light of the inferred triples from the system’s queries, it’s possible to implement

```

1 {
2   "@graph" : [ {
3     "@id" : "http://cea.list.papyrus4manufacturing/monitoring/turtle#BatteryTooLow",
4     "@type" : "http://cea.list.papyrus4manufacturing/monitoring#Warning"
5   }, {
6     "@id" : "http://streamreasoning.org/result/1702286076128",
7     "eventTime" : "1702286076128",
8     "processingTime" : "1702286076139"
9   } ],
10  "@context" : {
11    "processingTime" : {
12      "@id" : "http://streamreasoning.org/csparql/processingTime",
13      "@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
14    },
15    "eventTime" : {
16      "@id" : "http://streamreasoning.org/csparql/eventTime",
17      "@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
18    }
19  }
20 }

```

Listing 5.1: CSPARQL query outstream result

further adaptations to the system’s behavior. However, due to time constraints, we were unable to cover the complete MAPE-K loop within the scope of our current work. The incompleteness of this loop in our implementations points to an area for future exploration and development. In the conclusion section, we will discuss and outline potential directions for future research and development.

5.5 Conclusion

This chapter provides a complete explanation of how the CBSAM architecture was designed, developed, and used, utilizing several use cases from the testbed LocalSEA. How the AAS model was elaborated using P4M, how the capability checking was accomplished, how the AAS digital twin was deployed and orchestrated, and finally how the AAS digital twin was monitored are all described in this chapter. However, due to time constraints and the fact that all development is a proof-of-concept phase, the verification and comparison of performance is not covered in this chapter.

Chapter 6

Conclusion & Perspectives

In conclusion, this thesis emphasizes the upcoming transformation of manufacturing driven by Industry 4.0. The emergence of intelligent and adaptive manufacturing systems represents a significant advancement characterized by the ability to manage diverse and flexible production. Challenges related to the interoperability, adaptability, and robustness of manufacturing systems are at the heart of this transformation. The digital twin approach offers an opportunity to bring the physical world closer to the digital world, leading to better monitoring, more accurate planning, and efficient execution of production processes. The implementation of digital twins combined with the principles of model-driven engineering is a promising solution to these challenges. The proposed architecture CBSAM (Capability-Based Self-Adaptive Manufacturing) leverages the methodologies in software engineering and Industry 4.0, such as CBE (capability-based engineering), MDE (model-driven engineering), and MAPE-K (Monitor, Analyze, Plan, Execute over a shared Knowledge) structure to improve the responsiveness and flexibility of production systems.

This section gives a conclusion to the thesis, which will be outlined in two main parts. Firstly, this section recapitulates the contributions made by this research, which includes (1) the proposition of the CBSAM architecture to enable flexible manufacturing through digital twins, (2) the implementation of this architecture that brings the conceptual aspects into practical software applications, (3) the real-world application through a robotic testbed, (4) the dissemination of the research results through various publications and communication in the international conferences. The second part focuses on the future perspectives stemming from this research. This includes a discussion on the consolidation of the implementation and the generalization of the CBSAM architecture.

6.1 Contributions

6.1.1 Proposed Architecture

This CBSAM architecture aligns with the engineering phases that cover the entire lifecycle of the production system, from design to operation. Its design enables the ease of use, even for non-specialists, for the management of the manufacturing system through digital twins. This alignment with industry trends underlines its forward-thinking approach, which is crucial for meeting the demands of Industry 4.0.

1. To address the challenge of syntactic interoperability, the architecture incorporates the emerging standard AAS for digital twin modeling, accommodating different types of production processes, products, and resources. Since AAS is mainly focusing on structural digital twins modeling, we proposed to integrate the standard BPMN language for production processes modeling.

2. For semantic interoperability, the use of ontology enables systems to interpret and process data contextually, leading to a more automatic process-resource matchmaking procedure.
3. The adaptability challenge is enhanced through capability-based engineering, allowing it to adjust to various production plans seamlessly.
4. The incorporation of the MAPE-K loop structure is fundamental for robustness, ensuring self-adaptive system behavior and responding dynamically to changes in the production environment.

6.1.2 Software Implementations

The implementation phase is a crucial transition from conceptual exploration to practical realization, emphasizing the core contributions of the thesis research. The primary contributions revolve around the development of software tools and methodologies.

1. The innovative integration of the MDE technologies in the realm of digital twin modeling. The development of the P4M toolset provides the modeling environment that eases the creation and comprehension of AAS-compliant digital twins with the graphical diagrams. This thesis contributes to the production process modeling design by integrating the BPMN modeling plugin and the asset's internal structure modeling by extending the UML composite structure diagram.
2. The implementation of capability checking of CBE represents another contribution of this research. This implementation successfully bridges the integration of MDE with ontology, which not only provides rich semantic descriptions to digital twin models but also empowers model-driven engineering tools with enhanced automated reasoning capabilities. The capability checking procedure results to the automatic semantic alignment of the process requirements with the potential resources for the production.
3. The creation of a set of customized nodes in Node-RED simplifies the orchestration of process digital twins marks yet another important contribution. Transforming the static BPMN processes into executable and dynamic workflows demonstrates the practicality and applicability of the research in real-world manufacturing scenarios.
4. The implementation of monitoring and diagnosis module for dynamic AAS models using semantic computing technologies, which provides better interoperability between sensors that are supplied by different manufacturers with specific characteristics that can be taken into account by the diagnostic rules, from units of measurement to non-functional properties. This module is a crucial step for realizing a self-adaptive CBE system, enabling it to detect or anticipate system failures or potential issues, thus contributing to the awareness of the problem happening in production lines and enhancing the robustness.

6.1.3 Testbed Validation

In addition to the software development contributions, this thesis also contributes to the digital twin construction for an academic robotic testbed. The use of an academic robotic testbed allows a flexible environment where experimental and innovative ideas can be tested with less constraints and pressures than in industrial practice. This flexibility is essential for thorough testing and validation of new concepts and architectures.

This testbed serves as a proof of concept for the CBSAM architectural framework proposed in the thesis, and a mean for validating the various implementations that have been discussed. This testbed not only tests and validates the functionality and efficiency of the proposed methods but also provide insights into their potential improvements.

6.1.4 Publications

Throughout the course of this research, several conference articles have been published, contributing significantly to the communication and dissemination of the work. These articles encompass various aspects of the research, presenting the development and application of digital twin technology in the Industry 4.0 context.

1. ETFA 2021 [99]: This work-in-progress paper discusses the concept of Industry 4.0, emphasizing the challenges for the implementation of flexible production lines. It addresses the criticality of rapid reconfiguration for manufacturers, especially SMEs, to maintain commercial success. The paper analyzed the capability-based engineering approach to address interoperability and adaptability problems in flexible production lines.
2. IECON 2021 [100]: Focusing on the autonomy of future intelligent manufacturing systems, this article underlines the importance of monitoring the production process, rapid re-planning, and responding to unforeseen situations securely. It proposes a capability-based operation and engineering approach using the AAS standard and details the modeling concepts necessary for this approach.
3. ISIE 2022 [94]: This paper shares insights on an OPC UA-based robotic testbed for Industry 4.0 research, discussing the challenges of updating the information model with system design evolution. It outlines the testbed's development strategy, following the SysML and OPC UA standards.
4. Moddit'22 [101]: Addressing the gap in semantic interoperability for Industry 4.0 components, this article proposes a modeling approach for AAS-based digital twins using ontologies. It emphasizes the role of formal semantics and the general mapping between UML taxonomies and ontology taxonomies.
5. ONCON 2022 [97]: This paper presents an approach to bridge ROS 1 and ROS 2 robots to an OPC UA PubSub network, derived from the development of an OPC UA-based robotic testbed. It addresses the challenges of networking robots with OPC UA devices and enabling rapid system integration for quick experiments.
6. ICPS 2023 [102]: This article focuses on the interoperability gap in digital twins. It introduces an ontology-based approach for semantic capability checking and implements the capability checking step by enabling the semantic interoperability of AAS-based digital twin models, which transforms AAS-based plant models into ontology instances for resource matching. The implemented framework includes the transformation module, the matchmaking module, and the user interface.
7. ETFA 2023 [35]: This paper introduces the Papyrus4Manufacturing tool, which provides a model-based systems engineering approach to AAS. It supports the creation, deployment, and connection of AAS digital twins using the OPC UA protocol and evaluates its usability with an academic use case.

6.2 Future Perspectives

6.2.1 Short-Term Perspectives

One of the primary objectives is to implement and cover the architecture proposed in this thesis work. This involves enhancing the integration of the various components and processes to create a more cohesive and comprehensive system. The short-term future involved here are basically issues that have been studied but have not been completed.

Completing Real-Time Adaptation in the MAPE-K Framework

The implementation of the MAPE-K loop, which currently halts at monitoring and diagnosis, could be extended to incorporate real-time adaptive decision-making and action-taking. Envisioning a future where the planning component of MAPE-K is driven by an intelligent algorithm allows for dynamic and responsive system management. This advancement would enable systems to not only identify and diagnose issues in real-time but also to autonomously adapt and respond effectively.

Enhanced Feasibility Checking through AAS-Compliant Digital Twins

The concept of feasibility checking, currently focused on tests and validations within a simulation environment, stands on the brink of a significant evolution. An enhancement could be realized through the integration of AAS-compliant digital twins. This advancement would involve defining specialized simulation-related information within a submodel that will enable the automatic deployment of simulations using MDE methods. Such an approach promises to streamline the simulation process, allowing for more flexible and efficient feasibility assessments in complex system environments.

6.2.2 Long-Term Perspectives

The goal here is to enhance the proposed architecture in this thesis, ensuring its seamless integration and efficient operation in larger-scale systems. This involves refining the system to accommodate greater complexity and increased resource demands while maintaining its reliability and performance.

Global Event Correlation in MAPE-K

The central research challenge lies in advancing the MAPE-K framework to tackle the complex task of correlating global events across numerous interconnected devices and equipment. While current implementations demonstrate alerts only for single assets, a future approach would consider scenarios where multiple components collectively contribute to a specific outcome or effect. Developing capabilities for this kind of complex event correlation and analysis would vastly improve the system's ability to predict, diagnose, and respond to multi-faceted issues. Such an enhancement would significantly elevate the framework's utility in managing intricate and interconnected production lines, leading to more resilient operational management.

Advanced Implementation of Skill Execution in Plug-and-Produce Systems

In the realm of skill execution, the future requires a shift towards advanced “plug-and-produce” concepts. As the number of resources in the system continues to increase, the efficiency of centrally locating the corresponding resources decreases. Therefore, a corresponding approach is to give AAS digital twins with a degree of intelligence to endow them

with environmental awareness. This involves analyzing their own geometric location information and current capabilities to determine if they can handle a specific task required for the overall production process. Subsequently, these resources can autonomously self-organize to test whether they can form new production lines. The key research challenge lies in efficiently transitioning from centralized capability checking to distributed demand sensing while incorporating intelligence into AAS and optimizing resource self-organization for improved production efficiency. Each device in a system should, upon integration, become immediately aware of its role and configuration. This may require the establishment of a central registry for the digital twin models of AAS, where the digital twin models within the system can autonomously register or unregister themselves. Additionally, it would necessitate an enhanced task allocation controller equipped with greater intelligence to monitor the overall system. The development of such an intelligent system would enable seamless integration among devices, leading to more adaptable production processes.

6.2.3 Domain Perspectives

Another aspect of future research is to abstract and generalize the methods developed in this thesis. The aim is to extend the applicability of these methods beyond the specific requirements of the project, creating a universally applicable approach. This involves refining the techniques to be adaptable and relevant across different domains, enabling a wider range of applications for industrial digital twins in general.

Multiple Production Line

The first focus on the generalization involves integrating continuous command flows, encompassing planning, and maintenance of equipment across multiple production lines. This integration will provide a more holistic view of the entire manufacturing system. The challenge lies in creating a framework that is sufficiently flexible to accommodate the unique dynamics of different production environments while maintaining its core business objective.

Complete Management of Production Flow

Another aspect to consider is the facilitation of inter-company exchanges to manage production flows. This includes taking into account upstream supplier constraints and downstream customer demands. The future architecture should include also the representation of the product orders which trigger the process from the design to the delivery of the products. The development of algorithms to handle such complex, multi-faceted production networks is inherently challenging, with the potential for exponential complexity.

AAS for Logistics

Finally, the inclusion of the AAS with logistics management [103] in the future development of these methodologies is imperative. This integration is crucial for ensuring seamless interoperability and standardization across different systems and platforms. The AAS standard will provide a solid foundation for the development of more sophisticated and integrated digital twin models, paving the way for widespread adoption and implementation of the AAS standard in various industrial sectors.

In conclusion, the different types of perspectives discussed here underscore the enormous complexity of efforts required to realize the promise of Industry 4.0. These insights show that while significant progress has been made, there is still quite a long way to go.

Recognizing these challenges is critical as it frames the current discussion and identifies some possible directions for future efforts.

List of Figures

| | | |
|------|---|----|
| 1.1 | Mindmap for concepts appear in the thesis | 4 |
| 2.1 | RAMI model (reproduced from: Standardization Council Industrie 4.0) | 9 |
| 2.2 | AAS composite metamodel | 11 |
| 2.3 | Property value statement metamodel | 19 |
| 3.1 | Overview of capability-based self-adaptation manufacturing architecture - Part 1 | 27 |
| 3.2 | Overview of capability-based self-adaptation manufacturing architecture - Part 2 | 29 |
| 3.3 | Papyrus4Manufacturing modeling environment | 31 |
| 3.4 | Excerpt from the AAS UML profile | 32 |
| 3.5 | AAS Marco vocabularies mapping | 34 |
| 3.6 | Relations between capability and skill | 36 |
| 3.7 | Monitoring metamodel | 37 |
| 3.8 | Monitoring ontology object properties | 38 |
| 3.9 | Diagnosis metamodel | 39 |
| 3.10 | Capability-based reconfiguration approach | 42 |
| 3.11 | Capability checking architecture | 43 |
| 3.12 | Skill execution architecture | 44 |
| 3.13 | MAPE-K loop for a single AAS model | 46 |
| 3.14 | Ontology relation | 47 |
| 4.1 | AAS design diagram | 51 |
| 4.2 | Bill of material composition diagram | 52 |
| 4.3 | Workflow BPMN process diagram | 53 |
| 4.4 | Capability checking implementation workflow | 53 |
| 4.5 | Exemplary OML syntax | 55 |
| 4.6 | MaRCO screenshot in Protégé | 56 |
| 4.7 | MaRCO UML profile | 56 |
| 4.8 | Exemplary OWL instance | 57 |
| 4.9 | User interaction scenario | 58 |
| 4.10 | Papyrus4Manufacturing architecture | 60 |
| 4.11 | Node-RED flow representation of Process1 | 61 |
| 4.12 | Overall run-time monitoring & diagnosis implementation | 62 |
| 4.13 | Diagnosis rule modeling in P4M | 63 |
| 5.1 | LocalSEA cell | 72 |
| 5.2 | LocalSEA description | 74 |
| 5.3 | The architecture of ROS 1 and ROS 2 with UA bridges | 76 |
| 5.4 | REQs related to the AAS digital twin management | 78 |
| 5.5 | MaRCO structure | 79 |

| | | |
|------|---|-----|
| 5.6 | MaRCO transporting capability definition in Protégé | 80 |
| 5.7 | MaRCO required transporting capability definition in Protégé | 82 |
| 5.8 | LocalSEA OPC UA server configuration | 83 |
| 5.9 | Profile application to AAS model package | 87 |
| 5.10 | TurtleBot3_Type AAS model design | 87 |
| 5.11 | TurtleBot3_Type capability design | 88 |
| 5.12 | LittleTurtle AAS model design | 89 |
| 5.13 | Little Turtle property | 90 |
| 5.14 | Little Turtle behavior BPMN representation | 90 |
| 5.15 | Little Turtle monitor | 91 |
| 5.16 | Process AAS annotation design | 92 |
| 5.17 | Required capability (Transporting) | 92 |
| 5.18 | Assembly process BPMN diagram | 93 |
| 5.19 | Process monitor | 94 |
| 5.20 | Little Turtle behavior BPMN representation | 94 |
| 5.21 | AAS BaSyx server deployment example | 95 |
| 5.22 | An AAS2MaRCO generation of a required capability | 96 |
| 5.23 | Capability checking result window | 97 |
| 5.24 | Node-RED orchestration flow of Little Turtle | 97 |
| 5.25 | Project structure of process monitor | 98 |
| | | |
| A.1 | Concept de jumeau numérique pour une ligne de production | 120 |
| A.2 | Carte conceptuelle pour les concepts apparaissant dans la thèse | 123 |
| A.3 | Vue d'ensemble de l'architecture CBSAM - Partie 1 | 127 |
| A.4 | Vue d'ensemble de l'architecture CBSAM - Partie 2 | 128 |
| | | |
| B.1 | Installation Window | 131 |
| B.2 | Select Semantic Feature | 132 |
| B.3 | Trust Unsigned Contents | 132 |
| B.4 | Add MaRCO Profile | 133 |
| B.5 | Annotate AAS Models | 133 |
| B.6 | Capability Checking Command | 134 |
| B.7 | Matchmaking Requirement | 134 |
| B.8 | Result Window | 135 |
| B.9 | Project Explorer | 135 |

List of Tables

| | | |
|-----|--|----|
| 2.1 | Excerpt of AAS metamodel V3RC2 definition from [10] | 10 |
| 2.2 | Tools implementing AAS standard | 13 |
| 3.1 | Different transformations in CBSAM architecture | 28 |
| 3.2 | AAS Meta-model and UML Meta-model Mapping | 31 |
| 3.3 | General concept mapping | 33 |
| 3.4 | Definitions of concepts for monitoring | 38 |
| 3.5 | Definitions of concepts for diagnosis | 40 |
| 4.1 | BPMN and Node-RED nodes mapping | 61 |
| 5.1 | A summary of simple network communication evaluation for the testbed | 77 |
| 5.2 | Dynamic properties of Little Turtle in OPC UA server | 83 |
| 5.3 | OPC UA methods of LittleTurtle | 84 |
| 5.4 | Exemplary events of LocalSEA resources | 85 |
| 5.5 | Exemplary events of related to scenario process | 85 |
| 5.6 | Overview of AAS model design | 86 |

Bibliography

- [1] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242, August 2014.
- [2] Dr. Michael Grieves. Digital twin: Manufacturing excellence through virtual factory replication. Technical report, US Florida Institute of Technology, Melbourne (2014), 01 2014.
- [3] E H Glaessgen and D S Stargel. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. *Structural Dynamics*, 2012.
- [4] Azad Madni, Carla Madni, and Scott Lucero. Leveraging Digital Twin Technology in Model-Based Systems Engineering. *Systems*, 7(1):7, January 2019.
- [5] Nelly Bencomo, Sebastian Götz, and Hui Song. Models@run.time: a guided tour of the state of the art and research challenges. *Software & Systems Modeling*, 18(5):3049–3082, October 2019.
- [6] Abe Zeid, Sarvesh Sundaram, Mohsen Moghaddam, Sagar Kamarthi, and Tucker Marion. Interoperability in smart manufacturing: Research challenges. *Machines*, 7(2), 2019.
- [7] José Lameh, Alexandra Dubray, and Marija Jankovic. Towards a configuration management integration to feature models in model-based product line engineering. *Proceedings of the Design Society*, 3:3581–3590, 2023.
- [8] Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, 1990.
- [9] J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
- [10] Details of the asset administration shell - part1, (version 3.0rc02). *Plattform Industrie 4.0*, 5 2022. https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html.
- [11] OMG. Business process model and notation v2.0, 2010. <https://www.omg.org/spec/BPMN/2.0/>.
- [12] Eeva Järvenpää, Niko Siltala, Otto Hylli, and Minna Lanz. The development of an ontology for describing the capabilities of manufacturing resources. *Journal of Intelligent Manufacturing*, 30(2):959 – 978, 2019.
- [13] Describing capabilities of industrie 4.0 components. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Capabilities_Industrie40_Components.html, journalPlattform Industrie 4.0.

- [14] Gianvito Urgese, Paolo Azzoni, Jan van Deventer, Jerker Delsing, and Enrico Macii. An engineering process model for managing a digitalised life-cycle of products in the industry 4.0. pages 1–6, 04 2020.
- [15] Romina Eramo, Francis Bordeleau, Benoit Combemale, Mark van den Brand, Manuel Wimmer, and Andreas Wortmann. Conceptualizing digital twins. *IEEE Software*, 39(2):39–46, 2022.
- [16] Michael Grieves. *Virtually Intelligent Product Systems: Digital and Physical Twins*, pages 175–200. 07 2019.
- [17] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, 7:167653–167671, 2019.
- [18] Douglas C Schmidt et al. Model-driven engineering. *Computer-IEEE Computer Society-*, 39(2):25, 2006.
- [19] Adalberto Sampaio Junior, Fabio Costa, and Peter Clarke. A model-driven approach to develop and manage cyber-physical systems. volume 1079, 09 2013.
- [20] Francis Bordeleau, Benoit Combemale, Romina Eramo, Mark van den Brand, and Manuel Wimmer. Towards model-driven digital twin engineering: Current opportunities and future challenges. In Önder Babur, Joachim Denil, and Birgit Vogel-Heuser, editors, *Systems Modelling and Management*, pages 43–54, Cham, 2020. Springer International Publishing.
- [21] Andreas Wortmann, Olivier Barais, Benoit Combemale, and Manuel Wimmer. Modeling languages in Industry 4.0: an extended systematic mapping study. *Software and Systems Modeling*, 19(1):67–94, January 2020.
- [22] Roland Heidel. Industrie 4.0: The reference architecture model rami 4.0 and the industrie 4.0 component. 2019.
- [23] Relationships between i4.0 components – composite components and smart production, 6 2017.
- [24] Xun Ye and Seung Ho Hong. Toward industry 4.0 components: Insights into and implementation of asset administration shells. *IEEE Industrial Electronics Magazine*, 13(1):13–25, 2019.
- [25] Constantin Wagner, Julian Grothoff, Ulrich Epple, Rainer Drath, Somayeh Malakuti, Sten Grüner, Michael Hoffmeister, and Patrick Zimmermann. The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2017.
- [26] Giovanni di Orio, Pedro Maló, and José Barata. Novaas: A reference implementation of industrie4.0 asset administration shell with best-of-breed practices from it engineering. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 5505–5512, 2019.
- [27] Erich Barnstedt, Birgit Boss, E Clauer, D Isaacs, SW Lin, S Malakuti, P van Schalkwykm, and T Weber Martins. Open source drives digital twin adoption. *IIC J. Innov*, 2021.

- [28] Jonathan Fuchs, Jan Schmidt, Jörg Franke, Kasim Rehman, Manuel Sauer, and Stamatis Karnouskos. I4.0-compliant integration of assets utilizing the asset administration shell. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1243–1247, 2019.
- [29] Michael Jacoby, Michael Baumann, Tino Bischoff, Hans Mees, Jens Müller, Ljiljana Stojanovic, and Friedrich Volz. Open-source implementations of the reactive asset administration shell: A survey. *Sensors*, 23(11), 2023.
- [30] Ljiljana Stojanovic, Thomas Usländer, Friedrich Volz, Christian Weißenbacher, Jens Müller, Michael Jacoby, and Tino Bischoff. Methodology and tools for digital twin management—the fa3st approach. *IoT*, 2(4):717–740, 2021.
- [31] Andreas Orzelski, Michael Hoffmeister, and Marko Ristin. Eclipse aasx package explorer.
- [32] Sebastian Wolf, Kasim Rehman, Helge Dickel, Tobias Ostermann, Manuel Sauer, Philipp Huebner, and Yannik Schiebellhut. Sap/i40-aas.
- [33] Peter Drahoš Rudolf Pribiš, Lukáš Beňo. Asset administration shell design methodology using embedded opc unified architecture server. *Electronics*, 10(20), 2021.
- [34] Papyrus model driven workbench. <https://www.eclipse.org/papyrus/>.
- [35] Saadia Dhouib, Yining Huang, Asma Smaoui, Tapanta Bhanja, and Volkan Gezer. Papyrus4manufacturing: A model-based systems engineering approach to aas digital twins. 2023.
- [36] Saadia Dhouib, Asma Smaoui, Ibtehil Khemir, Tapanta Bhanja, and Volkan Gezer. Papyrus for manufacturing, 2023. <https://www.eclipse.org/papyrus/components/manufacturing/>.
- [37] Alexander Roth and Bernhard Rumpe. Towards product lining model-driven development code generators. In *In Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development*, pages 539–545. SciTePress, 2015.
- [38] Katrin Hölldobler, Judith Michael, Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. Innovations in Model-based Software and Systems Engineering. *The Journal of Object Technology*, 18(1), July 2019.
- [39] Fraunhofer IESE. Eclipse basyx. <https://www.eclipse.org/basyx/>.
- [40] CS Sauer and Holger Eichelberger. Performance evaluation of basyx based asset administration shells for industry 4.0 applications. *Softwaretechnik-Trends*, 43(1):47–49, 2023.
- [41] Felix Larrinaga, William Ochoa, Alain Perez, Javier Cuenca, Jon Legaristi, and Miren Illarramendi. Node-red workflow manager for edge service orchestration. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2022.
- [42] Julius Pfrommer, Miriam Schleipen, and Jurgen Beyerer. PPRS: Production skills and their relation to product, process, and resource. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–4, Cagliari, Italy, September 2013. IEEE.

- [43] Plattform I4.0. Information model for capabilities, skills & services. <https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/CapabilitiesSkillsServices.html>.
- [44] Roman Froschauer, Aljosha Köcher, Kristof Meixner, Siwara Schmitt, and Fabian Spitzer. Capabilities and skills in manufacturing: A survey over the last decade of etfa. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.
- [45] Giancarlo Guizzardi. *Conceptualizations, Modeling Languages, and (Meta) Models*. 2007. Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference, DB&IS'2006.
- [46] Luis Palacios Medinacelli, Florian Noyrit, and Chokri Mraidha. Augmenting model-based systems engineering with knowledge. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS '22*, page 351–358, New York, NY, USA, 2022. Association for Computing Machinery.
- [47] Doug Migliori (CloudBlue) Anto Budiardjo (Padi). Digital twin system interoperability framework. *Digital twin consortium whitepaper*, 2021.
- [48] Jan deMeer. Semantics for i4.0 smart manufacturing. In Ralf H. Reussner, Anne Koziolk, and Robert Heinrich, editors, *INFORMATIK 2020*, pages 289–298. Gesellschaft für Informatik, Bonn, 2021.
- [49] Birgit Vogel-Heuser, Felix Ocker, Iris Weiß, Robert Mieth, and Frederik Mann. Potential for combining semantics and data analysis in the context of digital twins. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2207):20200368, 2021.
- [50] Jürg Meierhofer, Lukas Schweiger, Jinzhi Lu, Simon Züst, Shaun West, Oliver Stoll, and Dimitris Kiritsis. Digital twin-enabled decision support services in industrial ecosystems. *Applied Sciences*, 11(23), 2021.
- [51] Marie Platenius-Mohr, Somayeh Malakuti, Sten Grüner, and Thomas Goldschmidt. Interoperable digital twins in iiot systems by transformation of information models: A case study with asset administration shell. In *Proceedings of the 9th International Conference on the Internet of Things, IoT 2019*, New York, NY, USA, 2019. Association for Computing Machinery.
- [52] A. Perzylo et al. Capability-based semantic interoperability of manufacturing resources: A basys 4.0 perspective. *IFAC-PapersOnLine*, 52(13):1590–1596, 2019.
- [53] Dominique Ernadote. Ontology-based pattern for system engineering. In *Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems, MODELS '17*, page 248–258. IEEE Press, 2017.
- [54] Maged Elaasar. Definition of modeling vs. programming languages. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*, pages 35–51, Cham, 2018. Springer International Publishing.
- [55] Markus Damm. D-2.1 Modulares Erweiterungskonzept der BaSys 4.0 Fähigkeitenontologie. https://wiki.eclipse.org/File:2020-02-28_BaSys42_D-2.1.C4I_Ontology_Model.pdf.

- [56] Anant Mital, Niko Siltala, Eeva Järvenpää, and Minna Lanz. Web-based solution to automate capability matchmaking for rapid system design and reconfiguration. *Procedia CIRP*, 81:288–293, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.
- [57] W3C. Owl, Feb 2017. <https://www.w3.org/OWL/>.
- [58] Sparql query language for rdf. <https://www.w3.org/TR/rdf-sparql-query/>, 2008.
- [59] Yingfeng Zhang, Cheng Qian, Jingxiang Lv, and Ying Liu. Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor. *IEEE Transactions on Industrial Informatics*, 13(2):737–747, 2017.
- [60] E. Francalanza, J. Borg, and C. Constantinescu. A knowledge-based tool for designing cyber physical production systems. *Computers in Industry*, 84:39–58, 2017.
- [61] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Abmann. Toward a framework for self-adaptive workflows in cyber-physical systems. *Software & Systems Modeling*, 18(2):1117–1134, April 2019.
- [62] Tao Ma, Shaukat Ali, and Tao Yue. Modeling foundations for executable model-based testing of self-healing cyber-physical systems. *Software & Systems Modeling*, 18:2843–2873, 2019.
- [63] Detlef Olschewski, Ulrich Epple, Birgit Boss, Torben Miny, Wilfried Hartmann, marco hoch, Christoph Legat, Ulrich Löwen, Daniel Stock, and Thomas Usländer. Din spec 92000 data exchange on the base of property value statements. 09 2019.
- [64] U. Epple, M. Mertens, F. Palm, and M. Azarmipour. Using properties as a semantic base for interoperability. *IEEE Transactions on Industrial Informatics*, 13(6):3411–3419, Dec 2017.
- [65] F.-N. Demers and J. Malenfant. Reflection in logic, functional and object-oriented programming: a short comparative study. In *IJCAI’95 Workshop on Reflection and Metalevel Architectures and their Applications in AI*, pages 29–38, 1995.
- [66] B.C. Smith. Reflection and semantics in lisp. In *Proceedings of the ACM POPL’84*, pages 23–35, 1984.
- [67] S. Frølund and J. Koistinen. Qml: A language for quality of service specification. Tech. Report HPL-98-10, Software Technology Laboratory, Hewlett-Packard, 1998.
- [68] N. Ben Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny. QoS-aware service composition in dynamic service oriented environments. In *Middleware 2009*. Springer, 2009.
- [69] P. Châtel, J. Malenfant, and I. Truck. QoS-based late-binding of service invocations in adaptive business processes. In *ICWS 2010*, pages 227–234, 2010.
- [70] O. Rogovchenko and J. Malenfant. Handling hardware heterogeneity through rich interfaces in a component model for autonomous robotics. In *SIMPAR 2010*, Springer LNAI 6472, pages 312–323, 2010.
- [71] Amina Bekkouche, Sidi Mohammed Benslimane, Marianne Huchard, Chouki Tibermacine, Fethallah Hadjila, and Mohammed Merzoug. QoS-aware optimal and automated semantic web service composition with user’s constraints. *Service Oriented Computing and Applications*, 11(2):183–201, June 2017.

- [72] L. Jaiem, L. Lapierre, K. Godary-Dejean, and D. Crestani. Towards performance guarantee for autonomous mobile robotic mission: an approach for hardware and software resource management. In *TAROS 2016*, pages 189–195, 2016.
- [73] World Wide Web Consortium (W3C). Resource Description Framework (RDF). <https://www.w3.org/RDF/>.
- [74] World Wide Web Consortium (W3C) Community Group. RDF Stream Processing Community Group. <https://www.w3.org/community/rsp/>.
- [75] Daniele Dell’Aglío, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein. Stream reasoning: A survey and outlook. *Data Science*, 1(1-2):59–83, 2017.
- [76] Riccardo Tommasini, Pieter Bonte, Femke Ongenaë, and Emanuele Della Valle. *RSP4J: An API for RDF Stream Processing*, pages 565–581. 05 2021.
- [77] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-sparql: Sparql for continuous querying. In *Proceedings of the 18th International Conference on World Wide Web, WWW ’09*, page 1061–1062, New York, NY, USA, 2009. Association for Computing Machinery.
- [78] Anh Le-Tuan, Manh Nguyen-Duc, Chien-Quang Le, Trung-Kien Tran, Manfred Hauswirth, Thomas Eiter, and Danh Le-Phuoc. Cqels 2.0: Towards a unified framework for semantic stream fusion, 2022.
- [79] Nathan Gruber and Birte Glimm. A comparative study of stream reasoning engines. In *European Semantic Web Conference*, pages 21–37. Springer, 2023.
- [80] Rahat Iqbal, Tomasz Maniak, Faiyaz Doctor, and Charalampos Karyotis. Fault detection and isolation in industrial processes using deep learning approaches. *IEEE Transactions on Industrial Informatics*, 15(5):3077–3084, 2019.
- [81] Dubravko Miljković. Fault detection methods: A literature survey. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 750–755, 2011.
- [82] Yuanfang Chi, Yanjie Dong, Z. Jane Wang, F. Richard Yu, and Victor C. M. Leung. Knowledge-based fault diagnosis in industrial internet of things: A survey. *IEEE Internet of Things Journal*, 9(15):12886–12900, 2022.
- [83] Márton Búr, Gábor Szilágyi, András Vörös, and Dániel Varró. Distributed graph queries over models@run.time for runtime monitoring of cyber-physical systems. *International Journal on Software Tools for Technology Transfer*, 22(1):79–102, February 2020.
- [84] Xilang Tang, Guo Chi, Lijie Cui, Andrew W. H. Ip, Kai Leung Yung, and Xiaoyue Xie. Exploring research on the construction and application of knowledge graphs for aircraft fault diagnosis. *Sensors*, 23(11), 2023.
- [85] Feixiang Xu, Xinhui Liu, Wei Chen, Chen Zhou, and Bingwei Cao. Ontology-based method for fault diagnosis of loaders. *Sensors*, 18(3), 2018.
- [86] Carlos Hernández, Julita Bermejo-Alonso, Ricardo Sanz, Craig Schlenoff, Stephen Balakirsky, and Henrik Christensen. A self-adaptation framework based on functional knowledge for augmented autonomy in robots. *Integr. Comput.-Aided Eng.*, 25(2):157–172, jan 2018.

- [87] Nadia Hammoudeh Garcia Harshavardhan Deshpande Gijs van der Hoorn Jon Tjergren Andrzej Wasowski Darko Bozhinoski, Mario Garzon Oviedo and Carlos Hernandez Corbato. Mros: runtime adaptation for robot control architectures. *Advanced Robotics*, 36(11):502–518, 2022.
- [88] ISO/IEC/IEEE. Systems and software engineering - architecture description. *ISO/IEC/IEEE 42010:2011(E)*, pages 1–46, 1 2011.
- [89] Omg, unified modeling language (uml) specification, version 2.5.1. <https://www.omg.org/spec/UML/>.
- [90] Tao Ma, Shaukat Ali, and Tao Yue. Modeling foundations for executable model-based testing of self-healing cyber-physical systems. *Softw. Syst. Model.*, 18(5):2843–2873, oct 2019.
- [91] Dubravko Miljković. Fault detection methods: A literature survey. pages 750–755, 2011.
- [92] Isa95, enterprise-control system integration part 1: Models and terminology, 2010. <https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95>.
- [93] Eeva Järvenpää, Otto Hylli, Niko Siltala, and Minna Lanz. Utilizing spin rules to infer the parameters for combined capabilities of aggregated manufacturing resources. *IFAC-PapersOnLine*, 51(11):84–89, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.
- [94] Quang-Duy Nguyen, Fadwa Rekik, Yining Huang, and Saadia Dhouib. Early lessons learned from the development of a local opc ua-based robotic testbed for research. In *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, pages 615–618, 2022.
- [95] R.G.J. Damgrave and E. Lutters. Smart industry testbed. *Procedia CIRP*, 84:387–392, 2019. 29th CIRP Design Conference 2019, 08-10 May 2019, Póvoa de Varzim, Portugal.
- [96] Tarik Eltaeib. Tcp/ip protocol layering. 3:415–417, 01 2015.
- [97] Quang-Duy Nguyen, Saadia Dhouib, Yining Huang, and Patrick Bellot. An approach to bridge ros 1 and ros 2 devices into an opc ua-based testbed for industry 4.0. In *2022 IEEE 1st Industrial Electronics Society Annual On-Line Conference (ONCON)*, pages 1–6, 2022.
- [98] Hyun Min Park and Jae Wook Jeon. OPC UA based Universal Edge Gateway for Legacy Equipment. In *2019 IEEE 17th International Conference on Industrial Informatics*, volume 1, pages 1002–1007, July 2019.
- [99] Yining Huang, Saadia Dhouib, and Jacques Malenfant. Aas capability-based operation and engineering of flexible production lines. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 01–04, 2021.
- [100] Yining Huang, Saadia Dhouib, and Jacques Malenfant. An aas modeling tool for capability-based engineering of flexible production lines. In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2021.

-
- [101] Yining Huang, Saadia Dhouib, Luis Palacios Medinacelli, and Jacques Malenfant. Enabling semantic interoperability of asset administration shells through an ontology-based modeling method. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS '22*, page 497–502, New York, NY, USA, 2022. Association for Computing Machinery.
- [102] Yining Huang, Saadia Dhouib, Luis Palacios Medinacelli, and Jacques Malenfant. Semantic interoperability of digital twins: Ontology-based capability checking in aas modeling framework. In *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–8, 2023.
- [103] Alan Harrison, Heather Skipworth, Remko I van Hoek, and James Aitken. *Logistics management and strategy*. Pearson UK, 2019.

Appendix A

Résumé en Français

A.1 Contexte Général

L'industrie future sera caractérisée par des systèmes de fabrication intelligents hautement autonomes et adaptatifs. Les systèmes de type "lot-size-one", ainsi que les concepts "plug-and-produce", impliquent la production d'une plus grande variété de produits d'une manière très flexible, et rendent la mise en service et la maintenance plus efficaces. Plus le lot de production est petit, plus le besoin de reconfiguration est fréquent, ce qui nécessite une réduction du délai entre les lots et l'automatisation du processus de reconfiguration de la ligne de production. Ces nouveaux systèmes de production doivent offrir des propriétés particulières pour leur adoption par les industriels. La rapidité avec laquelle les fabricants, en particulier les petites et moyennes entreprises (PME), peuvent reconfigurer la production en fonction d'une nouvelle série et ainsi répondre aux besoins des clients et éviter des temps d'arrêt coûteux des machines. De plus, l'intégration de ces PME dans des chaînes de production multi-entreprises optimisées, qui pousse encore davantage à l'automatisation. Ces caractéristiques sont essentielles pour maintenir le succès commercial et la rentabilité. Ces systèmes doivent présenter un haut degré d'autonomie pour gérer la reconfiguration des lignes de production et faire face à une grande variété de situations imprévues.

Le paradigme de l'industrie 4.0 (I4.0) vise à intégrer les technologies de l'information et de la communication dans des systèmes de fabrication traditionnels afin d'améliorer l'efficacité et la qualité des processus de production. Selon ce concept, l'autocontrôle des systèmes complexes est la base de la production industrielle moderne. Pour mettre en place un tel système auto-adaptable, la capacité d'orchestrer et de maintenir de manière autonome le système complexe est essentielle. La fabrication flexible et la robustesse de la gestion de la chaîne de production dans la vision de l'industrie 4.0 s'appuieront principalement sur la technologie du jumeau numérique [2].

Le paradigme du jumeau numérique [3] vise à fournir un retour d'information en temps réel entre le système physique et son équivalent numérique (partie cyber d'un système Cyber-Physique). L'ingénierie dirigée par les modèles (IDM)[4] est une approche qui a vu le jour dans le domaine du développement de logiciels qui promeut l'utilisation de modèles pour la conception et le développement de systèmes. Dans le contexte des jumeaux numériques industriels, les technologies IDM peuvent être utilisées pour créer et préserver le modèle de jumeau numérique tout au long du cycle de vie du système. Elles fournissent une base pour la construction de modèles de jumeaux numériques qui simplifient la conception et le développement de systèmes complexes. En utilisant des méthodes d'IDM précises, les fabricants peuvent développer des systèmes ouverts, extensibles et obéissants aux meilleurs principes du génie logiciel capables de contrôler, simuler et optimiser le processus de production. En outre, le concept de `models@run.time` [5] dans l'IDM vise à exploiter les avantages des modèles, tels que l'abstraction et la modularité,

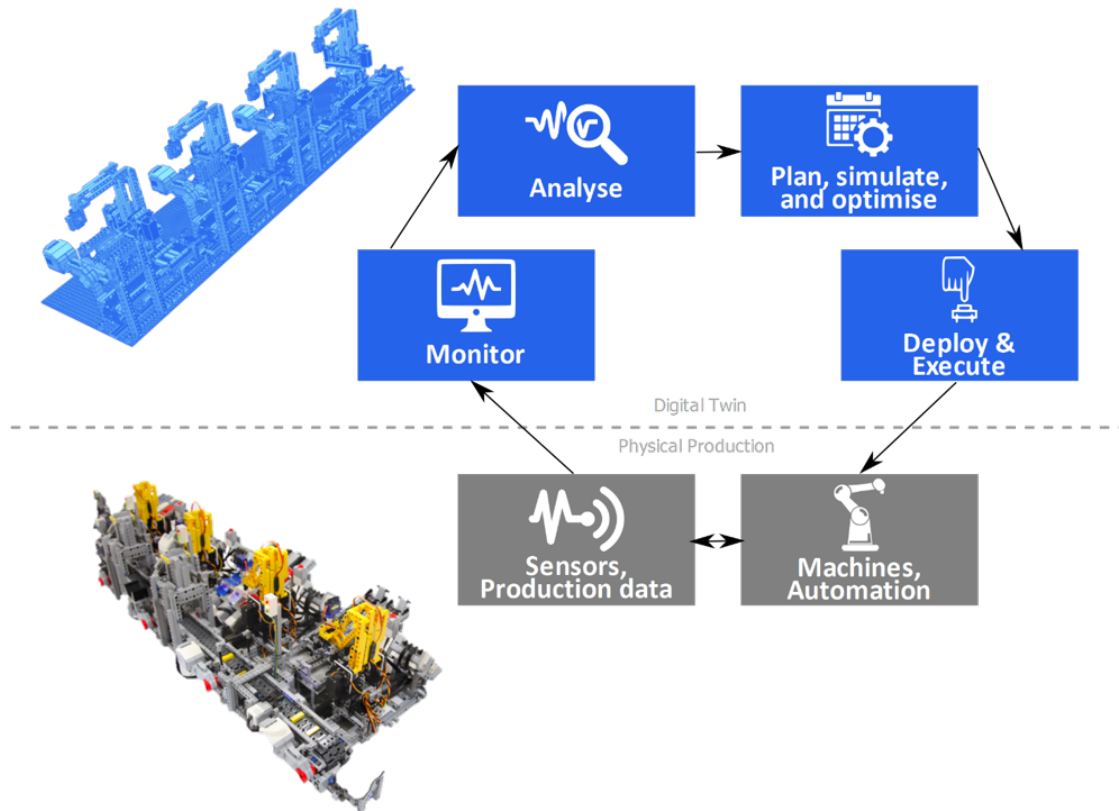


Figure A.1: Concept de jumeau numérique pour une ligne de production

tout au long du cycle de vie d'un système. L'idée de modèles exécutables répond à la définition du jumeau numérique lorsque les modèles restent en connexion active avec son jumeau physique pendant toute sa durée de vie. L'abstraction permet aux concepteurs et aux développeurs de se concentrer sur des concepts de haut niveau qui facilitent la compréhension humaine. La modularité garantit la flexibilité et l'évolutivité de la gestion du système. Les *models@run.time* reflètent l'état actuel du système, ce qui fournit un cadre idéal pour l'auto-adaptation. La boucle d'auto-adaptation se compose de quatre phases : Surveillance, Analyse, Planification et Exécution. Cette boucle de rétroaction est essentielle, car elle permet une adaptation dynamique aux changements et garantit la robustesse du système.

A.2 Problématiques de Recherche

Cependant, plusieurs défis et questions de recherche essentiels sont apparus dans le domaine des jumeaux numériques et de leurs applications à l'industrie. Outre les défis bien connus liés à la supervision de systèmes cyber-physiques complexes, la nécessité de fournir une représentation complète des produits, des processus de production et des ressources de l'usine d'une manière interopérable entre les équipements hétérogènes représente des défis significatifs. Trois problèmes centraux étudiés dans cette thèse sont *interopérabilité*, *adaptabilité* et *robustesse* des systèmes de fabrication à l'ère de l'industrie 4.0.

A.2.1 Interopérabilité

L'interopérabilité [6] fait référence à la capacité de différents systèmes et organisations à travailler ensemble de manière transparente. Elle peut être subdivisée en interopérabilité syntaxique et interopérabilité sémantique. L'interopérabilité syntaxique indique que les systèmes ont la capacité de communiquer et d'échanger des données, ce qui inclut les formats de données spécifiés, les protocoles de communication, les descriptions d'interface, etc. Les normes fournissent des règles, des guides, des templates, etc., pour la cohérence et la conformité d'un domaine spécifique, ce qui régit la construction du système d'une manière cohérente et transparente. Cependant, pour les domaines émergents tels que l'industrie 4.0 ou les jumeaux numériques, il n'est pas facile de trouver des normes appropriées et de les mettre en œuvre. Le défi consiste à identifier et à tirer parti des normes existantes pour spécifier l'interface unifiée de tous les participants à un système de jumeaux numériques de l'industrie 4.0, de sorte qu'ils ne soient plus dépendants des fournisseurs et de la technologie.

RQ1 Comment pouvons-nous identifier et exploiter les normes et technologies existantes que nous pouvons utiliser pour concevoir un modèle de jumeau numérique pour les plans de production et les ressources de l'usine d'une manière à la fois interprétable par les machines et qui facilite les échanges entre les différentes parties prenantes ?

L'interopérabilité sémantique quant à elle ne se limite pas à un échange efficace de données entre systèmes. Elle comprend également le fait que les données soient comprises et traitées dans leur signification propre et communément admise par toutes les parties. Bien que l'utilisation de normes assure l'interopérabilité syntaxique des actifs entre fournisseurs, le problème de l'interopérabilité sémantique demeure. De nombreux groupes et unités de recherche ont pris conscience de ce problème et ont commencé à étudier ce sujet. Cependant, la question de l'interopérabilité sémantique demeure spécifique à un domaine, et les expressions sémantiques dans différents domaines peuvent être très différentes. Par conséquent, en fonction des limites de notre projet, nous formulons principalement les deux questions suivantes.

RQ2 Comment pouvons-nous définir sémantiquement les modèles afin de déterminer la sélection la plus appropriée des ressources disponibles dans l'usine qui répondent aux exigences du procédé de production ?

RQ3 Comment traiter sémantiquement les données de surveillance obtenues à partir du jumeau numérique pendant l'exécution pour diagnostiquer de manière correcte et précise les risques dans le système de fabrication ?

A.2.2 Adaptabilité

L'adaptabilité du système fait référence à sa capacité à réagir de manière pertinente et dans les délais aux changements. Un système de fabrication intègre l'ensemble des ressources existantes, des produits souhaités et des processus de production. Par conséquent, un système de fabrication adaptatif doit être en mesure de répondre aux changements dans toutes ses composantes. Lorsque les demandes du marché évoluent ou que les objectifs internes changent, les lignes de production peuvent nécessiter des ajustements. Par conséquent, un aspect essentiel de l'adaptabilité consiste à réaliser automatiquement la recomposition et la replanification de la ligne de production lorsque le flux de travail change. Un autre aspect important est la capacité de configuration/reconfiguration rapide. Le système doit être conçu pour être facilement reconfigurable sans temps d'arrêt important. Ces défis soulèvent les questions de recherche suivantes :

RQ4 Quelles méthodologies peuvent être développées pour faciliter la recombinaison et la re planification dynamique des lignes de production de manière flexible et automatisée ?

RQ5 Quelle stratégie peut-on employer pour garantir une reconfiguration rapide et efficace du système en réponse à des besoins opérationnels variés ?

A.2.3 Robustesse

En outre, la robustesse est une caractéristique clé d'un système de fabrication; elle fait référence à la réactivité du système, à sa tolérance aux pannes et à sa capacité d'auto-adaptation. La différence entre un système adaptatif et un système auto-adaptatif tient au fait que le système comprend ou non une boucle de rétroaction (surveillance, analyse, planification et exécution). Ce dernier est capable d'ajuster son propre comportement en cours d'exécution afin d'atteindre les objectifs du système. La surveillance et le diagnostic des modèles de jumeaux numériques dynamiques sont essentiels à la réalisation des systèmes auto-adaptatifs. La surveillance met à jour le jumeau numérique avec des données du monde réel. Le module de diagnostic examine les données et utilise les informations transmises par le jumeau numérique pour prendre des décisions. Le jumeau numérique doit ensuite interagir avec le monde réel et prendre des décisions. Ces caractéristiques peuvent permettre aux systèmes de fabrication non seulement de continuer à fonctionner de manière optimale, mais aussi de fournir des plans de reprise en cas d'anomalies. Par conséquent, le processus d'auto-adaptation du système peut théoriquement être réalisé en s'appuyant sur les technologies de jumeaux numériques. Toutefois, la mise en œuvre pratique de la combinaison de ces théories de manière appropriée et peu coûteuse reste un défi à relever.

RQ6 Comment intégrer efficacement les modules de surveillance et de diagnostic en temps réel dans les systèmes existants afin de garantir un fonctionnement continu pendant les phases d'exécution ?

A.3 Contributions Principales

L'objectif principal de cette thèse de doctorat est de trouver des solutions aux problèmes susmentionnés. Elle vise à mettre en œuvre un prototype de méthode de gestion de production flexible basée sur la technologie du jumeau numérique dans un environnement IDM et à valider les théories sur un démonstrateur académique. La figure A.2 illustre la problématique analysée dans la section précédente, les approches adoptées pour façonner la méthodologie et les solutions incorporées dans la mise en œuvre.

- La principale contribution est de créer un environnement de modélisation conçu pour le développement de jumeaux numériques industriels. Cet environnement aborde la question de l'interopérabilité syntaxique en utilisant des méthodologies IDM. L'ensemble d'outils Papyrus4Manufacturing (P4M) est développé pour fournir une approche IDM à cet environnement de modélisation. Cet ensemble d'outils intègre des éditeurs conviviaux pour normaliser la création de modèles de jumeaux numériques des ressources et processus de la production. Le P4M fournit une fonctionnalité de déploiement automatique basée sur des transformations de modèles et la génération de codes. Le développement d'un ensemble d'outils spécifiques adaptés à ce langage de spécification est un aspect clé de cette contribution.

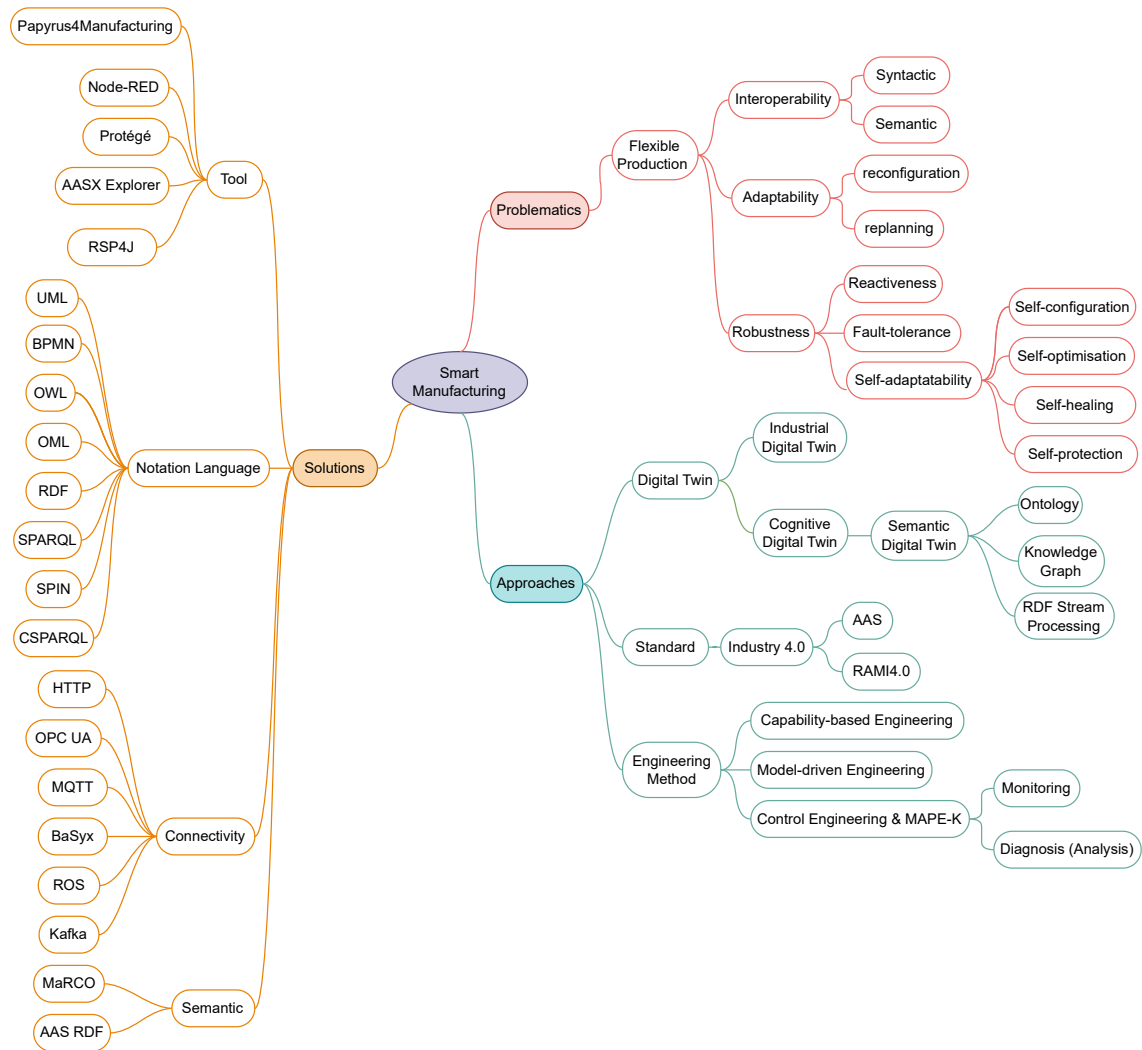


Figure A.2: Carte conceptuelle pour les concepts apparaissant dans la thèse

- La modélisation des jumeaux numériques industriels de manière standardisée. L’Asset Administration Shell (AAS) [10] est un candidat solide pour la normalisation des modèles de jumeaux numériques industriels, qui fournit une interface commune pour décrire tous les types de participants à la production.
- La modélisation des processus de production nécessite un langage normalisé pour la conception des processus. Le Business Process Model and Notation (BPMN) est un langage de modélisation graphique qui décrit les processus d’entreprise. La contribution comprend l’extension et l’intégration du plugin de modélisation BPMN dans l’environnement de modélisation du jumeau numérique.
- La norme AAS fournit une interface d’interopérabilité syntaxique pour tous les actifs impliqués dans les usines intelligentes. Cependant, il est encore nécessaire de combler les exigences concernant l’interopérabilité sémantique, afin de permettre une compréhension efficace entre les composants de l’industrie 4.0. Les ontologies définissent des modèles sémantiques de données combinés à des connaissances du domaine pertinentes et formulent des stratégies d’inférence. Il s’agit donc d’une approche très pertinente pour apporter une interopérabilité sémantique. Notre deuxième contribution consiste à proposer une méthode de modélisation AAS basée

sur l'ontologie qui permet l'interopérabilité sémantique des modèles de jumeaux numériques AAS. Cette approche franchit les barrières entre la représentation ontologique (spécifiquement OWL) et les modèles AAS dans P4M (modèles UML). Cette contribution porte sur l'annotation de la signification sémantique extraite des concepts de l'ontologie aux modèles AAS. Cette mise en œuvre contient principalement trois avantages.

- MaRCO (Manufacturing Resource Capability Ontology) [12] pour fournir des descriptions sémantiques des capacités de fabrication
 - Les transformations permettant de conserver l'alignement entre l'ontologie et des modèles
 - Compléter les outils d'ingénierie dirigé par les modèles avec un raisonnement automatisé.
- L'approche de l'ingénierie basée sur les capacités (IBC) [13] aborde les défis d'adaptabilité de la gestion des lignes de produits flexibles de l'industrie 4.0. L'IBC propose de transformer automatiquement une série de flux de production abstraits (les lots de production) présentant leurs capacités requises en plans de production qui sélectionnent, configurent et exploitent les ressources offrant des capacités correspondantes. Cette méthode d'ingénierie répond aux défis de la recherche sur l'adaptabilité mentionnés dans la section précédente. La limitation des algorithmes actuels de mise en correspondance des ressources uniquement syntaxiques a été surmontée en mettant en œuvre l'interopérabilité sémantique basée sur les ontologies *i.e.*, en transformant les modèles d'usine basés sur l'AAS en instances MaRCO. La principale contribution de cette partie est
 - d'affiner l'architecture de l'IBC de la conception du modèle à la phase d'exécution,
 - de mettre en œuvre la fonctionnalité de vérification automatique des capacités dans P4M, et
 - de permettre l'exécution du processus à distance.
 - Pour améliorer la robustesse, la boucle MAPE-K propose une méthode précise pour structurer et organiser de manière souple et évolutive un système auto-adaptatif. La mise en œuvre de cette boucle de rétroaction intègre des techniques d'informatique sémantique pendant l'exécution du modèle. L'informatique sémantique va au-delà des méthodes informatiques traditionnelles en donnant aux données une signification plus profonde et contextuelle, ce qui permet au système de prendre des décisions plus pertinentes et intelligentes. Plus précisément, nous utilisons des annotations sémantiques en cours d'exécution pour associer dynamiquement les données brutes aux concepts de l'ontologie. En outre, les données annotées servent d'entrée pour le traitement des flux RDF, ce qui nous permet d'interroger et de manipuler en permanence les flux de données sémantiques. Cette combinaison d'annotations sémantiques et de traitement de flux de données annotés sémantiquement (RDF) améliore la connaissance qu'a le système de ses performances en temps réel.
 - Afin de tirer parti des contributions précédentes, une architecture de système auto-adaptatif basé sur les capacités et piloté par un modèle (CBSAM) a été proposée. Cette architecture intègre la boucle de rétroaction MAPE-K à l'architecture IBC mentionnée précédemment. Elle fournit une méthodologie pour formuler un tel système de développement de jumeaux numériques basé sur l'AAS, de la phase de spécification à la phase d'exploitation et de maintenance.

- Un démonstrateur académique est construit pour présenter et valider les approches proposées pour les problèmes de recherche. Mes contributions au développement de ce démonstrateur consistent en
 - l'identification d'un scénario de processus de production précis pour le démonstrateur,
 - la conception de modèles de jumeaux numériques fonctionnels pour les composants et le processus,
 - le déploiement automatique et l'orchestration de modèles de jumeaux numériques fonctionnels pour exécuter les dispositifs en situation réelle.

A.4 L'architecture CBSAM

Cette section présente une approche générique pour la construction d'un tel système de production flexible et auto-adaptable. Les problèmes susmentionnés peuvent être résolus grâce à cette architecture qui intègre les différentes approches et méthodes avancées qui ont été analysées dans la section précédente. Cette architecture suit les phases d'ingénierie présentées par G.Urgese et al. [14], avec l'intention de couvrir le système de production depuis le début de la conception jusqu'au processus d'exploitation de la production. Les figures A.3 et A.4 illustrent la vue d'ensemble de l'architecture de cette approche de fabrication auto-adaptative basée sur les capacités (CBSAM).

- **Phase de spécification :**

Cette phase permet de collecter les informations relatives à la spécification du modèle. Afin d'assurer le bon déroulement des phases suivantes, certains éléments indispensables doivent être identifiés dès cette première phase. Les trois principaux aspects de la réalisation du CBSAM sont les informations relatives aux capacités, à l'exploitation et au suivi du modèle. La spécification de la capacité de chaque ressource doit comprendre la collecte et l'analyse des fiches de caractéristiques techniques de la ressource. La spécification de la capacité du processus consiste à identifier le scénario de travail. La représentation sémantique appropriée de la capacité de fabrication doit également être sélectionnée pour la spécification de la capacité. Les informations opérationnelles concernent les informations relatives à la connexion des actifs ; par exemple, le modèle d'information du réseau d'interconnexion (OPC UA) et les configurations de serveur peuvent constituer une bonne base. Les spécifications relatives à la surveillance et au diagnostic nécessitent l'établissement d'une relation entre les exigences sémantiques liées à la surveillance et les données dynamiques que le modèle de jumeau numérique peut surveiller. Il est essentiel de spécifier les événements potentiels qui vont déclencher des réactions, comme le dysfonctionnement d'équipements, pour le système de production avec l'aide des experts du domaine.

- **Phase de conception :**

Tous les modèles AAS (ressources, processus et produits) doivent être conçus au cours de cette phase en fonction des informations de spécification recueillies au cours de la phase précédente. Un modèle AAS est composé de sous-modèles, et les sous-modèles sont composés d'éléments de sous-modèles. Différents sous-modèles sont utilisés pour décrire différents aspects. Un "sous-modèle de capacité" décrit les capacités de fabrication au niveau abstrait en annotant l'élément de sous-modèle *Capacité* avec la sémantique de l'ontologie. Le "sous-modèle OperationalData" détaille les propriétés opérationnelles et les opérations exécutables. Il s'agit du

sous-modèle qui permet l'échange des données avec les biens physiques, y compris la lecture/écriture des valeurs dynamiques des propriétés et l'invocation des opérations disponibles. Le "Monitoring.Submodel" peut être conçu pour différents actifs, y compris différentes ressources de production et processus de production. Conformément à l'introduction précédente, nous devons concevoir les éléments à surveiller pour chaque bien à ce stade, les événements qui peuvent être déclenchés et les conditions de déclenchement des événements (les règles de diagnostic).

- **Phase d'ingénierie et de déploiement :**

Une base de connaissances ontologique globale peut être générée par une transformation du modèle en connaissances lors de la phase d'ingénierie. Cette base de connaissances contient des informations sur les modèles AAS spécifiés et conçus au cours des phases précédentes. Entre-temps, chaque modèle AAS peut être généré vers un serveur AAS pour le mode exécutable après la conception du modèle. Ces serveurs AAS générés permettent la connexion aux équipements et la transparence et l'interopérabilité de l'échange de données pendant l'exécution des équipements de production. Ensuite, un processus initial d'IBC doit être réalisé pour la première configuration du système, où le plan de travail de production peut être trouvé et validé. Le module de vérification des capacités utilise la base de connaissances globale initiale, qui automatise la sélection du plan de production à partir de la réserve de ressources. Les résultats des connaissances déduites obtenus lors de l'étape de vérification des capacités sont conservés en vue d'une maintenance ultérieure. Le module de vérification de la faisabilité ne se contente pas de valider les ressources candidates déduites par le module de vérification de la capacité, mais trouve également les paramètres appropriés pour la configuration du système. Les paramètres des ressources sont reconfigurés en fonction du plan validé. Lors de l'étape d'exécution des compétences, les paramètres précédemment déterminés sont configurés sur l'appareil. Un orchestrateur de processus de production interagit avec le modèle de ressources AAS en cours d'exécution conformément au plan de production validé. Cet orchestrateur invoque les opérations conformément à la conception du processus.

- **Phase d'opérations et de maintenance :**

Pendant la production, l'ensemble du processus peut être surveillé en collectant des données à partir de l'équipement sur la ligne de production. Des connaissances locales extraites en continu peuvent être extraites en transformant les données en temps réel en graphes de connaissances. Un diagnostic des défauts basé sur la connaissance est nécessaire dans cette phase pour analyser le flux de données en temps réel provenant de la ligne de production. Les défauts peuvent être détectés par des règles prédéfinies dans la base de connaissances. Selon les conseils de réparation recueillis lors de la phase d'analyse, une replanification du système est nécessaire pour garantir une gestion rapide et des méthodes d'ajustement dans les situations anormales afin de parvenir à un processus de production stable et continu. En fonction de l'avis de réparation, un actionneur local sera déclenché pour réinitialiser une opération ou l'actionneur central de récupération sera activé si une réaction plus importante est requise. Lorsque le module de planification décide de remplacer la ressource de la chaîne de production, le module d'exécution met à jour les connaissances locales dans la base de connaissances globale et invoque l'actionneur de récupération central. L'actionneur central de récupération lance un nouveau processus d'IBC et le nouvel appariement des capacités s'appuie sur la base de connaissances globale mise à jour.

Grâce à cette architecture, il est possible de construire un système de jumeau numérique

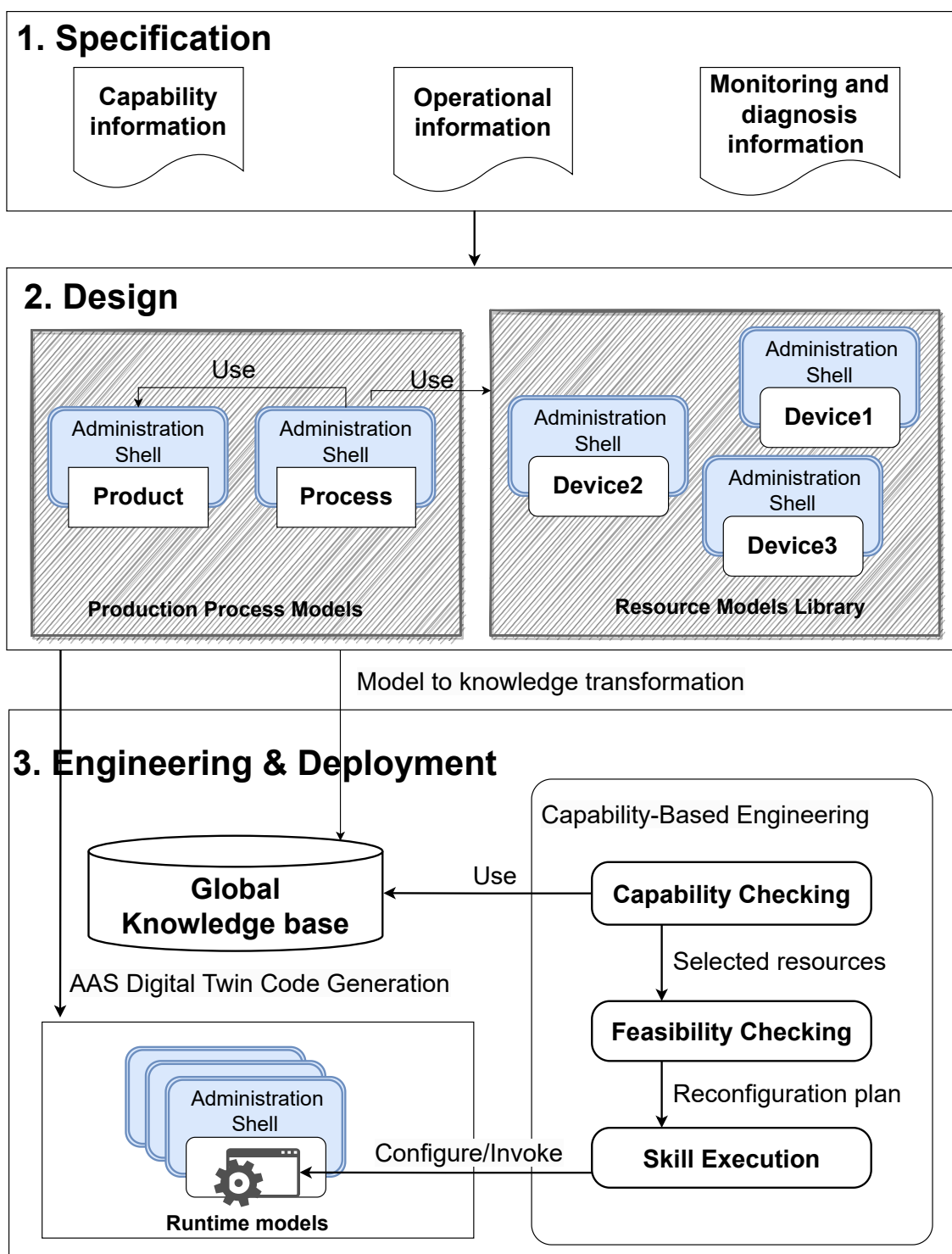


Figure A.3: Vue d'ensemble de l'architecture CBSAM - Partie 1

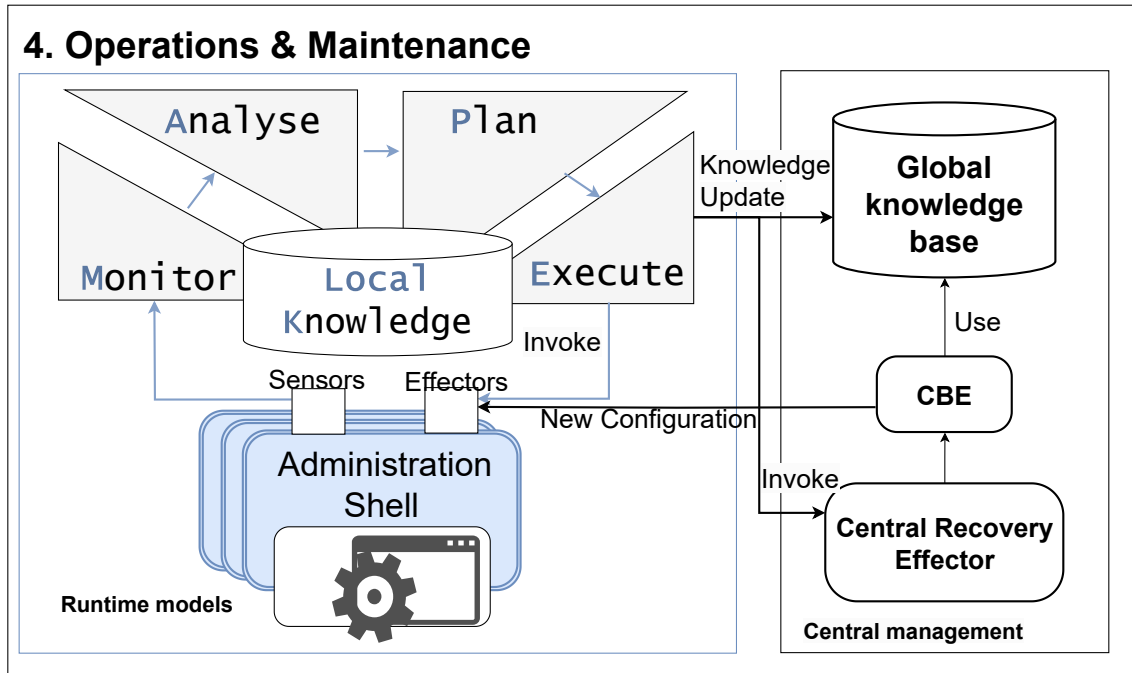


Figure A.4: Vue d'ensemble de l'architecture CBSAM - Partie 2

d'usine à partir de la base qui prend en charge l'interopérabilité syntaxique et sémantique. L'utilisation des technologies IDM facilite également le déploiement rapide de modèles exécutables grâce à l'automatisation. L'IBC permet la replanification et la reconfiguration rapides des lignes de production. Et la boucle MAPE-K permet au système de fabrication de prendre des décisions optimales et de répondre aux changements en fonction du comportement en cours d'exécution.

A.5 Contenu du manuscrit

L'introduction sur le contexte général et le problème de recherche a été mentionnée au début de ce chapitre. Le reste de cette thèse est organisé comme suit.

Comme indiqué dans la section précédente, nous avons intégré et mis en œuvre de nombreux concepts et technologies dans cette thèse. Le chapitre 2 compare les différentes technologies existantes et les solutions alternatives, y compris la mise en œuvre du jumeau numérique industriel suivant les technologies sur l'IDM, l'ingénierie basée sur les capacités et les compétences, l'ontologie et le métamodèle pour les jumeaux numériques sémantiques, et les systèmes cyber-physiques auto-adaptatifs. Ce chapitre étudie l'état de l'art des sujets susmentionnés et justifie les choix de la sélection de la méthodologie et de la technologie pour la mise en œuvre.

Le chapitre 3 présente une architecture d'extension de la méthode d'ingénierie basée sur les capacités avec une boucle de rétroaction fermée afin de construire un système de fabrication interopérable et auto-adaptable. L'architecture suit les phases d'ingénierie formelle [14] de la spécification à la maintenance. L'architecture proposée fournit une base théorique pour les questions de recherche précédentes.

La mise en œuvre est le processus qui consiste à mettre en pratique les méthodes explorées dans la section précédente. La phase de mise en œuvre implique l'application des concepts, approches et techniques examinés dans les chapitres précédents. Dans le chapitre 4, les détails de la mise en œuvre de l'approche CBSAM présentée au chapitre 3 peuvent être pleinement élaborés. Afin d'être cohérent, il est tout d'abord nécessaire d'examiner

brièvement la structure de l'architecture CBSAM ainsi que les objectifs généraux de la mise en œuvre. Ce chapitre met en lumière les contributions à la partie du sujet consacrée à l'implémentation.

Par essence, le système de jumeau numérique est une construction cyber-physique qui englobe des entités du monde réel et des entités numériques. Ainsi, au-delà de l'architecture conceptuelle, la capacité à la mettre en œuvre et à la valider dans la réalité devient une étape critique. Pour fournir une plateforme de validation des théories de recherche, Le chapitre 5 présente un cas d'usage. L'un des principaux objectifs est d'établir une base expérimentale robuste et flexible qui puisse s'adapter aux différents besoins de la recherche et aux trajectoires de l'innovation. La couche numérique, c'est-à-dire la partie d'intégration AAS, est une partie importante du cas d'usage pour compléter le développement des jumeaux numériques. En outre, l'utilisation d'un cas d'usage de l'industrie 4.0 favorise une compréhension approfondie des défis pratiques et des opportunités qui peuvent être rencontrés lors de la transition du concept à une application pratique.

Le chapitre 6 résume l'ensemble du manuscrit et discute des orientations futures de la recherche.

A.6 Conclusion et perspectives

En conclusion, cette thèse met l'accent sur la transformation à venir des systèmes de production sous l'impulsion de l'industrie 4.0. L'émergence de systèmes de production intelligents et adaptatifs représente une avancée significative caractérisée par la capacité à gérer une production diversifiée et flexible. Les défis liés à l'interopérabilité, à l'adaptabilité et à la robustesse des systèmes de fabrication sont au cœur de cette transformation. L'approche des jumeaux numériques offre la possibilité de rapprocher le monde physique du monde numérique, ce qui permet une meilleure surveillance, une planification plus précise et une exécution efficace des processus de production. La mise en œuvre de jumeaux numériques combinée aux principes de l'ingénierie dirigée par les modèles est une solution prometteuse pour relever ces défis. L'architecture proposée (CBSAM) s'appuie sur les méthodologies du génie logiciel, telles que l'ingénierie basée sur les capacités, l'IDM et l'approche MAPE-K, pour améliorer la réactivité et la flexibilité des systèmes de production.

Les travaux futurs peuvent être divisés en deux catégories :

- **Consolidation de l'implémentation** : L'une des principales perspectives est compléter le développement de l'architecture proposée dans ce travail de thèse. Il s'agit d'améliorer l'intégration des différents composants (tel que le module d'exécution dans MAPE-K) et processus afin de couvrir tous les modules architecturaux de l'approche MAPE-K.
- **Généralisation et abstraction** : Un autre aspect de la recherche future est l'abstraction et la généralisation de l'architecture. L'objectif est d'étendre l'applicabilité de cette méthode au-delà des exigences spécifiques du projet, en créant une approche universellement applicable. Cela implique d'affiner les techniques pour qu'elles soient adaptables et pertinentes dans différents domaines, permettant ainsi une plus large gamme d'applications pour les jumeaux numériques industriels en général.

Appendix B

Capability Checking Module

This appendix provides a detailed installation guide for the capability checking module plugin within the Papyrus4Manufacturing framework¹. It includes step-by-step instructions on how to successfully integrate the module into the existing framework.

B.1 Download extensions

1. Open P4M -> Help -> Install New Software. (Figure B.1)

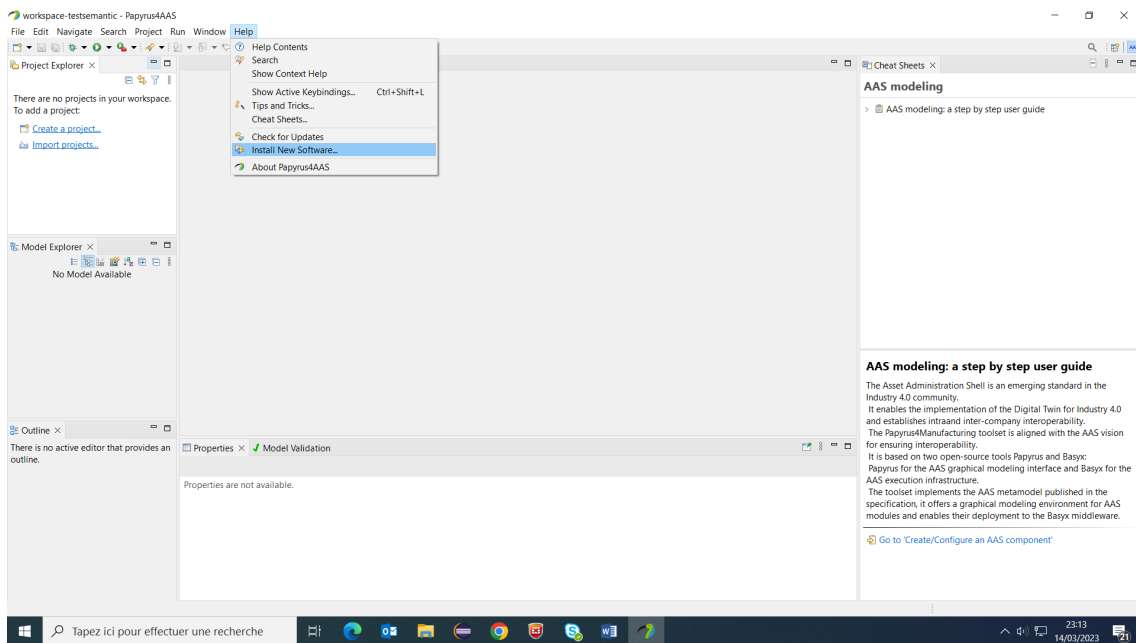


Figure B.1: Installation Window

2. Add.. -> Add a name to this software location -> Archive.
3. Select the eclipse-update-site-capability-checking.zip.
4. Select Papyrus4Manufacturing semantic feature and continue the installation until finish. (Figure B.2)

¹<https://eclipse.dev/papyrus/components/manufacturing/downloadaas.html>

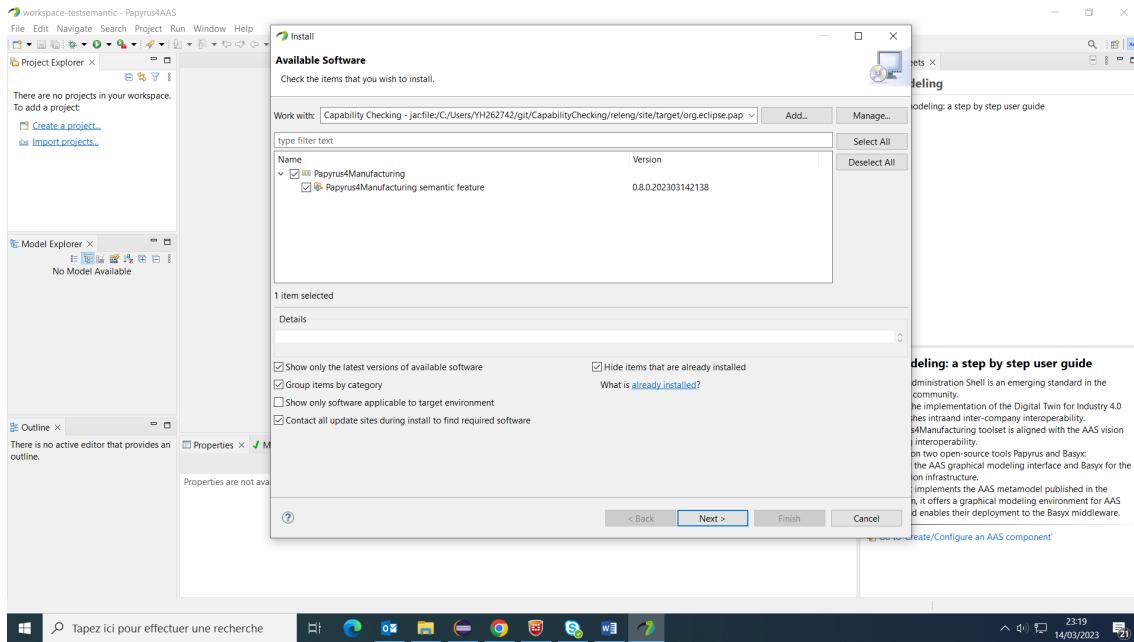


Figure B.2: Select Semantic Feature

5. Trust unsigned contents. (Figure B.3)

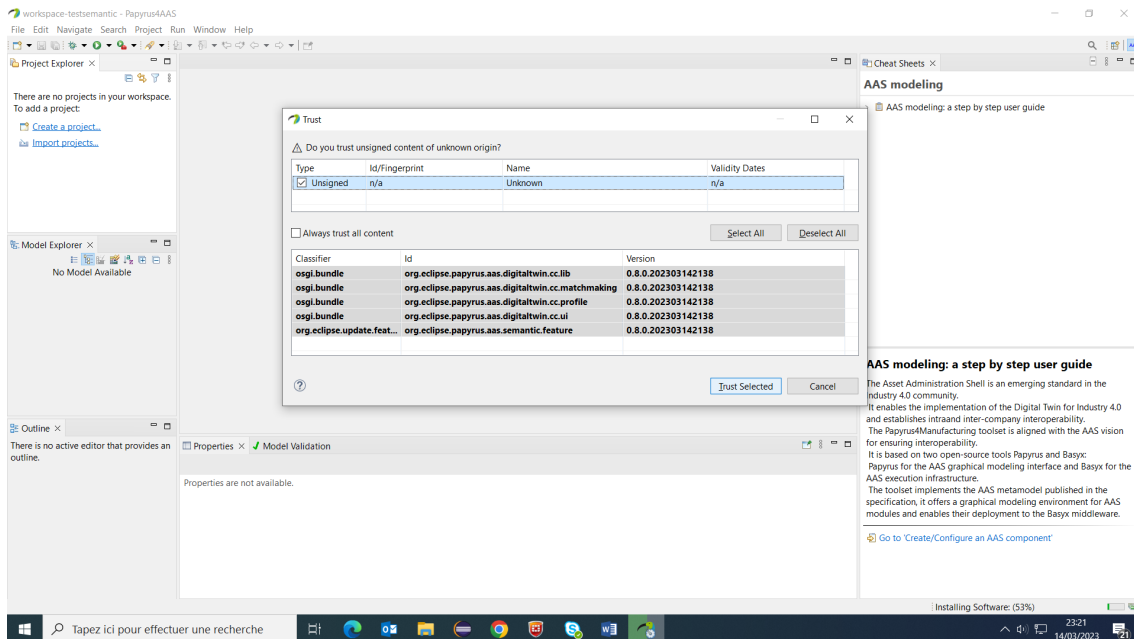


Figure B.3: Trust Unsigned Contents

6. Restart Application

B.2 Example

1. Create/import an AAS project. You can import the provided LocalSEA-AAS-Model.
2. Add MaRCO profile to root model. (Figure B.4)

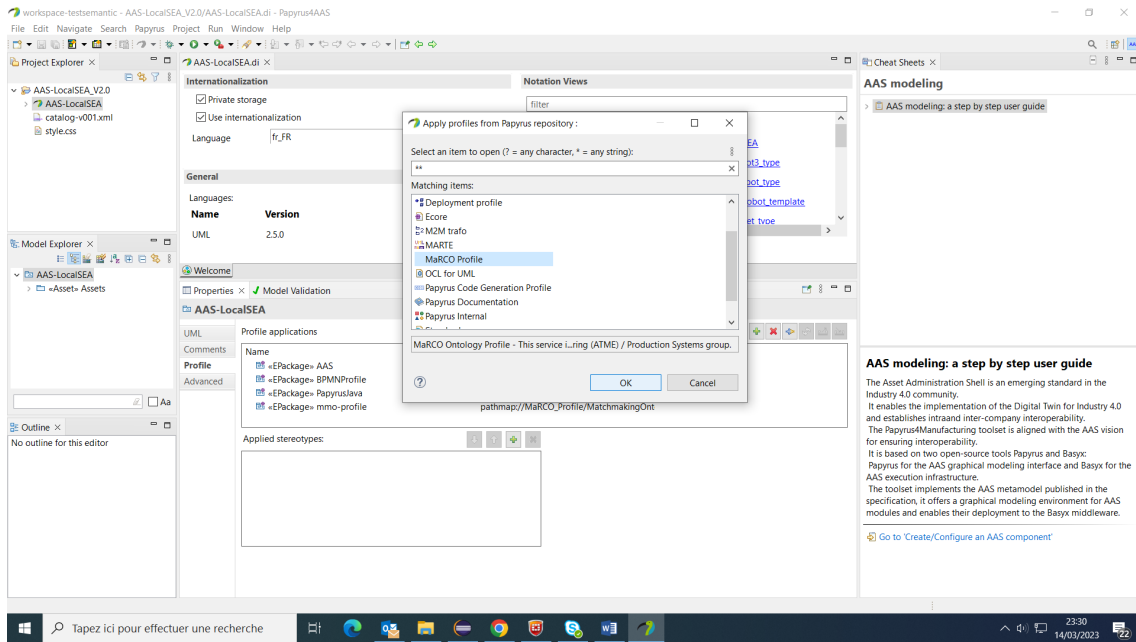


Figure B.4: Add MaRCO Profile

3. Annotate AAS models with MaRCO semantics and specify the attribute values. (Figure B.5)

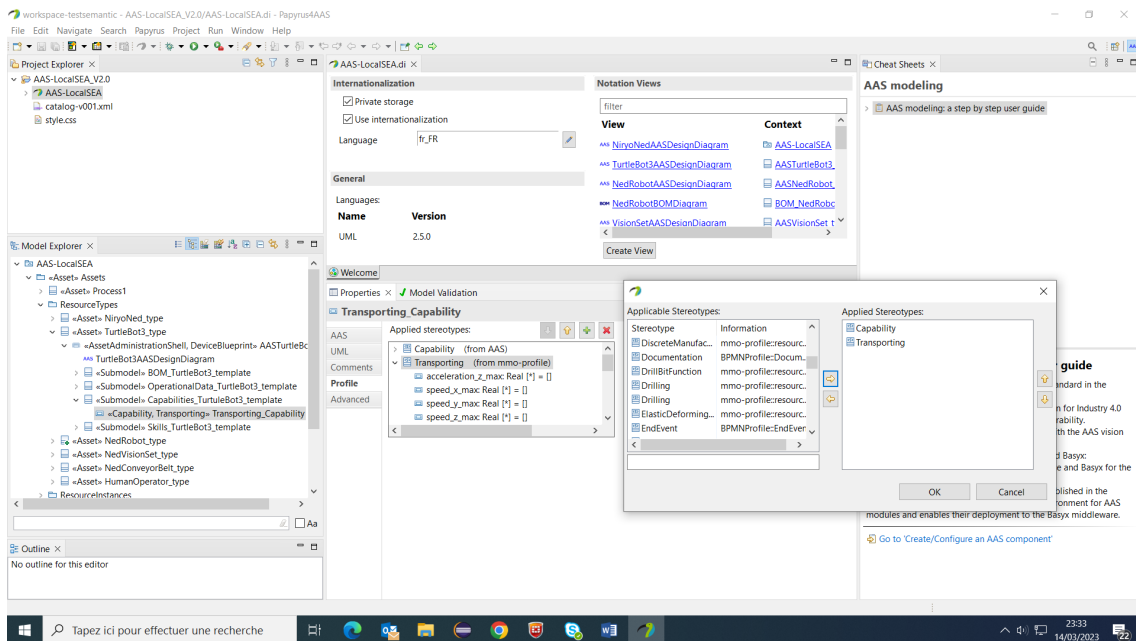


Figure B.5: Annotate AAS Models

4. Right click on root model -> Capability Checking This command permits the transformation the AAS models to MaRCO individuals in an OWL file named AASindividuals.rdf, which can be found in the same project location with the AAS model. (Figure B.6)

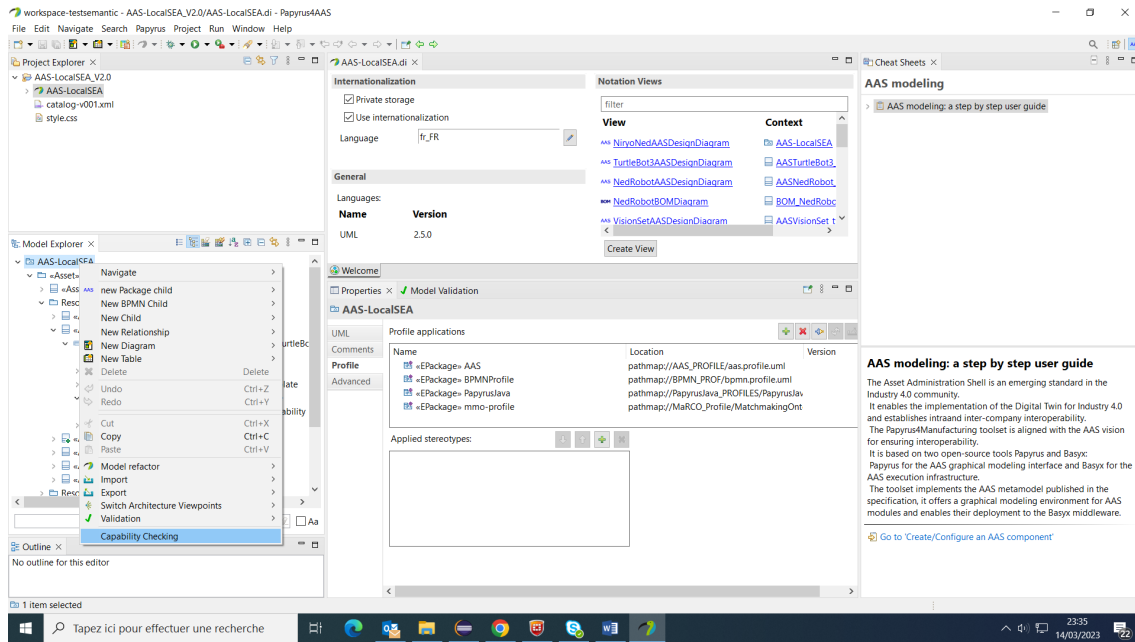


Figure B.6: Capability Checking Command

- Then a pop-up window will show up, and the user needs to select the products and processes that require matchmaking. (Figure B.7)

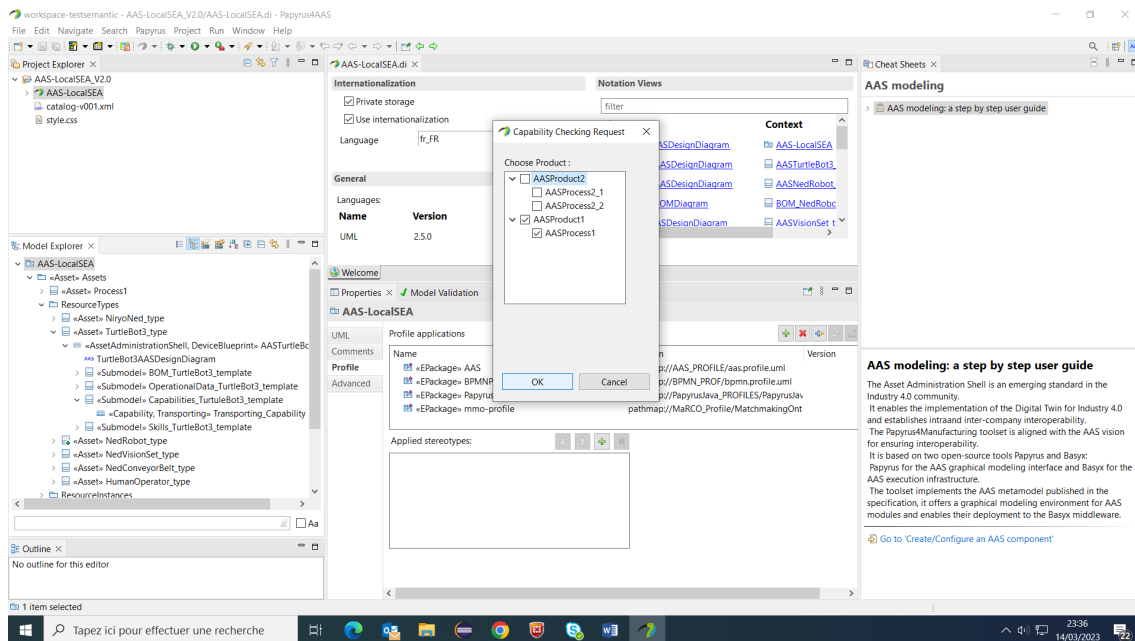


Figure B.7: Matchmaking Requirement

- The matchmaking process requires a bit of time, after the calculation a result window should pop-up. (Figure B.8)

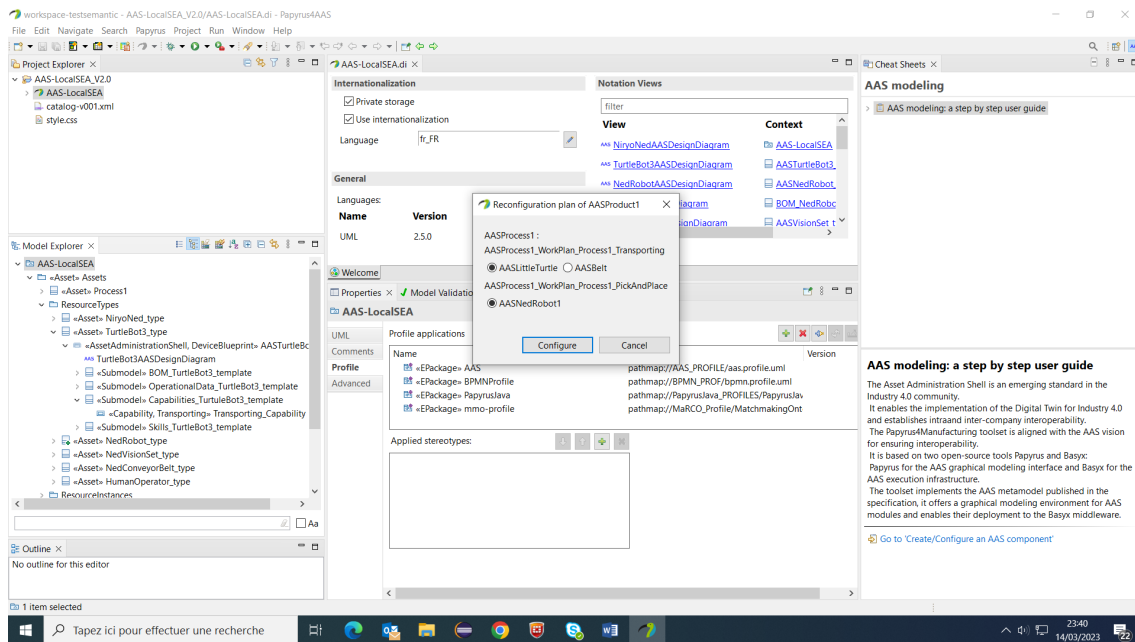


Figure B.8: Result Window

- The generated ontologies can be found in the same location with the AAS model after refreshing the Project Explorer. (Figure B.9)

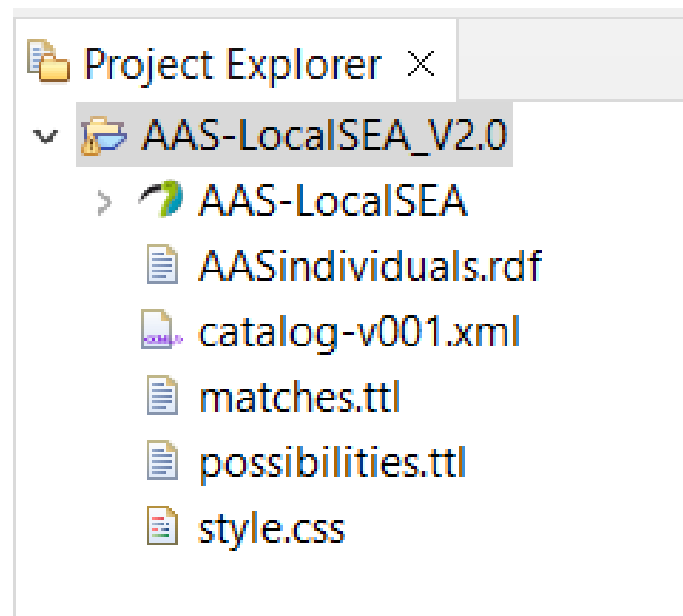


Figure B.9: Project Explorer

Appendix C

AAS BaSyx Node-RED Package

A collection of customized Node-RED nodes designed to facilitate communication and service integration using the AAS BaSyx framework. These nodes enable seamless interaction between Node-RED and BaSyx. The nodes provide user-friendly interfaces for connecting, managing, and exchanging data with AAS environments, simplifying the process of building complex industrial digital twin orchestration applications.

C.1 Prerequisite

To install these nodes, it's essential to have Node-RED installed and running correctly on your system. You can refer to the Node-RED Getting Started guide¹ for detailed instructions on setting up Node-RED locally. This guide provides comprehensive steps for installation, including system requirements, download procedures, and initial configuration settings. Once Node-RED is successfully installed and operational, you can proceed with the integration of the AAS BaSyx nodes into your Node-RED environment.

C.2 Install Steps

1. Run command under `./node-red`
`npm install path/to/your/project/node-red-aas-bpmn`
2. Start Node-RED

C.3 Workflow for List Days

Two flows has been created for the List Days Demonstration. They are also deployed on nuc8's `nodered.service`, so the URL is `http://192.168.56.8:1880/`. – CEA-List-Day View: provides the UI interface – CEA-List-Day Prepare: provides the implementation of orchestration The two flows can be found in the Tuleap repo.

C.3.1 Flow Deployment

1. Make sure you have Node-RED installed on your machine.
2. Install `node-red-aas-bpmn` from the `tuleap` repo.
3. Start Node-RED
4. Import the flows to Node-RED

¹<https://nodered.org/docs/getting-started/local>

