Carl von Ossietzky Universität Oldenburg

Faculty II - Computer Science, Business and Law
Department of Computer Science

An integrated environment for modeling and deploying Digital Twins

From the faculty of Computer Science, Business and Law at the Carl von Ossietzky University Oldenburg to obtain the degree and title of

Doctor of Engineering (Dr.-Ing.)

Accepted dissertation from

Msc. Charles Steinmetz

born on 02.05.1991 in Cerro Largo, Rio Grande do Sul (Brazil)

Evaluators:	Prof. Dr. Martin Fränzle
	Prof. Dr. Carlos Eduardo Pereira
	Prof. Dr. Achim Rettberg

Day of Disputation: 25.09.2024

Acknowledgments

The journey from a small farm in a small city in Brazil to completing my PhD in Computer Science in Germany has been filled with challenges and invaluable opportunities for learning and growth. At every stage of this journey, I have been fortunate to receive immense support, without which this achievement would not have been possible. I would like to express my deepest gratitude to those who have guided, encouraged, and helped me along the way.

First, I would like to thank my supervisor, Prof. Achim Rettberg. Thank you for giving me the opportunity to pursue this PhD, for believing in my work and abilities, and for constantly encouraging me to persevere. Your kindness and humanity create an environment where people feel welcome and supported—something especially important for those entering new phases in their lives. I am also deeply grateful for your advices, not only on academic matters but on life itself, demonstrating your unique ability to lead and teach with compassion. Simple things, like "in Germany, we say our last name when answering the phone," along with many other insights, were invaluable in helping me adapt and learn. Thank you, Achim!

I would also like to extend my sincere thanks to Prof. Carlos Eduardo Pereira. Your guidance since my Master's, countless opportunities, and invaluable advice have been instrumental in my growth, both professionally and personally. You have an admirable talent for connecting with people from different backgrounds and recognizing their strengths. I am always inspired by that. I deeply appreciate your mentorship over the past years and all the opportunities you have given me.

To my parents, Telmo Steinmetz and Lisete Catarina Krein Steinmetz, my grandparents, João Francisco Steinmetz and Nelsi Helena Kuhn Steinmetz, Soeli and Valdir Semprebon, and my extended family—only we know how hard it was to get here. There is a vast gap between working with farming tools and programming a computer. I can only imagine the difficulty you faced when advising me on whether to pursue Computer Science or something else, and in so many other life decisions that, from a farming perspective, seemed even harder than NP-complete problems. Thankfully, from you, I learned the most important values, which gave me the strength to overcome the chal-

lenges I faced on this journey. To me, you are the real doctors I am proud of. I am deeply grateful for your unwavering love, support, and sacrifices. I would not be where I am today without you.

Alyona Moskova, my girlfriend—you and your family have been my rock. You have been my biggest supporter throughout this journey. Your unwavering love, patience, and understanding have been my guiding light, and I am so grateful for your presence in my life. I am so lucky to have you by my side. I love you.

A heartfelt thank you to my friends and their partners: Andre Faria, Breno Menezes, Cintia Martins, Daniel Flores Bastos, Diego Siqueira, Henrique Martins, Laercio Antonio Krein, Lauren Bohner, Luis Filipe Araujo Pessoa, Meher Malik, Priscila Ferreira, and Rebeca Montenegro. You patiently listened to my struggles and were always there to offer help and encouragement. Your support has been my foundation. I am grateful for your friendship and for the countless moments of joy and laughter we shared. So many others also helped me, but I can't write another thesis only with names :/. You know who you are, and I am grateful for your support.

A special thank you goes to Greyce Schroeder and Alecio Binotto for their significant contributions to my PhD. Without your help, this thesis and the accompanying publications would not have reached the level of quality we achieved. I am truly grateful for your friendship and support.

To the most supportive and collaborative research group, my colleagues and friends at HSHL: Bettina Weiß, Christopher Beck, Fatima Idrees, Gido Wahrmann, Katrin Glöwing, Kathleen Strodick, Kristian Rother, Mehdi Azarafza, Narmada Ambigapathy, Peer Adelt, Pia Arens, Prof. Stefan Henkler and your group, and many others in this wonderful working environment. I am grateful for the countless discussions, brainstorming sessions, and collaborations that have enriched both my research and my life. I am proud to be part of this group. Thank you, Department 2, Lippstadt, for everything.

Finally, I would like to extend my sincere thanks to all the members of my PhD committee and the department secretary for taking the time to review and evaluate my work. Your valuable insights and thoughtful feedback have been instrumental, and I deeply appreciate your commitment and dedication.

Abstract

In recent years, advancements in digitalization technologies, such as Meta's Metaverse and Apple's Vision Pro, have redefined human-computer interaction, emphasizing the need for virtual representations of physical entities. The evolution of concepts such as Internet of Things (IoT) and Cyber Physical Systems (CPSs), Digital twins, serving as these virtual counterparts, optimize system performance, enhance decision-making, and drive innovation. However, challenges remain in creating a unified, extensible methodology and ensuring interoperability across different systems and stakeholders. This thesis investigates the modeling and deployment of semantic digital twins (DTs), focusing on their application across various domains.

The research explores five key areas: developing a generic and domain-independent modeling approach, enabling parallel operation of DTs at the edge, integrating ISO 23247 and Web of Things (WoT) standards, creating semantic models comprehensible to both machines and stakeholders, and simplifying the extension of existing applications with DT capabilities.

A universal modeling approach, implemented on Node-RED, demonstrates the feasibility of these objectives. The proposed nodes, adhering to ISO 23247 and WoT standards, enable interoperability and standardization. Implementing these models in diverse scenarios, such as industrial and smart city applications, validates their flexibility and extendability. The ability to execute models in parallel at the edge enhances performance and scalability, confirming the approach's utility in dynamic environments.

The thesis also highlights the importance of semantic modeling for better stakeholder collaboration and machine understanding. By utilizing knowledge graphs, the developed models facilitate accurate inferences and decision-making. However, limitations such as the lack of real-time capabilities and version control are noted.

Overall, the proposed methodology and framework significantly contribute to the field of digital twins, offering a scalable, interoperable, and semantically rich approach to modeling and deploying DTs. Future research directions include exploring machine learning for complex system modeling, enhancing version control, applying the methodology to new domains, and developing advanced tools for DT deployment.

Zusammenfassung

In den letzten Jahren haben Fortschritte bei den Digitalisierungstechnologien, wie Metas Metaverse und Apples Vision Pro, die Interaktion zwischen Mensch und Computer neu definiert und den Bedarf an virtuellen Darstellungen physischer Einheiten betont. Die Entwicklung von Konzepten wie Internet of Things (IoT) und Cyber Physical Systems (CPS), digitale Zwillinge, die als diese virtuellen Gegenstücke dienen, optimieren die Systemleistung, verbessern die Entscheidungsfindung und fördern die Innovation. Die Schaffung einer einheitlichen, erweiterbaren Methodik und die Gewährleistung der Interoperabilität zwischen verschiedenen Systemen und Akteuren stellen jedoch nach wie vor eine Herausforderung dar. In dieser Doktorarbeit werden die Modellierung und der Einsatz semantischer digitaler Zwillinge (DTs) untersucht, wobei der Schwerpunkt auf ihrer Anwendung in verschiedenen Domänen liegt.

Die Forschung untersucht fünf Schlüsselbereiche: die Entwicklung eines generischen und domänenunabhängigen Modellierungsansatzes, die Ermöglichung des parallelen Betriebs von DTs am Rande, die Integration von ISO 23247 und Web of Things (WoT)-Standards, die Erstellung semantischer Modelle, die sowohl für Maschinen als auch für Stakeholder verständlich sind, und die Vereinfachung der Erweiterung bestehender Anwendungen mit DT-Funktionen.

Ein universeller Modellierungsansatz, der auf Node-RED implementiert wurde, zeigt die Machbarkeit dieser Ziele. Die vorgeschlagenen Knoten, die sich an die Normen ISO 23247 und WoT halten, ermöglichen Interoperabilität und Standardisierung. Die Implementierung dieser Modelle in verschiedenen Szenarien, wie Industrie- und Smart-City-Anwendungen, bestätigt ihre Flexibilität und Erweiterbarkeit. Die Möglichkeit der parallelen Ausführung von Modellen am Netzwerkrand verbessert die Leistung und Skalierbarkeit und bestätigt den Nutzen des Ansatzes in dynamischen Umgebungen.

Die Arbeit unterstreicht auch die Bedeutung der semantischen Modellierung für eine bessere Zusammenarbeit der Beteiligten und ein besseres maschinelles Verständnis. Durch die Verwendung von Wissensgraphen erleichtern die entwickelten Modelle genaue Schlussfolgerungen und Entscheidungen. Allerdings gibt es auch Einschränkungen, wie z. B. das Fehlen von Echtzeitfunktionen und Versionskontrolle.

Insgesamt leisten die vorgeschlagene Methodik und das Framework einen wichtigen Beitrag zum Bereich der digitalen Zwillinge, indem sie einen skalierbaren, interoperablen und semantisch reichhaltigen Ansatz für die Modellierung und den Einsatz von digitalen Zwillingen bieten. Zukünftige Forschungsrichtungen umfassen die Erforschung des maschinellen Lernens für die Modellierung komplexer Systeme, die Verbesserung der Versionskontrolle, die Anwendung der Methodik auf neue Bereiche und die Entwicklung fortschrittlicher Werkzeuge für die DT-Bereitstellung.

Contents

1. Introduction

In recent years, a wave of transformative applications and technologies, such as Meta's Metaverse, Apple's Vision Pro, and others, have begun to redefine the landscape of human-computer interaction and system development. These innovations offer immersive experiences that blend the digital and physical realms, paving the way for novel ways of communicating, collaborating, and creating. As these technologies become increasingly integrated into daily life, they play a key role in the broader digitalization movement, pushing the boundaries of what's possible in virtual spaces. This shift towards more interactive and interconnected digital environments naturally aligns with the concept of Digital Twins (DTs), which serve as virtual representations of physical entities. DTs are becoming an essential concept in optimizing system performance, enhancing decision-making processes, and driving innovation in various industries. Together, these advancements are not only transforming how we interact with systems but also how systems are conceived, designed, and deployed, marking a significant milestone in the journey towards a more digitalized world.

Concepts like the Internet of Things (IoT) [AIM10] and Cyber-Physical Systems (CPS) have tractioned the development of systems that are more connected and that can interact with the physical world, enabling people to interact with these systems in different ways [BG11]. This is one of the key elements of Industry 4.0 [Las+14] and is the basis for the digitalization process of application.

This Chapter aims to introduce the context of the research, the motivation (Section 1.1), a short overview of the current state of the art and the hypotheses, the goals and contributions of the thesis, the structure of the thesis, and the publications produced from the thesis.

1.1. Motivation

The DT concept has gotten more attention since the development of new digitalization technologies and it has been used as a solution for applications that deal with big, distributed and isolated data [Liu+21]. Understanding what all this data means and how assets can be related to each other is essential for building smart systems. Additionally, handling dynamic and heterogeneous models is still an open challenge [Sah+21]. Therefore, adding semantics is crucial for adding meaning to the parts of the system since the modeling phase.

The digitalization opens a new set of challenges and opportunities for the development of systems. Data-driven approaches [BD24] and the increasing usage of AI-based solutions have been changing the way systems are developed and operated. Additionally, the advent of Generative AI [Bor+24] and Large Language Models (LLMs) has strengthened the importance of semantics in the development of systems and the integration of different data sources. The use of semantics is crucial for the development of systems that can be understood by different stakeholders and that can be integrated with other systems.

Applications based on the DT concept usually involve different stakeholders and different phases of the lifecycle of the system (as can be seen in Figure 1.1). The development of a unified methodology, modeling language, and architecture for semantic modeling of digital twins is essential to address the complexity and diverse requirements of stakeholders in IoT applications, ultimately enhancing the effectiveness and interoperability of these digital representations [Zha24].

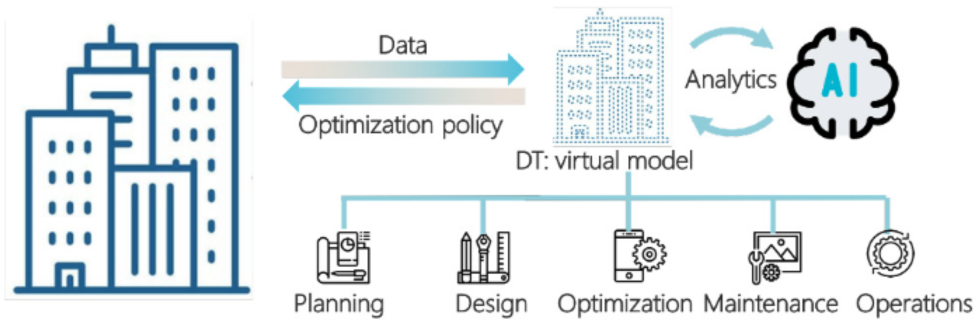


Figure 1.1.: Digital Twin model [Zha24].

1.2. Challenges

This section aims to give a brief overview of the current challenges identified in the field of digital twins and semantic modeling. A deeper analysis of the literature will be presented in Chapter 3.

Figure 1.2 illustrates the traditional scenario of systems development. It usually starts with the modeling phase, where the system is represented using different modeling languages and tools. These high-level representations will then be used to generate the instances that will run in the real world.

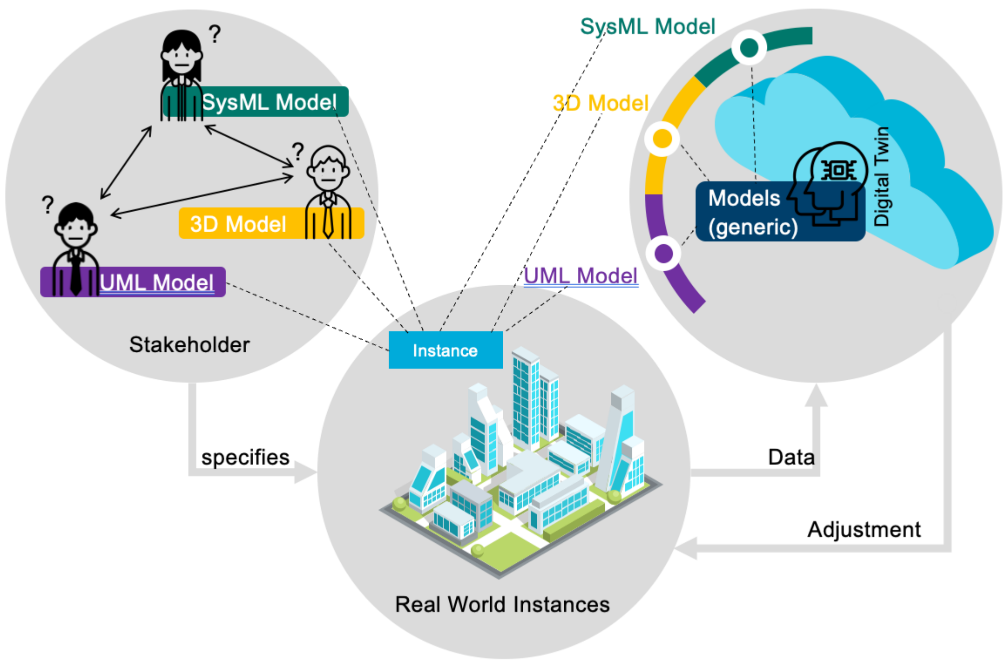


Figure 1.2.: Challenges in the development of digital twins.

When the instances start running in the real world, the data generated by these instances will be collected and analyzed. This data is then stored and used to generate insights that will be used to improve the system. This process is usually done by different stakeholders, each one using different tools and languages. This scenario is usually complex and hard to manage, especially when the system is large and involves different stakeholders since the knowledge is usually distributed and isolated.

One key challenge in the development of digital twins is the lack of a common language and approach for modeling the system. This is crucial for enabling different stakeholders to express their knowledge in a way that other stakeholders and machines can understand. The lack of a common language and approach can lead to misunderstandings and misinterpretations, which can result in errors and inefficiencies in the system.

Additionally, modern systems must be ready to overcome the challenge of living in silos [DJM22] [PI21], especially with the advent of AI-based solutions. The integration of different data sources and the development of systems that can be understood by different stakeholders is crucial for the development of systems that can be integrated with other systems and that can be used in different domains.

As DT applications are based on the IoT concept, it is important to consider how models can run in parallel and close to the edge. This is crucial for ensuring that the models are up-to-date and accurate, especially in dynamic and distributed environments. The use of semantics is essential for improving the representation of the system and ensuring that the models are comprehensible to both machines and stakeholders, particularly those situated close to the edge of the network. Bringing the model close to the edge devices can help to avoid misinterpretations at the higher level of the system since as soon as the data is generated, it can be correctly modeled with all necessary semantic information. For example, if a sensor is measuring the temperature of a machine, the data generated by this sensor can be directly modeled with the necessary semantic information (such as the unit in Celsius), avoiding misinterpretations at the higher level of the system.

Besides, the lack of a generic standard [YPK17] for modeling digital twins is another challenge [ZLK22]. This makes it difficult to develop systems that can be easily extended and used in different domains.

In this context, after understanding the motivation and the main challenges, the next section describes a vision of an ideal scenario where the development of systems based on the DT concept will be more accessible and collaborative.

1.2.1. Vision

The primary objective in integrating Digital Twins (DTs) seamlessly is to shift towards technological ecosystems that are more accessible, adaptable, and collaborative. Industries aim to fully leverage DTs for operational excellence and innovation, highlighting the need to simplify the transition of current applications to DT-enabled platforms. This ambition extends beyond the mere technical capability for such integrations; it envisions a scenario where diverse domains of knowledge and applications merge. Through this convergence, the advancement into a new era of digital innovation is propelled.

In this context, the vision of this thesis is that in the near future the development of systems based on the DT concept will be more accessible and collaborative. This vision is based on three main aspects:

- To create a vendor-independent and cross-domain solution for helping current applications implement the DT concept with less effort.
- stakeholders from different domains can express their knowledge in a way other stakeholders and machines can understand.

- enhancing the collaboration between different stakeholders and AI-based solutions.

In this context, several research questions and hypothesis have been defined to guide the development of the thesis. These questions and hypotheses will be presented in the next section.

1.3. Research Questions and Hypothesis

The foundation of this thesis is established through the definition of specific research questions and hypotheses. These elements are intended to guide the formulation of methodologies, the development of modeling languages, and the architectural design for semantic modeling of digital twins. Aimed at investigating the creation of a more inclusive, adaptable, and universally applicable framework for integrating digital twins across various fields, the following research questions and hypotheses have been developed in response to the identified objectives and challenges:

Research Question 1.3.1. *Can a generic and not domain-specific approach, comparable to the current state of the art in semantic modeling of Digital Twins, be developed to facilitate its application across different domains and ensure its extensibility?*

Hypothesis 1.3.1. *A universal modeling framework can be created, which, while maintaining the depth and utility of current domain-specific models, offers greater generic applicability and ease of extension across various industry sectors.*

Research Question 1.3.2. *Is it feasible to design DTs where models operate in parallel, close to the edge devices and are maintained by experts for each asset?*

Hypothesis 1.3.2. *By employing innovative strategies, Digital Twin models can be set up to function concurrently and operate on edge devices, with maintenance and updates performed by domain experts using a standardized language, thereby enhancing the relevance and accuracy of each model.*

Research Question 1.3.3. *How can the principles of ISO 23247 and the Web of Things (WoT) be integrated into the design of digital twins to promote standardization and interoperability?*

Hypothesis 1.3.3. *By embedding the guidelines of ISO 23247 and the WoT framework into the DT design process, a standardized and interoperable model can be achieved, facilitating seamless communication and data exchange across different platforms and devices.*

These research questions and hypotheses aim to direct the thesis, towards addressing key challenges in the field of digital twins, with a focus on creating a versatile and user-friendly environment for semantic modeling and deployment.

1.4. Goals and contributions of the thesis

The main goal of this thesis is to help existing IoT-focused applications apply the Digital Twin concept with less effort, enabling stakeholders from different domains to express their knowledge in a way that other stakeholders and machines can understand. This will be achieved by developing a unified methodology, modeling language, and architecture for semantic modeling of digital twins that can address the complexity and diverse requirements of stakeholders in IoT applications, ultimately enhancing the effectiveness and interoperability of these digital representations. This goal will be achieved by addressing the following research objectives:

- Define modeling elements that can be used to represent the structural aspects of digital twins, addressing the diverse needs and applications of different stakeholders within IoT ecosystems.
- Develop a methodology for modeling digital twins that can be applied across different domains and ensure its extensibility.
- Propose a layered architecture that can help system designers structure their systems based on the responsibilities of each component.
- Design digital twins where models operate concurrently, close to the edge devices, and are maintained by experts for each asset.

- Integrate the principles of ISO 23247 and the Web of Things (WoT) into the design of digital twins to promote standardization and interoperability.
- Enable the creation of semantic and connected models that are comprehensible to both machines and stakeholders, particularly those situated close to the edge of the network.

1.5. Structure of the thesis

The thesis is structured as follows. This chapter, the Introduction, provides an overview of the context, motivation, research questions, and hypotheses, as well as the goals, contributions of the thesis, and the publications produced during the development of the thesis.

In Chapter 2, the main concepts and technologies used in the thesis are presented. The chapter aims to provide a basis for understanding the concepts and technologies used in the thesis, including IoT, CPS, model-based design, and semantic technologies. The chapter also presents an overview of the DT concept, followed by a discussion of how semantic is applied in DTs, applications and modeling approaches. Finally, as the DT concept is still under construction, the chapter presents the definition that is used in this thesis.

Chapter 3 presents the state of the art with a focus on the digital twin concept, semantic modeling, and related technologies. The chapter aims to provide an overview of the current state of the art in these areas, highlighting the main challenges and opportunities for future research.

In Chapter 4, the proposed layered architecture is presented. It describes each layer in detail, which is important to understand how to design the system in a modular way.

Chapter 5 presents the proposed modeling elements that can be used to represent the structural aspects of digital twins. The chapter aims to provide a detailed description of the modeling elements and how they can be combined to represent knowledge. The formal definition of the model and the grammar for the modeling language are also presented. Additionally, an overview of the implementation of the concept in Node-RED is given.

Chapter 6 presents the proposed methodology for modeling digital twins. The chapter aims to provide a detailed description of the steps of the methodology and how it can be applied across different domains. The chapter also presents an overview of the architecture and the modeling elements used in the methodology.

In Chapter 7, two use cases are presented to demonstrate the application of the proposed approach in this thesis. The chapter aims to provide a detailed description of the use cases and how the proposed methodology, modeling elements and architecture can be applied in practice.

Chapter 8 presents the evaluation of the proposed approach, highlighting the main findings and limitations of the research. The chapter aims to provide an overview of the evaluation process and the results obtained.

Finally, Chapter 9 concludes the thesis, summarizing the key findings and contributions, and outlining potential future research directions.

1.6. Publications produced during the develop of the thesis

Also as a form of validation and opportunity for discussions and feedback, several publications have been produced from the content of this thesis. Below, the list of publications with their main contributions are listed.

- An IoT-based ontology for DTs has been proposed in [Ste+18b]. The use of ontologies and standard middleware for integration IoT systems in the context of Industry 4.0 have been explored in [Ste+18c]
- Applications and product-as-a-service based on DT have been explored in [Ste+21a] and considering the interaction with different users/stakeholders in [Ste+20].
- The proposed architecture and the key components for modeling DTs have been presented in [Ste+21b]. A study of the use of Hierarchical Colored Petri Nets has been conducted in [SSR22]. The connectivity topologies for building DTs have been proposed in [Sch+21a].
- The basis for the proposed methodology has been discussed in [Sch+21b] and extended in [Ste+22a], and support for Knowledge graphs using Node-RED have been presented in [Ste+23]. The use of Node-RED for implementing Digital Twins focused on the autonomous driving and smart city context was done in [Ste+22b].
- Finally, the overall concept of this thesis has been presented in the PhD Forums at DATE 2023 and DAC 2023.

2. Foundations

This chapter aims to give an overview of the foundations of the research. It starts by discussing the IoT and its relation to CPS. Then, it presents the Industry 4.0 and its relation to the previous concepts. After that, it discusses the Model-based Design and, then, it presents the concept of semantic models and knowledge graphs, and their relation to the previous concepts.

2.1. Internet of Things

The Internet of Things (IoT) was first introduced by Kevin Ashton in 1999, who described IoT as a network of objects that are uniquely identifiable and interconnected through radio-frequency identification (RFID) technology, enabling them to communicate [Ash+09]. Nevertheless, the precise definition of IoT remains under development, evolving based on the various perspectives considered [LXZ15].

IoT is distinguished by the utilization of interconnected devices that integrate physical components, software, and embedded technologies. These components are capable of interacting and collaborating with one another to achieve a unified goal [AIM10]. This enables these devices to produce, share, and use data with little to no human involvement [REC15].

As outlined by [AIM17], there are three primary evolutionary stages of IoT (Figure 2.1): (a) the era of tagged objects, focusing on the identification and tracking of physical items; (b) the era of device interconnectivity through web technologies; and (c) the era of social objects, which emphasizes semantic data representation and the integration of cloud technologies, facilitating more sophisticated interactions and data management within the IoT ecosystem.

An important point to consider in the adoption of IoT is the heterogeneity of its system components [AIM10]. It's essential that interoperability is ensured to facilitate service provision and data exchange. Addressing this challenge involves the implementation of middleware that acts as a communicative bridge between devices and applications,

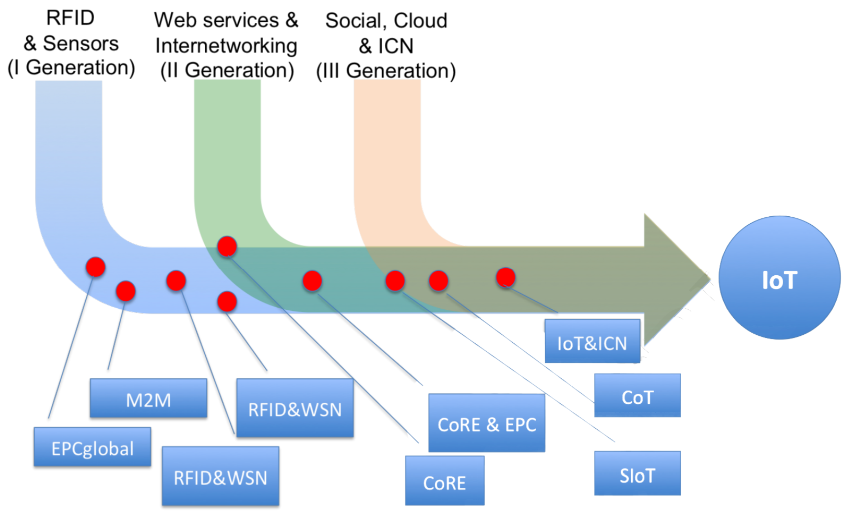


Figure 2.1.: Evolution of the IoT [AIM17]

enabling interaction across various devices, operating systems, and architectural frameworks [Far+17].

Nowadays, IoT is being used in several domain areas such as smart cities, smart homes, smart healthcare, and smart agriculture. Figure 2.2 shows the use of IoT in different domains.

These applications are not always isolated in a single domain and they can impact other domains. For example, a smart city application can impact the smart healthcare domain, as it can provide data for the healthcare system to make decisions. This opens a new set of challenges and opportunities for the development of systems that are not isolated in silos.

The IoT concept works as a basis for the development of Cyber-Physical Systems (CPS), connecting the physical world and the cyber world. More details about CPS are given in the next section.

2.2. Cyber Physical Systems

Cyber-Physical Systems (CPS) are systems that link computer-based algorithms with physical processes. Through embedded computing devices and networks, these systems actively monitor and manage physical operations, creating dynamic interactions where the physical state influences the computation and vice versa. This interdisciplinary field necessitates innovative computing and networking solutions to address its inherent com-

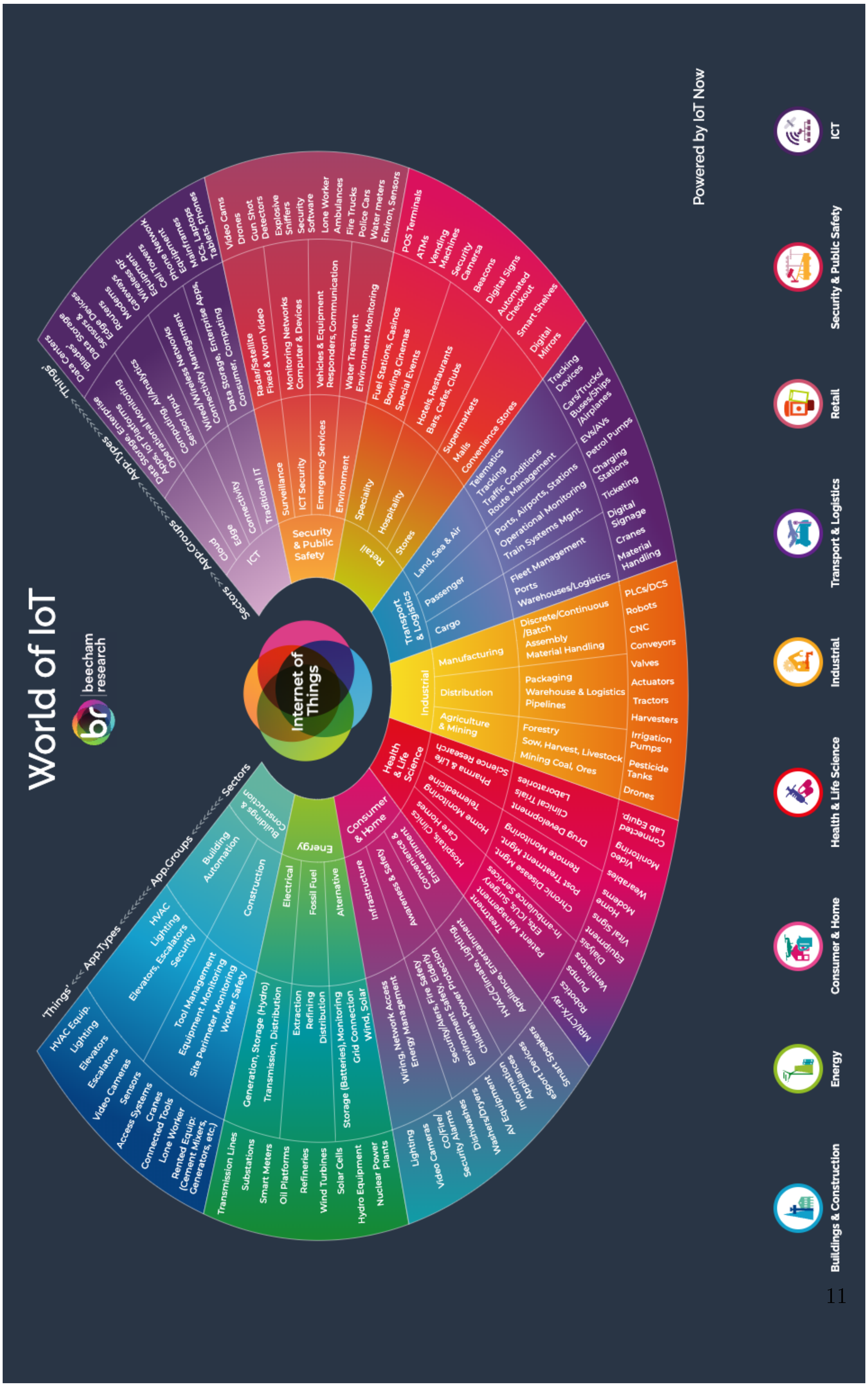


Figure 2.2.: IoT in different domains ©IoT Now

plexities, including safety and timing constraints, across a wide range of applications like transport systems, healthcare monitoring, and infrastructure control [Lee08]. It integrates the acting ability of the real world with the intelligence and computational power of the cyber world, enabling the development of smart systems that can adapt to changing conditions and optimize their performance.

Figure 2.3 illustrates a simplified architecture of a typical CPS, which contains two main layers: the physical where sensors and actuators are located; and the cyber where the software part is hosted. The usual flow is that measurements are taken from the physical layer and sent to the cyber layer, where they are processed and used to control (via commands) the physical layer. This interaction is what makes CPS a unique field, as it requires the integration of both layers to work properly. The seamless operation of these components within a unified architecture is essential for the functionality of CPS, enabling sophisticated applications such as autonomous vehicles and real-time patient monitoring systems [Son+16].

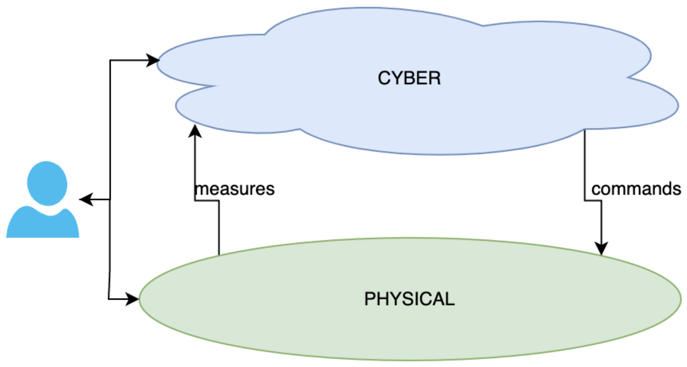


Figure 2.3.: Simplified CPS Architecture

The authors of [LS16] describe CPS as integrative mechanisms that orchestrate computation with physical processes. Through this integration, CPS enables real-time monitoring, control, and analysis of the physical world via cyber infrastructure. They also provide a foundational understanding of CPS, emphasizing their role in enhancing interaction between digital and physical domains across various sectors, including healthcare, transportation, and manufacturing.

CPS systems are pivotal in the development and advancement of various other technologies such as smart vehicles, autonomous driving in urban environments, and medical devices controlled by brain signals. CPS requires a multidisciplinary approach, blending insights from system science, engineering, networking, software, and human interaction to foster the creation, integration, and efficient operation of these complex systems [BG11].

CPS presents several challenges such as the complexity of CPS due to their intrinsic heterogeneity, concurrency, and timing sensitivity. The authors of [DLV11] highlight that hybrid system modeling, concurrent and heterogeneous models of computation, domain-specific ontologies, and the joint modeling of functionality and implementation architectures can help to address these challenges.

The adoption of CPS raises critical ethical and societal questions, particularly concerning privacy and the potential displacement of jobs due to automation. The ethical use of CPS, especially in sensitive applications such as personal health monitoring, requires careful consideration of privacy concerns and the development of stringent data protection measures. Moreover, policy frameworks that balance technological advancement with societal values are essential for ensuring the equitable distribution of CPS benefits [Hum+17]. This also highlights the need for bringing models closer to the physical world, avoiding misuse or misunderstanding of data at the higher levels of the system.

The integration of the physical and the cyber world is a key concept in the Industry 4.0 domain [Jaz14]. A deeper overview of the Fourth Industrial Revolution is given in the next section.

2.3. Industry 4.0

The concept of Industry 4.0 was first introduced in 2011 through a collaborative effort by the German government, universities, and private entities, aiming to advance manufacturing systems to bolster the productivity and efficiency of the country's industrial sector [KWH+13]. This initiative signifies a transformative phase in manufacturing, incorporating emerging and converging technologies to enrich the entire product lifecycle [Dal+18]. Additionally, it calls for a socio-technical shift in the role humans play within production systems, moving towards smart working practices across the value chain, all rooted in the foundations of information and communication technologies (ICTs) [Sto+18].

Industry 4.0 marks a huge shift in manufacturing, propelled by IT advancements that integrate digital technologies with physical production processes. This convergence fosters a move from product-centric to service-oriented models within traditional industries, potentially leading to the emergence of new business types that play specialized roles in the manufacturing and value creation networks. The development and management of these complex, dynamic, and integrated systems present both a significant challenge and opportunity for the Business and Information Systems Engineering (BISE) discipline, aiming to enhance the competitiveness of industrial enterprises through innovative integration, automation, and decentralization strategies [Las+14].

This concept touches different knowledge areas such as Automation, CPS, Cloud computing, IoT, the Internet of People, Digital Twins, Semantic technologies, Smart products and more. It is a complex and multidisciplinary field that requires a deep understanding of the physical world and the digital world, and how they interact with each other [Gho20]. Figure 2.4 shows an architectural design of Industry 4.0.

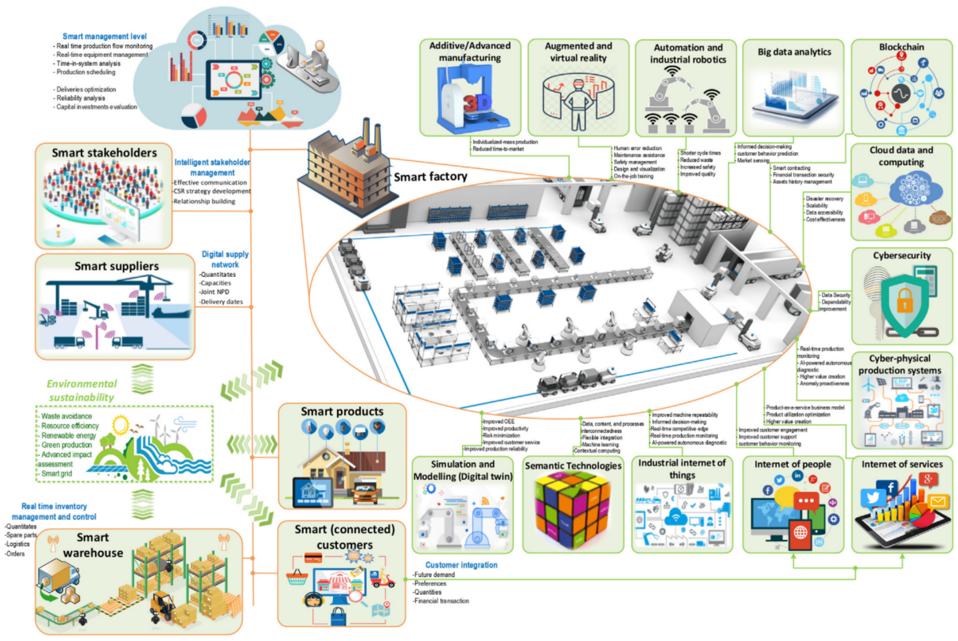


Figure 2.4.: Industry 4.0 Architectural design: [Gho20]

To build applications in this new industrial revolution, it is important to have a good understanding of the system in early development stages, enabling early design validation, and facilitating complex simulations, optimizing resource allocation. This is where the Model-based Design comes in. The next section presents an overview of this concept.

2.4. Model-based Design

Model-Based Design (MBD) is a systematic approach to engineering that uses virtual models for the design, simulation, verification, and validation of complex systems and products. Central to MBD is the use of mathematical and visual models as the foundation of all phases of development, from conceptual design through to implementation. This methodology enables engineers to explore and refine systems in a virtual environment, significantly reducing the need for physical prototypes and accelerating the development process. By facilitating early detection of design issues and supporting system-level optimization, MBD enhances product quality and development efficiency. The core principles of MBD include abstraction, where complex system behaviors are represented in simplified forms; automation, enabling automatic code generation and testing; and verification, ensuring the model meets specified requirements before physical realization [Eke+03].

The adoption of MBD across various industries, including automotive, aerospace, and electronics, has been instrumental in addressing the challenges of designing complex systems. Tools and standardized languages such as MATLAB/Simulink [Cha17], Modelica [Til01], and UML (Unified Modeling Language) [BRJ97] play a pivotal role in MBD by providing environments for simulation, model verification, and automatic code generation. They support the MBD methodology by allowing for the iterative refinement of models, integration with legacy systems, and the simulation of system behavior under different scenarios. The benefits of employing MBD extend beyond development efficiency to include improved system reliability, faster time-to-market, and enhanced capability to handle increasing system complexity [Lee08]. As such, MBD represents a paradigm shift in engineering design, moving away from traditional document-based approaches to a more integrated and holistic model-centric methodology.

MBD approach offers numerous benefits compared to traditional methods such as early validation and verification through simulation, allowing engineers to identify and address potential issues early in the design phase [FM08]. Additionally, it facilitates downstream activities such as automated programming of tools and equipment, ultimately leading to improved product quality, reduced development time, and enhanced cost-effectiveness.

While model-based design provides a framework for visualizing, simulating, and validating system designs and behaviors, semantic modeling takes this a step further by adding a layer of meaning to the models. It focuses on the relationships and data within these systems, enabling a deeper understanding of how components interact and the significance of these interactions. The next section presents an overview of semantic models and knowledge graphs, highlighting their role in enhancing the capabilities of model-based design and digital twins.

2.5. Semantic models and Knowledge graphs

While semantic modeling provides a foundation by defining relationships and meanings within data, knowledge graphs build on this by creating an intricate network of interconnected information. This allows for a more nuanced understanding and utilization of data, facilitating advanced analysis and insights. In essence, knowledge graphs extend the capabilities of semantic modeling, offering a powerful tool for synthesizing and leveraging knowledge across various domains.

Knowledge graphs have been gaining popularity since 2012 when Google announced that they were using it in their search engine, and since then it has gotten several definitions published in literature [EW16].

They can be defined as an interlinked set of concepts that describe things from the real world such as entities, events, and their relationships within a context that is understandable by humans and machines [BHW21].

In [Wan+17] a knowledge graph is defined as a multi-relational graph where entities are represented as nodes and relations as edges that connect two nodes. Figure 2.5 shows an example of two nodes (*dog* and *animal*) and their relationship (is).

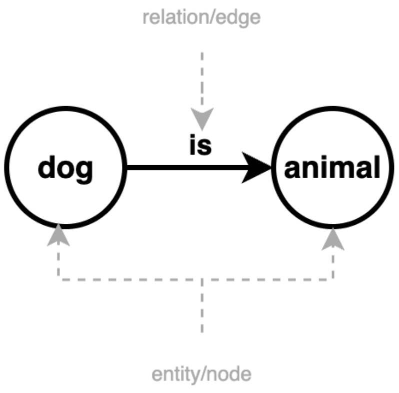


Figure 2.5.: Example of two nodes and one edge: *dog is animal*

This form of organizing data can be found in many services of companies around the world. Uber Eats, for instance, uses graph technology to learn which food is more likely to appeal to a specific user [Jai+22].

As a Knowledge graph can be used to describe things from the real world, this technology has also been used in IoT applications, making lots of connected devices easily discoverable via semantic queries [Le-+16]. It has also been applied to solve the heterogeneity problem of the different devices that might be connected via the IoT [Xie+21].

There are several databases available in the market nowadays that support knowledge graphs [FB18] such as Neo4j, TypeDB, and others. Neo4j [Neo22], for example, stores the connections between data nodes which enables efficient queries in Cypher language to be executed.

2.6. Digital Twin

This section presents the concept of DT and its applications. It also presents the modeling and deployment of DT systems. Finally, a definition of DT is presented for this thesis.

2.6.1. Concept

The Digital Twin concept can be seen as an evolution of the IoT and CPS. IoT enables the connection of physical objects to the internet, while CPS connects physical and computational systems. The Digital Twin concept goes beyond these concepts by creating a virtual representation of observable assets from the real world, enabling, for example, the simulation of its behavior and the prediction of its future states. This virtual representation is continuously updated with real-world, enabling the monitoring and control of the physical object [AIA22].

This concept has been applied in different areas, as for example, to manage the Product lifecycle [Gri14]. In 2010 the National Aeronautics and Space Administration (NASA) started investigating this concept and in 2012 they reviewed and defined it as a "multiphysics, multiscale, probabilistic, ultra fidelity simulation that reflects the state of an asset", composed by current and historical data and physical models [GS12]. Later, in 2017, Grieves formally defined the concept of a digital twin in a white paper, where he introduced the fundamental model of digital twins. This model encompasses physical objects, their virtual counterparts, and the data connection that bridges the physical and virtual realms [GV17]. Figure 2.6 shows the timeline of the DT concept proposed by the authors of [Zha24].

Digital Twin can also be seen as a set of virtual information that mimics the structure, context, and behavior of an asset or group of assets. It is dynamically updated real-world data from its physical twin during its whole life cycle, informing decisions that bring value [AIA22].

In [RSK20], authors define DT as a virtual representation of a physical asset that, via data and simulators, enables prediction, optimization, monitoring, controlling, and improvements of decision-making. The virtual representation should be continuously updated with real-world data.

It is possible to notice that the concept of DT is still under development and research, and there is no single and unique definition that defines this concept and is agreed upon by the community. Therefore, it is expected that different contributions will be made aiming to construct a global understanding of this new digitalization approach.

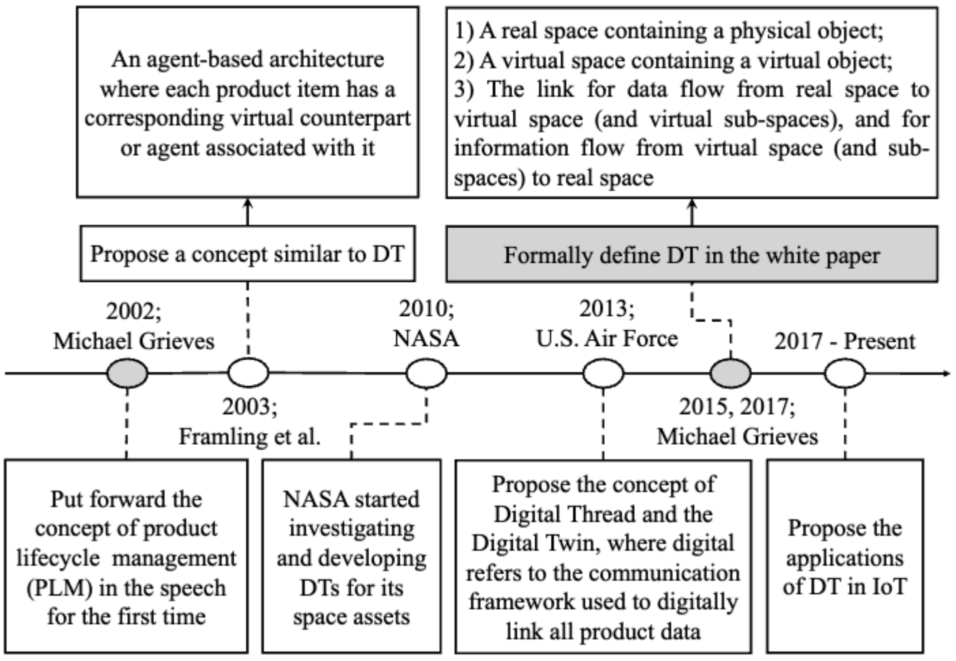

An agent-based architecture where each product item has a corresponding virtual counterpart or agent associated with it
Propose a concept similar to DT
1) A real space containing a physical object;
2) A virtual space containing a virtual object;
3) The link for data flow from real space to virtual space (and virtual sub-spaces), and for information flow from virtual space (and sub-spaces) to real space
Formally define DT in the white paper
2002; Michael Grieves
2003; Framling et al.
2010; NASA
2013; U.S. Air Force
2015, 2017; Michael Grieves
2017 - Present
Put forward the concept of product lifecycle management (PLM) in the speech for the first time
NASA started investigating and developing DTs for its space assets
Propose the concept of Digital Thread and the Digital Twin, where digital refers to the communication framework used to digitally link all product data
Propose the applications of DT in IoT

Figure 2.6.: Digital Twin timeline [Zha24]

Within the Digital Twin concept, a specialized subset known as the Cognitive Digital Twin incorporates Artificial Intelligence, emphasizing the critical role of semantic models. The following section will provide a concise overview of this concept.

2.6.2. Cognitive Digital Twin

Currently, there is a move in research from traditional Digital Twins to Cognitive Digital Twins (CDT) [DAm+22]. Figure 2.7 shows this advent where authors are adding semantics to Digital Twins.

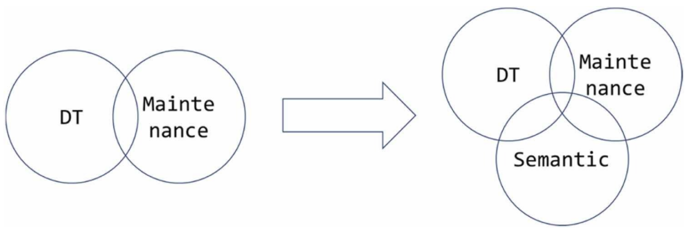


Figure 2.7.: Searches for Digital Twin [DAm+22]

The concept of the Cognitive Digital Twin has been introduced as a solution to the challenge of integrating different domains, enhancing digital twins with advanced semantic capabilities [Jin+22]. CDT serves as a comprehensive digital counterpart and intelligent partner to its physical twin, encompassing all subsystems and spanning every phase of its life cycle and evolutionary stages [Adl16].

The CDT is a specialized subset of the Digital Twin concept that incorporates AI and emphasizes the critical role of semantic models. It uses AI to learn from the data generated by the DT and other sources to improve its performance and decision-making capabilities. It can also be used to predict future events and optimize the operation of the system [ZLK22].

This subset of DT has a big synergy with the proposal of this thesis, since both deals with the use of semantics to improve the performance of the DT. In the next section, some of the main applications of DT are presented to better understand how this concept is used and its potential.

2.6.3. Digital Twin definition for the thesis

As it was possible to see, the DT has been the focus of research by many researchers in academia and industry. However, no single and unique definition still defines this concept and is agreed upon by the community. One of the reasons is that the concept is still under development and research; therefore, it is expected that different contributions will be made aiming to construct a global understanding of this new digitalization approach.

This thesis also contributes with a definition, which is partly extracted from the literature and other parts defined by the author's experience and point of view. It is important to define how the digital twin is understood to have a better comprehension of the next chapters of this work.

Figure 2.8 shows the main elements and how they interact with each other to form a DT representation.

An **asset** represents an instance of the real world that can be observed. An asset can also be composed of other assets. For example, a car is an asset, and its engine is another asset. Assets have **properties**, which can be defined directly or indirectly (e.g. through data fusion) by sensor readings. For example, an engine can have one or many temperature sensors that provide the temperature of this observable element or part of it.

Assets can also perform **actions** that enable them to interact with the real world. For example, a car's engine can be turned on or off. Actions can impact other assets as well, and not necessarily the asset performing the action. **Events** can be generated by the asset or by the environment. For instance, a car can generate an event when its engine

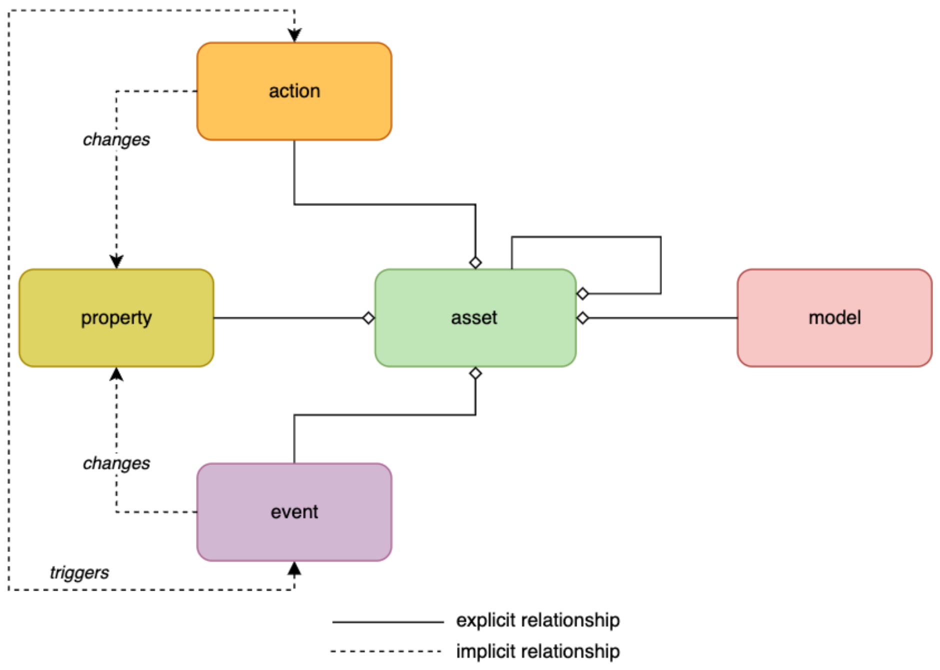


Figure 2.8.: Digital Twin conceptual diagram

is turned on. Actions and events can trigger each other and cause changes in the asset's state (properties).

Additionally, **models** represent the asset from different perspectives, such as behavior, geometry, and context.

In this context, *a DT is a combination of models that represents an observable asset from the real-world (device, process, or anything that can be observed) in different perspectives (behavior, geometry -3D models-, . . .), is continuously updated and can interact with the real-world. It can enable simulations, and learning mechanisms and track the whole life-cycle of an asset.*

3. State of the Art

This chapter presents the most related works in the literature, with the aim of identifying the main contributions of this thesis. The related works are divided into sections based on their areas. An analysis comparing these works with the thesis is shown at the end of this chapter.

3.1. Internet of Things technologies and applications

As already discussed in previous chapters, the Digital Twin concept is built on top of the IoT, enabling a new interaction paradigm between the cyber and the physical world. In this context, this section presents the state of the art related to IoT, technologies and its applications.

There are many fields of application for IoT, such as Healthcare, Manufacturing, Smart cities, Agriculture, Smart homes, and more [ARJ19] [Ans+21]. Figure 3.1 illustrates some of the applications areas of this concept.

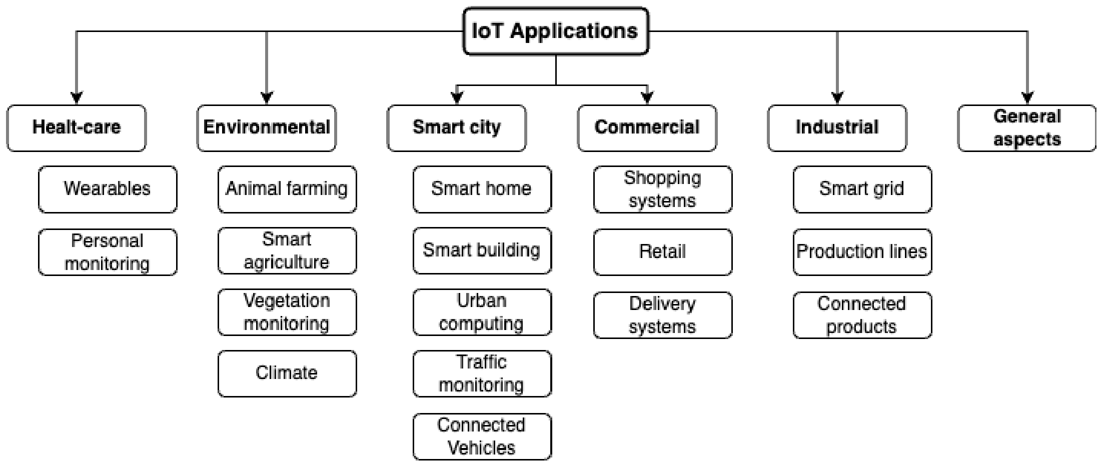


Figure 3.1.: Taxonomy of IoT applications (adapted from [ARJ19])

This variety of application domains shows, on the one hand, the importance of this concept, but on the other hand, it brings several challenges, such as interoperability, discoverability, data representation and storage, integration, etc., that need to be addressed [Niž+20] [Man+22] [Niž+20].

Addressing the semantic issues of most available approaches in the literature, the thesis of [Thu22] proposes a semantic extension to the Node-RED tool in the context of Industrial automation and IoT applications. This extension incorporates semantic definitions, particularly those from *iot.schema.org*, into Node-RED to address the challenge of semantic interoperability in IoT applications.

The main goals of this work are:

- To guide non-experts in semantic technologies, including device vendors and machine builders, in configuring device semantics consistently within IoT applications.
- To enable engineers and IoT application developers to design and develop semantically interoperable IoT applications with minimal effort, ultimately enhancing the ease of creating applications that can work across different vendor devices and ecosystems.
- To accelerate the application development process by introducing semantic application templates, referred to as "Recipes," which automate complex tasks such as skill matching between Recipes and existing components, facilitating the creation of complex IoT orchestrations.

The outcomes of the work [Thu22] have been used as a basis for the present thesis, which also aims to provide a semantic-based approach for modeling systems. However, this thesis extends that work by providing a concept of digitalizing real-world systems and assets with an emphasis on a common language and methodology that different stakeholders can use to express their knowledge.

In the agriculture field, the IoT concept is also applied to help evaluate soil state, atmospheric conditions, control and monitor variables such as temperature, humidity, vibration, or shocks during the transportation of products [Tal+17]. This paper suggests that while current IoT solutions in the agriculture domain rely on heterogeneous components and wireless sensor networks, future advancements may pivot towards cloud services and improved connectivity to realize a more integrated IoT ecosystem, which highlights the need for a common semantics to integrate these data sources.

In the Smart City context, several researches on IoT have been made exploring how to improve quality of life of the society [Sye+21]. The paper [Gha+21] explores the transformative potential of AI and IoT in fostering smart healthcare system within smart cities. By examining the application of sensor networks, IoT, and machine learning, the study reveals their crucial role in enhancing disease diagnosis and alleviating the burdens on healthcare professionals. The research anticipates a future where even everyday health items are IoT-enabled, contributing to significant improvements in quality of life and lifesaving capabilities. The authors highlight that the integration of smart technologies is not just an opportunity but a requisite for all healthcare stakeholders, aiming to optimize smart city living and improve the healthcare experience through AI innovations. This shows the need of having approaches for semantic modeling that integrate knowledge from different applications and stakeholders.

The IoT works as a basis for the DT implementation, and, therefore, the reviewed papers in this section will help to understand the path in which the next steps have to be taken. In the next section, several research works related to DT topic are discussed in detail.

3.2. Digital Twin applications and models

Digital Twins are virtual instances of real-world assets. They can be represented in different perspectives (in other words, models), and are continuously interacting with their physical twin. This section presents some of the most related research available in the literature.

The authors of [RSK20] present a comprehensive review of the benefits and challenges associated with modeling digital twins. It highlights values such as real-time monitoring, optimization, and enhanced decision-making while highlighting the challenges of data integration and scalability. However, the paper's exploration of the critical role of semantics in digital twins is somewhat limited. While it acknowledges the importance of accurate and meaningful data representation, a deeper analysis of semantic modeling (essential for ensuring the contextual relevance and interoperability of digital twin data) is still needed. This gap highlights a promising area for future research, particularly in developing semantically-rich digital twin models that can more effectively mirror the characteristics of their physical counterparts.

In the industrial context, DT models are being used for different purposes, such as focusing on production, predictive maintenance, and after-sale services. Implementation challenges in digital twin models can be identified, particularly in extending their role in various application domains. A more detailed exploration of semantic modeling would enrich the understanding of how DTs can accurately represent complex industrial systems and processes, ensuring more effective interoperability and contextual relevance in industrial settings [MDR20].

Modeling DT is the focus of several researchers, especially in the context of Industry 4.0 [JU20][RSK20]. However, other researchers still point out that standards for modeling and deploying DTs are still needed [YPK17]. In this section, some of the related works are presented.

Different models can be used for modeling DTs. One well-known industrial modeling language is AutomationML (Automation Markup Language - AML). This language has been used in a methodology for modeling data exchange for digital twins [Sch+16]. Furthermore, it has also been used for modeling DTs in several other researchers [Sie+18], [UWQ17] and [ZYW20].

Models can have relationships between them called dependencies. These dependencies show the process taken to create a solution, and they help to understand the implications of changes at any point of the process [BCT05].

Semantics are an important feature when knowledge has to be added to the model or to define relationships between elements [Xu+15] [Ste+18a]. Also in this context, a methodology based on ontologies uses the concept of *part digital twin* for supporting suppliers and assembly workshops[Bao+22].

Currently, there is no universally agreed-upon approach for Digital Twin (DT) modeling. Previous research has not fully encompassed the five key dimensions of DTs: physical component, virtual component, data, connectivity, and service modeling. Consequently, there is a significant need for more generic and comprehensive modeling methodologies and processes since modeling is the core of DTs [Tao+18].

3.2.1. Asset Administration Shell

In the context of digitalization in Industry 4.0, the Asset Administration Shell (AAS) [Mar+18] has been proposed. It consists of a set of submodels that contain information about a given asset, providing a virtual representation of this object. Figure 3.2 shows the AAS structure from a high-level point of view.

It is possible to see that the AAS also uses the idea of model composition (in the body section) for representing an industrial asset. Each model is composed of a set of properties that can be monitored from the real world. This is a common point with the thesis, since, as explained in the DT chapter, its concept definition is based on the combination of different models that represent an observable asset.

The AAS is mainly used in the manufactory context. The autors of [TA17] propose combining World Wide Web Consortium (W3C) specifications with Plattform Industry 4.0 guidelines to standardize communication interfaces and enhance CPS interoperability using asset administration shell. This approach aims to achieve interoperability between

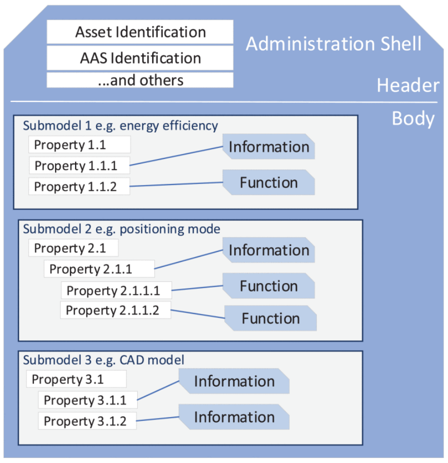


Figure 3.2.: AAS structure from a high-level point of view [CS20]

Industry 4.0 components and the IoT. The paper demonstrates the application of this structure in a use case involving the adaptation and remote maintenance of a production robot. In this research, stakeholders with different backgrounds and knowledge were not considered, which is a common fact when talking about system integration and the new industrial revolution.

There are technologies that support the implementation of AASes. Motivated by the large amount of computational resources AAS needs to be implemented, the paper [PBD21] presents a review of open-source solutions available in the literature and market, introducing a methodology on how to combine AAS and OPC-UA for achieving global communication and semantic interoperability. The authors claim that their approach can significantly reduce the resources needed for this implementation, enabling it to run on some embedded devices.

AAS has a similar idea as the one provided by this thesis; however, the proposed approach aims to be generic enough to allow designers to model any part of the system, from the physical devices up to the processes and workflows, using the same modeling language. Therefore, it might not offer the flexibility needed to seamlessly connect with varied external stakeholders, who may use heterogeneous systems, protocols, and vocabularies, potentially hindering interoperability and collaboration across different ecosystems.

When implementing complex systems, with many components and relationships between them, it is important to organize the system's parts by responsibilities. This is usually done via a definition of an architecture. The next section will present the main related works that discuss the use and definition of architectures in IoT and DT.

3.3. IoT and DT architectures

IoT architecture encompasses an array of components including physical objects, sensors, actuators, and developers, along with cloud services, various communication layers, user interfaces, business frameworks, and specific IoT protocols. Due to the extensive range of objects connected to the internet, there is no universally accepted standard for IoT architecture. Various researchers have proposed different architectural frameworks, reflecting the diverse nature of IoT applications [Jab+20].

Some of the most common IoT architectures are composed of 3 and 5 layers. Figure 3.4 shows one example of these common architectures.

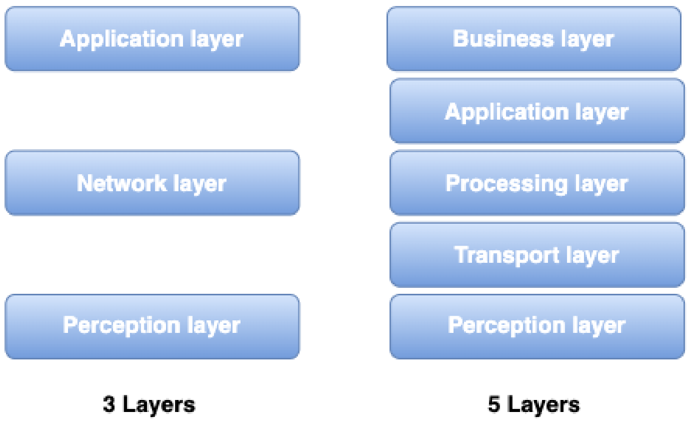


Figure 3.3.: 3 and 5 layers IoT architectures [CS20]

The 2 approaches differ basically only on the level of detail. The 3-layered one contains only the main parts of an IoT system whereas the 5-layered one divides the application layer into processing, application and business layers.

Architectures with 4 layers are also common in the IoT context. The authors of [SRP20] proposed a 4-layered architecture based on blockchain technology. The main goal was to enable a secure and distributed IA system. In this work, the four layers are Device intelligence, Edge Intelligence, Fog intelligence and Cloud intelligence. This approach can be reused in other works where distributed systems are needed, for example, distributed DTs.

In the industrial context (IIoT), architectures are also defined focusing on connecting industrial equipment for data acquisition, exchange and analytics. The research of [Qiu+20] highlights the significant role of edge computing in IIoT, particularly in reducing decision-making latency, conserving bandwidth, and enhancing privacy. The paper discusses the advantages of edge computing in IIoT and suggests a prospective architectural framework for the future. The proposed architecture is divided into 3 layers: The device layer (containing the physical devices), the Edge layer (that enables the cyber and physical to interact) and the Cloud Application Layer (that contains the applications).

Several initiatives to standardize DTs have already been proposed [Wan+22]. The Reference Architectural Model Industrie 4.0 (RAMI 4.0) [HR15] provides a three-dimensional layer model that describes aspects of the fourth industrial revolution. Some aspects of this standard are commonly shared with the proposed approach in this thesis such as the six layers to describe the composition of a machine as well as the use of semantics to enable interoperability between cross-vendor assets.

Based on the RAMI 4.0, the work [Ahe+21] proposes a concept of Digital Twin as a Service (DTaaS). The Digital Twin reference architecture model presented in this context is a multi-layered, three-dimensional structure, simplifying complex interrelations into more manageable segments. This model leverages the DIKW (Data, Information, Knowledge, Wisdom) matrix but can bypass its linear hierarchy if necessary, using smart sensors, IoT, Big Data, and cloud computing to achieve autonomous actions. This approach is demonstrated in the application of DTaaS for predictive maintenance, where an API is used to integrate DTaaS outcomes directly.

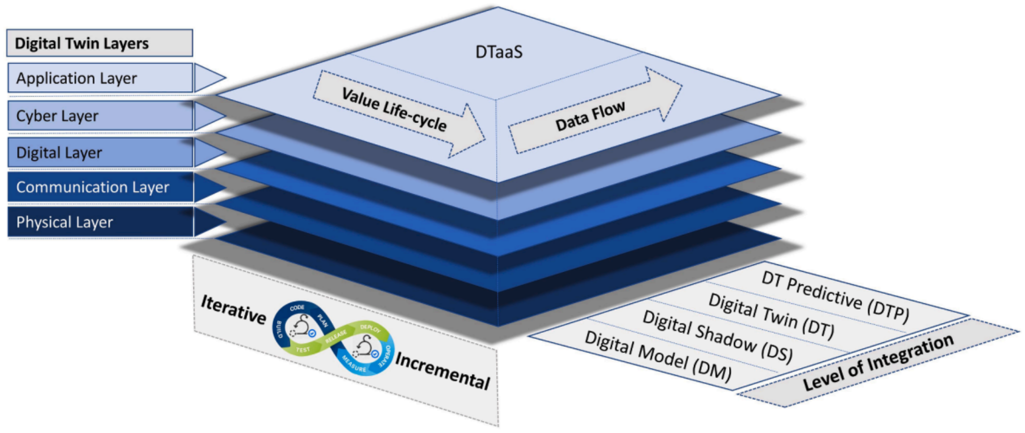


Figure 3.4.: A Digital Twin Reference Architecture Model in Industry 4.0 [Ahe+21]

An expansion of the original definition of DT from the manufacturing context has been presented in [MLC20] including developments in augmented and virtual reality, multiagent systems, and virtualization. This analysis identifies an extensive set of DT features, emphasizing the 'softwarization' of physical objects. The article proposes foundational properties to establish a shared, consolidated definition of DT, focusing on its essential characteristics. It then explores the technical and business value of DT, discussing its applicability and opportunities in different scenarios. The article illustrates this through four application scenarios, demonstrating the practical use of DT. These scenarios also provided a generic DT architectural model that is based on software architecture models and guidelines.

Another important aspect when implementing DT and IoT applications is the supporting technologies that help users realize their solutions. The next chapter presents related works in this context.

3.4. Digital Twin and IoT technologies

For developing DT and IoT applications, several technologies have been proposed by private companies and open-source communities to facilitate the development of IoT applications. New methods, tools and technologies have been proposed private companies and open-source communities to facilitate the development of IoT applications. Microsoft proposes Azure Digital Twin [Mic21]. It is a platform that enables users to create IoT applications and modeling elements of the system and their relationships. They provide a modeling language called *Digital Twins Definition Language* (DTDL) which is used to import/export models. This platform also supports the monitoring of data streams coming from the physical world into the cyber world, as well as it has a querying feature to run against the model.

Another possible option available in the market is the AWS IoT TwinMaker [Ama22]. It offers features to integrate 3D models to the platform which can be useful for use cases like smart buildings, production lines, factories, and other use cases this kind of visualization is helpful.

Node-RED [Nod22] is also a well-known platform in the IoT area. It provides a high-level programming tool (or so-called low-code) that allows connecting hardware devices, APIs, services and applications. It provides a graphical user interface with nodes that can perform a specific task. It is possible to connect nodes to each other, allowing them to exchange messages. Flows can also be built to orchestrate parts of the system.

Additionally, Node-RED can run on devices such as a Raspberry Pi, which allows it to work directly on edge devices. In this context, the same technology, standard and methodology can be used for edge devices up to the high level of the whole system. However, Node-RED has limitations regarding adding semantics to the model. This feature is important when it comes to analyzing and extracting information from the model and avoiding applications to work in silos.

Several supporting approaches and tools have been provided for enabling DTs [Qi+21]. Figure 3.5 shows some of the available technologies in this context, divided into four categories of models: Geometric model, Physical model, Rule model, and Behavioral model.

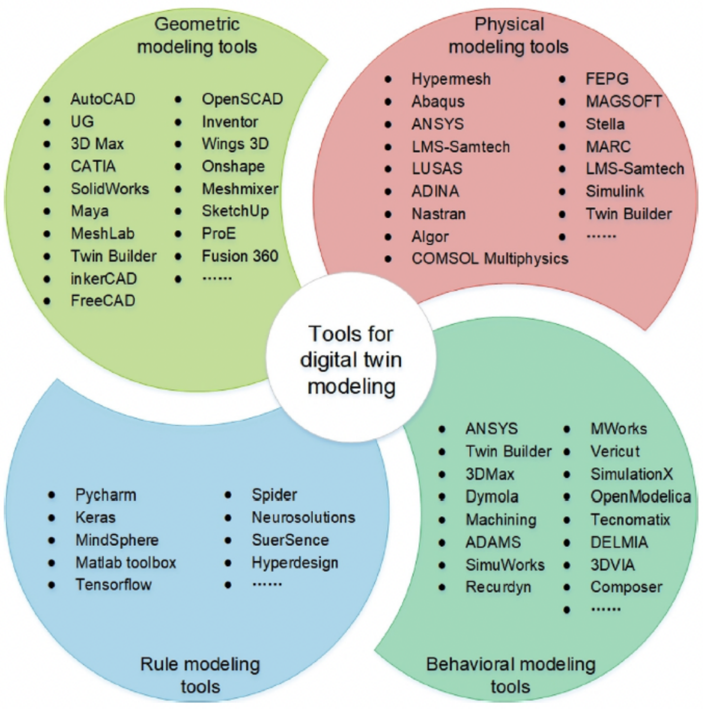


Figure 3.5.: Enabling technologies for DTs [Qi+21]

While it covers the technological and tool-based enablers extensively, a more thorough exploration of how these technologies handle the semantics of the data they process would provide a more holistic understanding of the challenges and solutions in the field of DTs. Therefore, a methodology for guiding users can be necessary to create a standardized model.

3.5. Digital Twin methodologies

Methodologies are developed for several reasons such as providing structured frameworks for tackling complex tasks, ensuring standardization, consistency, and the incorporation of best practices across different projects. They play a crucial role in risk management, and quality control, and facilitate efficient knowledge transfer. Additionally, methodologies are adaptable since they can continuously evolve based on feedback and changing needs. This evolution can bring innovation and effective problem-solving within a defined and efficient framework.

Implementing DTs can also be a complex task, and, therefore not only requires methodologies to support the implementation but also need that these methodologies are continuously updated and improved. Several methodologies have already been developed in the context of DT, as can be seen in [PM23].

A flexible and generic methodology, based on model-driven engineering, for designing DTs is proposed in [Sch+20]. The approach is divided into two steps: first, a DT is defined as a composition of components that provide basic functionalities like storage, communication, identification, data management, HMI, simulation and security. Then, it is possible to define the DTs by aggregating them with other DTs.

While the work of [Sch+20] and [Sch+16] uses AutomationML to implement their methodology, this thesis extends this methodology and adds generic modeling elements to define the structure of the digital representations.

Considering the need for flexibility, the authors of [Sto+21] explore flexible management methods to handle this complexity efficiently. It discusses the challenges in engineering and managing DTs and proposes conceptualizing methods and tools supporting the entire DT lifecycle. They introduce the FA3ST toolkit, a generic and flexible architecture for DT management. This toolkit is designed to integrate with existing DT frameworks and Industrie 4.0 standards like AAS, OPC UA, and AutomationML. Additionally, the paper highlights the need for DT ecosystems to support sustainability and resilience in supply chain networks, suggesting future DT architectures and tools should align with initiatives like GAIA-X to maintain control over DT usage.

In [Qam+21] the authors introduce a methodology for DT development in manufacturing, grounded in the System Development Life Cycle (SDLC) process. The object-oriented paradigm is used to provide capabilities such as scalability and extensibility. Similarly to [Sch+20], the authors also use the approach of having a first step to define the devices of the physical world and a second step to aggregate this representation with others, forming a combined twin.

It is possible to identify that there are several approaches proposing methods for implementing DTs. However, most of the approaches are isolated and use-case-specific, making it difficult to extend or use in different domains. For enabling different domains to interact and exchange information in a clear way, the use of semantics plays a crucial role. The next section will present related works that use semantics in their approaches.

3.6. Semantic models and data specification

Semantics in designing models is important because it brings clarity, precision, and context into the representation of a system. It helps to define clear meanings for system elements and their relationships. Semantics ensures a common understanding among all stakeholders. This is also applicable when mapping real-world applications into system models [BR23].

Using semantics can also facilitate the integration of different applications and data sources. The research of [Jia+23] explores the application of DTs and web semantics for controlling building fire protection (BFP) systems. It highlights the use of data fusion and various control mechanisms derived from these technologies. A specifically designed BFP ontology serves as the semantic model for fusing static building geometric information with dynamic sensor data. This information is integrated into a DT data model, representing a virtual mapping of the physical space. To ensure intelligent control and synchronization between the DT data model and the physical environment, rule and process models are developed.

The authors of [Yan+23] introduce a meta-model-based method for constructing shop-floor digital twins, using an architecture based on RAMI 4.0. This meta-model offers new ways to describe manufacturing resources and their status. The proposed shop-floor DT architecture involves three core components: 1- constructing the meta-model; 2- data modeling (including data interactions between cyber and physical spaces), and; 3- developing various integration-level models for the shop-floor DT, guided by iterative feedback between model demands and development.

Ontologies can also be used to model and represent knowledge. The authors of [Göp+21] explore the use of ontologies for specifying industrial components in the manufacturing domain. The paper provides a pipeline for modeling and deploying digital twins based on ontologies. This approach goes in a similar direction as the present thesis, however, it does not deal with a fully connected model with the real application. Additionally, the work provides a solution for a specific use case (manufacturing), while this thesis proposes a generic approach that aims to combine different assets' perspectives (domain-specific models) into one single model.

Numerous studies have explored the incorporation of semantics in DT models, yet these tend to be domain-specific and lack a standardized methodology applicable across various domains. This limitation presents a significant challenge for future applications aiming to interconnect diverse applications. Consequently, there remains a pressing need for innovative approaches, particularly in terms of facilitating integration among stakeholders from varied backgrounds and with distinct knowledge bases.

3.7. Discussion of the state of the art

The state-of-the-art is rich in diversity and innovative ideas regarding the topic of the digital twin, its design and implementation. It is possible to see, that there is a natural evolution from IoT concept, architecture and applications towards the concept of digital twin and the use of semantics for creating these virtual replicas. Similar to IoT, the DT is applied in a variety of areas, and, therefore, touches a heterogeneous group of users and stakeholders.

These varied applications also bring forth challenges in interoperability and data integration, emphasizing the need for standardized, semantic-rich approaches. The thesis by Aparna, which proposes a semantic extension to Node-RED for IoT applications, particularly in industrial automation, aligns well with these challenges, offering insights into the importance of semantic interoperability and ease of application development across diverse IoT ecosystems.

Still in the industry 4.0 context, the AAS is a well-known standard and, as mentioned before, it has several similarities to this work. However, as it is specifically designed for industry 4.0 applications, it can be complex for users without deep knowledge of the standard to realize their models.

Furthermore, the majority of existing approaches tend to focus on a single facet of Digital Twins' conceptualization, be it an ontology, an architecture, or a methodology. None of the reviewed works present an integrated approach that is generic to support different kinds of applications from various domains.

The table 3.1 shows some of the related works and their similarities to the present thesis.

In summary, the state-of-the-art works collectively underscore the imperative of semantic integration and standardized methodologies in IoT and DT realms. These findings lay a foundational understanding for this thesis, guiding the proposed approach towards addressing the highlighted gaps and challenges in the field.

In this context, this proposal aims to provide a generic approach for modeling and deploying DTs. It provides an architecture that helps to organize the system, a modeling language that enables semantic models, and a methodology that guides users in the process of modeling DTs. The target of this approach is generic systems with different stakeholders. As standards, it uses the ISO 23247 and the WoT standard, that can be

Research	Type	Target	Standard	External Models	Extendable	Connected to real-world
[Thu22]	architecture, method, use case	industrial	iot.schema .org	not directly	yes	yes
[Göp+21]	method, use case	manufacturing	ontology	yes	yes	no
[Sch+20]	architecture, method, use case	industry 4.0	AML	links, not data propagation	yes	No
[SG22]	methodology, architecture	generic	not covered	not covered	yes	yes
[GSH20]	architecture	industry 4.0	RAMI4.0	not covered	yes	yes
[Bic+20]	methodology, simulation	Naval unmanned system, maintenance	not mentioned	not mentioned	domain-specific	yes
[Hug+20]	methodology	generic	AADL/C code generation	not covered	yes	yes
[Kal+21]	methodology, semantics	logistics	not mentioned	not covered	yes	no
[DO21]	architecture	manufacturing	ontology	yes	yes	not mentioned
[Bor+21]	architecture	Healthcare	not mentioned	yes	yes	yes
[Gou+23]	architecture	energy	not mentioned	no	domain specific	yes
[Rod+23]	methodology	drone applications	communication standards	no	domain specific	yes
This work	architecture, modeling elements, methodology	generic, different stakeholders	ISO 23247, WoT	yes	yes	yes

Table 3.1.: Comparison of related works with the present thesis

used in different domains. This approach also covers the possibility of connecting the model to the real-world system, meaning that the model contains the latest state of the physical counterpart. This also means that the model is connected to the physical world, and, therefore, it is possible to make precise queries and inferences, since the current state of the application will also be semantically reflected in the model. Finally, the approach is extendable, meaning that it can be used in different use cases, not only in the Industry 4.0 context.

The next sections will present the open challenges and main contributions of this thesis, aiming to fill the lacks identified in the literature.

3.8. Open challenges

In the literature, there are several identified open challenges in the context of implementing digital twins. The main challenges pertinent to this thesis are described below.

- **Need for standards:** One of the primary challenges is the lack of standardized methodologies that can be applied across various domains.
- **Generic way of modeling:** There is a need for a generic way of modeling within the context of digital twins, the focus is on creating a framework that can effectively integrate domain-specific models. This integration is crucial for extracting and synthesizing knowledge from various stakeholders, ensuring a comprehensive understanding of the system or process being modeled. Such a framework should also provide a structured approach for feeding machine learning algorithms with the necessary data. This structured approach is essential for enhancing the effectiveness of ML algorithms, as it enables them to learn from a diverse range of inputs and apply this learning to improve the accuracy and efficiency of the digital twin. By tackling these aspects, the generic modeling approach should facilitate a more versatile and adaptive DT, capable of handling complex, multi-faceted scenarios.
- **Methods based on the ISO 23247:** The ISO 23247 aims to standardize the concept of DT in the manufacturing domain, however, it can be reused in different domains. Methods and approaches that easier its implementation are needed to bring it to the real world.
- **Semantically connect the model with the real system:** Semantic models are crucial to bringing interoperability between different entities. However, they often get out-of-date. Additionally, connecting them to the real system can enable more

precise queries and inferences, since the current state of the application will also be semantically reflected in the model.

- **User-Friendly approaches:** Many good approaches can be proposed, but if they are not user-friendly and close to the end users, they will probably remain on paper and their potential will be hidden. Therefore, it is important that new approaches also consider their ease and efficient use.

While frameworks such as Azure DT and AWS TwinMaker provide powerful platforms for specific use cases, they face challenges related to semantic modeling and cross-domain interoperability. These platforms often struggle to integrate seamlessly with heterogeneous systems, particularly at the edge, where resource constraints limit their effectiveness. To address these challenges, the proposed framework leverages ISO 23247 and WoT, providing a universally applicable semantic model that can bridge different application domains and reduce vendor lock-in.

Similarly, tools like Node-RED excel in accessibility but lack comprehensive semantic support, restricting their use for complex digital twin systems. By integrating semantic models directly into a low-code platform, the proposed approach retains the ease of use for non-experts while significantly enhancing the capacity for semantic interoperability, online updates, and cross-domain application.

An overview of the proposed approach, relating these contributions, can be found the the next section.

3.9. Overview of the proposed approach

This chapter aims to provide an overview of the proposed approach. Figure 3.6 illustrates the main steps of the methodology placed next to the architecture in the corresponding layers. Also, some examples of technical components are shown on the right side.

At the foundational Observe layer (Section 4.1), the physical realm is captured through the identification of observable assets (Section 6.1), which are crucial for model creation. Typically, this encompasses devices, sensors, and actuators.

Transitioning to the Collect layer (Section 4.2), the bridge between the tangible and digital worlds is constructed. Within this sphere, the communication interfaces are established (Section 6.6), accommodating elements like databases and communication protocols.

The Model layer (Section 4.3), which is the centerpiece of this dissertation, is dedicated to the formulation of the DT model (6.4) using the basic elements proposed in Chapter

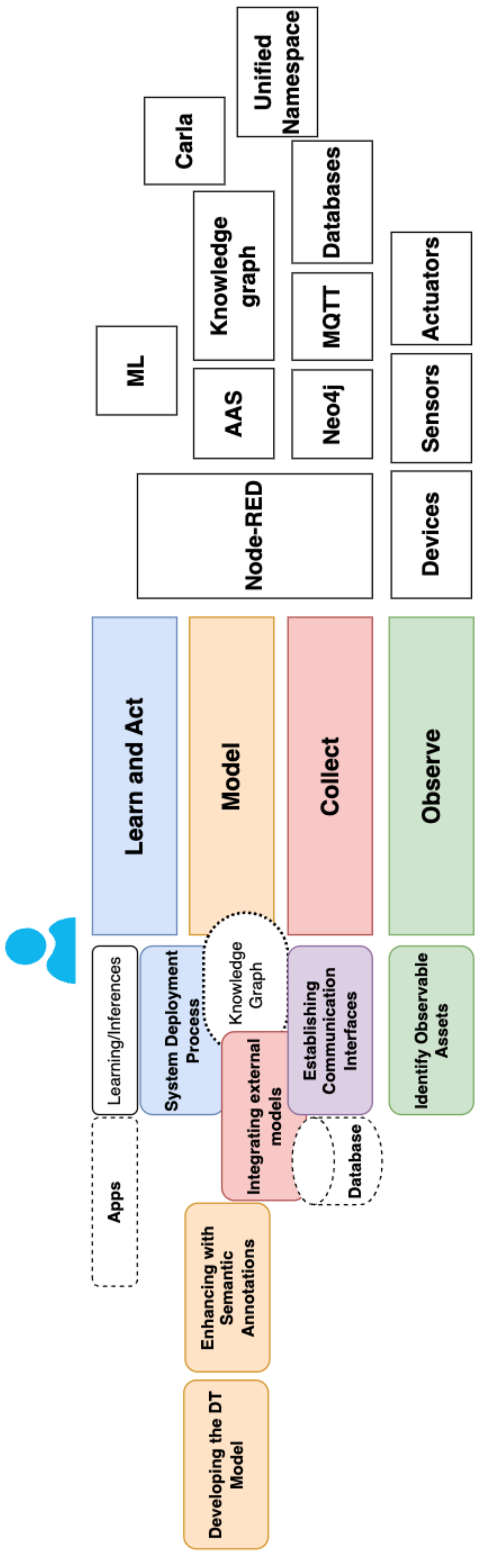


Figure 3.6.: Schematic representation of the proposed architecture with methodology steps and examples of technical components

5. This stage is crucial for semantic enhancement (6.3) and integration with external models (6.5). The output of this phase is the generation of a Knowledge Graph that systematically organizes assets and elucidates their interconnections. The foundational elements proposed herein are applied to construct the model, and a Node-RED-based implementation is proffered to seamlessly integrate the model with its physical counterparts.

The final layer is Learn and act (Section 4.4), designated for interaction, is the domain where applications and machine learning frameworks are sculpted to engage with the user interface. These components translate the complex data and processes of the lower layers into actionable insights and user-responsive actions.

This thesis proposed a harmonized approach that seamlessly integrates architecture, modeling elements, and methodology into a unified integration environment. This fusion facilitates system designers in creating their virtual representations, providing comprehensive support across various domains of the system design.

In this context, the next chapter describes the main contributions of this thesis, trying to fill the lacks identified in the literature.

3.10. Contributions

As identified in the literature review, there are still gaps to be addressed in the process of modeling and deploying Digital Twins. Especially in the era of Large Language Models (LLM), that are able to connect and understand different sources of information, it is important to have a common language and methodology that different stakeholders can use to express their knowledge.

Most of the existing approaches are isolated and use-case-specific, making it difficult to extend or use in different domains. For enabling different domains to interact and exchange information in a clear way, the use of semantics plays a crucial role. This thesis aims to provide a generic approach for modeling and deploying DTs, based on the ISO 23247, that can be used in different domains. This approach is based on a four-layered architecture, a set of basic elements for modeling DTs, and a methodology for guiding designers in the process of modeling DTs. The main contributions of this thesis are:

- **Architecture for DT-based systems:** A four-layered architecture based on ISO 23247 has been defined to identify the responsibilities of each part of the system [Ste+21b] [Ste+21a]. The architecture is presented in the Chapter 4.

- **Basic elements for modeling DTs:** These basic elements were defined to provide a standard way of modeling DTs. These components are generic and can be used to model any DT [Ste+22b] [Ste+22a]. The elements are presented in the Chapter 5.
- **Methodology for modeling DTs:** steps were defined to guide designers in the process of modeling DTs [Ste+20] [Sch+21b] [Ste+22a] [Ste+23]. The methodology is presented in the Chapter 6.

Additionally, the semantic models that are combined to work as a virtual replica of a real-world asset are also connected to the real-world system, meaning that the model contains the latest state of the physical counterpart.

3.11. Conclusion of the chapter

This chapter has presented the current state of the art by introducing the most related works. As DT is based on the IoT concept, an introduction to the basis has been done. Then applications and models that are already DT-specific were presented and discussed.

Aiming to understand how these kinds of applications are built, a review of architectures for both IoT and DT was carried out. It was possible to identify that most approaches work with 3, 4 or 5 layers, varying the details of each use case. To realize these applications, several tools and applications can support designers and engineers.

It is possible to identify the path taken from IoT and CPS, passing to DT and reaching the semantic DT, aiming to enhance interoperability and enabling more seamless communication across diverse systems and domains. This shift marks a significant stride towards more intelligent and unified system integration.

This chapter sets the basis for the upcoming sections of the thesis, which aim to contribute to this evolving field by proposing novel solutions and methodologies that address these identified gaps and challenges, ultimately pushing the boundaries of what is currently achievable in DT design and implementation.

4. Architecture for DT-based systems

One of the contributions of this thesis is an architecture that can be used to design DT-based systems. An architecture can serve as a comprehensive blueprint of the system by highlighting its components and relationships as well as separating them by responsibilities. It is an essential part of guiding the development team and helps to identify possible problems in the early stages of development.

An architecture can also help to enhance the scalability and flexibility of the system since each component is classified by its responsibilities, it is possible to replace or modify specific elements of the system without requiring a complete reimplementation. In the DT context, it is particularly important due to the variety of heterogeneous systems and the amount of data they can generate. For example, if the system needs more storage capabilities, the components responsible for that can be upgraded and the rest of the system remain working normally. In other words, architectures are an important peace for keeping the system maintainable.

In this context, this thesis proposes an architecture that is inspired by ISO 23247 [ISO20] and defines the main 4 layers for DT applications. This ISO also defines a set of assets types that might exist in systems: Equipment, Personnel, Material, Process, Facility, Environment, Supporting documents, and Product. A so-called Generic type has been added to the proposed concept in case none of the previous ones fit the asset.

The ISO 23247 is one of the first standardization initiatives in the context of DTs. It is mainly meant to be used in the manufacturing domain, but it is also generic enough to be adapted in other areas.

Figure 4.1 illustrates the proposed architecture.

- **Observe**: this layer is composed of the real-world elements that will be observed and might also have acting capabilities.

- **Collect**: in this layer, all communication details are defined. This layer enables the IoT and connects all elements of the real world with the cyber part of the system.

- **Model**: assets, their characteristics, and relationships are defined in this layer. This virtual representation is connected to its corresponding asset from the real-world and provides a high-level instance to the upper layer of the architecture.

- **Learn & Act**: interfaces that allow interaction with users, machine learning algorithms, simulations, reports, and so on are defined in this layer.

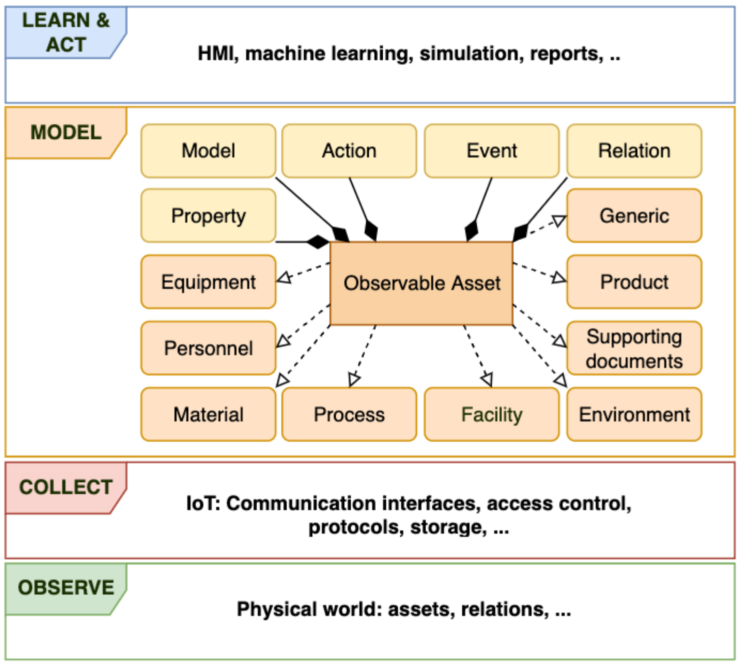


Figure 4.1.: Architecture for DT-based systems

Data flows from the lower layers up to the top feeding models, applications, and enabling learning and monitoring mechanisms. Meanwhile, commands and adjustments are sent from the top layers down to the bottom layers where the real world is contained.

This architecture can also be compared to layers of the RAMI4.0 reference architecture model [HR15]. **Observe** layer is equivalent to the *Asset* layer of RAMI, since they represent the real-world elements of the system. The layers **Collect** and **Model** can be related to the three layers *Integration*, *Communication*, and *Information*, since they are responsible of providing virtual representations of the assets as well as dealing with communication and storage capabilities. Finally, the **Learn & Act** layers can be compared to the *Functional* and *Business* layers.

The proposed approach, however, provides modeling elements that can be used throughout all layers of the system, enabling different stakeholders to use the same language and avoiding misunderstandings or incompatibilities.

In the next sections, each layer of the architecture will be detailed.

4.1. Observe layer

The Observe layer forms the foundational level of the (DT) architecture, representing the real world in its entirety, including physical devices, environment, processes, stakeholders and any other instance that can be observed.

For instance, in a manufacturing setting, this layer would include sensors and monitoring equipment attached to machinery, capturing data on their operational status, environmental conditions, and other critical parameters.

Assets can be composed of other assets. For example, an asset **car** is composed of several assets such as **engine**, **brake system**, **infotainment system** and so on. Environmental variables such as weather forecast, road condition or current driver (user) are common examples of assets that can be related to the asset car. Actions such as braking, turning or driving could be represented as processes that are not physical (hardware) but can be observed and are crucial parts of such use cases.

The composition of assets allows to define systems with different levels of granularity. From the device level up to processes and management levels that touch the end users.

4.2. Collect layer

The Collect layer primarily deals with the connection between the cyber and physical worlds, acting as a bridge that facilitates the flow of data from the physical assets to the digital platform. In this layer, technologies such as IoT devices, edge computing, and communication protocols play a vital role. For example, in a smart city DT, the Collect layer would involve the aggregation of data from various sources like traffic cameras, weather stations, and pedestrian footfall sensors. The primary function of this layer is to ensure that the data collected from the physical world is transmitted effectively, securely, and efficiently to the digital environment, where it can be further processed and analyzed.

Communication interfaces, access control, protocols and storage are some of the key components in this layer. They enable data from the physical world can be transmitted to the cyber world while commands and recommendations can be sent back to the real assets.

This layer is crucial because it ensures that the digital replica is continuously fed with up-to-date and accurate information from the physical world, forming the basis for all subsequent analysis and decision-making processes within the DT.

4.3. Model layer

The Model layer is the core of the DT architecture, where the actual Digital Twin is created and maintained. This layer involves the integration of external models, addition of semantic information, and the construction of a detailed digital replica. Here, various modeling techniques, such as geometrical models for physical structures or process simulation models for industrial workflows, are integrated. Additionally, semantic technologies like ontologies and knowledge graphs are used to add context and meaning to the data, enhancing the interpretability and usefulness of the DT.

One example in the industrial manufactory domain could be a CAD model of a robotic assembly line, where each robot, conveyor belt, and tool is precisely represented in the digital space. This digital representation is augmented with real-time operational data, like machine speed, temperature, vibration levels, and output rates, allowing for a dynamic and accurate reflection of the physical assets.

4.4. Learn and act

The Learn and Act layer hosts applications and user interfaces that enable stakeholders to interact with the DT, extract insights, and make informed decisions. These applications can also present to users results and reports generated from the systems or AI models and allow them to generate commands that will be sent to the physical counterpart.

One application within this layer could be a predictive maintenance tool that continuously learns from data related to engine performance, tire wear, brake conditions, and other critical components. The DT can predict when parts are likely to fail or require service. This predictive analysis enables car owners and service centers to proactively address maintenance needs, thereby reducing the likelihood of breakdowns and extending their product's lifetime.

4.5. Conclusion of the chapter

This chapter has presented the proposed architecture for implementing DT applications, delineating its four distinct layers: Observe, Collect, Model, and Learn and Act. Each layer plays a critical role in the seamless functioning and efficacy of the DT, with the architecture designed to ensure robust data collection, efficient processing, and intelligent response mechanisms. The Model layer, the focal point of this thesis, stands as the key element in this structure. It's here that the DT is semantically modeled and brought to life, bridging the gap between raw data and actionable insights.

Many existing IoT architectures are structured in 3, 4, or 5 layers, depending on the level of granularity they aim to provide. The most common examples include architectures that focus on the basic components—physical devices, communication protocols, data processing, and business logic. These architectures have been instrumental in various applications of IoT, but they often lack a clear connection to the specific needs of DT systems, where a precise reflection of real-world assets is essential.

The novelty of this thesis lies in the integration of ISO 23247 elements to define assets and WoT elements to describe assets and their relationships within the context of a four-layer architecture designed specifically for DT applications. This architecture is a solution where the Observe and Model layers are representations of each other, providing a robust structure for online synchronization between the physical and digital worlds.

In the context of DT, this architecture also enables that the Model layer goes beyond merely representing physical assets by enabling the link to simulation environments, predictions, and even combinations of properties that do not exist in the current state of the physical system. In some cases, the Model layer may contain more detailed or enriched data than the Observe layer, due to its ability to simulate future events or infer missing properties or relationships through advanced algorithms and data processing techniques. This approach enhances the capacity of DT systems to not only reflect but also anticipate real-world changes, ensuring that the DT can act as both a mirror and a predictor of the system it represents.

The architecture outlined in this chapter lays the groundwork for a comprehensive understanding of how Digital Twins can be efficiently structured and operated. It emphasizes the importance of a layered approach, ensuring that each aspect of a DT, from data acquisition to user interaction, is considered for performance and scalability. This is particularly crucial in the context of rapidly evolving technologies and the increasing complexity of systems that DTs are expected to replicate and interact with.

The next chapter will dive into the basic modeling elements that are central to defining the structure of DTs. This exploration will include an in-depth discussion of the modeling language developed specifically for this purpose, highlighting how it complements the architecture. The forthcoming chapter aims not only to elucidate the technical specifics of these modeling elements but also to showcase their practical applications and the value they add to the task of creating digital twins.

5. Basic elements for modeling Digital Twins

The advent of Digital Twins in the Web of Things [ZGC11] landscape marks a significant evolution in how we understand and interact with physical systems through their digital counterparts. At the heart of this evolution is the necessity for a semantic modeling approach, which ensures that digital twins not only replicate physical entities but also meaningfully interpret and utilize the data associated with them. This semantic layer is vital for bridging the gap between the raw data collected from various sources and the actionable insights required for effective decision-making [Kal+20]. It enhances the DT's ability to not only represent physical entities but to also understand their interactions, dependencies, and behaviors in a context-rich environment.

In this exploration of Digital Twins, the focus is placed on how semantic modeling can be effectively applied using the Web of Things concepts. This approach defines basic elements for modeling the structure of digital twins. Figure 5.1 shows the main elements that cover the concept of DT.

A DT is a combination of models that represents an observable asset from the real-world (device, process, or anything that can be observed) in different perspectives (behavior, geometry -3D models-, ...), is continuously updated and can interact with the real world and digital world. It can enable simulations, and learning mechanisms and track the whole life-cycle of an asset.

5.1. Asset

In the context of digital twins, assets ("dt-asset") are key elements that mirror observable entities from the real world [Tao+18]. They can be constituted from other assets, creating a combined and/or hierarchical and interconnected structure. For instance, consider the asset "valve," which might be linked to another asset, "pipe." These individual assets can then collectively form a part of a larger asset, such as a "refinery." This composition is represented through a "composition link". The possibility of combining assets allows designers to model systems with different levels of granularity.

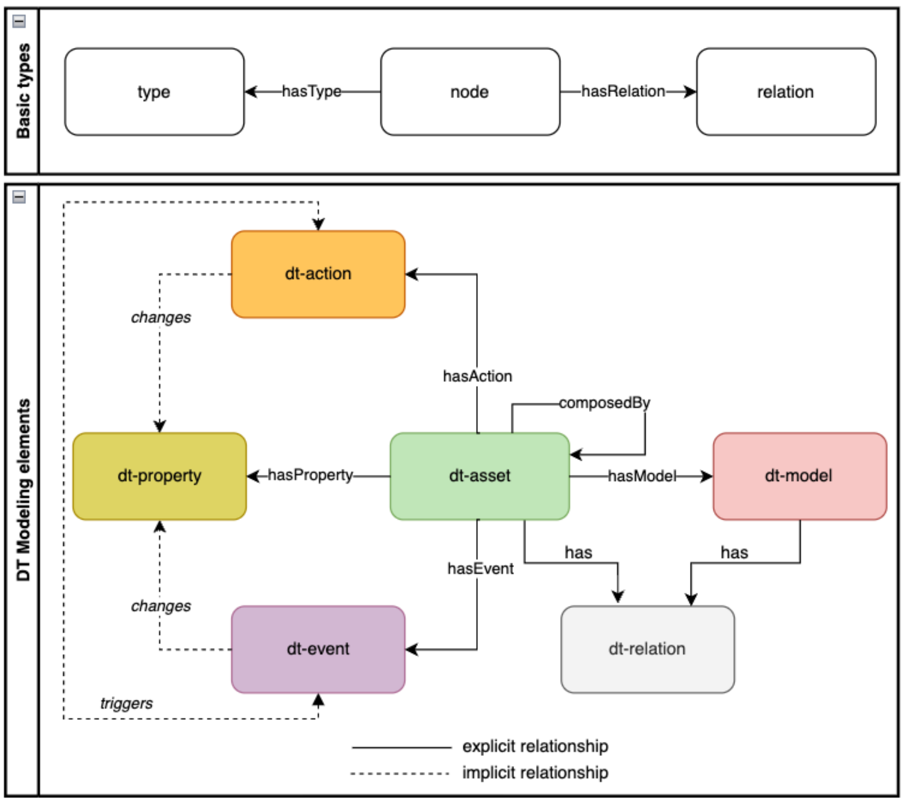


Figure 5.1.: Basic modeling elements

Each asset can encapsulate metadata that can enrich the data related to this entity. This includes the asset's name, context, identification, and type. The type category is particularly versatile, encompassing options like "Generic" or other classifications defined in the ISO 23247 standards [ISO20]. These standards include varied asset types such as parts, products, equipment, materials, processes, facilities, environments, personnel, and supporting documents. Such a structured approach ensures that every asset is comprehensively represented and accurately linked within the digital twin environment.

Assets can be used to also represent non-physical entities, including processes, workflows, external software, and similar elements.

Relationships between assets are expressed using semantic nodes termed "dt-relation". These nodes serve to connect different assets and nodes to semantically manifest their associations.

5.2. Property

As assets' properties are usually linked to a variable from the real world and can carry several metadata, an extra node to represent them is needed. These properties are continuously monitored via sensor readings and define the asset's state. For instance, an asset "valve" has a temperature sensor that provides the temperature variable. The temperature property could be enriched with metadata such as measurement units (e.g., Celsius or Fahrenheit), precision level, timestamp of the reading, and the sensor's location.

This node also allows the inclusion of a *context* metadata, which can be used to define the property's datatype. Metadata plays a vital role in the system such as avoiding mistaken conversion in the subsequent layers of the system (e.g., user interface or ML algorithms), or that parts of the system work in silos.

Data can be annotated as soon as it enters the cyber world, avoiding misinterpretations at later levels of the system. It also allows system designers to restrict data usage and privileges needed to access it in the other parts of the system. For instance, the temperature can be accessed by ML models to estimate the device's health condition. However, user-related data such as email, messages or identifiers should be annotated as sensible data, not allowing it to be sent to external systems.

5.3. Action

Assets within a digital twin concept can possess Actions (dt-action), also known as methods. These actions have the capacity to transmit commands from the digital world to the physical world. This functionality can result in changes in the state of a device. A classic example is a lamp, which might have actions such as *on()*, *off()*, *setColor()*, or *setBrightnessLevel()*. These actions enable remote control of the lamp's various states and features from the digital interface.

In addition to influencing physical devices, actions in assets can also be designed to initiate internal procedures within the digital model itself. These procedures operate within the boundaries of the digital environment and do not directly impact the physical counterpart of the asset.

Such internal actions are crucial for the maintenance and management of the digital twin system. They may include tasks like updating system logs, recalibrating virtual models, or running diagnostic checks. These actions ensure that the digital twin remains accurate, efficient, and reflective of its physical counterpart, even though they don't manifest any direct physical changes.

5.4. Event

Events (dt-events) are a significant component of digital twin, functioning as triggers or indicators of specific occurrences or conditions. These events can be internal, originating within the digital model, or external, stemming from changes or conditions in the physical world. Internal events might be generated by mechanisms like machine learning algorithms within the model. For instance, a predictive maintenance algorithm might determine a new maintenance date for a piece of equipment, signaling this as an event. Metadata can be added to events, such as its duration, frequency, timestamps, or the conditions that trigger it.

External events, on the other hand, are often linked to real-world changes or thresholds being met or exceeded. A typical example of this could be an asset reaching a critical temperature level. This kind of event is usually detected by sensors and then communicated to the digital twin, where it is processed and logged. External events are crucial for real-time monitoring and response, allowing the digital twin to reflect current conditions accurately and promptly. For handling the challenge of communication delays between the sensor readings and the digital twin, the event can be annotated with a timestamp, indicating when the event occurred in the real world.

The interaction between events and actions within a digital twin is dynamic and interconnected. When an event occurs, it can trigger specific actions related to that event. For instance, if an event signifies that a machine's temperature has exceeded safe operating limits, it might trigger an action to shut down the machine or activate a cooling system. Similarly, events can lead to changes in the properties of assets, like adjusting the status or operational parameters. This interconnectedness ensures that the digital twin responds appropriately to both internal predictions and external realities, maintaining the system's integrity and functionality.

5.5. Model

In the domain of digital twins, the concept of a 'Model' (dt-model) plays a central role in representing assets. An asset within a digital twin environment can be depicted through various models, each offering a different perspective or aspect of the asset. This multi-model approach allows system designers to create a comprehensive and multifaceted digital representation of each asset. For example, a single asset could be represented by a 3D model that visualizes its physical structure and geometry [Dan+18]. Simultaneously, it could also have a requirements model that details the expected operational parameters and functionalities of the asset [Moy+20]. The consistency problem between these models can be partially solved by using the DT-model constructed with the proposed elements of this thesis as a central point of reference and all the other models connected to it can be updated when the DT-model is updated.

Additionally, assets might be represented through specialized models like machine learning algorithms. These models are particularly valuable as they can continuously assess the asset's condition, predicting maintenance [EBA20] needs or potential failures. Such predictive models contribute significantly to proactive management of the asset, enhancing its efficiency and longevity. When an algorithm (e.g. machine learning model) makes a new prediction, it can generate events within the digital twin system, ensuring that the system remains responsive and up-to-date.

To effectively integrate these diverse models, it's crucial to implement drivers that facilitate communication between the model and the digital twin system. These drivers are necessary especially when the models are developed using different software tools or are hosted on separate machines. This interoperability is key to maintaining a seamless flow of information and updates. Moreover, models can be dynamically updated with real-world data, ensuring that the digital representation remains in sync with its physical counterpart. This continual updating process solidifies the digital twin as a reliable and current source of truth, regardless of the models' physical locations or the platforms on which they are built.

Figure 5.2 illustrates the relationships between assets and models. One asset can have models. It is also possible that models have dependencies. For allowing the twin state is propagate through the other models, it might be necessary to implement custom drivers that enable this communication. Additionally, events can be generated by these models and might cause some reactions in the system.

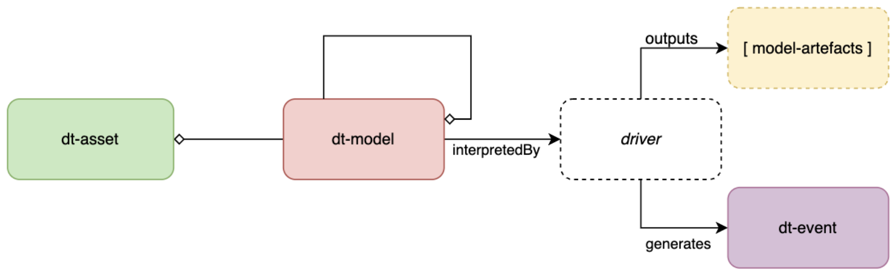


Figure 5.2.: Concept of model integration

A common example of an artifact is a simulation output. Consider a model designed to simulate the operational dynamics of a manufacturing process. This model could generate a time-series visualization artifact [BM24], depicting various parameters like production rate, energy consumption, or machine wear over time.

Models can be hosted externally, which means they reside outside the primary digital twin platform. This external hosting can occur on various platforms or servers, separated from the main system. The advantage of this approach is that it allows for specialized handling and management of the model, often using tools and software specifically designed for that model's needs. For instance, a complex machine learning model might be hosted on a server equipped with the necessary computational power and resources for its operations.

Alternatively, models can also be imported directly into the proposed digital twin environment. This method involves integrating the model into the core platform of the digital twin. The benefit of this approach is the close integration it offers, allowing for seamless communication and interaction within the digital twin environment. Imported or local models are directly accessible within the system, facilitating easier management and synchronization with other components of the digital twin.

The choice between external hosting and importing models depends on several factors, such as the complexity of the model, the resources available, and the specific requirements of the digital twin system. External hosting is often chosen for models that require specialized environments or computational resources, while importing is preferred for models that need to be closely integrated with the digital twin's core functionalities. In both scenarios, the aim is to ensure that the models contribute effectively to the accurate and dynamic representation of the physical assets they mirror.

Extending the model node

As the model node serves as a generic abstraction of various domain-specific models, it enables the creation of specialized nodes that encapsulate the technical intricacies of each model type. This capability represents a significant contribution of this thesis, as it contrasts with many approaches in the literature that predominantly focus on developing domain-specific models tailored to specific application contexts. By offering a flexible yet detailed approach, this work facilitates the representation of a wide range of models while maintaining extensibility across different domains.

One particularly useful example of this is the creation of models that incorporate state machines. State machines are computational models used to identify and document the various states an entity can be in and the transitions between these states based on certain inputs or conditions.

By integrating state machines into a digital twin model, designers can create sophisticated representations that more accurately mirror the dynamic behavior of the physical asset. For instance, a state machine model could be used to track and predict the operational status of industrial machinery during its lifecycle (5.3). In this setup, the state machine would delineate different states of the machinery, such as 'idle', 'running', 'maintenance mode', and 'error state'. Transitions between these states would be triggered by specific conditions or inputs, like sensor readings or time schedules.

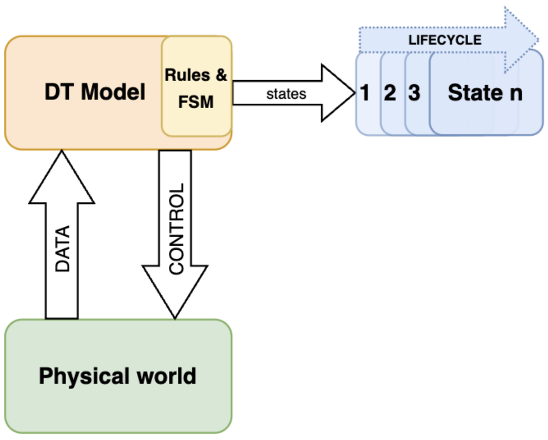


Figure 5.3.: State machines and digital twins

Each new state captured within the system serves as a valuable record of the lifecycle of assets. This accumulation of data can be utilized to understand and analyze the behavior of these assets over time. Moreover, insights gained from these state transitions can be instrumental in enhancing the workflow of the corresponding physical entities, leading to more efficient and effective operations.

The bachelor thesis developed by Aditya Kumar, at the Hochschule Hamm-Lippstadt, has implemented the feature of creating state machines inside the proposed DT environment for modeling DTs that this thesis proposes. Figure 5.4 illustrates how the *dt-* are used to build and update the state machine.

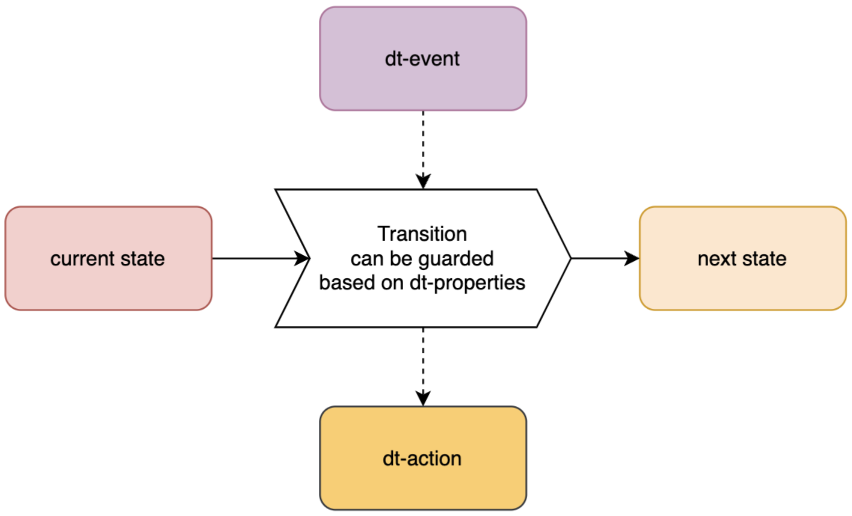


Figure 5.4.: DT nodes used integrated with state machines

By reading the properties of the assets, it is possible to have the current state of the system. In this implementation, only events (dt-event) were used to trigger transitions and then generate a new state. Furthermore, as previously discussed, such transitions can also potentially activate new actions, creating a dynamic, responsive system that adapts to evolving conditions and inputs.

The state machine can be modeled differently based on the use case and the system designer. For instance, an event can trigger an action, causing a transition to the state executing this action. The action can then generate a new event (e.g., finished) that triggers a transition to another state.

However, when considering state machines for modeling digital twins within CPS, it is essential to acknowledge the limitations that arise from relying on instantaneous transitions. From a physical perspective, many CPS applications are bound by physical conservation laws (such as energy, mass, or momentum) that may not be adequately represented through discrete state transitions alone. Instantaneous transitions, which are characteristic of transition-system-based models, can inadvertently violate these conservation laws. This issue poses a particular challenge in digital twin models, where the digital representation must remain faithful to the underlying physical processes. To mitigate the challenge of maintaining physical conservation laws in state machine models for digital twins, a more robust approach is required. As part of the methodology (section 6.1), it is essential to first identify the key physical laws, such as the conservation of energy, relevant to the system. Transitions between states should be designed to include appropriate conditions that attempt to uphold these laws. This can be achieved by introducing intermediary continuous states that simulate gradual changes in physical systems, or by embedding differential equation-based components to capture ongoing dynamics. For example, when modeling an industrial valve transitioning from a closed to an open state, the model should reflect the gradual changes in pressure and flow rate. By integrating continuous dynamics alongside discrete state transitions, designers can ensure that digital twins remain accurate and reliable, making them effective tools for monitoring and control. In this thesis, which focuses on the fundamental aspects of modeling digital twins, the state machine was implemented with instantaneous transitions. Future work will explore incorporating continuous dynamics into the state machine model, aiming to improve the fidelity and accuracy of the digital twin representation.

5.6. Relationship

The relationship is another element for modeling DTs. It enables designers to add semantic annotations that connect two assets. This is a basic feature used to define knowledge graphs and makes it possible to add as many semantics as necessary. Every use case/application has its own requirements, and, therefore, the amount of semantics of the models is highly related to the application goal and user expectations.

Figure 5.5 illustrates the potential application of the node relationship. This node is capable of linking two distinct assets and enhancing their connection through the addition of semantic descriptions. Additionally, it serves a crucial role in associating different models. This is particularly useful in scenarios where demonstrating the interrelation between two models is essential, employing terms like "specializedBy", "extends", "overrides", among others, to describe their connections.

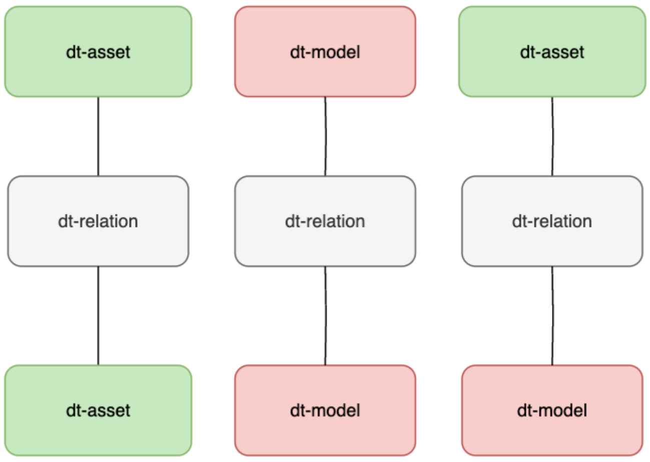


Figure 5.5.: Use of the relationship node

The nature of relationships within a digital twin framework is characterized by their directionality, which is a crucial aspect to define for accurate representation and interaction between assets. These relationships can be categorized into three types:

- *forward*: A –>B: This type of relationship indicates a one-way connection where asset A influences or is linked to asset B, but not vice versa. It represents a directional flow of influence or dependency from A to B.
- *backward*: A <– B: In contrast to the forward relationship, a backward relationship suggests that asset B has an influence or connection to asset A. This type of relationship signifies a reverse directional influence, where the flow is from B to A.
- *bidirectional*: A <–>B: This relationship type denotes a two-way connection between assets A and B. It implies that each asset influences or is linked to the other, creating a reciprocal relationship. This is particularly relevant in systems where assets interact or depend on each other mutually.

These directional relationships are essential in defining the structure and dynamics of interactions within the digital twin system. They help in accurately mapping out the network of asset interdependencies, which is critical for the correct understanding of the model.

5.7. Common aspects to all elements

Each element can have annotations associated with it, allowing adding metadata to different parts of the system. For instance, annotations such as *assumption* and *guarantee* are important in contract-based design of safety-critical systems[SDP12]. Contracts can be used to describe the expected behavior of a specific part of the system. Annotations concerning data privacy can also be added to nodes, allowing designers to specify which group of users have access to certain information.

An additional node called **dt-graph** is also provided. This node is used for monitoring the whole model and generating a graph of all elements, enabling, for instance, the persistence of the updated model in a database. Every time real-world data comes in, or when the model is changed and deployed, the *dt-graph* outputs an updated graph with the system's current state. Distributed models are also listened to by this node, e.g. models that are running in different locations can be linked together to a central model. For example, in a scenario of an industrial plant that contains five machines and one central station, each machine can have its own model running on it while the central station listens to updates coming from these distributed entities, integrating everything into one main model.

The next chapter will formalize the proposed model, defining the nodes and relationships that constitute the knowledge graph. This formalization is essential for ensuring consistency and clarity in the model's structure, facilitating effective communication and validation of the models.

5.8. Formal definition of the model

The proposed model is based on the concept of a knowledge graph. A knowledge graph is a graph-based knowledge representation that captures complex relationships between entities and their attributes. It is a powerful way of organizing and representing data, enabling the creation of a comprehensive and interconnected web of knowledge. This approach is particularly well-suited for digital twins, as it allows for the accurate and detailed representation of the relationships and attributes of the various elements within the system.

This chapter aims to formalize the proposed model, defining the nodes and relationships that constitute the knowledge graph. This formalization is essential for ensuring consistency and clarity in the model's structure, facilitating effective communication and validation of the models.

Knowledge Graph Definition

Figure 5.6 illustrates the concept of nodes and edges in a knowledge graph.

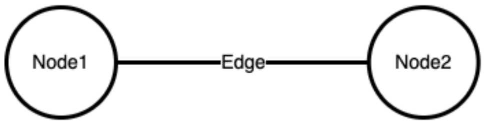


Figure 5.6.: Example of nodes and edge in KGs

Definition 5.8.1 (Knowledge Graph). *A Knowledge Graph is a knowledge base that connects entities (such as objects, events, or concepts) in a network, representing the relationships and properties among them. It encodes knowledge in a machine-readable format, enabling semantic queries and inference, thereby facilitating a more intuitive understanding of complex datasets. By leveraging nodes (entities) and edges (relationships), knowledge graphs provide a comprehensive framework for integrating, analyzing, and visualizing data across various domains and applications.*

Definition 5.8.2 (Node). *A node in a knowledge graph represents an entity, such as an asset, property, action, event, or model, and encapsulates metadata that captures its attributes and characteristics. Each node is uniquely identified and can be connected to other nodes through relationships, forming a network of interconnected entities.*

Definition 5.8.3 (Edge). *An edge in a knowledge graph represents a relationship between two nodes, capturing the connections and interactions between entities. Each edge is characterized by its directionality, name, and associated metadata, providing a structured representation of the relationships within the knowledge graph.*

Knowledge graphs can be defined as $KG = (N, Ed)$, where:

- N is the set of nodes representing assets, properties, actions, events, and models, each with its associated metadata.
- Ed is the set of edges representing relationships between these nodes, also with associated metadata.

Knowledge graphs play a pivotal role in enhancing the understanding, integration, and utilization of data across various domains, particularly in the context of semantic

Digital Twins. By encapsulating the semantics of data — its meaning, relationships, and properties — knowledge graphs facilitate a more natural, human-like comprehension of complex information systems. This semantic foundation enables the integration of heterogeneous data sources, overcoming silos and disparate data formats to create a unified, accessible view of information. Additionally, knowledge graphs enhance search and query capabilities, moving beyond simple keyword searches to more context-aware, intent-driven queries that yield relevant and precise results.

By providing structured, contextual data, knowledge graphs support the training of more sophisticated models, enabling nuanced reasoning and enhanced natural language processing capabilities. Furthermore, they facilitate dynamic system evolution, allowing for the seamless adaptation of digital twins as real-world conditions change. This adaptability, combined with the capability to foster interdisciplinary collaboration through a common semantic framework, underscores the transformative impact of knowledge graphs on digitalization and automation efforts.

Modeling-Elements Specification

Definition 5.8.4. ***Assets (A)****: is represented with a node. It has attributes such as:*

- ***id****: a UUID field that uniquely identifies the asset.*
- ***name****: a String field that contains the name of the asset.*
- ***context****: a String field that contains the context of the asset.*
- ***DTType****: a String field that contains the type of the asset defined by the ISO 23247 (parts, products, equipment, materials, processes, facilities, environments, personnel, and supporting documents or generic) [ISO20].*

Definition 5.8.5. ***Properties (P)****: is represented with a node in the KG. It has attributes such as:*

- ***id****: a UUID field that uniquely identifies the property.*
- ***name****: a String field that contains the name of the property.*
- ***accessGroup****: a String field that contains the access group of the property. This field can be used to define the group of users who have access to the property in further layers of the system.*

- ***context***: *a String field that contains the context of the property. This field can be used to define the datatype or schemas of the property.*
- ***type***: *a String field that contains the type of the property.*

Definition 5.8.6. ***Actions (Ac)***: *is represented with a node in the KG. It has attributes such as:*

- ***id***: *a UUID field that uniquely identifies the action.*
- ***name***: *a String field that contains the name of the action.*
- ***topic***: *a String field that contains the topic (mainly MQTT-based) of the action.*
- ***payload***: *a String field that contains the payload of the action. This field usually contains a complex data structure in JSON format that can be deserialized and used as an object.*

Definition 5.8.7. ***Events (E)***: *is represented with a node in the KG. It has attributes such as:*

- ***id***: *a UUID field that uniquely identifies the event.*
- ***name***: *a String field that contains the name of the event.*
- ***topic***: *a String field that contains the topic (mainly MQTT-based) of the event.*

Definition 5.8.8. ***Models (M)***: *is represented with a node in the KG. It has attributes such as:*

- ***id***: *a UUID field that uniquely identifies the model.*
- ***name***: *a String field that contains the name of the model.*
- ***link***: *a String field that contains the link to the model. This field usually contains a URL that points to the location of the model.*

Definition 5.8.9. ***Relationships (R)***: *is represented with an edge in the KG. It has attributes such as:*

- ***id***: *a UUID field that uniquely identifies the relationship.*
- ***name***: *a String field that contains the name of the relationship.*

- ***direction**: a String field that contains the direction of the relationship. It can be one of the following values: $\rightarrow$, $\leftrightarrow$, $\leftarrow$.*

Nodes creation and Edges: For each element type Node x in $\{A, P, Ac, E, M\}$, a function f_x that maps elements of type x to nodes in the knowledge graph is defined, capturing the specified attributes.

$$\begin{aligned} f_A &: A \rightarrow N, \\ f_P &: P \rightarrow N, \\ f_{Ac} &: Ac \rightarrow N, \\ f_E &: E \rightarrow N, \\ f_M &: M \rightarrow N \end{aligned}$$

Each f_x incorporates the relevant attributes into the metadata (attributes) of the created node.

Edges creation is defined by the function $g : R \rightarrow Ed$, where each relationship r in R is mapped to an edge in Ed. For each relationship r with attributes id, name, and direction, the function $g(r)$ uses these attributes to create an edge and its metadata in the knowledge graph.

Given a relationship r with attributes:

$$\text{id} = \text{"r1"}, \quad \text{name} = \text{"ConnectedTo"}, \quad \text{direction} = \text{"->"}$$

The function $g(r)$ creates an edge $e \in Ed$ capturing this relationship as:

$$e = g(r) = \text{"edge"}(\text{id} = \text{"r1"}, \text{name} = \text{"ConnectedTo"}, \text{direction} = \text{"->"}, \text{from} = \text{"a1"}, \text{to} = \text{"a2"})$$

Consider an asset $a \in A$ representing a car with the following attributes:

$$\text{id} = \text{"car001"}, \text{name} = \text{"Passat"}, \text{context} = \text{"Automotive"}, \text{DTType} = \text{"Product"}$$

The transformation function $f_A(a)$ would create a node $n \in N$ in the knowledge graph with metadata capturing these attributes:

$$\begin{aligned} n = f_A(a) = \text{"node"}(\text{id} = \text{"car001"}, \text{name} = \text{"Passat"}, \text{context} = \text{"Automotive"}, \\ \text{DTType} = \text{"Product"}) \end{aligned}$$

Assuming the car has a property for fuel level with attributes:

$$\text{id} = \text{"prop001"}, \text{name} = \text{"Fuel Level"}, \text{accessGroup} = \text{"Driver"}, \text{context} = \text{"Fuel"}, \\ \text{type} = \text{"Percentage"}$$

The transformation for this property into a node in the KG would be represented as:

$$p = f_P(\text{property}) = \text{"node"}(\text{id} = \text{"prop001"}, \text{name} = \text{"Fuel Level"}, \text{accessGroup} = \text{"Driver"}, \\ \text{context} = \text{"Fuel"}, \text{type} = \text{"Percentage"})$$

For an action of starting the car, with attributes:

$$\text{id} = \text{"act001"}, \quad \text{name} = \text{"Start Car"}, \quad \text{topic} = \text{"Engine"}, \quad \text{payload} = \text{"Ignition On"}$$

The corresponding node creation is:

$$ac = f_{Ac}(\text{action}) = \text{"node"}(\text{id} = \text{"act001"}, \text{name} = \text{"Start Car"}, \text{topic} = \text{"Engine"}, \\ \text{payload} = \text{"Ignition On"})$$

If there's a relationship indicating the car "has" a fuel level property, represented as:

$$\text{id} = \text{"rel001"}, \quad \text{name} = \text{"hasProperty"}, \quad \text{direction} = \text{"->"}$$

The edge creation capturing this relationship is:

$$e = g(r) = \text{"edge"}(\text{id} = \text{"rel001"}, \text{name} = \text{"hasProperty"}, \text{direction} = \text{"->"}, \\ \text{from} = \text{"car001"}, \text{to} = \text{"prop001"})$$

This example formalizes the mapping of an automotive asset (a car) and its related elements into a structured knowledge graph using specified transformation functions.

Given the transformation functions for creating nodes and edges, the overall transformation T of the detailed model into a knowledge graph can be formalized as a combination of these transformations:

$$T = \left(\bigcup_{x \in \{A, P, Ac, E, M\}} \bigcup_{y \in x} f_x(y) \right) \bigcup \left(\bigcup_{r \in R} g(r) \right) \tag{5.1}$$

Where:

- A, P, Ac, E, M represent the sets of assets, properties, actions, events, and models, respectively.
- R represents the set of relationships.
- $f_x(y)$ represents the function that transforms elements y of type x into nodes in the knowledge graph, capturing the specified attributes for each element type.
- $g(r)$ represents the function that transforms relationships r into edges in the knowledge graph, defining the connections between nodes based on the specified attributes of each relationship.

The formula provided describes a transformation process, T, that converts a detailed model (comprising various types of elements such as assets, properties, actions, events, and models) into a knowledge graph. The transformation is achieved by applying specific transformation functions to both the elements of the model and the relationships among these elements.

In the next section, an overview of how the modeling elements can be combined together is given in the form of BNF Grammar.

5.9. BNF Grammar

The proposed modeling elements allow system designers to model their system's structure with different granularities. Therefore, it is important to combine and connect the elements to represent the desired system.

Each of the elements has its purpose (Sections 5.1 - 5.6) and can be combined with others to create a complex system. "A complex system is literally one in which there are multiple interactions between many different components" [Rin99].

In this context, it is also important to define a grammar that allows the creation of complex models using the proposed elements. This grammar is essential for ensuring consistency and clarity in the model's structure, facilitating effective communication and validation of the models.

A Backus-Naur Form (BNF) grammar has been created to express the rules for modeling using the proposed modeling elements (Figure 5.1) of this approach. Figure 5.7 shows the BNF grammar for modeling DT proposed by this thesis.

An asset can be followed by any other of the proposed nodes. This connection could be enriched by using the relationship node, allowing to add semantic information regarding this relationship. Finally, models can also be connected together setting their relationship, such as dependencies, generalizations and so on.

```
<dt-asset> ::= <dt-property>
             | <dt-action>
             | <dt-event>
             | <dt-relation>

<dt-relation> ::= <dt-asset>
                | <dt-model>
```

Figure 5.7.: BNF grammar for modeling with the basic elements

This grammar is important because it plays a fundamental role in defining the language's structure, ensuring consistency and clarity, facilitating automation, and supporting effective communication and validation of models.

For example, in a scenario where a manufacturer1 produces and sells vehicle1, the model could be represented as two assets (Def. 5.8.4) connected by two relationships (Def. 5.8.9). According to the grammar, relationships can not be connected together, which would be an incorrect model. This is an example of how grammar can be used to validate the model and ensure its consistency. Figure 5.8 illustrates this example of an invalid model.

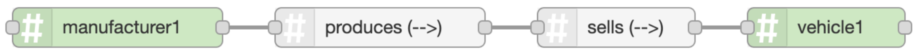


Figure 5.8.: Example of an invalid model

Figure 5.9 shows the valid model for the example given. The model is valid because it follows the grammar rules.

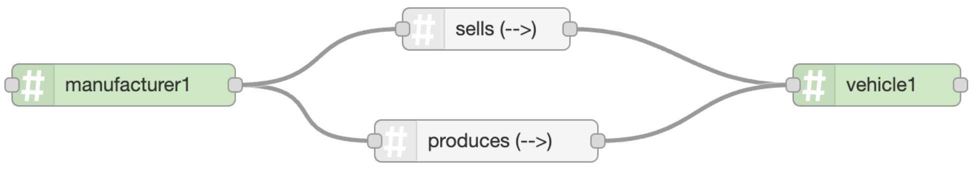


Figure 5.9.: Example of a valid model

At this point, the formal definition of the model has been established, providing a structured representation of the proposed elements and their relationships. This formalization is essential for ensuring consistency and clarity in the model's structure, facilitating effective communication and validation of the models. The BNF grammar further enhances the model's expressiveness and utility, enabling the creation of complex models using the proposed elements. This grammar is fundamental for ensuring the correctness and coherence of the models, supporting the development and validation of digital twins in diverse domains and applications. These aspects are needed to start the implementation of the proposed model, which is the next step in this thesis.

5.10. Implementation of the elements on top of Node-RED

The proposed approach could have been implemented in any platform that supports the creation of nodes and edges. However, the implementation was done on top of Node-RED. It is a well-known IoT platform, which is renowned for its open-source nature and versatile flow-based programming environment.

The implementation in Node-RED not only validates the concept but also showcases the potential for broader applications in the field of IoT and digital twins.

Overview of the implementation

The selection of Node-RED as the platform for implementing the thesis' approach was driven by the inherent synergy between DTs and the IoT. Node-RED is known for its strong foundations in IoT and it provides a set of ready-to-use nodes that facilitate building IoT applications. Additionally, this platform has been utilized by several applications in different domains which can now be extended with a semantic layer to build twins. This choice ensures that the transition to a more sophisticated DT-based system is not only seamless but also leverages the existing IoT infrastructure, thereby enhancing efficiency and reducing the need for extensive redevelopment. Node-RED's adaptability and IoT-centric design thus play a pivotal role in the successful implementation and realization of the proposed DT approach.

Figure 5.10 shows a screenshot of Node-RED with the implemented nodes. The nodes palette is located on the left side of the screen, and the 7 proposed elements (defined in Defs. 5.8.4 - 5.8.9) are shown. The central part of the platform is where the models can be built by dragging and dropping the nodes. On the right side, the debug screen is shown. This feature can show printed debug messages when the system is running.

Another important feature is the deployment (red button on the top right). When *Deploy* is clicked, the model is validated against the grammar defined in Section 5.9 and then deployed: the changes made are activated, with all nodes restarting to reflect the latest configurations. This process initiates the flow's execution, including triggering any 'inject' nodes set to activate on startup. Additionally, the configuration is saved for future restarts, and the user interface updates to show the current state. The deployment also involves error checks, alerting users to any potential issues in the flow.

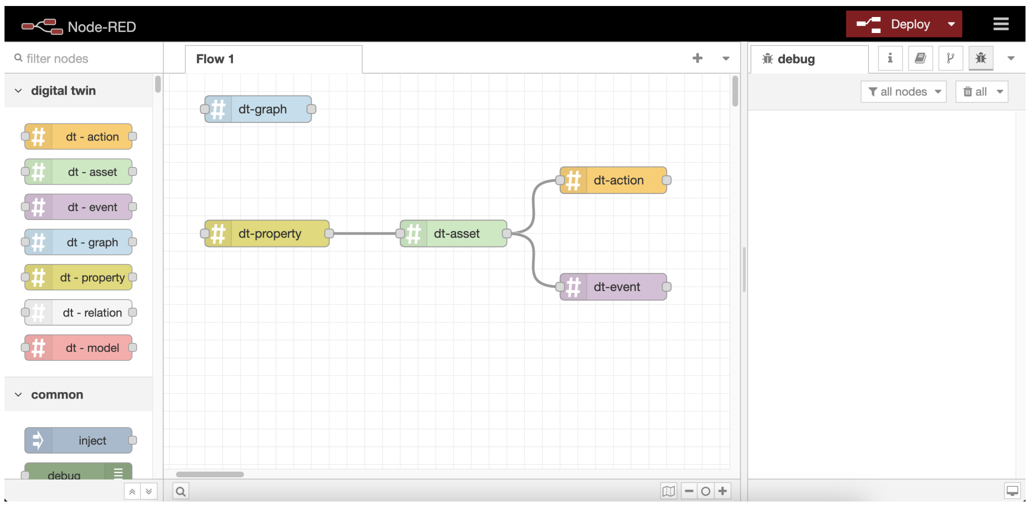


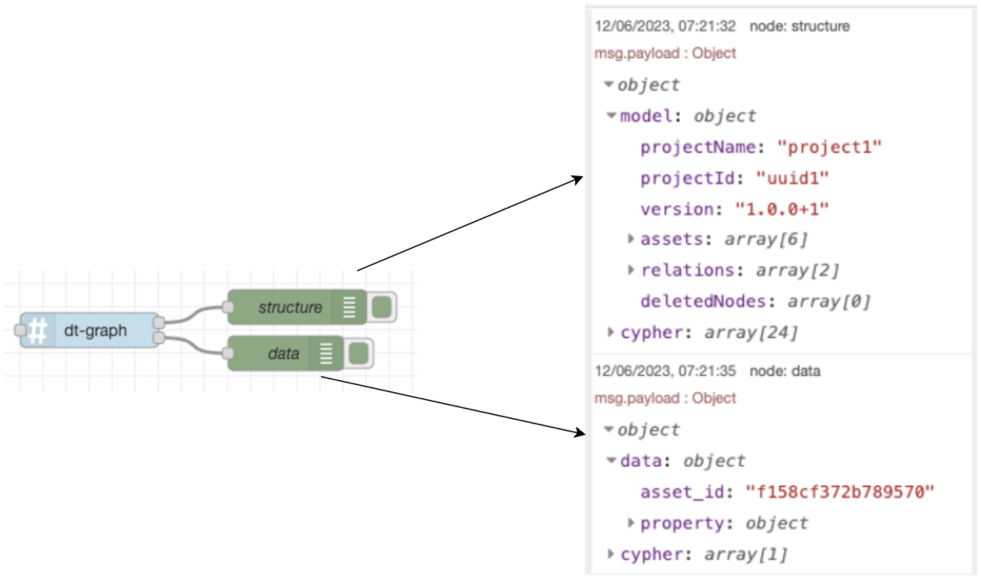
Figure 5.10.: Node-RED screen with the implemented nodes

When building a model, there must be one node *dt-graph* since it is responsible for monitoring changes in the model and emitting updates when new data comes from the real world.

There are two kinds of updates that a DT model can have: structural updates and new data received from the real world, as can be seen in Figure 5.11.

- **Structural updates:** Structural updates refer to the modifications made to models, which result in generating new updates each time a change occurs. These updates are crucial in ensuring that the models accurately represent the current state of the system. When a model is altered – whether it's a change in parameters, addition or removal of components, or any modification to its structure – an update is triggered. This update process is essential for maintaining the integrity and relevance of the data stored in databases or used by inference algorithms.

  An example of a structural update could be in a digital twin model of a manufacturing process. If a new machine is added to the production line, the model must be



```
12/06/2023, 07:21:32   node: structure
msg.payload : Object
▾object
 ▾model: object
    projectName: "project1"
    projectId: "uuid1"
    version: "1.0.0+1"
   ▸assets: array[6]
   ▸relations: array[2]
    deletedNodes: array[0]
 ▸cypher: array[24]
12/06/2023, 07:21:35   node: data
msg.payload : Object
▾object
 ▾data: object
    asset_id: "f158cf372b789570"
   ▸property: object
 ▸cypher: array[1]
```

Figure 5.11.: dt-graph updates

updated to include this machine. This structural update would then be reflected in the database, ensuring that any inference algorithms or analytics tools using this data are working with the most current representation of the manufacturing process. Such updates are fundamental in dynamic environments where continuous learning and adaptation are key to efficiency and effectiveness.

The output of structural updates is a knowledge graph in the JSON-LD format. This format is particularly well-suited for representing complex, interlinked data structures like knowledge graphs, which are essential for capturing the detailed relationships and attributes of the updated models. Whenever a model undergoes a structural change, the resulting update is systematically encoded into this knowledge graph, ensuring that the JSON-LD representation is both current and accurate. This approach not only facilitates the efficient storage and retrieval of complex data but also enhances the interoperability with other web-based technologies and systems, leveraging the structured, machine-readable format of JSON-LD.

- **Data updates:** Data updates happen when new sensor readings from the real world are received by the system. These updates are a critical aspect of maintaining runtime accuracy in the model. For instance, consider an asset in the real world whose temperature is being monitored. When the temperature changes and this new reading is captured by the sensors, the information is immediately transmitted to the model. As soon as this data is received, a data update is generated. This update includes both the asset in question and its corresponding property, in this case, the updated temperature value. This process ensures that the model reflects the states of the real-world asset in runtime, allowing for more precise monitoring and decision-making based on current data. This implementation does not address real-time requirements, as fulfilling these would necessitate different technologies. However, the model supports the description of real-time characteristics, which can subsequently be leveraged to generate platform-specific code designed to manage real-time aspects effectively.

The implementation of these concepts in existing IoT applications is crucial for their transition into the realm of Digital Twins. Here, Node-RED plays a pivotal role, serving as a bridge that enables existing IoT applications to migrate seamlessly to Digital Twin models. Node-RED's versatility and wide adoption in IoT make it an ideal platform for this transition, ensuring that the shift to more sophisticated, semantically-rich DT models is both efficient and effective. This integration highlights the potential of enhancing existing IoT infrastructures with advanced DT capabilities, opening up new avenues for innovation and optimization in the digital modeling space.

Distributed models

This section discusses the potential of running the proposed model on distributed systems. The implementation of the model on such devices can significantly enhance its utility in various applications, particularly in the domain of Digital Twins. The Research Question 1.3.2 is addressed in this section.

Node-RED's compatibility with compact computing devices like the Raspberry Pi significantly enhances its utility in the domain of Digital Twins. This flexibility allows Node-RED to operate on smaller, more cost-effective hardware, which is particularly beneficial for deploying distributed Digital Twin models. Running Node-RED on devices like Raspberry Pi enables the creation of decentralized, yet interconnected digital twin systems. Each individual unit can host and manage the digital twin of a specific machine or sector within a larger network, such as a manufacturing line.

In a practical application, each machine or sector in a manufacturing factory could run Node-RED and contain the specific digital twin model. This setup allows for localized processing and management of data, reducing latency and enhancing responsiveness. Simultaneously, these individual models can be connected to a central model, enabling a holistic view of the entire manufacturing process. Figure 5.12 depicts a setup where three distinct machines are each connected to their respective Raspberry Pi units. These units are then interconnected, as well as linked to a central computer. Additionally, there is the possibility of connecting these systems to a storage solution, enabling data preservation and archiving.

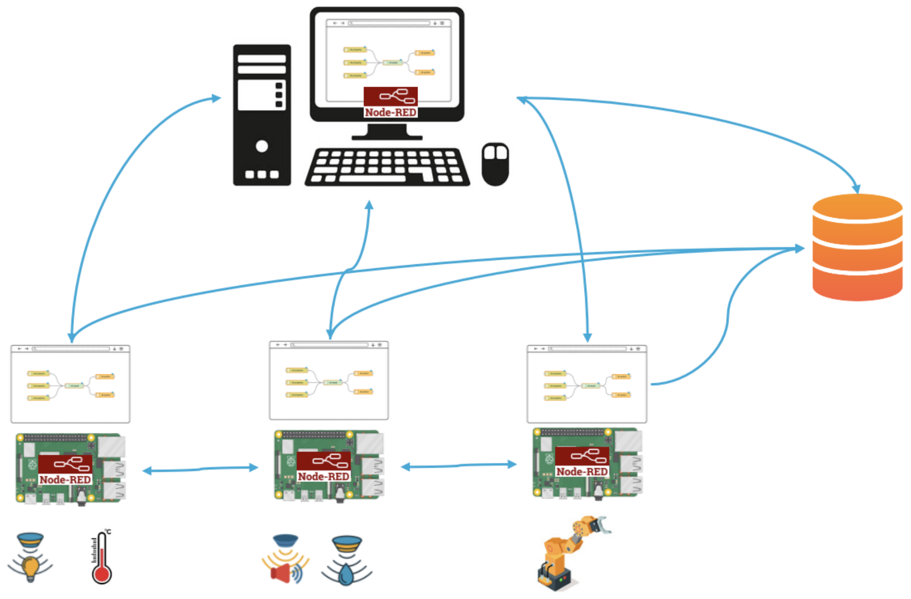


Figure 5.12.: Parallel models illustration

In the envisioned scenario (Figure 5.12), each machine within the system can be modeled and managed by a domain expert, harnessing their specialized knowledge for optimal operation and efficiency. Crucially, despite the diversity of expertise and the uniqueness of each machine, all experts can utilize the same unified modeling language as proposed in this thesis. This common language fosters a collaborative environment where different specialists can work cohesively, ensuring that their individual models integrate seamlessly into the broader system. This approach not only enhances the accuracy and effectiveness of each digital twin but also streamlines communication and coordination among the various experts, leading to a more harmonious and efficient overall system.

An innovative approach to further enhance the efficiency and scalability of distributed Digital Twin models on Raspberry Pi devices is the use of containerization technologies, such as Docker. Containerization allows for the encapsulation of the Node-RED (namely, the DT model) environment and its dependencies into a compact, isolated unit, making it easier to deploy and manage across multiple devices. By encapsulating the Node-

RED environment and its dependencies within a container, it ensures consistent behavior across various Raspberry Pi devices, regardless of their individual configurations. This standardization is particularly vital when expanding the system to encompass additional machinery or sectors, as it streamlines deployment and reduces complexity.

Furthermore, the isolated nature of containers enhances the overall security and stability of the system. Each container operates independently, ensuring that processes and issues within one container do not impact others. This isolation is crucial in a distributed environment, as it minimizes the risk of systemic failures and security breaches, thereby maintaining the integrity and reliability of the entire network.

Lastly, containerization simplifies the processes of updating and maintaining the Node-RED environment and the digital twin models. Modifications or upgrades can be made to a single container image, which can then be easily and uniformly distributed across all devices in the network. This approach not only ensures consistency across the system but also significantly reduces the effort and complexity involved in system maintenance.

This distributed approach can enable systems to be effectively scaled, accommodating additional machines or sectors with ease. It exemplifies the power of this approach in creating a flexible, scalable network of DTs, where each node contributes to a comprehensive understanding and control of the entire system. The ability to run these models on small devices like Raspberry Pis opens up numerous possibilities for efficient and advanced digital twin implementations in various small, medium and large industries.

5.10.1. Technical Details of Node-RED Development

The development of nodes in Node-RED incorporates various technical aspects, including supported operating systems and programming languages. This section gives an overview of these technical details.

Node-RED is designed to be highly compatible across multiple operating systems, ensuring broad accessibility and flexibility. Key supported operating systems include Windows, Linux (including the Raspberry Pi versions), and MacOS. Node-RED's compatibility with Raspberry Pi is particularly noteworthy, as it enables the deployment of lightweight, distributed Digital Twin models in various applications.

The creation of custom nodes in Node-RED leverages a combination of programming languages and technologies to ensure both functionality and user-friendliness. JavaScript, known for its versatility and widespread use in web development, serves as the core language for Node-RED development, with the option to use TypeScript for more structured and type-safe coding. To enhance the visual appeal and usability of the Node-RED user interface, Cascading Style Sheets (CSS) are utilized for styling purposes. Additionally, Hypertext Markup Language (HTML) plays a crucial role in structuring the content and layout of the nodes, particularly in configuring their properties and user interface elements. This blend of JavaScript/TypeScript, CSS, and HTML allows for the creation of efficient, effective, and aesthetically pleasing custom nodes in Node-RED.

Node-RED runs on top of Node.js, a JavaScript runtime. This underlying Node.js framework provides a robust and scalable foundation for Node-RED, enabling it to handle asynchronous events and a wide range of network tasks. Running on Node.js allows Node-RED to benefit from its event-driven architecture, making it highly efficient for event-based applications. This combination enhances Node-RED's capabilities in managing data flows and integrating various services and APIs seamlessly. The use of Node.js also contributes to Node-RED's lightweight nature and its ability to run on various devices, including low-power hardware like the Raspberry Pi, making it a flexible choice for diverse IoT environments and digital twin implementations.

5.11. Conclusion of the chapter

In conclusion, this chapter has introduced the basic modeling elements for defining the structure of Digital Twins. Emphasizing the need for a semantic modeling approach reveals how DT can evolve beyond simple digital replicas into context-aware, intelligent systems. These systems are designed to interact seamlessly with their physical counterparts, showcasing a revolution in the way digital and physical entities coexist and collaborate. The incorporation of semantic layers into DTs ensures a deeper understanding of data, leading to more accurate and actionable insights. This approach not only enhances the fidelity of DTs but also elevates their role in predictive analysis, decision-making, and monitoring.

The proposed modeling elements are fully aligned with the WoT standard, enabling seamless integration with modern web-based applications. Incorporating widely recognized standards like WoT into DT modeling represents a significant advancement, as it facilitates interoperability and allows for the integration of web technologies. This approach also supports the use of existing DT frameworks, such as the Asset Administration Shell (AAS) [Pak+21], further enhancing the flexibility and applicability of the proposed models across various domains.

Moreover, the application of this approach in existing IoT infrastructures, facilitated by tools like Node-RED, is a significant step towards the evolution of IoT systems. By enabling existing IoT applications to migrate effortlessly to Digital Twins, the proposed approach ensures a smooth transition to more advanced, semantically-enriched models.

This not only preserves existing investments in IoT but also paves the way for future innovations and enhancements.

To support system engineers in building their DTs system, a methodology has been introduced in the next chapter. This methodology serves as a roadmap, offering step-by-step guidance on how to utilize the previously discussed semantic modeling elements and architecture to construct semantic digital representations of physical entities.

6. Digital Twin Modeling Methodology

To support the design and implementation of DTs, using the previously explained architecture and modeling elements, this chapter introduces a novel methodology that systematically guides the development process from conceptualization to deployment. By following this structured approach, practitioners can effectively bridge the gap between theoretical models and practical applications, creating DTs that are both functional and contextually relevant. This approach is designed to assist engineers and system designers in the creation and implementation of their DT models. The methodology is graphically represented in Figure 6.1, which outlines the specific modeling steps to be followed.

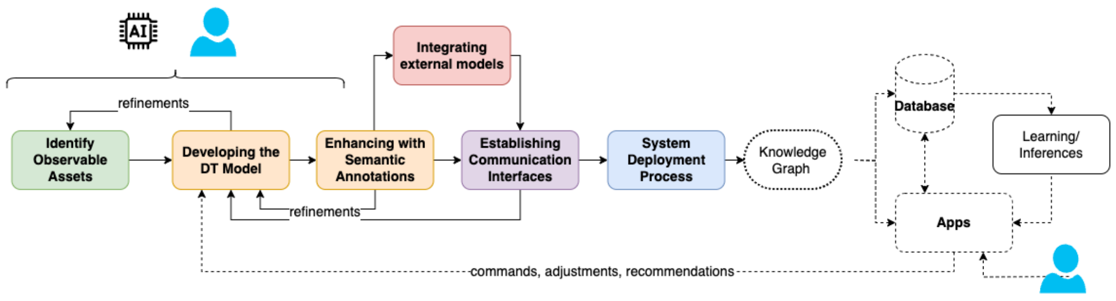

AI
Integrating external models
refinements
Identify Observable Assets
Developing the DT Model
Enhancing with Semantic Annotations
Establishing Communication Interfaces
System Deployment Process
Knowledge Graph
Database
Learning/ Inferences
Apps
refinements
commands, adjustments, recommendations

Figure 6.1.: Steps of the mothodology

This methodology follows the Model-based design approach, by starting with the system models that are built at the beginning of the process. Each step is described as follows:

6.1. Identify Observable Assets

The initial phase in modeling Digital Twins involves a detailed analysis of the real-world components that are observable or interactable. These components are referred to as Observable Assets (Def. 5.8.4). An Observable Asset is not just a standalone entity; it can encompass a composite structure made up of other assets, forming a network of interrelated entities. For instance, in a manufacturing setting, an observable asset like a conveyor belt system might include individual components such as motors, sensors, and control units, each of which is an asset in its own right.

In this crucial step, it's important to meticulously identify the properties and actions associated with each asset. Properties refer to the static and dynamic characteristics of an asset, such as size, color, temperature, or speed. For example, a motor might have properties like rotational speed and operating temperature. Actions, on the other hand, are the functions or operations that an asset can perform. Continuing with the motor example, possible actions could include 'start', 'stop', or 'adjust speed'.

The process of identifying these assets, their properties, and actions requires close collaboration with domain experts and thorough observation of the operational environment [Liu+21]. This ensures that the digital twin accurately mirrors the complexities and nuances of each physical asset. By comprehensively mapping out these observable assets, their properties, and actions, a solid foundation is laid for constructing a detailed and functional model.

6.2. Developing the DT Model

Following the identification of observable assets, the next step involves the modeling of each asset, along with the integration of their respective actions and properties. This modeling process is conducted using the specific nodes outlined in the previous section, as visualized in Fig. 5.1. In this phase, each asset is not just represented digitally but is also enriched with its unique characteristics and functionalities. For example, if an asset is a temperature sensor, its model will include properties like current temperature readings, operating range, and accuracy, along with actions like 'activate' or 'reset'.

In addition to these individual representations, semantic relationships between assets are established to reflect their real-world interactions and dependencies. For instance, in a smart building context, a thermostat (an asset) might have a semantic relationship with HVAC systems (another asset), indicating a control or feedback loop.

A critical aspect of this step is the flexibility to revisit and refine the models. As insights are gained or changes occur in the real-world setup, it becomes necessary to loop back to the initial step of identifying observable assets. This iterative process allows for the reorganization and updating of asset models to ensure they continue to accurately mirror the physical world. Such iterative refinements are key in maintaining the relevance and accuracy of the Digital Twin model, ensuring it adapts to evolving real-world conditions and requirements.

6.3. Enhancing with Semantic Annotations

Beyond the initial semantics integrated into relationships and asset compositions and DT structure modeling, this stage focuses on enriching properties, actions, and assets with more detailed semantic annotations. This enhancement aligns the model with a universal standard, ensuring consistency and clarity across all elements. For instance, consider an asset with a temperature property, typically represented by a single float number. Without semantic annotation, the unit of measurement for this temperature—whether Celsius, Fahrenheit, or Kelvin—remains unspecified. However, leveraging insights from the research by [Thu+20], designers can utilize standards like *iot.schema.org* to annotate assets with precise semantic definitions. This means the temperature property would be clearly defined with its unit of measurement, eliminating any ambiguity.

As defined in Section 5.8, the semantic annotations are added to the relationships in asset compositions, properties, actions, and assets.

Furthermore, the relationships between assets can be established according to the specific needs of the use case. For example, in a smart building scenario, an asset named 'room1' could have a 'belongsTo' relationship with another asset, 'building1'. This not only defines the physical association but also implies a semantic connection, such as 'room1' being a part of 'building1'.

Once these semantic relationships are established, a comprehensive graph of the model is generated, encapsulating all the semantic annotations. This graph serves as a visual and interactive representation of the model, illustrating the intricate web of connections and dependencies between different assets. Importantly, the process remains flexible, allowing for iterative modifications and refinements. Designers can revisit and adjust the model as required, ensuring that it continually aligns with real-world changes and maintains its semantic integrity. This ongoing process of enhancement and refinement is vital in keeping the Digital Twin model both accurate and semantically rich.

6.4. Process of defining the DT models

The initial three steps—Identifying Observable Assets, Developing the DT Model, and Enhancing with Semantic Annotations—are executed collaboratively by system designers from various fields, using the common language and methodologies outlined in earlier sections. This interdisciplinary approach is crucial, as it brings together diverse perspectives and expertise, ensuring a comprehensive and accurate representation of the structure of the real-world system in the digital twin.

Relating these steps to the architecture, Figure 6.2 shows the transfer of information from the observable layer to the model layer. This is one of the main steps in the digitalization process and helps business models to be shifted to a data-driven approach since the data is one of the main assets of modern companies.

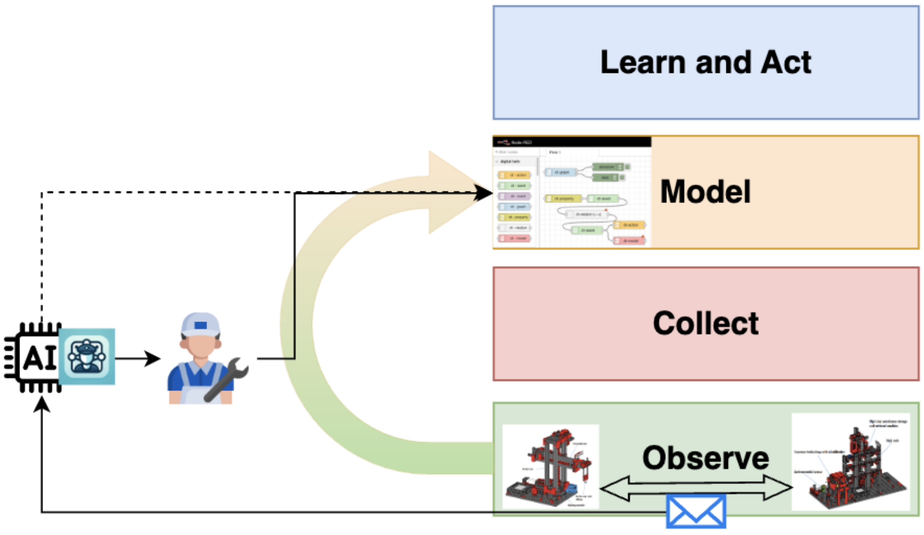


Figure 6.2.: Transfer of information from the observable layer to the model layer

Similarly to current technologies such as GitHub Copilot [NN22] that helps programmers to write code, Grammarly [Fit21], which helps to write better texts and other AI tools that help in the creative process, AI can also be used to help in the DT modeling process. The goal of this thesis is not to provide a complete solution for applying AI to DT modeling but to show that it is possible to use AI to help in the process, possibly called DT-Copilot. For this thesis, an API of OpenAI was used to generate Assets based on MQTT messages. The API receives a message and returns a list of assets that could be generated based on the message. This is a simple example of how AI can help in the process of defining the DT models and working as a DT-Assistant.

The evolution of AI and Natural Language Processing (NLP) technologies can play a significant role in this process [Sun+22]. Advanced AI algorithms assist in analyzing complex systems and identifying key components and their interactions. A designer may manually craft the component models, or these models can be generated through AI algorithms that are capable of processing a set of requirements. These algorithms can use NLP to analyze text descriptions, interpret machine messages, and deduce the relationships and characteristics of various devices within the system. Alternatively, a combined approach can be employed where human insights and AI capabilities are integrated. In this collaborative model, the precision and analytical strength of AI complement the nuanced understanding and creative problem-solving skills of the human designer. This partnership aims to construct a more accurate and efficient DT system, harnessing the best of both worlds: the intuitive, contextual intelligence of humans and the speed and scalability of AI.

By listening and analyzing messages exchanged by the physical assets, the AI can help to identify the assets and their relationships. For example, if a message is sent from a sensor to a controller, the AI can understand that these two assets are related and can help to define the relationship between them. This is a very complex task and is not in the scope of this thesis, but it is important to mention that it is possible to use AI to help in the process of defining the DT models. AI can also help system designers by suggesting new assets, properties, and actions based on the messages exchanged by the physical assets.

In the scope of this thesis, an API of OpenAI was used to generate Assets based on MQTT messages. The API receives a message and returns a list of assets that could be generated based on the message. This is a simple example of how AI can help in the process of defining the DT models and working as a DT-Assistant.

While AI-based approaches have made significant strides, they still face critical challenges such as hallucinations, where the AI generates information that is not grounded in reality, or incorrect suggestions, as noted by [Li+24]. Therefore, it remains essential to keep a human in the loop to validate and refine these AI-generated outputs, ensuring the reliability and accuracy of the final model. As this is a rapidly evolving field, future research and development are expected to further refine and enhance the capabilities of AI in DT modeling, making it an indispensable tool for system designers.

According to the methodology, at this point, the main structure of the model is already defined, and the next steps are to integrate the external models. This is the focus of the next section.

6.5. Integrating External Models

This phase of the methodology is centered around the integration of external models, which provide diverse and unique perspectives of each asset. Considering that an asset can be represented through various models and technologies [TTA23], this step involves aggregating these models and establishing semantic connections to their respective

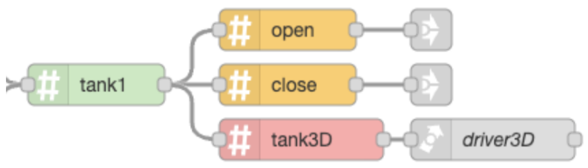


Figure 6.3.: Integration of external models in the DT model through drivers

components within the system. For example, a motor in an industrial setup might be represented by a mechanical model detailing its physical structure, an electrical model outlining its circuitry, and a performance model predicting its efficiency over time.

Each of these models adheres to its specific standards and formats. As a result, it becomes necessary to implement custom drivers that facilitate communication with these external models. These drivers act as interpreters, translating the data and commands between the system's language provided in this thesis and the model-specific language. For instance, a driver could translate sensor readings into specific instructions for 3D modeling software that represents the physical structure of the motor.

Drivers can be developed for specific types of models and then reused for similar models. For example, a driver implemented in JavaScript to interact with a REST API that updates a 3D model of a motor can be reused for other models that also use REST APIs. Figure 6.3 shows the integration of external models in the DT model.

The successful linking of these models within the Digital Twin framework enhances the depth and accuracy of the asset representation. By ensuring that all models are connected and accessible, a comprehensive, multi-dimensional view of each asset is achieved. This not only enriches the system's understanding and simulation capabilities but also allows for more informed decision-making and predictive analysis. The integration of external models, therefore, is a crucial step in building a robust and dynamic Digital Twin environment, capable of reflecting the complexities of real-world assets while keeping all linked models updated with the current state of the system.

After adding external models, the next step is to define the communication interfaces to allow the model to be connected to the real world.

6.6. Establishing Communication Interfaces

Once the Digital Twin model is defined and enriched with semantic annotations (relationship between the assets that are reflected in the KG), the focus shifts to setting up *communication interfaces*. This critical phase bridges the gap between the digital model and the physical world, ensuring seamless interaction between the two realms. The configuration of these interfaces involves detailing how the digital (cyber) and physical components of the system will communicate. This step is crucial as it enables the continuous flow of data from the physical world into the digital model, and conversely, allows the digital model to influence the physical world by sending adjustments and commands.

To facilitate this two-way communication, Node-RED plays a significant role. It offers a diverse range of nodes, many of which are community-developed, supporting various well-established communication technologies. These technologies include MQTT (a lightweight messaging protocol for small sensors and mobile devices), REST APIs (allowing for web-based requests and data exchange), sockets (for stream of data), and direct file access, among others.

By leveraging these technologies, the digital twin can maintain an up-to-date representation of the physical assets, reflecting changes and states. Conversely, it can also exert control over the physical assets, enabling actions like altering settings, initiating processes, or even triggering emergency shutdowns. This level of interaction is crucial for the effective functioning of DTs, as it ensures that the models are not only reflective of the current state of their physical counterparts but are also capable of actively participating in their operation and management.

Having established the communication interfaces, the next step is to deploy the system, making it operational and connected to the physical twin. The next section details the deployment process.

6.7. System Deployment Process

Reaching the end of the setup process, the DT model is ready for deployment. This pivotal stage marks the transition from a theoretical model to an operational system [Sch+21b]. Upon deployment, Node-RED plays a crucial role in saving the model locally within its environment. There's also the flexibility to export the model for external use or backup purposes. This step is essential as it signifies the activation of communication channels between the digital twin and the real world, enabling the system to start functioning based on the data and interactions.

Concurrently, a comprehensive knowledge graph in JSON-LD format is generated, encompassing all the defined elements of the model. This graph format is particularly beneficial as it aligns with web standards, making the data universally understandable and accessible. The JSON-LD graph can then be stored in a graph database, such as

TypeDB or Neo4j. These databases are capable of handling complex, interrelated data structures, making them ideal for managing the intricate networks of relationships and properties inherent in DTs.

Furthermore, the deployed system allows for the execution of queries and inferences on the updated graph. This capability is vital for extracting meaningful insights from the data, as it enables the exploration and analysis of the semantic layers within the model. By querying and analyzing the graph, users can gain a deeper understanding of the relationships, patterns, and trends that define the digital twin's operation and interactions with the physical world. In essence, the deployment phase not only brings the Digital Twin model to life but also sets the stage for ongoing analysis, optimization, and adaptation of the system.

Details of what happens after the deployment and its output are explained in the next section.

6.8. After deployment stage

After the deployment of the model is done, the Digital Twin model is live and ready for ongoing analysis, optimization, and adaptation of the system. Every time a change is made in the model, it must be deployed to bring this modification to life, and then a knowledge graph of the assets and relationships is generated and can be stored on databases and used with ML algorithms to learn and perform inferences. Finally, all this information can be presented to users to help in the decision-making and/or result in commands to the physical twin.

6.8.1. Knowledge Graph

The KG (defined in Def. 5.8.1) in a DT model is a semantically rich, graph-based data structure that encapsulates the relationships and properties of all the system's elements. It provides an intuitive visualization of how different parts of the system are interconnected, making it easier to understand complex dependencies and operational flows. This detailed representation is crucial for accurately mirroring the real-world system and providing actionable insights.

It is important to notice that every time there is a change in the model (data change or structural change), a new snapshot of the model is created and can be stored, keeping track of the system life cycle. It works as a snapshot of a specific timestamp and can be reused not only for learning patterns but also for visualizing previous states of the system after a crash.

For example, if an Asset called *gripper* would have the properties *position, pressure, temperature, and load*, and the action *moveToStorage*, it could be modeled like in Figure 6.4.

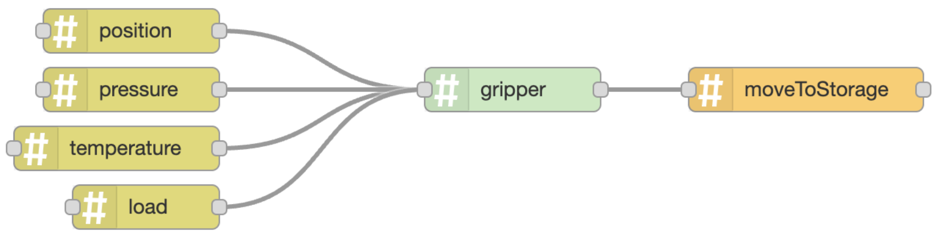


Figure 6.4.: Example of a model

This model would generate a graph in JSON-LD similar to the following (the complete JSON can be found in the appendix A.1):

```
    {
  "model": {
    "projectName": "project1",
    "projectId": "uuid1",
    "version": "1.0.0+1",
    "assets": [
      {
        "id": "8caf970c47b84ea6",
        "type": "dt-asset",
        "name": "gripper",
        "aTtype": "Generic",
        "context": "",
        "actions": [
          {
            "id": "dc665aa4cf49dac3",
            "type": "dt-action",
            "name": "moveToStorage",
            "topic": "",
            "payload": ""
          }
        ],
        "properties": [
          {
            "id": "e5ba81d75d4b83c8",
            "type": "dt-property",
            "name": "position",
            "accessGroup": "",
            "context": "",
            "aType": "float"
          }
          ...
        ]
      }
    ]
  }
}
```

It is possible to see that this JSON contains information about the model in general, such as:

- **project name:** a string that represents the name of the project (e.g. "Gripper Machine").
- **project id:** a unique identifier for the project (e.g. a UUID identifies).
- **version:** a string that represents the version of the model (e.g. "1.0.0+1"). Every time a change is made in the model, this version is updated, allowing the tracking of the model life cycle.

Within the model, there is an array of assets, and each asset has its properties and actions. For example, the asset *gripper* has the action *moveToStorage* and the property *position*.

Assets, according to the definition 5.8.4 have attributes such as id, name, context, and type (aTtype), and properties and actions. Properties and actions are defined in the definition 5.8.5 and 5.8.6, respectively. Properties have attributes such as id, name, context, and type (aType), and actions have attributes such as id, name, topic, and payload.

The attribute “type” is used by Node-RED to identify which kind of node it is, therefore, to represent the type (defined in Defs. 5.8.4, 5.8.5) the attribute “aType” is used.

Based on the DT model structured in this way, it is possible to convert it into different formats such as Cypher using the transformation defined in Section 5.8. This format is usually used by knowledge graph databases as a query language. Figure 6.5 shows the knowledge graph stored on Neo4J.

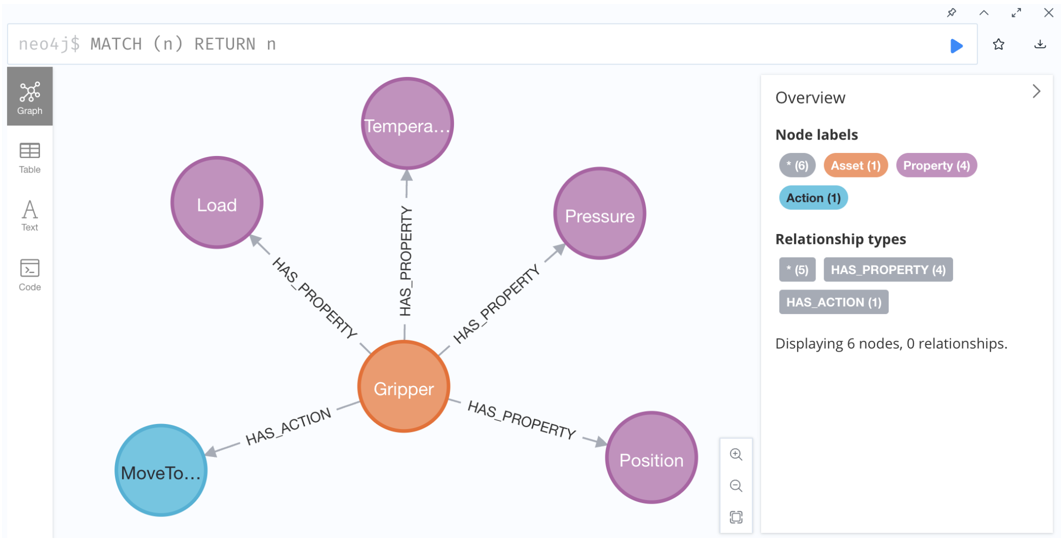

```
neo4j$ MATCH (n) RETURN n
```

Figure 6.5.: Example of a model

If designers do not specify the relationships between assets with properties and actions, these relationships are always referred to as "has_property" and "has_action".

In this context, if the asset *gripper* has the property *position*, the relationship between them is "has_property", and if the formula defined in (5.1) is applied, the result in a KG-way would be:

1. Define the Elements:

   - Let the Asset (defined in 5.8.4) $A = \{\text{"Gripper"}\}$
   - Let the Property (defined in 5.8.5) $P = \{\text{"Temperature"}\}$
   - Let the Relationship (defined in 5.8.9) $R = \{\text{"has\_property"}\}$ where "has_property" is the relation between "Gripper" and "Temperature".

2. Transform the Elements:

- For $f_x(y)$ where $x = A$ and $y =$ "Gripper", this function would transform "Gripper" into a set of triples [1] that define it as an asset in the KG [2]. Similarly, for $x = P$ and $y =$ "Temperature", it transforms "Temperature" into a set of triples [3].
- For $g(r)$ where r is the "has_property" relation between "Gripper" and "Temperature", this function would transform this relation into a set of triples that represent the relationship.

3. Apply the Formula:
    - The transformation T results in a knowledge graph that includes nodes for the Gripper and its Temperature, along with an edge representing the "has_property" relationship between them.

While KGs offer a semantically rich and intuitive representation of system elements and their interconnections, databases provide the infrastructure for storing, querying, and managing these representations over time. The next section discusses the role of databases in the DT modeling process.

6.8.2. Database

The choice of database for a DT system is contingent on the specific needs of the application. A multi-faceted approach, combining different types of databases, often provides the most comprehensive solution.

Graph Databases for DT Structure: Graph databases are particularly well-suited for storing the DT model's structure and current state. They excel in representing complex relationships and interdependencies among various assets, mirroring the KG's structure.

Time-Series Databases for Historical Data: Time-series databases are ideal for tracking the historical data of the system (and its assets). They efficiently store and manage chronological data, such as sensor readings or operational logs. This historical data is crucial for analytics like predictive maintenance, enabling the system to learn from past patterns and anticipate future needs.

[1] In the context of knowledge graphs, transforming an entity into a set of triples allows for a structured representation of information. A *triple* consists of a subject, predicate, and object, forming the basic building block of semantic data models.

[2] Subject: Gripper, Predicate: type, Object: Asset, indicating that the "Gripper" is of type "Asset"

[3] Subject: Gripper, Predicate: hasProperty, Object: Temperature, indicating that the "Gripper" has a property "Temperature".

The choice of databases is usually highly dependent on the use case and the specific requirements of the system, therefore, it is not in the scope of this thesis to define which database should be used.

However, storing data is crucial for the system to be able to learn and perform inferences. The next section gives an overview of this concept.

6.8.3. Learning and inferences

The DT model, supported by its comprehensive database and KG, becomes a fertile ground for advanced learning algorithms and inference mechanisms.

Machine learning algorithms can analyze the accumulated data to detect patterns, predict future states, and suggest optimizations. These algorithms can learn from both the current state of the system (as represented in the KG) and its historical data.

In the scope of this thesis, through the use cases in Chapter 7, inferences on a Knowledge Graph (KG) are performed using the Cypher query language. Cypher is a declarative query language for graph databases, allowing for complex queries and pattern matching. By executing Cypher queries on the KG, users can extract valuable insights, identify relationships, and uncover hidden patterns within the system. However, it is important to note that complex algorithms can also be used to perform inferences and learn from the data.

The insights and predictions derived from these algorithms can be translated into actionable commands. These commands can be communicated to end-users through applications or directly implemented to effect changes in the physical assets, optimizing performance and preempting issues. The next section discusses the applications that can be used to interact with the DT model.

6.8.4. Applications

Applications serve as the user interface of the DT system, translating complex data and insights into an accessible and actionable format for end-users.

These applications can provide a user-friendly interface to interact with the DT system. They display information derived from the KG and database analyses in a comprehensible manner, aiding in decision-making processes.

Updates and Commands: Through these applications, users can receive updates on the system's status, view predictions and recommendations, and issue commands. These commands, in turn, are relayed back to the physical assets, closing the feedback loop between the digital and physical twins.

6.9. Conclusion of the Chapter

The methodology presented in this chapter delineates a comprehensive framework for DT modeling, crucial to the evolution of Industry 4.0. The structured approach outlined here bridges the gap between theoretical constructs and their practical applications, ensuring that DTs are not only conceptually robust but also functionally relevant and contextually adaptable.

Beginning with the identification of observable assets, this methodology highlights the importance of understanding and capturing the real-world components that the DT will reflect. It then transitions into the development of the DT model, emphasizing the need to integrate both actions and properties of each asset. This includes the critical task of semantic annotation, which ensures clarity and consistency across the DT environment by standardizing definitions and relationships within the model.

The process is reinforced by the integration of external models, each offering a different perspective of the asset, and the establishment of communication interfaces that enable real-world interactions. The deployment phase makes the model come to life, enabling runtime data flow and interactive management of the physical assets.

The culmination of this process is a live DT model that can be continuously analyzed, optimized, and adapted. This dynamic model is ever-evolving, with changes and updates reinforcing its accuracy with the physical counterpart. The knowledge graph that emerges from this model serves as a structured representation of the system's complexity and the relationships of its components.

By adopting this approach, system designers are equipped with a powerful toolkit that not only aids in the design and implementation of DTs but also in the ongoing evolution of these systems. The convergence of AI, machine learning, and human expertise within this framework heralds a new era of DT modeling—one that is more intuitive, predictive, and responsive to the needs of both the system and its users.

Thus, this thesis sets forth a methodological paradigm that is expected to significantly contribute to the field of DTs, offering system designers a path that is as innovative as it is practical. It is a pathway marked by continuous refinement, learning, and adaptation, ensuring that DTs remain at the forefront of new advanced systems.

7. Application of the proposed approach

To illustrate the utility and versatility of the proposed approach presented in the previous chapters, two distinct use cases have been developed. The first use case is situated within the Industry 4.0 domain, demonstrating the application of DT in a manufacturing and production context. The second use case explores the automotive domain, showcasing the potential of DT models in vehicle design, performance optimization, and maintenance.

7.1. Industry 4.0

In the context of Industry 4.0, the introduced concept is applied to a comparatively conventional setting where the system structure remains mainly static, and the main changes are observed in the asset properties rather than their fundamental configurations or functions.

DTs are extensively utilized in Industry 4.0 to replicate physical assets in a virtual environment, enabling monitoring, simulation, and analysis. This digital mirroring facilitates a deeper understanding of asset performance under various conditions, predictive maintenance, and optimization of production processes. The essence of DT in Industry 4.0 lies in its ability to create a seamless bridge between the physical and digital worlds, enhancing operational efficiency, reducing downtime, and fostering innovation.

By mirroring the factory modules such as the storage and retrieval station, vacuum gripper robot, high-bay warehouse, and others in a virtual environment, stakeholders can have a digital representation of the system, experiment with new settings, analyze reports and so on. This hands-on experience with DT models prepares users for the complexities of modern manufacturing environments, emphasizing the importance of data-driven decision-making and the integration of IoT technologies for optimized performance based on current data. Additionally, in the era of LLMs, industries must have ways of integrating different information sources, feeding decision-making systems with the knowledge of the various stakeholders that can be involved.

Thus, this use case underscores the practical benefits of adopting the proposed DT approach, demonstrating its relevance and applicability in enhancing the understanding and optimization of manufacturing processes within the context of Industry 4.0.

More specific details about the developed use case are given in the next subsection.

Description of the use case

For this use case, the Industry 4.0 simulator from Fischer Technik has been used to demonstrate how the proposed concepts of this thesis can be applied in this kind of application. It is suitable for vocational training, academic research, and professional development in universities, corporations, and IT sectors. It offers a simulated environment showcasing the digital and interconnected stages of ordering, manufacturing, and delivery processes. Figure 7.1 shows the mentioned simulator.

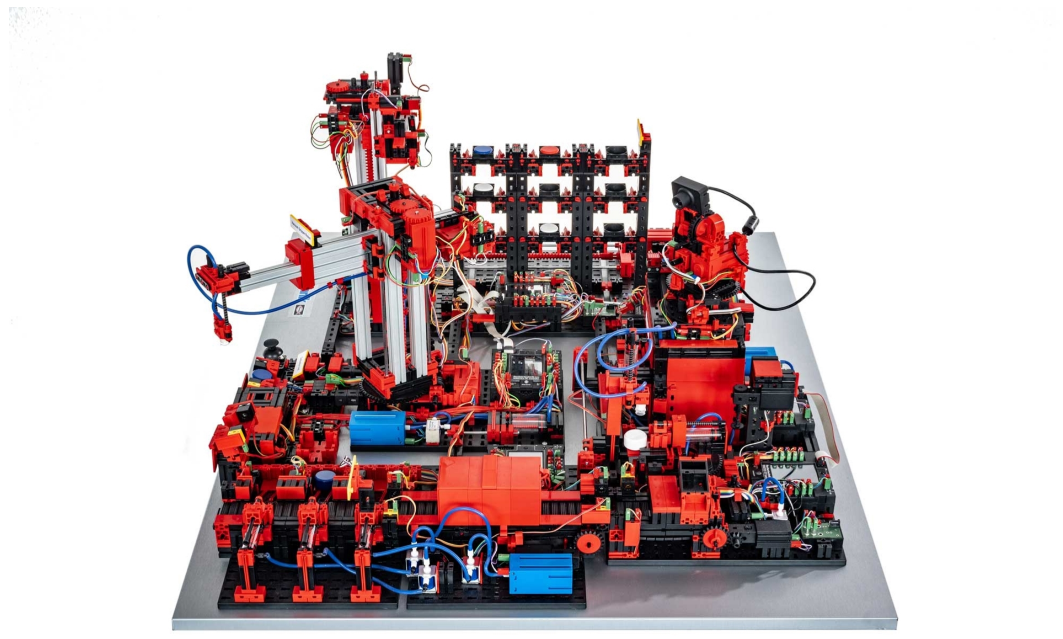


Figure 7.1.: Fischer Technik Industry 4.0 simulator ©fischertechnik GmbH

This interactive Training Factory features modules such as a storage and retrieval station, a vacuum gripper robot, high-bay storage, a multi-functional station with an oven, a sorting conveyor with color detection, an environmental sensor, and a pivoting camera.

From the moment raw materials are delivered and stored in the high-bay warehouse, through to processing orders via the dashboard, each step is visualized in runtime on the dashboard. The environmental sensor enhances the setup by monitoring temperature, humidity, air pressure, and air quality, while the camera's extensive pivot range supports comprehensive web-based remote surveillance.

Each component is monitored using NFC technology, assigning a unique ID to every item to track and display its progress through the production line.

The Factory operates on a 24V (or 9V) system and is controlled by a PLC (programmable logic controller), which is compatible with various brands, though not included.

The aim of this use case is to illustrate the practical application of the thesis's proposed concepts by integrating them within the Industry 4.0 applications. By doing so, it seeks to demonstrate the effectiveness of using a knowledge graph to represent the structural and operational aspects of an industrial plant. This approach enables the execution of precise inferences based on actual data collected from the various components of the simulator. Through this real-world application, the use case aims to highlight how digital twin models can be effectively utilized to enhance decision-making processes, optimize performance, and predict future outcomes in an Industry 4.0 setting. For example, if one of the machines is broken or will stop working soon, inferences can be made to find out what is the current best strategy for replacing this equipment.

This industrial plant is composed of several sectors/components that work together to execute different processes. For this use case, the process of Inbound Delivery and Storage has been executed to demonstrate the proposed concepts. Some of the main components of this process are the Delivery and Pickup Station, the Gripper Robot, and the Warehouse Storage.

Figure 7.2 shows these three components with some of their parts.

- **1. Delivery and Pickup Station:** The Delivery and Pickup Station is a critical component of the factory, serving as the initial entry and final exit point for workpieces. It is divided into four main work areas, integrating several functionalities to facilitate the processing of materials as they come in and out of the production line. These areas include:

  Input/Output Unit: This area manages the introduction of raw materials into the system and the dispatch of finished products. Equipped with a light barrier, it detects the presence of workpieces, signaling the system to initiate subsequent processing steps. **Color Detection**: Once a workpiece is identified, it passes

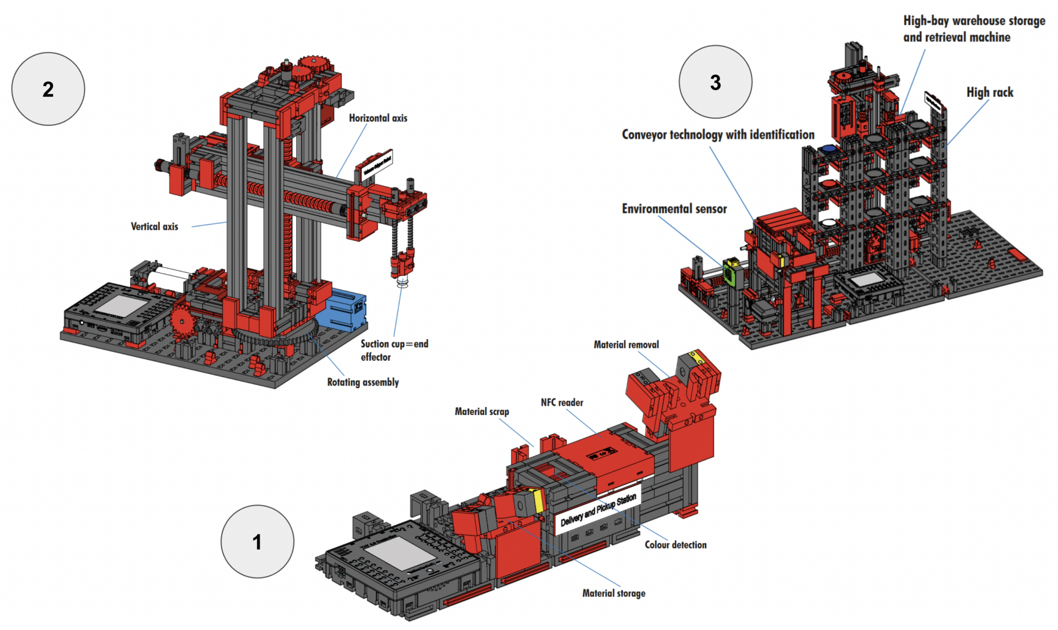


Figure 7.2.: Inbound Delivery and Storage Process ©fischertechnik GmbH

through the color detection area. Here, a color sensor assesses the workpiece's color, gathering essential data for the production process. This information can be used to sort workpieces or tailor the manufacturing process to specific requirements. **NFC Reader**: After color detection, the vacuum gripper positions the workpiece onto the NFC reader. This step involves clearing the NFC tag's memory and marking the item as raw material, before proceeding to write production-related data onto the NFC tag. This data includes, but is not limited to, the workpiece's color, processing steps completed, and quality control metrics. **Output Area**: Completed workpieces are moved to the output area, where they can be organized for further processing or prepared for shipping. This area also allows for additional data to be stored on the NFC tag, providing comprehensive traceability of the workpiece's journey through the production line.

- **2. Gripper Robot:**The Gripper Robot is an automated mechanism designed for precise handling and manipulation of workpieces within the factory. It plays a pivotal role in moving materials from one station to the next, ensuring smooth transitions between different phases of the production process. Key functions include:

  Picking and Placing: The robot's primary task is to pick up workpieces from their incoming position and place them accurately for processing, whether it be for storage, color detection, or further manufacturing steps. **Interacting with NFC Reader**: The Gripper Robot also interfaces with the NFC reader, positioning workpieces so that their embedded NFC tags can be accurately read and written to, facilitating the tracking and management of production data.

- **3. Warehouse Storage:**Warehouse Storage represents the backbone of the factory's logistics and inventory management system. It is where raw materials, work-in-progress items, and finished goods are systematically stored and retrieved, optimizing space utilization and ensuring the availability of materials for continuous production. Key aspects include:

- **High-Bay Storage**: This area is designed for efficient storage, maximizing vertical space to accommodate a large volume of workpieces in a compact footprint. It ensures that materials are safely stored until needed for production.

- **Automated Retrieval System**: Integrated with the factory's control system, the warehouse storage features an automated retrieval system that efficiently locates and moves items to and from the production line. This system is critical for maintaining a seamless flow of materials, reducing wait times, and increasing overall productivity.

Each of these components was modeled individually and their model was also running on different instances. This allows technicians with domain-specific knowledge can create and maintain the machine models that they are responsible for.

To illustrate one of the several benefits of this approach. For demonstration of the benefits of knowledge graphs and inferences that can be made based on them, a failure in color detection has been added.

The next section explains in more detail how the Inbound Delivery and Storage Process works.

Inbound Delivery and Storage Process

For this thesis, the inbound delivery storage process has been implemented using the proposed approach. Figure 7.3 illustrates this process with its main steps.

- **1. Introduction of Workpiece**: Detected by a light barrier, signaling delivery.

- **2. Activation of Vacuum Gripper Robot:** Moves to pick up the workpiece.

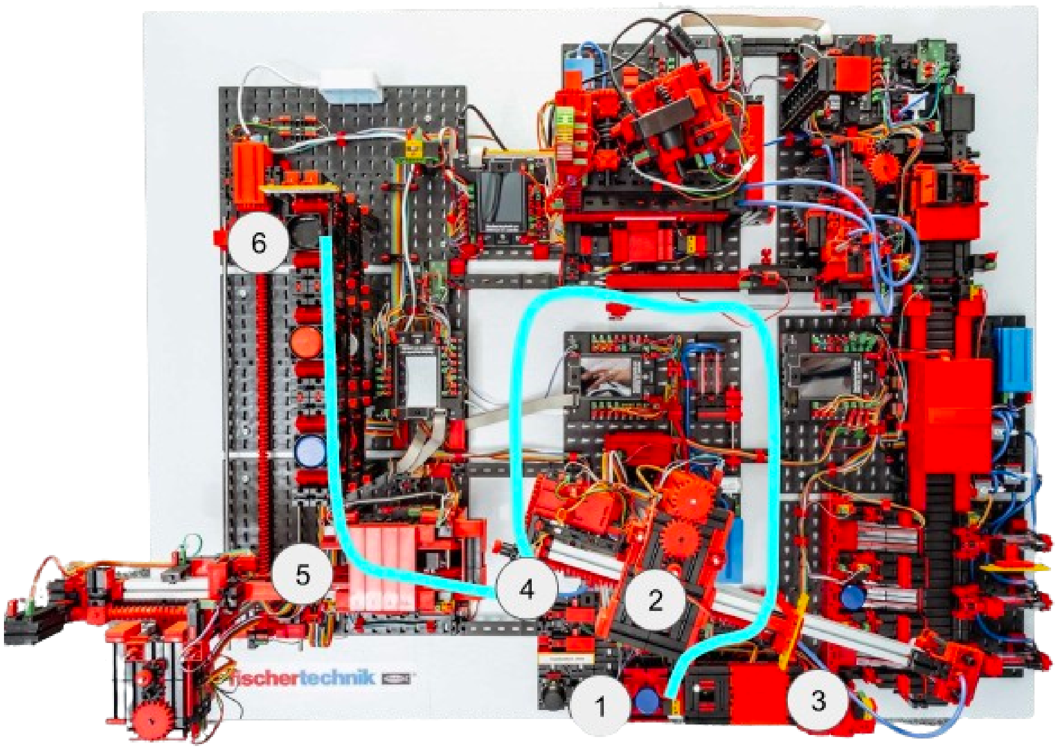


Figure 7.3.: Inbound Delivery and Storage Process

- **3. Color Detection:** Workpiece color identified by the sensor. **NFC Tag Processing:** Data on the tag is erased and then rewritten with new information including color, delivery specifics, and quality control data.
- **4. Gripper transports the piece to storage:** After the piece is identified, it is transported by the robot to the storage warehouse.
- **5. Container Handling:** An empty container is provided, filled with the workpiece. and then stored in the high-bay warehouse.
- **6. The piece is stored:** Finally the piece is stored in the warehouse.

In the next section, details on how the proposed concepts of this thesis can be applied are presented.

Application of the proposed approach

The first step defined in the methodology (Chapter 6) is to **Identify the Observable Assets** (Section 6.1). For this use case, the first identified assets are the main stations: The delivery and Pickup Station, the Gripper Robot, the Warehouse Storage, and the Work Piece that enters the line and is stored in the storage.

The "Pickup station" as referenced in Figure 7.4, is strategically positioned at the commencement of the process flow, serving as the gateway for items entering the system. It is engineered to seamlessly handle the initial identification and sorting of workpieces, leveraging detection technologies. This station is equipped with sensors and readers to ensure identification from the moment the workpieces are introduced into the system. The integration of a light barrier, an NFC (Near Field Communication) reader/writer, and color recognition capabilities are the main parts of this station.

The primary operations of the "Pickup station" are represented in its two main actions: *scanNFC* and *scanColor*. The *scanNFC* function facilitates the identification and verification of workpieces through embedded NFC tags, enabling the station to interact with each workpiece's digital data. The *scanColor* action, on the other hand, allows for the visual categorization of workpieces based on their color, supporting subsequent sorting and processing activities.

This station is also adept at managing system flow control through the generation of specific events. The *reqQuit* event allows for the interruption of the process flow, providing a mechanism to halt operations when necessary. NFC-related events, including *nfcTagOk*, *emptyTag*, and *nfcError*, offer detailed feedback on the NFC scanning process, highlighting successful scans, untagged workpieces, and errors, respectively. Similarly,

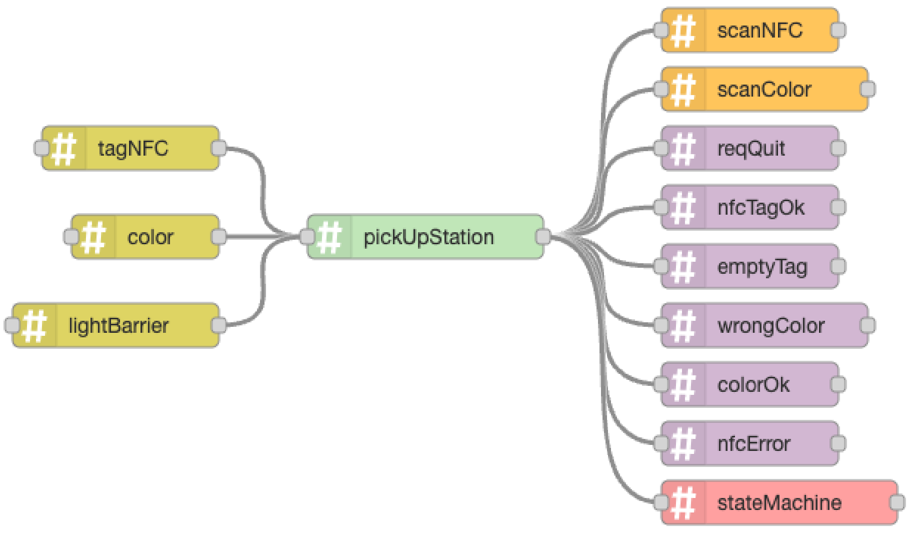
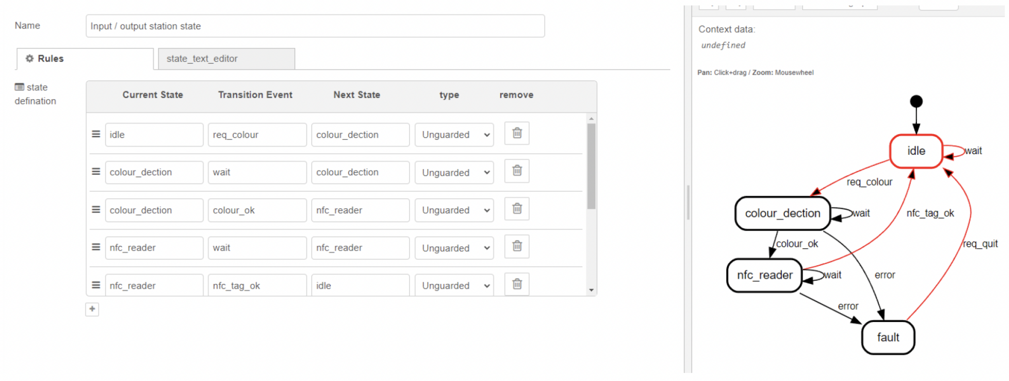


Figure 7.4.: Model of the Pickup Station

color-related events such as *wrongColor* and *colorOk* deliver immediate insights into the success or challenges encountered during color verification.

The "Pickup station" thus plays an integral role in setting the stage for the processing of workpieces throughout the system. Its functionalities not only ensure the smooth transition of workpieces into the system but also lay the foundation for their subsequent handling and processing, exemplifying its critical position at the outset of the workflow. The image 7.5 shows its behavior modeled in a state machine. This is part of the methodology step **Integrating External Models** (Section 6.5).


Name: Input / output station state

Rules | state_text_editor

state defination

Current State	Transition Event	Next State	type	remove
idle	req_colour	colour_dection	Unguarded	🗑
colour_dection	wait	colour_dection	Unguarded	🗑
colour_dection	colour_ok	nfc_reader	Unguarded	🗑
nfc_reader	wait	nfc_reader	Unguarded	🗑
nfc_reader	nfc_tag_ok	idle	Unguarded	🗑

Context data:
undefined

Pan: Click+drag / Zoom: Mousewheel

Figure 7.5.: Pickup station - State Machine

The state machine created for this asset has 4 states: *idle, colour_ detection, nfc_ reader, and fault.* When the system starts, it goes to the state *idle*. The transitions between these states are based on the events that are generated by the asset (see left table in Figure 7.5). For example, when the *scanNFC* action is performed, the asset generates the *nfcTagOk* event, which triggers the transition from *nfc_ reader* to the *idle* state. This state machine is updated with live data coming from the real-world twin and therefore, it shows the current state of the system.

An extension of the model (defined in 5.8.8) node in Node-RED has been developed to allow the creation of state machines. This extension allows the creation of state machines within the same environment the DT-models are created, using the same concepts that were used to create the DT-models, such as the events (defined in 5.8.7) and actions (defined in 5.8.6) of an asset. This extension was developed within the scope of a bachelor thesis at the University of Applied Sciences Hamm-Lippstadt (HSHL) under the supervision of the author and supervisor of this thesis.

Linking different models to form a more comprehensive representation of the system is an important aspect of DTs. Furthermore, keeping all these models updated with live data from the real world is crucial to ensure that the digital twin is an accurate representation of the physical system. With the proposal approach, whenever there is a change in the system's state, these modifications are propagated to the external models (such as the state machine), enabling their timely update.

The Gripper is illustrated in the Figure 7.6. It is composed of the properties *position*, *pressure*, *temperature*, and *load*. It has the actions *moveToColorDetection* (that moves the piece to the color detection), and the *moveToStorage* (that moves the piece to the storage). It also has the events *requestQuite, fetched and next*.

The asset labeled "storage", Figure 7.7, serves as a crucial component in the management and organization of workpieces within the system. After the gripper transfers a workpiece into the storage section, the storage's main function is to allocate space for these items based on its current availability and capacity. The properties integral to the storage asset include "*workpieces*," which lists the items currently stored; "*history*," detailing the sequence of stored and retrieved items over time; and "*capacity*," defining the maximum number of workpieces it can accommodate.

In terms of operational capabilities, the storage can perform actions such as "*store-Container*," allowing it to accept a container either with or without a workpiece, and "*fetchWorkPiece*," which facilitates the retrieval of workpieces as needed. This asset is also designed to emit two specific events: "*workPieceStored*" signifies the successful stor-

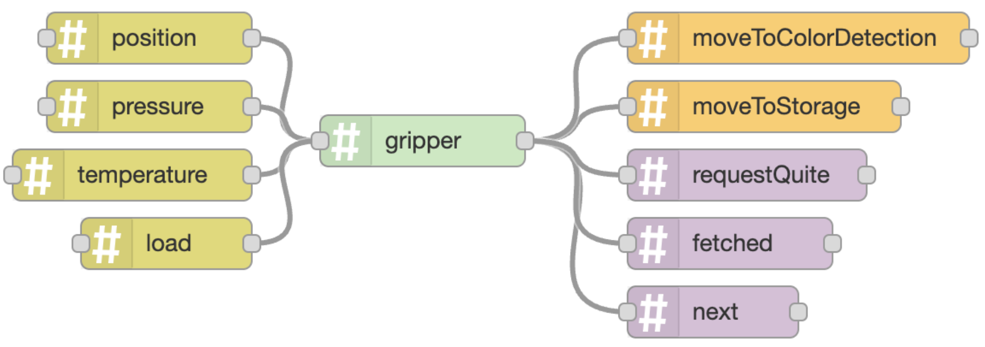


Figure 7.6.: Model of the Gripper

age of a workpiece, and "*error*" indicates any issues or malfunctions that occur during the storage process.

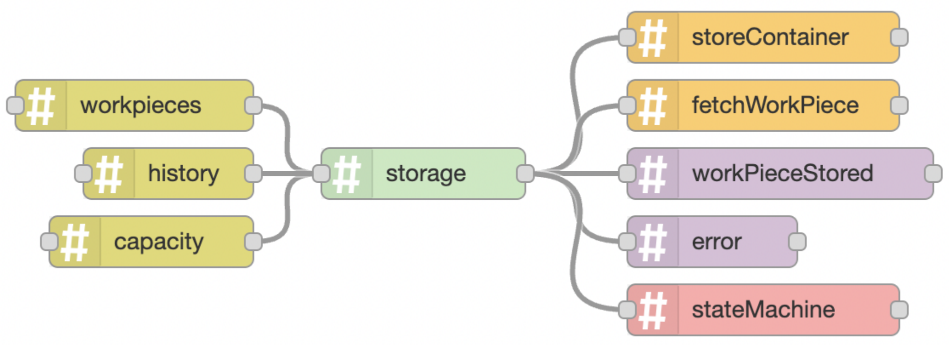
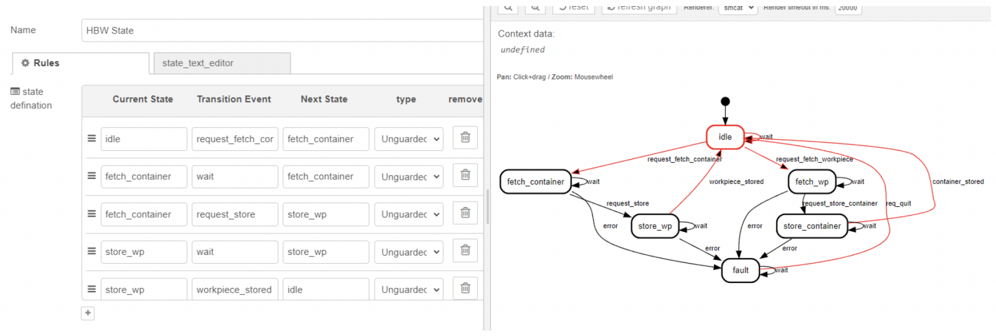


Figure 7.7.: Storage model

To further encapsulate its functionality and operational logic, the storage asset is complemented by an external model in state machine format, as shown in Figure 7.8. This model delineates the various states the storage can be in, such as empty, partially filled, or full, and the transitions between these states based on the actions performed, like storing or fetching workpieces. The state machine model provides a structured and clear representation of the storage's behavior, enhancing the predictability and efficiency of its operations within the system.


Name: HBW State

Rules | state_text_editor

state defination

	Current State	Transition Event	Next State	type	remove
≡	idle	request_fetch_cor	fetch_container	Unguardec	🗑
≡	fetch_container	wait	fetch_container	Unguardec	🗑
≡	fetch_container	request_store	store_wp	Unguardec	🗑
≡	store_wp	wait	store_wp	Unguardec	🗑
≡	store_wp	workpiece_stored	idle	Unguardec	🗑

Renderer: smcat

Render timeout in ms: 20000

Context data:
undefined

Pan: Click+drag / Zoom: Mousewheel

Figure 7.8.: Storage state machine model

The state machines are updated with live data coming from the real-world twin and therefore, it shows the current state of the system.

The asset named "workpiece" 7.9 is a central element within the system, characterized by distinct properties that include *type, color, and history*. The type property categorizes the workpiece, differentiating it based on its intended function or role within the system. The color property provides a visual identifier, enabling the system to apply color-based sorting or processing rules. Lastly, the history property tracks the workpiece's journey through the system, documenting interactions, processes undergone, and any changes in status. This comprehensive set of properties ensures that each workpiece can be accurately identified, tracked, and managed throughout its lifecycle in the system, facilitating efficient processing and quality control.

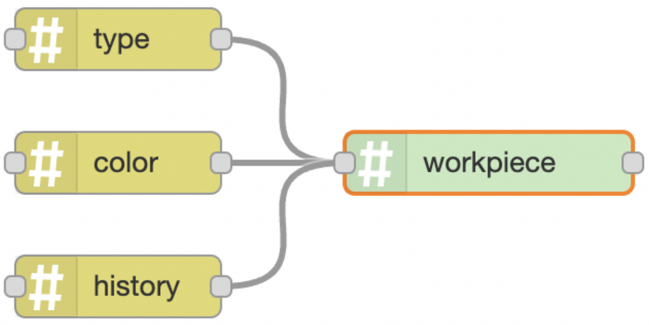


Figure 7.9.: Work piece model

After the main assets are identified and modeled, it is possible to add their relationships via semantic annotations. This represents the step "**Enhancing with Semantic Annotations**" of the methodology (Section 6.3).

Figure 7.10 presents a visual overview of the key assets within the system and the web of relationships that connect them. This model serves a critical function by abstracting away from the details of individual asset specifications to concentrate on how these assets interact within the broader system architecture. Such a high-level perspective is important in fostering an understanding of the system's functional dynamics without becoming overwhelmed by the complexity of each component's details.

This strategic focus on relationships rather than on asset details plays a pivotal role in the collaborative effort of system design and optimization. It enables stakeholders from various domains to contribute effectively to the system's development. For instance, by highlighting the connections between assets, the model allows a production engineer to identify and optimize the workflow and interdependencies of different sectors. This could involve streamlining the movement of workpieces between stations or ensuring efficient communication protocols are in place for asset coordination.

On the other hand, technicians or support engineers, with their rich technical expertise, can leverage the same diagram to understand how the assets they are responsible for fit into the larger operational scheme. This understanding is crucial for troubleshooting, maintenance, and the fine-tuning of machine performance, as it situates their technical interventions within the context of the system's overall functionality.

Moreover, this level of abstraction facilitates cross-disciplinary collaboration and knowledge sharing. It acts as a bridge between the macro-level understanding of system operations and the micro-level technicalities of asset functionality. Doing so ensures that the insights and innovations contributed by domain experts are harmonized toward the common goal of enhancing system performance.

In essence, the diagram presented in Figure 7.10 is more than just a schematic; it is a tool for strategic planning, a facilitator of interdisciplinary collaboration, and a catalyst for innovation. It underscores the importance of a holistic view in system design, where understanding the relationships between assets is just as crucial as understanding the assets themselves. This step is important to show how the assets are connected and how they can interact with each other.

It is possible to see on the model that either the technician or the pickUpStation can trigger the gripper to move the workpiece to the storage. In case this component is broken, the technician can trigger the gripper to move the workpiece to the storage. This is an example of how the relationships can be used to make inferences on the system.

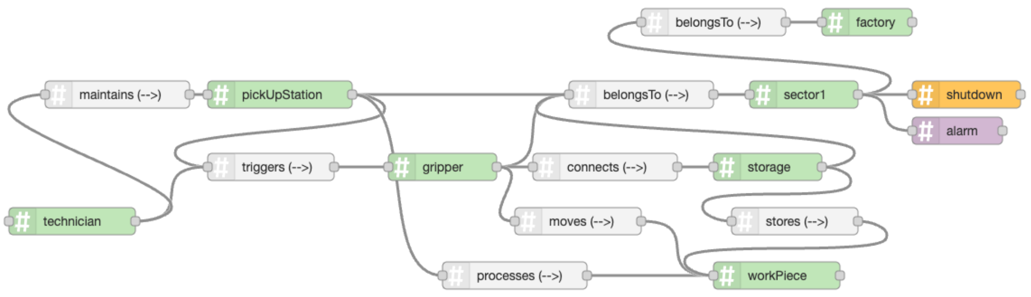


Figure 7.10.: Assets and their relationships

The gripper is connected to the storage and can move the workpiece to this location. All these assets belong to the *section1*, which contains an action shutdown and an event alarm. Finally, the *sector1* is connected to the asset factory.

As described in the previous chapters, an asset can also be something from the real world that is not physical, as for example, a process. In this case, the process of Inbound Delivery and Storage has been modeled as an asset, as can be seen in Figure 7.11.

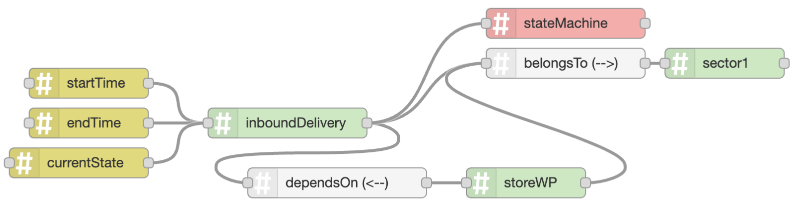


Figure 7.11.: Inbound Delivery and Storage Process

The Inbound Delivery process has properties such as start and end time as well as its current state. In the Figure, it is also possible to see that dependencies between processes can be modeled through semantic annotations like *dependsOn*. This is important to show how the processes are connected and how they can interact with each other.

These models were a simplified version of how a model of the factory could look like. The real model would have more details and more assets, but this is a good example of how the proposed concepts can be applied in a real-world scenario.

The next step of the methodology is to **Stablish the communication interfaces** with the real world. In this use case, the communication interface is established through the MQTT protocol.

Each asset has a topic in which it publishes its data. The data is published in a JSON format and it is sent to a broker. The broker is responsible for receiving the data and sending it to the subscribers. The subscribers are the assets that are interested in the data. In this scenario, the DT model is a subscriber of this data and it is responsible for updating the state of the assets based on the received data.

This step involves a considerable amount of technical work, as it deals with all the IoT details on how to communicate with the real world. Existing applications already have this kind of communication established, which makes it easier to integrate the proposed concepts. These existing applications can add a layer of abstraction to their existing communication interfaces so that the proposed concepts can be integrated.

After the communication channels with the real world are defined, the next step is to **System Deployment**. This step is responsible for deploying the models and the communication interfaces in the real world. This step outputs a knowledge graph of the current structure of the system, which can be used to make inferences and optimize the system.

In case the deployment is successfully done, both worlds should be connected and communicating. The real world should be sending data to the DT model and the DT model should be updating the real world based on the inferences made. Every time the real world changes, the DT model should be updated and a new version of the knowledge graph is generated.

Figure 7.12 shows a knowledge graph generated from the models (according definition described in Section 5.8) of the assets and their relationships. This knowledge graph is a representation of the current state of the system and it can be used to make inferences and optimize the system. For example, if the pickUpStation is broken, the technician can trigger the gripper to move the workpiece to the storage. This is an example of how the relationships can be used to make inferences on the system.

The transformation of the model of the assets and their relationships into a knowledge graph was done using the formal definitions explained in the previous sections. For example, the asset *technician* can trigger the asset *gripper* to move the workpiece to the storage. Transformation:

- **Assets -> Nodes:**
    - **pickUpStation**
        - ("pickUpStation", "id", "pickUpStation01")
        - ("pickUpStation", "name", "pickUpStation")
        - ("pickUpStation", "context", "Warehouse A")

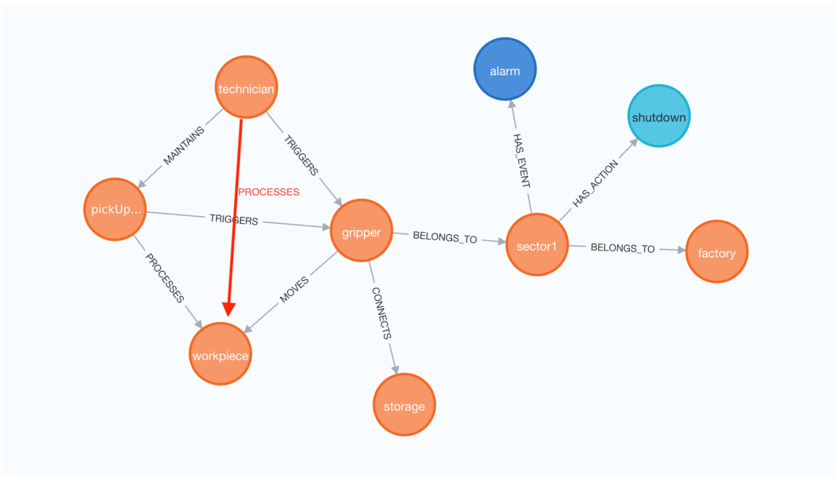


Figure 7.12.: Knowledge Graph of the Inbound Process

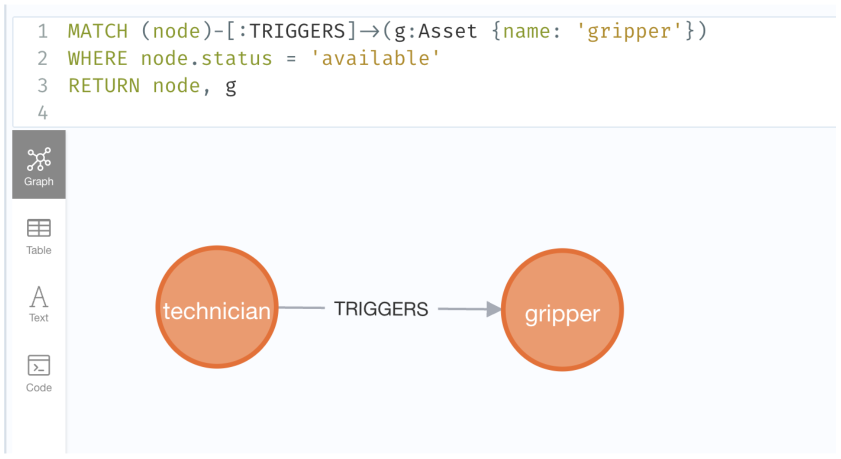
    * ("pickUpStation", "DTType", "Equipment")

- **gripper**
    * ("gripper", "id", "gripper01")
    * ("gripper", "name", "gripper")
    * ("gripper", "context", "Warehouse A")
    * ("gripper", "DTType", "Equipment")

- **Relationships -> Edges:**
  - **triggers**
    * ("triggers", "id", "trigger01")
    * ("triggers", "name", "triggers")
    * ("triggers", "direction", "->")

Queries and inferences can be performed on this knowledge graph of the system. For example, if the technician and the pickUpStation can trigger the gripper to move the workpiece to the storage, it is possible to make an inference to find out what is the current best strategy for replacing this equipment. And, as the KG is updated with live data, the inferences are very precise based on the current state of the system. Another inference that an AI algorithm can make is the relationship between the technician and the workpiece (in this case, it could be possible to infer that the technician processes the workpiece).

Figure 7.13 shows a query on the knowledge graph of the Assets and their relationship (model illustrated in Figure 7.10), forming a structural view of the system. It shows how the system can find out that the technician can trigger the gripper to move the workpiece to the storage because the pickUpStation is broken.



```
MATCH (node)-[:TRIGGERS]→(g:Asset {name: 'gripper'})
WHERE node.status = 'available'
RETURN node, g
```

Figure 7.13.: Query on the Knowledge Graph of the Assets and their relationships

The output of the methodology is the knowledge graph of the system. It can then be used by other applications that work as an interface to end users. For example, a dashboard that shows the current state of the system and the inferences made. This dashboard can be used by the technicians to make decisions based on the inferences made by the system.

Some of the possible inferences that could be made using the knowledge graph of the use case are shown in Table 7.1.

After the decision is made, the dashboard can send the decision to the DT model, which will update the real world based on the decision made. This completes the loop of the DT model, where it receives data from the real world, makes inferences, and updates the real world based on the inferences made or user decisions.

The next section shows how this use case can be mapped into the architecture proposed in this thesis.

Mapping components into the architecture

This section aims to show how the components of the use case can be mapped into the proposed architecture (Chapter 4). Figure 7.14 illustrates this mapping.

In the **Observe** layer, the real world is represented. According to ISO 23247, these assets could be Equipment, Personnel, Material, Process, Facility, Environment, Sup-

Asset (s)	Inference (s)	Description
Gripper Robot	Predictive Maintenance and Fault Diagnosis: Early Warning Signals	By monitoring the operational data (like temperature, pressure, and load) from the Gripper Robot and comparing it with historical patterns, the knowledge graph can be used to infer potential equipment failures before they occur. For instance, if the temperature and load on the gripper's motor exceed normal ranges, it could predict an impending failure, prompting preemptive maintenance actions.
Any asset	Predictive Maintenance and Fault Diagnosis: Fault Diagnosis	In the event of a breakdown, such as the color sensor malfunctioning, the knowledge graph can help diagnose the issue by correlating the failure with recent system changes or similar past incidents. This accelerates troubleshooting and repair, minimizing downtime.
Delivery and Pickup Station, Gripper Robot, and Warehouse Storage, Workpiece	Process Optimization: Bottleneck Identification	By analyzing the flow of workpieces through the Delivery and Pickup Station, Gripper Robot, and Warehouse Storage, the knowledge graph can be used to identify bottlenecks in the production process. For example, if workpieces are consistently delayed at the color detection stage, it might infer that the color sensor is a throughput-limiting factor and suggest process adjustments or equipment upgrades.
Warehouse Storage	Process Optimization: Resource Allocation	The knowledge graph can be used to infer optimal resource allocation strategies by analyzing the current workload and performance of the warehouse storage system. If the system detects an uneven distribution of workpieces, suggesting potential overloading of certain storage areas, a recommendation for reallocating the balance of the load to improve efficiency could be made.
Inbound Process	Scenario Planning and Simulation:	Before implementing changes in the production line, the knowledge graph can be used to simulate the impact of those changes, inferring potential outcomes based on historical and current operational data. This helps in making informed decisions about equipment upgrades, process changes, or new workflows.

Table 7.1.: Possible inferences that can be done using the knowledge graph of the use case

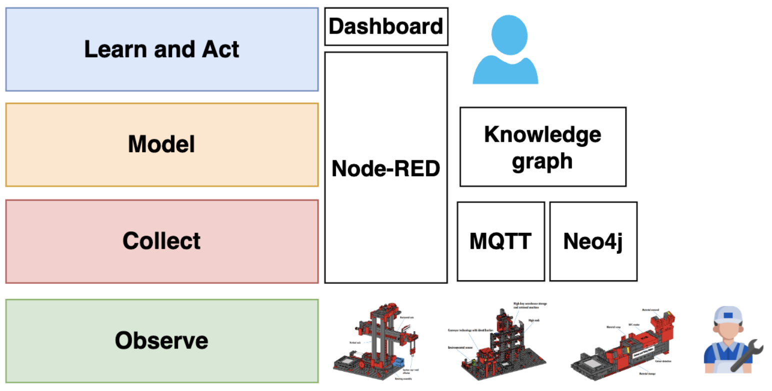


Figure 7.14.: Mapping of the use case into the proposed architecture

porting documents, and Product (also represented in Figure 4.1). A Generic type has been added to cover cases that might not fit in one of these classifications. For this use case, the assets are the Delivery and Pickup Station, the Gripper Robot, the Warehouse Storage, and the Work Piece. The Inbound Delivery and Storage Process is also an asset in this layer as well as the technician of the pickUpStation.

In the layer **Collect**, the communication and storage capabilities are represented. For this use case, Node-RED has been used to realize the IoT capabilities in a low-code way. The MQTT protocol has been used to communicate with the real world. The data and models are stored in a database Neo4j, which is a graph database that is suitable for storing and querying graph data.

For creating the semantic **Model**, Node-RED and the concepts and the extension developed in this thesis have been used. This allows the creation of semantic models within one environment. The models are then converted into a knowledge graph that supports representing nodes and their connections/relationships.

The last layer is the **Learn and Act**. In this layer the applications that work as an interface to end users are represented. For this use case, a dashboard from Node-RED shows the current state of the system. A state machine viewer has also been used to show the current state of the asset.

Having a layered architecture can help to separate the concerns and make the system more maintainable and scalable. It also helps to understand the system better and to replace or add new components without affecting the other layers.

Conclusion of the use case

The DT concept has become a key element in the advancement and implementation of Industry 4.0 paradigms, offering important insights into physical systems through their virtual counterparts. In the context of our use case, DT's application has been oriented towards a scenario characterized by more static structural compositions, where changes as new assets being added or removed occur infrequently.

The methodology proposed in this thesis was adopted for this use case, ensuring a comprehensive and systematic approach to the integration of DT within the specified scenario. Through this application, the use case was successfully mapped onto the proposed DT architecture, highlighting the seamless fusion of theoretical principles with practical implementation.

The knowledge graph serves as a potent tool for inference-making, offering a sophisticated mechanism to analyze, understand, and optimize the system. It encapsulates the relationships and dependencies within the system, enabling the identification of optimization opportunities and the anticipation of potential issues before they arise.

An example of inference was presented where it was possible to identify the best strategy for replacing a broken equipment. This demonstrates the practical utility of the knowledge graph in making informed decisions based on real-time data and historical patterns. Several other inferences were also described, as shown in Table 7.1, highlighting the diverse applications of the knowledge graph.

It is acknowledged that the presented use case is a simplification of a potentially more complex system. This simplification was intentional, designed to elucidate the core concepts and demonstrate the viability of applying DT technology in a real-world context. While the use case showcased a relatively straightforward application, DT's versatility allows for its extension to encompass more intricate systems, involving a broader spectrum of components and interactions. This adaptability underscores DT's potential to revolutionize system design, monitoring, and optimization across diverse industrial landscapes.

7.2. Automotive application in the context of smart city

This section presents the second use case, which is in the automotive and smart city context. This use case aims to showcase a more dynamic scenario, where the assets, relationships, and properties can change during the time. For example, now one car can have a relationship to another one, or a parking lot, but in some minutes this relationship is gone and the model has to adapt itself to this change.

This use case was developed in the context of the GAIA-X 4 AGEDA project, which is a German-funded project that aims to enable, via innovative middleware, vehicles to be part of the GAIA-X ecosystem.

In the next subsection, a deeper description of the use case is given.

Description of the use case

For this thesis, a simplified version of the use case Collective Vision and Control of the GAIA-X 4 AGEDA project has been used to demonstrate how the proposed concepts of this thesis can be applied in this kind of dynamic application.

The main idea of the whole use case is to enable vehicles can have a collective vision of the surrounding environment and can then perform a control based on this vision. The services and participants are registered and discoverable via GAIA-X services. The services are registered in the GAIA-X services and the participants are registered in the GAIA-X participants. The project GAIA-X 4 AGEDA aims then to enable the vehicles to participate in a GAIA-X dataspace, which is a federated data infrastructure that enables secure, trustworthy, and sovereign data exchange between different participants.

This middleware will enable vehicles to connect, consume and provide services with other participants, such as other vehicles, infrastructure, and service providers. This is especially important in the context of smart cities, where vehicles can communicate with each other and with the infrastructure to optimize traffic and to provide better services to the citizens. It can also enable traditional physical products such as vehicles to be offered as services as previously explored in [Ste+21b].

Figure 7.15 illustrates a scenario of a smart city where vehicles are connected to each other and to the infrastructure. All connected elements can interact and exchange data with each other.

The use case contains two main services: the Traffic Awareness Service that runs in the cloud and collects data from different participants to construct a collective vision; and the Object Detection Service that runs on the vehicles and uses its sensors to detect objects in the environment and then share with the Traffic Awareness Service.

This computational resource could also be provided inside a specific area (as presented in Figure 7.16) by another moving participant, such as a car, that has free resources at a specific time. This is an example of how the DT can be used to simulate new scenarios and make inferences based on the current state of the system. Queries and inferences such as illustrated in the first use case (Figure 7.13) can be similarly applied in this context.

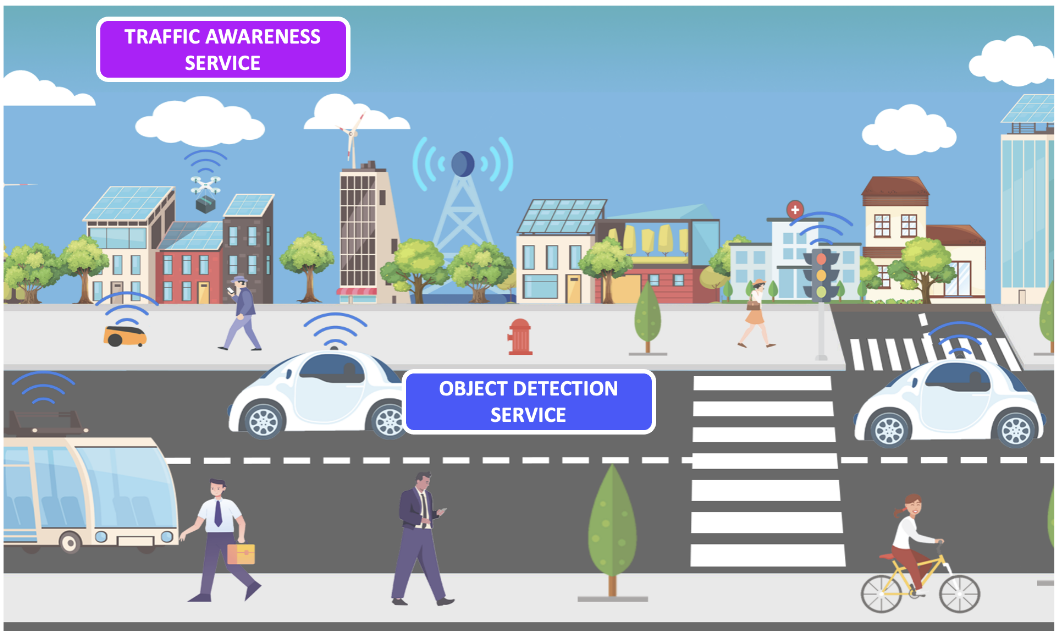


Figure 7.15.: Smart City scenario

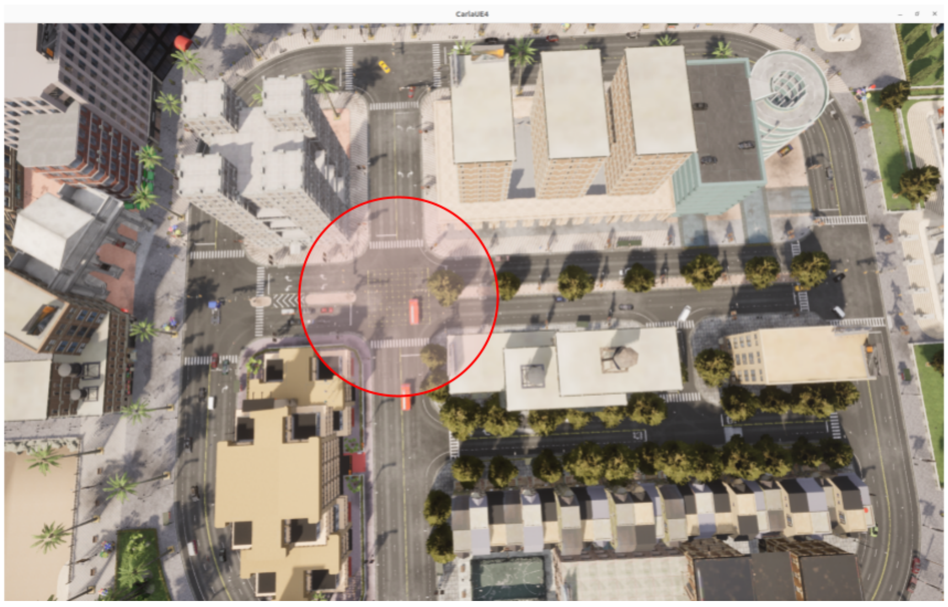


Figure 7.16.: CARLA simulator - Crossing with a service offer

Application of the proposed approach

As mentioned before, this is a simplification of the original use case from the project. The main idea is to show how the proposed concepts can be applied in a dynamic scenario.

The first step defined in the methodology (Chapter 6) is to **Identify the Observable Assets** (Section 6.1). For this use case, the first identified asset would be the vehicle. The vehicle has properties such as *position, speed, direction, timestamp, and vehicleResources*, as can be seen in Figure 7.17. Vehicles can also have a service asset called *objectDetectionService* that runs on the edge and sensors data to detect objects in the environment. This model represents the asset and its details, so it is a low-level model.

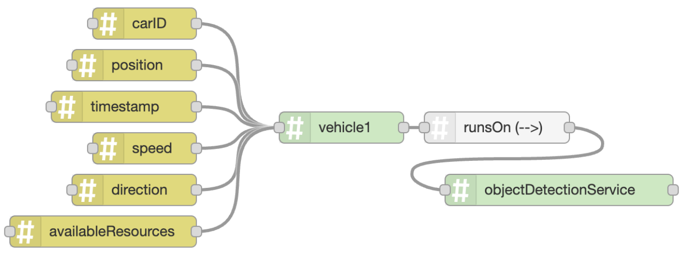


Figure 7.17.: Vehicle model

Figure 7.18 illustrates the relationship between the vehicles and the cloud-based service called *trafficAwarenessService*. This service is responsible for collecting data from different participants to construct a collective vision.

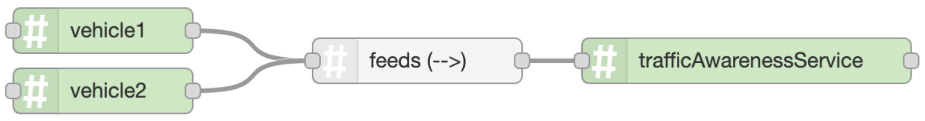


Figure 7.18.: Model of vehicles and the Traffic Awareness Service

In this use case, it is possible that vehicles come and go and dynamically participate in the system perceiving the surroundings and sending it to the cloud. As DTs deal with unique instances, models should also be able to adapt to these changes.

One possible way of handling this challenge is to use AI-based algorithms to try to automatically update the model based on the messages that are being exchanged in the system.

For example, if the system contains messages such as those described below in the Listing 7.1.

Listing 7.1: JSON message of one vehicle

```
    [
        {
          "carID": "Car_01",
          "position": { "latitude": 40.7128, "longitude":
             -74.0060 },
          "timestamp": "2024-02-20T12:00:00Z",
          "speed": 50,
          "direction": "north",
          "vehicleResources": [
            "camera", "lidar", "radar",
            "cpu": {"cores":4, "availability":35}
          ]
        }
    ]
```

In case two more vehicles are added to the system, the messages would be as shown in the Listing 7.2, and this data could be used to update the model of the vehicles by creating new instances of the vehicle model.

Listing 7.2: JSON message of three vehicles

```
    [
        {
            "carID": "Car_01",
            "position": { "latitude": 40.7128, "longitude":
               -74.0060 },
            "timestamp": "2024-02-20T12:02:00Z",
            "speed": 50,
            "direction": "north",
            "vehicleResources": [
            "camera", "lidar", "radar",
            "cpu": {"cores":4, "availability":50}
            ]
        },
        {
            "carID": "Car_02",
            "position": { "latitude": 40.7129, "longitude":
               -74.0061 },
            "timestamp": "2024-02-20T12:02:00Z",
            "speed": 55,
            "direction": "east",
            "vehicleResources": [
            "camera", "lidar", "radar",
            "cpu": {"cores":4, "availability":55}
            ]
        },
        {
            "carID": "Car_03",
            "position": { "latitude": 40.7130, "longitude":
               -74.0062 },
            "timestamp": "2024-02-20T12:02:00Z",
            "speed": 60,
            "direction": "south",
            "vehicleResources": [
            "camera", "lidar", "radar",
            "cpu": {"cores":4, "availability":10}
            ]
        }
    ]
```

By calling, for example, LLM-based solutions such as Chat-GPT4, it is possible to identify the assets from the exchange messages and create these instances in the model.

The activity diagram in Figure 7.19 illustrates the logic of the dynamic reconfiguration of the model based on the messages exchanged in the system.

In this use case, the MQTT protocol was used to communicate with the real world. The activity diagram for the dynamic reconfiguration of the model starts when a new message is received by the system. When a new message is received, the system checks if the message belongs to a model node by checking the id contained in the data structure.

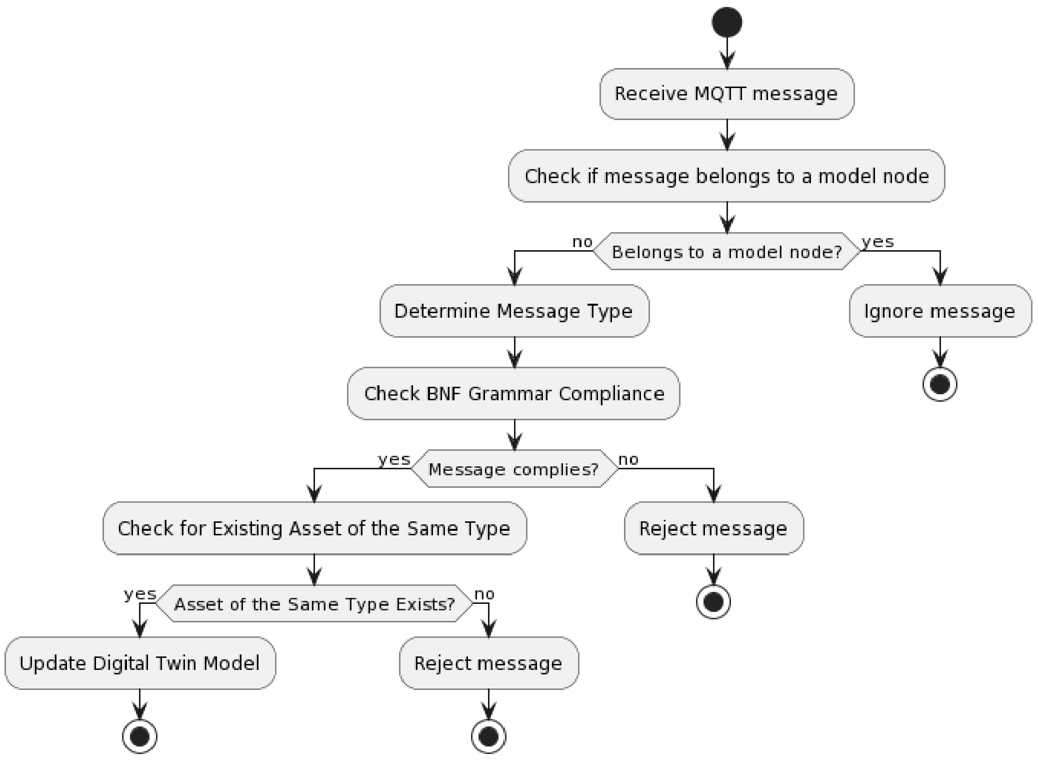

Receive MQTT message
Check if message belongs to a model node
no
Belongs to a model node?
yes
Determine Message Type
Ignore message
Check BNF Grammar Compliance
yes
Message complies?
no
Check for Existing Asset of the Same Type
Reject message
yes
Asset of the Same Type Exists?
no
Update Digital Twin Model
Reject message

Figure 7.19.: Activity diagram of the dynamic reconfiguration of the model based on the messages exchanged in the system.

If the message belongs to an existing node, it is ignored by the dynamic reconfiguration process.

If the message does not belong to an existing node, the system determines the message type. This is done via a call to LLMs pre-trained with the modeling elements proposed in this thesis. This algorithm receives an MQTT message and returns a model with de modeling elements defined in Chapter 5.

This returned model is then checked against the current model to see if it complies with the BNF grammar (defined in Section 5.9). If it does not comply with the grammar, (e.g. the returned model contains two assets connected without an explicit relationship), the message is rejected.

If the message complies with the grammar, the system checks if there is an existing asset of the same type (e.g. the LLM suggested a model of a product, then it is checked if this type is already known in the model if there are other products in the model). If there is an existing asset of the same type, the system updates the digital twin model. If there is no existing asset of the same type, the message is rejected. This technique can be highly optimized and other strategies can be used to validate the message without the need of a human in the loop.

In this context, it is possible to formalize when a message is valid or not. Let M be the set of all messages exchanged by elements in the system. The validation function could be defined as $Valid : M \to \{True, False\}$ that determines whether a given message m from M is valid based on its adherence to a grammar G (Section 5.9) and the presence of other objects of the same type within the model.

G represents the BNF grammar defining the valid structure of messages. A message m is said to comply with G if it can be derived from the start symbol of G using the production rules of G.

Let $Type(m)$ return the type of the asset described by message m (e.g., "product"). Let $Assets_{Type}$ be the set of all currently modeled assets of the type returned by $Type(m)$.

The formal validation logic can then be expressed as follows:

$$Valid(m) = \begin{cases} True & \text{if } m \in G \text{ and } |Assets_{Type(m)}| > 0, \\ False & \text{otherwise.} \end{cases}$$

This logic ensures that a message m is considered valid if and only if it complies with the BNF grammar G and there exists at least one asset of the same type as $Type(m)$ in the current model.

At each valid change in the system, a new state of the model is generated and can be saved to keep track of the whole lifecycle of this dynamic system.

Based on the lifecycle, stakeholders can run learning algorithms to understand the behavior of the system and make inferences based on the current state of the system.

Conclusion of the use case

This use case aimed to demonstrate how the proposed concepts can be applied in a dynamic scenario. The main idea was to show how assets, relationships, and properties can change over time.

The level of detail was not so deep as in the first use case, but it was enough to show how the proposed concepts can be applied in this kind of dynamic application.

With the DT it is also possible to simulate new scenarios and make inferences based on the current state of the system, for example, add a cluster as a service offering in a crossing and see how the cars would react to this new service. It would enable, for example, cars with less processing capabilities could use the processing capabilities of the cluster to make decisions.

The idea of having automatic model updates based on the messages exchanged in the system is a promising approach to handling the dynamic nature of this kind of application. It is important to notice that this is a simplification of the original use case from the project. Additionally, in a real-world application, the updates and new versions of the models should be monitored and validated by domain experts to ensure that the model is correctly updated. Keeping the human in the loop is crucial to ensure that the model is correctly updated and that the inferences made are correct, since LLMs can make mistakes and generate incorrect answers.

There is a clear need for more research in this area to understand how to handle the dynamic nature of the assets and relationships in the DT. This is a promising area for future research and development. Also, not known asset structures can be an issue to be classified and modeled.

8. Evaluation

This chapter presents the evaluation of the proposed approach. The evaluation is done in three parts: the first part presents a feature-based comparison with other tools, the second part evaluates the modeling approach, and the third part evaluates the integrated environment based on Node-RED in terms of execution time and scalability.

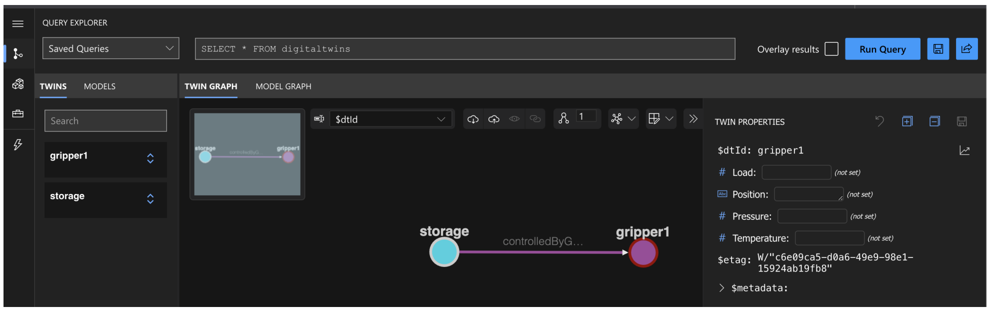
8.1. Feature-based comparison with other tools

As seen in the related works, there are several tools, concepts and methods that can be used to model digital twins available in the market. This section aims to compare the main features of the proposed approach with other tools and methods that can support the creation of DT applications.

The Azure IoT platform (Figure 8.1) is a cloud-based solution that provides a set of services for building and managing IoT applications. It provides a set of tools for building DTs, including a modeling language (DTDL), a set of APIs for interacting with the DT, and a set of tools for monitoring and managing the DTs. As it is a proprietary solution, it requires some effort for setting up as well as adding financial information. It is well integrated with other Azure services, which can be a good point for companies that are already using Azure. However, it is not flexible and it is not possible to run the model on the edge. It also does not support events. This thesis approaches all these points in a different way, as it is open-source, it is possible to run the model on the edge, it is flexible and it supports events.

Azure Digital Twins enables the use of semantics and works with KGs, however, it does not follow the WoT standard nor ISO 23247, which are two advantages of this thesis.

Comparing the current approach with AWS IoT TwinMaker it is possible to identify that the main focus is different from both approaches. The AWS solution focuses more on the integration of 3D models with IoT data and it does not emphasize semantics and interoperability. The AWS solution is also not open-source, which is a disadvantage when compared to the proposed approach.



```
SELECT * FROM digitaltwins
```

```
$dtId: gripper1
# Load: (not set)
Position: (not set)
# Pressure: (not set)
# Temperature: (not set)
$etag: W/"c6e09ca5-d0a6-49e9-98e1-
       15924ab19fb8"
> $metadata:
```

Figure 8.1.: Azure Digital Twins tool

MindSphere from Siemens is a cloud-based IoT platform that, contrary to the proposed approach, can not run on edge devices. It is also not open-source and it is not flexible. It is well integrated with Siemens' hardware and software solutions, which can be a good point for companies that are already using Siemens. It also does not support events, which is a disadvantage when compared to the proposed approach.

The ability of the system to make inferences based on the current state of the system is a powerful tool for decision-making and is not present on most of the available tools in the literature. Models should be able to be updated and adapted as the system changes, and not only be a static representation of the system at a given time and only the properties being updated. This feature is also a key point for the proposed approach, compared to the other tools. The updated models can then be used to perform precise queries and inferences. It is important to notice that the inferences are only as good as the data that is being fed to the system. Therefore, it is important to have good data and model quality.

8.2. Evaluation of the Modeling Approach

This section evaluates the basic elements for modeling Digital Twins (DTs) and their implementation on Node-RED. The evaluation addresses four research questions and their corresponding hypotheses, providing detailed insights into the feasibility, applicability, and effectiveness of the proposed approach.

8.2.1. Generic and Extensible Modeling Approach (RQ1 & H1)

The proposed nodes are generic elements (Chapter 5) designed to model the structure of DT applications across various domains. This genericity aligns with Hypothesis 1.3.1, suggesting that a universal modeling approach can be created while maintaining the depth and utility of current domain-specific models.

The implementation of these nodes in diverse scenarios, such as industrial and smart city applications (Chapter 7), demonstrates their flexibility. Compared to existing approaches, this universal framework allows for the design of systems from multiple domains using the same modeling language, significantly reducing the learning curve for users and lowering development time across projects.

In addition to flexibility, the approach supports extensibility (Section 5.5). New nodes can be added to the model to represent additional assets or relationships, making it adaptable to evolving requirements or new use cases. This ability to extend the model ensures that it remains relevant and can be updated as necessary without requiring a complete overhaul, which is a significant advantage in maintaining long-term scalability. The implementation of these nodes in Node-RED (Section 5.10) further demonstrates their practical applicability, especially given Node-RED's capacity to integrate seamlessly with real-world IoT systems.

Moreover, Node-RED's connectivity to physical devices ensures that the DT models remain synchronized with their real-world counterparts, enabling online updates. This connectivity enhances the overall utility of the proposed framework by ensuring continuous alignment between the digital model and the physical world.

8.2.2. Parallel Operation and Edge Deployment (RQ2 & H2)

The ability to execute models in parallel, particularly close to edge devices, is a key advantage of the proposed approach. This capability, as discussed in Section 5.10, aligns with Hypothesis 1.3.2, which suggests that DT models can operate concurrently on edge devices and be maintained by domain experts. Implementing these nodes on Node-RED enables models to run efficiently in parallel, significantly enhancing both scalability and model fidelity. By allowing maintenance directly on edge devices, system experts can ensure that the models are kept up-to-date and reflect real-world conditions accurately.

In contrast, many existing tools face limitations when it comes to edge deployment. For example, platforms like AWS IoT TwinMaker and Microsoft Azure Digital Twins are heavily cloud-dependent and struggle with deployment on edge devices due to high computational requirements and the need for continuous internet connectivity. These tools often rely on proprietary technologies that demand centralized processing power, which makes running them on resource-constrained edge devices infeasible. Additionally, the closed nature of some proprietary platforms makes it difficult for domain experts to maintain or update models independently without relying on external support or vendor-specific modifications.

In comparison, the proposed approach using Node-RED is lightweight and capable of running on edge devices with limited computational capabilities. Node-RED's support for decentralized deployment enables online operation and parallel execution of models, even on devices like Raspberry Pi or other constrained hardware. This flexibility, combined with the use of standardized languages, allows domain experts to directly interact with and update the models on-site, without the bottlenecks typically associated with cloud-reliant solutions. Thus, the proposed method not only demonstrates feasibility but also outperforms existing tools in terms of adaptability to edge environments, thereby validating Hypothesis 1.3.2.

8.2.3. Standardization and Interoperability (RQ3 & H3)

The proposed approach adheres to ISO 23247 and WoT standards, which is a key-player for achieving interoperability in heterogeneous environments. This compliance ensures a consistent representation of assets and their properties, enabling seamless communication and data exchange across different platforms and devices. This standardization aligns with Hypothesis 1.3.3, which posits that embedding ISO 23247 and WoT principles into DT design promotes interoperability. The implementation examples in Chapter 7, such as Figures 7.6, 7.7, and 7.9 show device-level models, while Figure 7.10 illustrates a higher-level model of the assets and their relationships, and Figure 7.11 depicts a process model. All these models use the same language, demonstrating the approach's capability to maintain interoperability across different levels and types of models.

In contrast to many existing approaches that rely on proprietary protocols or domain-specific standards, which restrict interoperability across industries and platforms, the proposed approach, by embedding ISO 23247 and Web of Things (WoT) principles, provides a more open and flexible solution. While ISO 23247 is primarily designed for the manufacturing sector, its framework is adaptable across various domains, ensuring that asset data is consistently represented in a standardized format. The WoT standard further enhances this by enabling uniform interaction between devices and platforms, regardless of their underlying protocols. This universal model allows for seamless integration and data exchange across different layers and systems, fostering a more cohesive and scalable DT ecosystem.

8.2.4. Semantic and Connected Models

The development of semantic models that are connected to the real world and interpretable by both machines and human stakeholders has been demonstrated as feasible and effective. This validation is rooted in the practical implementation of the proposed approach using Node-RED (Section 5.10), which supports the creation and deployment of such models. The semantic nature of these models allows for automated processes and analytics by machines, while the standardized modeling language and graphical interface of Node-RED make the models accessible and understandable to human stakeholders. This accessibility fosters deeper and more intuitive interactions, particularly for those situated close to the edge of the network. Additionally, having the model connected to the physical twin enables the system to perform precise inferences based on actual data, as exemplified in Table 7.1 and in the dynamic of the use case in Smart City (Section 7.2).

While the scope of this work focused on non-real-time applications, this limitation should be noted. Real-time requirements were not covered, meaning the virtual representation cannot react within a defined deadline. Addressing this limitation in future work could involve exploring methods to enhance the system's real-time capabilities.

8.3. Evaluation of the integrated environment based on Node-RED

To evaluate the performance of the proposed approach, a set of tests were conducted. The tests were performed on a Raspberry Pi 3 Model B with 1GB of RAM and a 16GB microSD card. The operating system used was Raspbian GNU/Linux and the Node-RED version used was v3.0.0-beta.4, running on Node.js version: v19.7.

The tests were performed by injecting data into the model and measuring the time it took to generate the knowledge graph. The data was injected in the model and the KG was generated. The time was measured from the moment the data was injected until the KG was generated. IoT-related times were not considered.

Each asset used in this test was a simple object with five properties. The property was a string with a random value. The tests were performed with different numbers of assets: 1, 10, 50, 100, 500, and 1000.

One thousand executions were performed for each test and the average time was calculated. The jitter was calculated as the standard deviation of the execution times. The minimum and maximum times were also recorded. The table 8.1 and Figure 8.2 show how the Node-RED reacts when adding more nodes to the model.

Assets	AVG Execution Time	Jitter	Min	Max
1x	3.48	1.69	1.53	9.12
10x	19.77	2.12	9.64	23.19
50x	72.91	14.69	65.00	122.01
100x	186.10	19.55	173.23	275.78
500x	594.21	58.30	522.00	734.06
1000x	1739.31	74.99	1585.00	1957.42

Table 8.1.: Execution time (in ms) of the software with different parameters

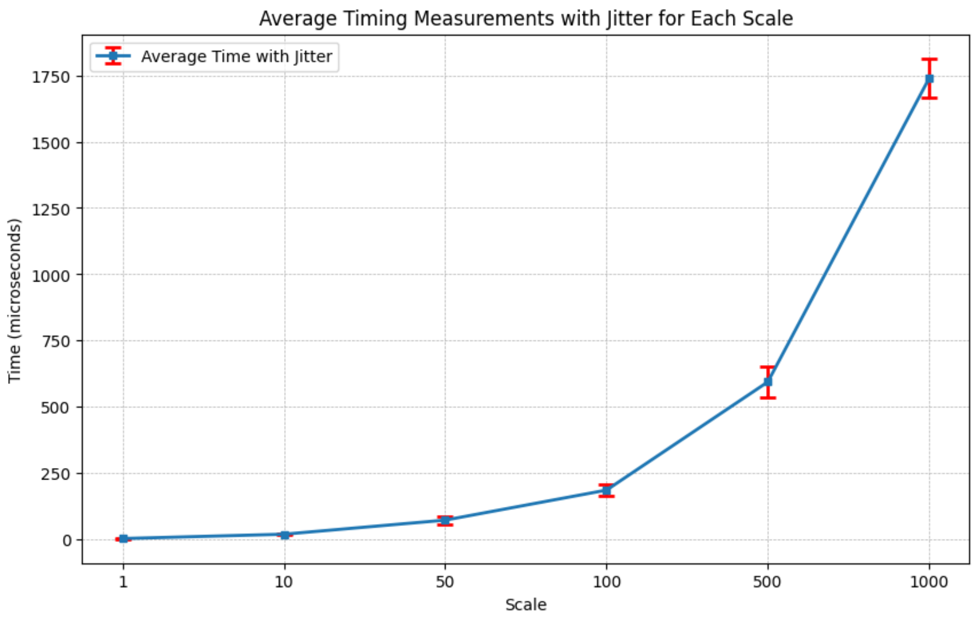


Figure 8.2.: Execution time (in ms) of the software with different parameters

As the number of assets increases, the average execution time increases significantly. This is expected in most systems as more assets typically mean more data to process, more computations, and potentially more network traffic in the case of distributed models implementations.

The jitter increases with the number of assets, but not linearly. The relative increase in jitter from 1x to 10x assets is smaller compared to the jump from 100x to 500x assets. This indicates that as the system scales, the variability in execution time becomes more significant, suggesting that the system might be facing scaling challenges at higher loads.

The minimum and maximum execution times provide insight into the best and worst-case scenarios. The variability increases in execution time as the system is scaled. This is particularly evident in the jump from 100x to 500x and then to 1000x assets, where the maximum execution time nearly doubles from 275.78ms at 100x assets to 734.06ms at 500x assets, and then more than doubles again to 1957.42ms at 1000x assets.

In summary, your Node-RED implementation's performance degrades as the number of assets increases, with both the average execution time and variability in execution times increasing. This suggests that while the system can handle smaller numbers of assets relatively efficiently, it faces significant challenges in scaling, potentially requiring optimization for better resource management, parallel models, or other scaling techniques to improve performance at higher asset counts. However, big models could also lead to readability and maintainability issues, which is a trade-off that should be considered.

Real-time applications can be monitored in runtime and criterias that are semantically defined in the model can be checked. This feature can be useful for defining future actions that do not need to be within a deadline. For example, an asset *task1* has a property *execution time* that is continuously taking longer than expected. An action could be taken by reviewing why this behaviour is happening and a solution can be found for future executions.

Security and safety aspects were not considered in this work. The proposed approach does not have any security features, such as authentication, authorization, and encryption. It is important to notice that for using this approach in a real-world scenario, it is important to have a good safety and security strategy in place, which might lead to additional execution time.s

9. Conclusion

This chapter concludes the thesis, summarizing the main findings and contributions, and suggesting future research directions.

9.1. Summary of the Work

This thesis aimed to investigate the modeling and deployment of semantic digital twins and their application across various domains. The primary objective was to propose an integrated environment for modeling and deploying semantic digital twins and to validate it through real-world use cases.

The literature review (Chapter 2) presented concepts related to digitalization, IoT, DT, and the state-of-the-art in modeling and deployment techniques. It was identified that digital twins are a promising technology for current and future digitalized systems, necessitating standards and methodologies for effective modeling and deployment.

In Section 2.6, a deeper examination of the DT concept was conducted, providing a definition focused on the use of semantic technologies for modeling and deployment.

Chapter 3 presented the state-of-the-art in modeling and deployment techniques for digital twins. It was found that there are several modeling techniques for digital twins, such as AutomationML, AAS, ontologies, and others. There are also several deployment strategies for digital twins, but most of them focus on cloud-based solutions and do not consider the edge as part of the twin. Additionally, there is still a need for standards and methodologies for modeling and deploying digital twins that consider the use of semantic technologies such as WoT, and standards such as the ISO 23247 which focuses specifically on DTs.

To fill this gap, the thesis proposes an architecture that can support in the building and maintenance of the system (Chapter 4), a set of modeling elements, compatible with the WoT and the ISO 23247, that can be used to semantically model digital twins in different domains and levels of detail (Chapter 5), and a methodology for deploying semantic digital twins in different environments (Chapter 6).

The research conducted in this thesis successfully addressed all the formulated research questions. First, the generic and extensible modeling approach (RQ1) was validated, demonstrating that the proposed method allows for the creation of a universal modeling framework while maintaining depth and utility across different domains, as seen in its successful application in industrial and smart city scenarios. Regarding parallel operation and edge deployment (RQ2), it was confirmed that DT models can indeed operate concurrently on edge devices, providing significant advantages in terms of scalability and online system management. The research also demonstrated that the proposed approach adheres to ISO 23247 and WoT standards (RQ3), enhancing interoperability in heterogeneous systems, seamless data exchange across platforms, and promoting a more flexible, future-proof DT ecosystem. Finally, the proposed methodologies and tools simplified the process of extending existing applications with digital twin capabilities, which can lead to reduced development time. Through these findings, each research question was thoroughly examined and answered, confirming the validity of the hypotheses outlined at the start of the thesis.

9.2. Future Research Directions

The work presented in this thesis opens up several future research directions. Some of the possible future research directions include:

- Exploring new modeling techniques for semantic digital twins, such as the use of machine learning and deep learning for modeling complex systems.
- Models evolve during the lifecycle of the system. It is important to investigate how version control can be implemented in the proposed methodology.
- Applying the proposed methodology to other domains, such as smart agriculture and even try to combine different domains.
- Explore the use of containerization and orchestration tools for deploying semantic digital twins in different environments.
- Developing new tools and frameworks for modeling and deploying semantic digital twins, and evaluating their performance in real-world use cases.
- Investigating the use of semantic digital twins for enabling new applications, such as augmented reality, virtual reality, and mixed reality.

- Integrating models without losing consistency and coherence as well as automating this task is also a future research direction.
- Incorporating continuous dynamics into the state machine models to enhance the accuracy and fidelity of digital twins, particularly for systems governed by physical conservation laws, such as energy and fluid dynamics.

Research is needed to strengthen the human-in-the-loop concept, particularly by evaluating the benefits of semantic models as a common language for both humans and machines. This direction would focus on developing connected models that can be easily understood by stakeholders at different levels, especially those working near the edge of the network. By using knowledge graphs to bridge the gap between physical systems and their digital counterparts, future work can enhance operational transparency and decision-making.

Additionally, methods for extending existing tools and approaches to better support the Digital Twin concept should be studied. This research can focus on strategies to reduce development time and costs, which would facilitate broader adoption of Digital Twin technology across various domains. Simplifying the integration process and making it more accessible would help popularize the concept, promoting innovation while maintaining efficiency across industries.

These future research directions are expected to contribute to the advancement of the field of semantic digital twins, and to the development of new applications and services for digital application and beyond.

A. Annex

A.1. JSON example of a model

```json
    {
  "model": {
    "projectName": "project1",
    "projectId": "uuid1",
    "version": "1.0.0+1",
    "assets": [
      {
        "id": "8caf970c47b84ea6",
        "type": "dt-asset",
        "z": "52c43a8ac210a9ef",
        "name": "gripper",
        "dtType": "Generic",
        "aContext": "",
        "aId": "",
        "actions": [
          {
            "id": "dc665aa4cf49dac3",
            "type": "dt-action",
            "z": "52c43a8ac210a9ef",
            "params": {},
            "name": "moveToStorage",
            "topic": "",
            "payload": ""
          }
        ],
        "properties": [
          {
            "id": "e5ba81d75d4b83c8",
            "type": "dt-property",
            "z": "52c43a8ac210a9ef",
            "name": "position",
            "accessGroup": "",
            "aContext": "",
            "aId": "",
            "aType": ""
          },
          {
            "id": "13ace7450252822f",
            "type": "dt-property",
            "z": "52c43a8ac210a9ef",
            "name": "temperature",
            "accessGroup": "",
            "aContext": "",
            "aId": "",
            "aType": ""
          },
          {
            "id": "9cab9d7591b4ba84",
            "type": "dt-property",
            "z": "52c43a8ac210a9ef",
            "name": "pressure",
            "accessGroup": "",
            "aContext": "",
            "aId": "",
            "aType": ""
          },
          {
            "id": "e4073145f20f4d8b",
            "type": "dt-property",
            "z": "52c43a8ac210a9ef",
            "name": "load",
            "accessGroup": "",
            "aContext": "",
```

```json
            "aId": "",
            "aType": ""
          }
        ]
      }
    ]
  }
}
```

List of Figures

List of Tables

Bibliography

[Adl16] Ahmed El Adl. *The Cognitive Digital Twins: Vision, Architecture Framework and Categories.* Technical Report. 2016. URL: https://www.slideshare.net/slideshow/embed_code/key/JB60Xqcn.

[Ahe+21] Shohin Aheleroff et al. "Digital Twin as a Service (DTaaS) in Industry 4.0: An architecture reference model". In: *Advanced Engineering Informatics* 47 (2021), p. 101225.

[AIA22] Aerospace Industries Association (AIAA). *DIGITAL TWIN: DEFINITION & VALUE.* Accessed: 2022-03-30. 2022. URL: https://www.aiaa.org/docs/default-source/uploadedfiles/issues-and-advocacy/policy-papers/digital-twin-institute-position-paper-(december-2020).pdf.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[AIM17] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm". In: *Ad Hoc Networks* 56 (2017), pp. 122–140.

[Ama22] Amazon. *AWS IoT TwinMaker (Preview).* 2022. URL: https://aws.amazon.com/iot-twinmaker/.

[Ans+21] Seema Ansari et al. "Internet of Things: Technologies and applications". In: *Introduction to Internet of Things in Management Science and Operations Research: Implemented Studies* (2021), pp. 1–30.

[ARJ19] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. "Internet of Things applications: A systematic review". In: *Computer Networks* 148 (2019), pp. 241–261.

[Ash+09] Kevin Ashton et al. "That 'internet of things' thing". In: *RFID journal* 22.7 (2009), pp. 97–114.

[Bao+22] Qiangwei Bao et al. "Ontology-based modeling of part digital twin oriented to assembly". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 236.1-2 (2022), pp. 16–28.

[BCT05] Alan W Brown, Jim Conallen, and Dave Tropeano. "Introduction: Models, modeling, and model-driven architecture (MDA)". In: *Model-Driven Software Development*. Springer, 2005, pp. 1–16.

[BD24] Lina Bariah and Merouane Debbah. "The Interplay of AI and Digital Twin: Bridging the Gap between Data-Driven and Model-Driven Approaches". In: *IEEE Wireless Communications* (2024), pp. 1–7. DOI: `10.1109/MWC.133.2200447`.

[BG11] Radhakisan Baheti and Helen Gill. "Cyber-physical systems". In: *The impact of control technology* 12.1 (2011), pp. 161–166.

[BHW21] J Barrasa, AE Hodler, and J Webber. *Knowledge Graphs: Data in Context for Responsive Businesses*. O'Reilly Media, Newton, 2021.

[Bic+20] Jason Bickford et al. "Operationalizing digital twins through model-based systems engineering methods". In: *Systems Engineering* 23.6 (2020), pp. 724–750.

[BM24] Rebekka Benfer and Jochen Müller. "Semantic digital twin creation of building systems through time series based metadata inference–A review". In: *Energy and Buildings* (2024), p. 114637.

[Bor+21] Andrea Borghesi et al. "IoTwins: Design and implementation of a platform for the management of digital twins in industrial scenarios". In: *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. 2021, pp. 625–633.

[Bor+24] Maria Bordukova et al. "Generative artificial intelligence empowers digital twins in drug discovery and clinical trials". In: *Expert Opinion on Drug Discovery* 19.1 (2024), pp. 33–42.

[BR23] Manfred Broy and Bernhard Rumpe. "Development Use Cases for Semantics-Driven Modeling Languages". In: *Communications of the ACM* 66.5 (2023), pp. 62–71.

[BRJ97] Grady Booch, Jim Rumbaugh, and Ivar Jacobsen. "UML: unified modeling language". In: *Versão* (1997).

[Cha17] Devendra K Chaturvedi. *Modeling and simulation of systems using MATLAB and Simulink*. CRC press, 2017.

[CS20] Salvatore Cavalieri and Marco Giuseppe Salafia. "Asset Administration Shell for PLC Representation Based on IEC 61131–3". In: *IEEE Access* 8 (2020), pp. 142606–142621. DOI: `10.1109/ACCESS.2020.3013890`.

[Dal+18] Lucas Santos Dalenogare et al. "The expected contribution of Industry 4.0 technologies for industrial performance". In: *International Journal of Production Economics* 204 (2018), pp. 383–394. ISSN: 0925-5273. DOI: `https://doi.org/10.1016/j.ijpe.2018.08.019`. URL: `https://www.sciencedirect.com/science/article/pii/S0925527318303372`.

[DAm+22] Rosario Davide D'Amico et al. "Cognitive digital twin: An approach to improve the maintenance management". In: *CIRP Journal of Manufacturing Science and Technology* 38 (2022), pp. 613–630. ISSN: 1755-5817. DOI: `https://doi.org/10.1016/j.cirpj.2022.06.004`. URL: `https://www.sciencedirect.com/science/article/pii/S1755581722001158`.

[Dan+18] NgocSon Dang et al. "3D digital twin models for bridge maintenance". In: *Proceedings of the 10th International Conference on Short and Medium Span Bridges, Quebec City, QC, Canada*. Vol. 31. 2018.

[DJM22] Adeline Deprêtre, F Jacquinod, and A Mielniczek. "Exploring digital twin adaptation to the urban environment: comparison with CIM to avoid silo-based approaches". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4 (2022), pp. 337–344.

[DLV11] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. "Modeling cyber–physical systems". In: *Proceedings of the IEEE* 100.1 (2011), pp. 13–28.

[DO21] Juan Manuel Davila Delgado and Lukumon Oyedele. "Digital Twins for the built environment: learning from conceptual and process models in manufacturing". In: *Advanced Engineering Informatics* 49 (2021), p. 101332.

[EBA20] Itxaro Errandonea, Sergio Beltrán, and Saioa Arrizabalaga. "Digital Twin for maintenance: A literature review". In: *Computers in Industry* 123 (2020), p. 103316.

[Eke+03] Johan Eker et al. "Taming heterogeneity-the Ptolemy approach". In: *Proceedings of the IEEE* 91.1 (2003), pp. 127–144.

[EW16] Lisa Ehrlinger and Wolfram Wöß. "Towards a definition of knowledge graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4 (2016), p. 2.

[Far+17] Amirhossein Farahzadi et al. “Middleware technologies for cloud of things-a survey”. In: *Digital Communications and Networks* (2017). ISSN: 2352-8648. URL: `http://www.sciencedirect.com/science/article/%20pii/S2352864817301268`.

[FB18] Diogo Fernandes and Jorge Bernardino. “Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB.” In: *Data*. 2018, pp. 373–380.

[Fit21] Tira Nur Fitria. “Grammarly as AI-powered English writing assistant: Students’ alternative for writing English”. In: *Metathesis: Journal of English Language, Literature, and Teaching* 5.1 (2021), pp. 65–78.

[FM08] Gaia Franceschini and Sandro Macchietto. “Model-based design of experiments for parameter precision: State of the art”. In: *Chemical Engineering Science* 63.19 (2008), pp. 4846–4872.

[Gha+21] Taher M Ghazal et al. “IoT for smart cities: Machine learning approaches in smart healthcare—A review”. In: *Future Internet* 13.8 (2021), p. 218.

[Gho20] Morteza Ghobakhloo. “Industry 4.0, digitization, and opportunities for sustainability”. In: *Journal of cleaner production* 252 (2020), p. 119869.

[Göp+21] Amon Göppert et al. “Pipeline for ontology-based modeling and automated deployment of digital twins for planning and control of manufacturing systems”. In: *Journal of Intelligent Manufacturing* (2021), pp. 1–20.

[Gou+23] Sri Nikhil Gupta Gourisetti et al. “A theoretical open architecture framework and technology stack for digital twins in energy sector applications”. In: *Energies* 16.13 (2023), p. 4853.

[Gri14] M Grieves. “Digital twin: Manufacturing excellence through virtual factory replication”. In: *URL http://www. apriso. com* (2014).

[GS12] Edward Glaessgen and David Stargel. “The digital twin paradigm for future NASA and US Air Force vehicles”. In: *53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA*. 2012, p. 1818.

[GSH20] Mezzour Ghita, Benhadou Siham, and Medromi Hicham. “Digital twins development architectures and deployment technologies: Moroccan use case”. In: *International Journal of Advanced Computer Science and Applications* 11.2 (2020).

[GV17] Michael Grieves and John Vickers. “Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems”. In: *Transdisciplinary perspectives on complex systems: New findings and approaches* (2017), pp. 85–113.

[HR15] Martin Hankel and Bosch Rexroth. “The Reference Architectural Model Industrie 4.0 (RAMI 4.0)”. In: *ZVEI* 2.2 (2015), pp. 4–9.

[Hug+20] Jerome Hugues et al. “TwinOps-DevOps meets model-based engineering and digital twins for the engineering of CPS”. In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 2020, pp. 1–5.

[Hum+17] Abdulmalik Humayed et al. “Cyber-physical systems security—A survey”. In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1802–1831.

[ISO20] ISO. *Automation systems and integration — Digital Twin framework for manufacturing*. en. Standard ISO 23247-1:2020. International Organization for Standardization, 2020. URL: https://www.iso.org/standard/75066.html.

[Jab+20] Mohammad Ali Jabraeil Jamali et al. “IoT architecture”. In: *Towards the Internet of Things: Architectures, Security, and Applications* (2020), pp. 9–31.

[Jai+22] A Jain et al. *Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations*. Accessed: 2022-03-30. 2022. URL: https://eng.uber.com/uber-eats-graph-learning/.

[Jaz14] Nasser Jazdi. “Cyber physical systems in the context of Industry 4.0”. In: *2014 IEEE international conference on automation, quality and testing, robotics*. IEEE. 2014, pp. 1–4.

[Jia+23] Liu Jiang et al. “Intelligent control of building fire protection system using digital twins and semantic web technologies”. In: *Automation in Construction* 147 (2023), p. 104728.

[Jin+22] Lu Jinzhi et al. “Exploring the concept of Cognitive Digital Twin from model-based systems engineering perspective”. In: *The International Journal of Advanced Manufacturing Technology* 121.9 (2022), pp. 5835–5854.

[JU20] Michael Jacoby and Thomas Usländer. “Digital twin and internet of things — Current standards landscape”. In: *Applied Sciences* 10.18 (2020), p. 6519.

[Kal+20] Elem Güzel Kalaycı et al. "Semantic integration of Bosch manufacturing data using virtual knowledge graphs". In: *The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II 19*. Springer. 2020, pp. 464–481.

[Kal+21] Kostas Kalaboukas et al. "Implementation of cognitive digital twins in connected and agile supply networks—An operational model". In: *Applied Sciences* 11.9 (2021), p. 4103.

[KWH+13] Henning Kagermann, Wolfgang Wahlster, Johannes Helbig, et al. "Recommendations for implementing the strategic initiative INDUSTRIE 4.0". In: *Final report of the Industrie* 4.0 (2013), p. 82.

[Las+14] Heiner Lasi et al. "Industry 4.0". In: *Business & information systems engineering* 6.4 (2014), pp. 239–242.

[Le-+16] Danh Le-Phuoc et al. "The graph of things: A step towards the live knowledge graph of connected things". In: *Journal of Web Semantics* 37 (2016), pp. 25–35.

[Lee08] Edward A Lee. "Cyber Physical Systems: Design Challenges". In: *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. IEEE. 2008, pp. 363–369.

[Li+24] Johnny Li et al. "Banishing LLM hallucinations requires rethinking generalization". In: *arXiv preprint arXiv:2406.17642* (2024).

[Liu+21] Mengnan Liu et al. "Review of digital twin about concepts, technologies, and industrial applications". In: *Journal of Manufacturing Systems* 58 (2021), pp. 346–361.

[LS16] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. 2nd. The MIT Press, 2016. ISBN: 0262533812.

[LXZ15] Shancang Li, Li Da Xu, and Shanshan Zhao. "The internet of things: a survey". In: *Information systems frontiers* 17 (2015), pp. 243–259.

[Man+22] Monika Mangla et al. "Real-life applications of the Internet of Things: Challenges, applications, and advances". In: (2022).

[Mar+18] Petr Marcon et al. "The Asset Administration Shell of Operator in the Platform of Industry 4.0". In: *2018 18th International Conference on Mechatronics - Mechatronika (ME)*. 2018, pp. 1–5.

[MDR20] Tsega Y Melesse, Valentina Di Pasquale, and Stefano Riemma. “Digital twin models in industrial operations: A systematic literature review”. In: *Procedia Manufacturing* 42 (2020), pp. 267–272.

[Mic21] Microsoft. *Azure Digital Twins.* 2021. URL: `https://azure.microsoft.com/de-de/services/dig%20ital-twins/`.

[MLC20] Roberto Minerva, Gyu Myoung Lee, and Noel Crespi. “Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models”. In: *Proceedings of the IEEE* 108.10 (2020), pp. 1785–1824.

[Moy+20] James Moyne et al. “A requirements driven digital twin framework: Specification and opportunities”. In: *Ieee Access* 8 (2020), pp. 107781–107801.

[Neo22] Neo4j. *The Neo4j Graph Data Platform.* 2022. URL: `https://neo4j.com`.

[Niž+20] Sandro Nižetić et al. “Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future”. In: *Journal of cleaner production* 274 (2020), p. 122877.

[NN22] Nhan Nguyen and Sarah Nadi. “An empirical evaluation of GitHub copilot’s code suggestions”. In: *Proceedings of the 19th International Conference on Mining Software Repositories.* 2022, pp. 1–5.

[Nod22] NodeRED. *Node-RED Low-code programming for event-driven applications.* Accessed: 2021-08-15. 2022. URL: `https://nodered.org`.

[Pak+21] Harish Kumar Pakala et al. “Integration of asset administration shell and Web of Things”. In: *https://opendata. uni-halle. de//handle/1981185920/41505* (2021).

[PBD21] Rudolf Pribiš, Lukáš Beňo, and Peter Drahoš. “Asset administration shell design methodology using embedded OPC unified architecture server”. In: *Electronics* 10.20 (2021), p. 2520.

[PI21] Dessislava Petrova-Antonova and Sylvia Ilieva. “Digital twin modeling of smart cities”. In: *Human Interaction, Emerging Technologies and Future Applications III: Proceedings of the 3rd International Conference on Human Interaction and Emerging Technologies: Future Applications (IHIET 2020), August 27-29, 2020, Paris, France.* Springer. 2021, pp. 384–390.

[PM23] Foivos Psarommatis and Gokan May. “A literature review and design methodology for digital twins in the era of zero defect manufacturing”. In: *International Journal of Production Research* 61.16 (2023), pp. 5723–5743.

[Qam+21] Yassine Qamsane et al. “A methodology to develop and implement digital twin solutions for manufacturing systems”. In: *IEEE Access* 9 (2021), pp. 44247–44265.

[Qi+21] Qinglin Qi et al. “Enabling technologies and tools for digital twin”. In: *Journal of Manufacturing Systems* 58 (2021), pp. 3–21.

[Qiu+20] Tie Qiu et al. “Edge computing in industrial internet of things: Architecture, advances and challenges”. In: *IEEE Communications Surveys & Tutorials* 22.4 (2020), pp. 2462–2488.

[REC15] Karen Rose, Scott Eldridge, and Lyman Chapin. “The Internet of Things: An Overview”. In: *The internet society (ISOC)* 80 (2015), pp. 1–50.

[Rin99] David Rind. “Complexity and climate”. In: *science* 284.5411 (1999), pp. 105–107.

[Rod+23] Mario Sanz Rodrigo et al. “Digital Twins for 5G Networks: A modeling and deployment methodology”. In: *IEEE Access* (2023).

[RSK20] Adil Rasheed, Omer San, and Trond Kvamsdal. “Digital twin: Values, challenges and enablers from a modeling perspective”. In: *Ieee Access* 8 (2020), pp. 21980–22012.

[Sah+21] Nada Sahlab et al. “Knowledge Graphs as Enhancers of Intelligent Digital Twins”. In: *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. 2021, pp. 19–24. DOI: 10.1109/ICPS49255.2021.9468219.

[Sch+16] Greyce N Schroeder et al. “Digital Twin data modeling with automationml and a communication methodology for data exchange”. In: *IFAC-PapersOnLine* 49.30 (2016), pp. 12–17.

[Sch+20] Greyce N Schroeder et al. “A methodology for digital twin modeling and deployment for industry 4.0”. In: *Proceedings of the IEEE* 109.4 (2020), pp. 556–567.

[Sch+21a] Greyce N Schroeder et al. “Digital Twin connectivity topologies”. In: *IFAC-PapersOnLine* 54.1 (2021), pp. 737–742.

[Sch+21b] Greyce N. Schroeder et al. “A Methodology for Digital Twin Modeling and Deployment for Industry 4.0”. In: *Proceedings of the IEEE* 109.4 (2021), pp. 556–567. DOI: 10.1109/JPROC.2020.3032444.

[SDP12] Alberto Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. “Taming Dr. Frankenstein: Contract-based design for cyber-physical systems”. In: *European journal of control* 18.3 (2012), pp. 217–238.

[SG22] Mariana Segovia and Joaquin Garcia-Alfaro. “Design, modeling and implementation of digital twins”. In: *Sensors* 22.14 (2022), p. 5396.

[Sie+18] Seppo Sierla et al. “Automatic assembly planning based on digital product descriptions”. In: *Computers in Industry* 97 (2018), pp. 34–46.

[Son+16] Houbing Song et al. *Cyber-physical systems: foundations, principles and applications.* Morgan Kaufmann, 2016.

[SRP20] Sushil Kumar Singh, Shailendra Rathore, and Jong Hyuk Park. “BlockIoTIntelligence: A Blockchain-enabled Intelligent IoT Architecture with Artificial Intelligence”. In: *Future Generation Computer Systems* 110 (2020), pp. 721–743. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2019.09.002`. URL: `https://www.sciencedirect.com/science/article/pii/S0167739X19316474`.

[SSR22] Adam Sulak, Charles Steinmetz, and Achim Rettberg. “Communication Layer Architecture for a Production Line Digital Twin Using Hierarchical Colored Petri Nets”. In: *International Embedded Systems Symposium.* Springer. 2022, pp. 41–50.

[Ste+18a] Charles Steinmetz et al. “Internet of Things Ontology for Digital Twin in Cyber Physical Systems”. In: *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC).* 2018, pp. 154–159. DOI: `10.1109/SBESC.2018.00030`.

[Ste+18b] Charles Steinmetz et al. “Internet of things ontology for digital twin in cyber physical systems”. In: *2018 VIII Brazilian symposium on computing systems engineering (SBESC).* IEEE. 2018, pp. 154–159.

[Ste+18c] Charles Steinmetz et al. “Using Ontology and Standard Middleware for integrating IoT based in the Industry 4.0”. In: *IFAC-PapersOnLine* 51.10 (2018), pp. 169–174.

[Ste+20] Charles Steinmetz et al. “A digitalization concept for the interaction between users and car-as-a-service”. In: *2020 X Brazilian Symposium on Computing Systems Engineering (SBESC).* IEEE. 2020, pp. 1–8.

[Ste+21a] Charles Steinmetz et al. “Enabling and supporting car-as-a-service by digital twin modeling and deployment”. In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2021, pp. 428–433.

[Ste+21b] Charles Steinmetz et al. “Key-Components for Digital Twin Modeling With Granularity: Use Case Car-as-a-Service”. In: *IEEE Transactions on Emerging Topics in Computing* 10.1 (2021), pp. 23–33.

[Ste+22a] Charles Steinmetz et al. “A methodology for creating semantic digital twin models supported by knowledge graphs”. In: *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2022, pp. 1–7. DOI: `10.1109/ETFA52439.2022.9921499`.

[Ste+22b] Charles Steinmetz et al. “Digital Twins modeling and simulation with Node-RED and Carla”. In: *IFAC- PapersOnLine* 55.19 (2022), pp. 97–102.

[Ste+23] Charles Steinmetz et al. “Semantische digitale Zwillinge: Eine Methodik zur Digital-Twin-Erstellung mit Hilfe von Wissensgraphen”. In: *atp magazin* 65.5 (2023), pp. 69–77.

[Sto+18] Tim Stock et al. “Industry 4.0 as enabler for a sustainable development: A qualitative assessment of its ecological and social potential”. In: *Process Safety and Environmental Protection* 118 (2018), pp. 254–267. ISSN: 0957-5820. DOI: `https://doi.org/10.1016/j.psep.2018.06.026`. URL: `https://www.sciencedirect.com/science/article/pii/S0957582018303677`.

[Sto+21] Ljiljana Stojanovic et al. “Methodology and tools for digital twin management—The FA3ST approach”. In: *IoT* 2.4 (2021), pp. 717–740.

[Sun+22] Jiao Sun et al. “Investigating explainability of generative AI for code through scenario-based design”. In: *Proceedings of the 27th International Conference on Intelligent User Interfaces*. 2022, pp. 212–228.

[Sye+21] Abbas Shah Syed et al. “IoT in smart cities: A survey of technologies, practices and challenges”. In: *Smart Cities* 4.2 (2021), pp. 429–475.

[TA17] Erdal Tantik and Reiner Anderl. “Integrated data model and structure for the asset administration shell in Industrie 4.0”. In: *Procedia Cirp* 60 (2017), pp. 86–91.

[Tal+17] Jesús Martín Talavera et al. “Review of IoT applications in agro-industrial and environmental fields”. In: *Computers and Electronics in Agriculture* 142 (2017), pp. 283–297.

[Tao+18] Fei Tao et al. “Digital twin in industry: State-of-the-art”. In: *IEEE Transactions on industrial informatics* 15.4 (2018), pp. 2405–2415.

[Thu+20] Aparna Saisree Thuluva et al. “Semantic Node-RED for rapid development of interoperable industrial IoT applications”. In: *Semantic Web* 11.6 (2020), pp. 949–975.

[Thu22] Aparna Saisree Thuluva. “Semantic Driven Approach for Rapid Application Development in Industrial Internet of Things”. Available at `https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-791751`. PhD thesis. Dresden, DE: Technische Universität Dresden, May 2022.

[Til01] Michael Tiller. *Introduction to physical modeling with Modelica.* Springer Science & Business Media, 2001.

[TTA23] Valerian Vanessa Tuhaise, Joseph Handibry Mbatu Tah, and Fonbeyin Henry Abanda. “Technologies for digital twin applications in construction”. In: *Automation in Construction* 152 (2023), p. 104931.

[UWQ17] Jumyung Um, Stephan Weyer, and Fabian Quint. “Plug-and-Simulate within Modular Assembly Line enabled by Digital Twins and the use of AutomationML”. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 15904–15909.

[Wan+17] Quan Wang et al. “Knowledge graph embedding: A survey of approaches and applications”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743.

[Wan+22] Kai Wang et al. “A review of the technology standards for enabling digital twin”. In: *Digital Twin* 2.4 (2022), p. 4.

[Xie+21] Cheng Xie et al. “Multilayer Internet-of-Things Middleware Based on Knowledge Graph”. In: *IEEE Internet of Things Journal* 8.4 (2021), pp. 2635–2648. DOI: `10.1109/JIOT.2020.3019707`.

[Xu+15] Wenjun Xu et al. “Dynamic modeling of manufacturing equipment capability using condition information in cloud manufacturing”. In: *Journal of Manufacturing Science and Engineering* 137.4 (2015).

[Yan+23] Xiaolang Yang et al. “Meta-model-based shop-floor digital twin architecture, modeling and application”. In: *Robotics and Computer-Integrated Manufacturing* 84 (2023), p. 102595.

[YPK17] Seongjin Yun, Jun-Hong Park, and Won-Tae Kim. “Data-centric middleware based digital twin platform for dependable cyber-physical systems”. In: *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE. 2017, pp. 922–926.

[ZGC11] Deze Zeng, Song Guo, and Zixue Cheng. “The web of things: A survey”. In: *J. Commun.* 6.6 (2011), pp. 424–438.

[Zha24] Yan Zhang. *Digital Twin: Architectures, Networks, and Applications.* Springer Nature, 2024. URL: `https://library.oapen.org/handle/20.500.12657/87638`.

[ZLK22] Xiaochen Zheng, Jinzhi Lu, and Dimitris Kiritsis. “The emergence of cognitive digital twin: vision, challenges and opportunities”. In: *International Journal of Production Research* 60.24 (2022), pp. 7610–7632.

[ZYW20] Haijun Zhang, Qiong Yan, and Zhenghua Wen. “Information modeling for cyber-physical production system based on digital twin and AutomationML”. In: *The international journal of advanced manufacturing technology* (2020), pp. 1–19.

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Charles Steinmetz