TU WIEN Informatics

Digital Twinning for Industrial Energy Systems Utilizing Semantic Web Technologies

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

DI DI Gernot Steindl, BSc
Registration Number 00625241

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.Dr. Wolfgang Kastner

The dissertation has been reviewed by:

Alexander Fay	Martin Wollschlaeger

Vienna, 1st September, 2021

Gernot Steindl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

DI DI Gernot Steindl, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. September 2021

Gernot Steindl

Acknowledgements

First of all, I want to thank Wolfgang Kastner deeply for providing me the opportunity to work on this thesis at the Research Unit Automation Systems. His guidance and expertise had an inestimable value for me personally and for my work!

Next, I want to thank René Hofmann as scientific coordinator of the doctoral school Smart Industrial Concept! (SIC!) and my colleagues for all the discussions leading to new ideas and research directions.

I also want to thank all my friends who have accompanied me for such a long time. Without naming all of you, I am glad that you have been with me on this journey and that we are still having so much fun together.

Last but not least, I want to thank my family – my wife Martina, for supporting me whatever I am doing, and my sons Maximilian and Florian, who inspire me every day and make my life even more joyful!

Kurzfassung

Zukünftige industrielle Energiesysteme müssen sich an neue Anforderungen anpassen, die durch eine flexible Produktionsumgebung, die Nutzung volatiler erneuerbarer Energien und die Integration in Energiemärkte entstehen. Für solche industrielle cyber-physische Systeme (ICPS) sind digitale Zwillinge (DZ) eine Schlüsseltechnologie zur Optimierung des Betriebs dieser Energiesysteme, indem sie Dienste (Services) für deren Überwachung, Diagnose, Vorhersage und Steuerung bereitstellen. Obwohl das Konzept des DZ nicht neu ist, fehlen noch immer Frameworks und Methoden, um sie und ihre Services effizient zu erstellen. Dies führt aktuell zu Realsierungen, die auf anwendungsspezifische Lösungen abzielen und den anfallenden Daten keine Semantik verleihen. OPC UA, als Technologie für Industrie 4.0 Anwendungen bietet die Möglichkeit, die Daten durch Informationsmodellierung semantisch zu beschreiben. Obwohl OPC UA Informationsmodelle maschinenlesbar sind, basieren sie nicht auf formaler Logik, wodurch ein automatisiertes Schlussfolgern verhindert wird. Außerdem mangelt es ihnen an semantischer Ausdruckskraft und auch die Abfrage und Suche in Informationsmodellen ist aktuell nur eingeschränkt möglich. Diese Nachteile können durch die Kombination von OPC UA mit Semantic Web Technologien beseitigt werden. Dadurch können auch die Fähigkeiten von DZ in Bezug auf Interoperabilität und Anpassungsfähigkeit weiter verbessert werden.

Die vorliegende Arbeit liefert ein technologieunabhängiges Architektur-Framework für die Erstellung von DZ und deren Services im Bereich industrieller Energiesysteme. Dazu werden Kontextinformationen und Laufzeitdaten in einem verteilten Knowledge-Graphen verknüpft. Es wird eine Methode gezeigt, wie bestehende OPC UA Informationsmodelle genutzt werden können, um domänenspezifische Ontologien zu instanziieren. Diese Ontologien liefern die Semantik für die Daten des DZ und seiner Services. Außerdem wird eine Ontologie-basierte Datenzugriffsmethode für OPC UA-Laufzeitdaten vorgestellt, um Zeitreihendaten in einen Knowledge-Graphen zu integrieren. Die Leistungsfähigkeit des entwickelten Ansatzes in Bezug auf Datenabfragen wurde mit verwandten Möglichkeiten zur semantischen Integration von Sensordaten verglichen und evaluiert. Die gespeicherten Informationen im Knowledge-Graphen wurden außerdem genutzt, um ein datengetriebenes Simulationsmodell automatisch zu identifizieren.

Die Ergebnisse der Arbeit zeigen die Vorteile der Kombination von OPC UA und Semantic Web-Technologie im Kontext eines DZ. Der DZ wird an seine Umgebung anpassbar und semantische Interoperabilität zwischen den Services des DZ dadurch ermöglicht.

Abstract

Future industrial energy systems have to adapt to new requirements, caused by a flexible production environment, the usage of volatile renewable energy sources, and the integration into energy markets. Consider these energy systems as Industrial Cyber-Physical Systems (ICPSs), Digital Twins (DTs) are the key enabling technology for optimizing the operation by providing services for advanced monitoring, diagnosis, prediction, and control. Although the concept of DTs is not new, there are still missing architectural patterns and methods for creating them and their services in the domain of industrial energy systems efficiently. This results in implementations targeting application-specific solutions which provide no semantics to the accruing data. Thus, these data are hardly reusable for other tasks.

On the other hand, OPC UA, as a promising technology used in Industry 4.0 applications, provides additional semantics to the data by its information modeling capability. Though such information models are machine-readable, they still lack semantic expressiveness, are not based on formal logic, and inhibit automated reasoning. Also, information retrieval is limited in OPC UA by its current implementation. However, these drawbacks can be overcome by combing OPC UA with Semantic Web technology further enhancing the capabilities of DTs regarding interoperability and adaptability.

As an outcome, the thesis provides architectural guidelines for creating DTs and their services in the domain of industrial energy systems, considering RAMI 4.0 and utilizing existing OPC UA infrastructures to provide context information and run-time data stored in a federated knowledge graph. Therefore, a generic DT architecture is presented and a method is shown, how existing OPC UA information models can be used to instantiate domain-specific ontologies to provide context information for a DT. Also, an ontology-based data access method for OPC UA run-time data is developed to integrate time series data into a knowledge graph. The query performance of the developed approach is evaluated in comparison with other semantic sensor data integration methods. To showcase the applicability of the stored context information, the information is used to identify a data-driven simulation model automatically. In the end, functional and non-functional requirements and a service framework architecture for DTs are presented.

The results of the thesis show the benefits of combining OPC UA and Semantic Web technology in the context of an architectural DT framework by making the DT adaptable to its environment and providing semantic interoperability between the DT's services.

Contents

Preface

Publications

The present dissertation is carried out in a cumulative manner. A compilation of six peer-reviewed journal and conference papers is presented and forms the integral part of this thesis. The publications contributing to the thesis are listed below together with the author's contribution to each paper stated by the CRediT taxonomy [1]:

- *G. Steindl, M. Stagl, L. Kasper, W. Kastner, and R. Hofmann, "Generic Digital Twin Architecture for Industrial Energy Systems," Applied Sciences, vol. 10, no. 24, p. 8903, Dec. 2020, doi: 10.3390/app10248903*

  CRediT author statement: Conceptualization, G.S., M.S. and L.K.; Methodology, G.S.; Software, M.S.; Validation, G.S. and L.K.; Writing — Original Draft Preperation, G.S., M.S. and L.K.; Writing - Review and Editing, W.K. and R.H.; Visualization, G.S. and M.S.; Supervision, W.K. and R.H.

- *G. Steindl and W. Kastner, "Transforming OPC UA Information Models into Domain-Specific Ontologies" 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), 2021.*

  CRediT author statement: Conceptualization, G.S.; Methodology, G.S.; Software, G.S.; Validation, G.S.; Writing — Original Draft Preperation, G.S.; Writing - Review and Editing, W.K.; Visualization, G.S.; Supervision, W.K

- *G. Steindl, T. Frühwirth and W. Kastner, "Ontology-Based OPC UA Data Access via Custom Property Functions," 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019, pp. 95-101, doi: 10.1109/ETFA.2019.8869436.*

  CRediT author statement: Conceptualization, G.S.; Methodology, G.S.; Software, G.S. and T.F.; Validation, G.S.; Writing — Original Draft Preperation, G.S.; Writing - Review and Editing, W.K. and T.F.; Visualization, G.S. and T.F; Supervision, W.K.

- *G. Steindl and W. Kastner, "Query Performance Evaluation of Sensor Data Integration Methods for Knowledge Graphs," 2019 Big Data, Knowledge and Control Sys-*

tems Engineering (BdKCSE), 2019, pp. 1-8, doi: 10.1109/BdKCSE48644.2019.9010668.

CRediT author statement: Conceptualization, G.S.; Methodology, G.S.; Software, G.S.; Validation, G.S.; Writing — Original Draft Preperation, G.S.; Writing - Review and Editing, W.K.; Visualization, G.S.; Supervision, W.K. .

- *G. Steindl and W. Kastner, "Ontology-Based Model Identification of Industrial Energy Systems," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), 2020, pp. 1217-1223, doi: 10.1109/ISIE45063.2020.9152386.*

  CRediT author statement: Conceptualization, G.S.; Methodology, G.S.; Software, G.S.; Validation, G.S.; Writing — Original Draft Preperation, G.S.; Writing - Review and Editing, W.K.; Visualization, G.S.; Supervision, W.K.

- *G. Steindl and W. Kastner, "Semantic Microservice Framework for Digital Twins," Applied Sciences, vol. 11, no. 12, p. 5633, Jun. 2021, doi: 10.3390/app11125633.*

  CRediT author statement: Conceptualization, G.S.; Methodology, G.S.; Software, G.S.; Validation, G.S.; Writing — Original Draft Preperation, G.S.; Writing - Review and Editing, W.K.; Visualization, G.S.; Supervision, W.K.

The published articles are presented in Chapters 2 - 7. The order of these chapters are chosen to provide a logical flow for the reader. The layout of the papers is aligned to maintain readability and unified referencing, but the content of each chapter is identical to the original publication. A reference to the underlying publication is given at the beginning of each chapter.

Doctoral School Smart Industrial Concept! (SIC!)

Smart Industrial Concept![1] is a joint venture of the three scientific partners, TU Wien, Austrian Institute of Technology GmbH (AIT), and Montanuniversität Leoben, together with the industrial partners EVN, evon, FunderMax, ILF Consulting Engineers, and OSIsoft. The research activities of SIC! are mainly focusing on promoting digitalization and decarbonization in industry. The doctoral school's overarching goal is to create methods for energy-optimized industrial plant operation, energy conversion, distribution, storage, and interactions with energy markets. The four main research areas of SIC!, namely data processing and preparation, sector coupling and energy markets, optimal design of energy supply, and deployment and optimization, are depicted in Figure A.

[1] https://sic.tuwien.ac.at

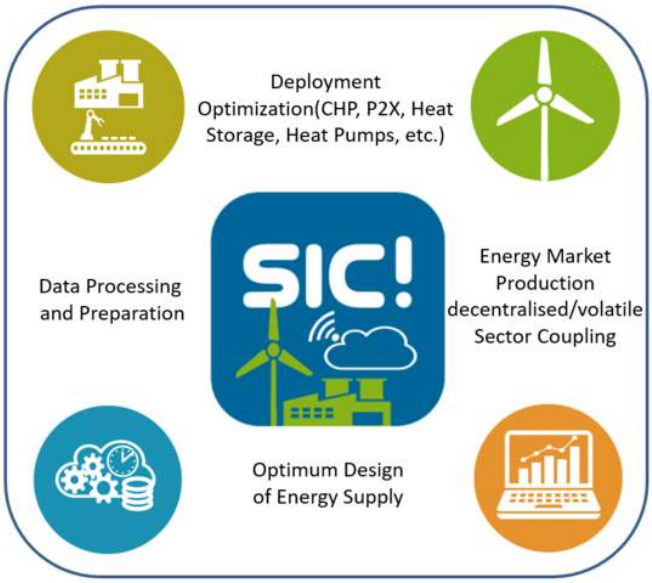


Figure A: Main pillars of the doctoral school SIC!

As part of the doctoral school, the thesis contributes mainly to the area of data processing and preparation. It provides methods and concepts for semantic data integration as well as knowledge engineering and management, which can be used to build a Digital Twin of an industrial energy system. The presented methods and concepts in this thesis can be seen as a foundation for further research and development work in the other areas of the SIC! consortium.

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

In industry, energy is consumed during the production process to transform raw materials into products. This energy has to be transformed, transported, and stored within the facility. Depending on the current production plan, electric power has to be provided to different equipment, like fans, pumps, or conveyors. Additionally, such industrial energy systems also transform fuel and power into other energy utilities, like heat, steam, and hot fluids used for production [35].

Currently, industrial energy systems have to adapt to changing demands. One reason is the goal of a highly optimized and customized production, as well as increased automation and adaptability of the production process [41]. These goals should be reached within the so-called fourth industrial revolution or Industry 4.0. The concepts of Industrial Cyber-Physical Systems (ICPSs) in combination with Industrial Internet of Things (IIoT) technology are the main enablers for this revolution. The industrial energy systems are tightly coupled with the production process, thus they have to fulfill similar requirements. Another reason is decarbonization in industry, which leads to the integration of renewable and distributed energy sources. Industrial energy systems have to compensate for the volatile energy production by facilitating storage capabilities and enhancing energy management capabilities. The third reason is the development of new business models, which are enabled through such a flexible industrial energy system. Participating in energy markets adds new values to the business, but also new constraints must be met because the energy supply chain has to be considered beyond the plant's boundary.

Transforming industrial energy systems into flexible ICPSs will be necessary for future businesses in Industry 4.0. The components of an industrial energy system are tightly coupled with the production process and become elementary assets of the business. Thus, Reference Architecture Model Industry 4.0 (RAMI 4.0), which was designed as solution

space for production systems, can also be applied to these industrial energy systems. In this context, RAMI 4.0 seems to be more suitable than other architecture models typically used in the energy domain, like Smart Grid Architecture Model (SGAM). SGAM is quite similar to RAMI 4.0 as it also provides a three-dimensional solution space, where the two dimensions "Interoperability Layers" and "Zones" are more or less equivalent to the RAMI 4.0 dimensions of "Layers" and "Hierarchy Levels". However, the third dimension in SGAM called "Domains" is concerned with energy transmission, including bulk generation, transmission, distribution, distributed energy resources, and custom premises. As in this thesis, industrial energy systems are viewed from the perspective of a plant operator RAMI 4.0 seems to be more suitable than SGAM.

A key enabling technology for ICPSs to provide services like monitoring, diagnosis, prediction, and control to optimize their operation are Digital Twins (DTs) [54]. According to [32], a DT is a "formal digital representation of some asset, process, or system that captures attributes and behaviors of that entity suitable for communication, storage, interpretation or processing within a certain context". To this definition, it has to be added that a DT can also include more than one asset, process, or system, which are somehow logically connected by the ICPS. Thus, an ICPS can consist of many DTs that interact with each other by sharing information to reach a common goal. This interaction can happen in various ways. In this context, the Platform Industrie 4.0[1] has defined the concept of an Asset Administration Shell (AAS), which can be seen as a DT, and also defined a language for Industry 4.0 components [56] to enable their interaction.

However, even if the concept of DTs is not new, their development is still a time-consuming and laborious task. This is because even if most implementations are goal-driven, their development is rarely accompanied by following architectural templates [27]. Thus, methods and concepts are needed in the domain of industrial energy systems to facilitate the DT development process and support the digital transformation in that area.

Even if some architectural concepts and frameworks for DT exist in the literature, they are missing a common terminology and understanding. The Industry 4.0 initiative has defined the RAMI 4.0 as a solution space to locate standards and technologies in its three dimensions. The clear definition of these dimensions helps to create a common understanding for new technologies, like the DT. In this context, a DT architecture should be aligned with RAMI 4.0 and its dimensions. Thus, the following research question emerges:

RQ1 - What is an appropriate system architecture for Digital Twins of industrial energy systems considering the Reference Architecture for Industry 4.0 (RAMI4.0)?

To reach the needed flexibility in ICPS and enable informed decision making, information from the whole life-cycle is needed. The DT acts as an interface to this information which can be managed to utilize machine-readable knowledge representation. This

[1] https://www.plattform-i40.de

knowledge representation can also be used to provide semantic interoperability within the DT's services as well as between the DT and client applications. The Semantic Web stack provides a comprehensive tool-set for building and querying such knowledge representations in the form of ontologies, utilizing Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), Web Ontology Language (OWL) and SPARQL Protocol and RDF Query Language (SPARQL). These technologies are standardized by World Wide Web Consortium[2] (W3C). In combination with reasoning engines and the incorporation of external information sources, so-called knowledge graphs can be created.

Next to these web technologies, there is a well-established industrial standard called OPC UA, which also provides information modeling capabilities. With these information models, structure and semantics are provided to the data within an ICPS, but with less semantic expressivity than provided by formal ontologies. Thus, a combination of OPC UA and the Semantic Web stack brings advantages regarding tooling, semantic expressivity by means of reasoning, browsability, and the ability of interlinking with other information to build an knowledge graph inside the DT. These OPC UA information models are sometimes already available and contain valuable information. Thus, methods are needed to reuse these models and utilize them inside a DT. Therefore, the following research question shall be answered:

RQ2 - What is an appropriate method for providing semantics for the data of a Digital Twin utilizing existing OPC UA information models?

Ontologies are suitable to create a semantic integration layer that provides a higher level of abstraction and facilitates the data integration process from heterogeneous sources [42]. However, ontologies are not suitable to store a large amount of time-series data, as created by sensors and events in an ICPS. Ontologies are stored in so-called triple stores, which are specialized RDF databases. If time-series are directly stored in RDF, new triples have to be created for every new observation. This causes performance issues for data retrieval if the triple store grows over time. However, the use cases of historical data access are quite common for a DT application. To overcome the performance problem, the concept of Ontology-Based Data Access (OBDA) exists, where the data can stay in their original storage and – for reasons of performance optimization – only mapped into the ontology on demand. Similar concepts have to be applied and evaluated for the integration of OPC UA, which is a common data source in industrial applications [16]. Thus, the following research question is stated:

RQ3a - How can OPC UA run-time data be semantically integrated into a Digital Twin architecture considering enhanced data retrieval performance?

Regardless of its suitability, time-series data are sometimes stored directly in RDF for certain use cases, as presented in [19]. For relational databases, frameworks like Ontop [10] exist and are already applied for industrial use cases to facilitate data access [36].

[2] https://www.w3.org

A firm evaluation of the available approaches and a comparison with the one which is developed for OPC UA data (RQ3a) is needed. Thus, the following research question shall be answered:

RQ3b - What is the query performance of the OPC UA run-time data integration approach compared to other semantic sensor data integration approaches found in the literature?

Simulation models are an important part of a DT in the domain of industrial energy systems, as the knowledge of the plant's physical behavior is required for optimizing the operation. So-called white-box models require a detailed understanding of the underlying physical principles. However, some physical conditions are often unknown or hard to get. They make no use of actual data from the process. Differences between the initial assumptions and the actual implementation result in a mismatch between the simulation and the actual physical behavior of the process. Even if this is a suitable approach for some applications, white-box models will not easily adapt to changes in the process without human intervention.

Thus, two other modeling approaches seem to be more appropriate for most applications, which rely on data produced by the underlying process. The first approach is called gray-box modeling and is based on physical principles which provide some predefined structure to the model. The parameters of the model are determined using historical data. Black-box models rely solely on data. Typically Machine Learning algorithms are used to build such models. As a result, prior understanding of the physical principles is not required as the model does not incorporate this information. The model is only built on past data produced by the system's behavior. However, domain expertise, like knowledge about the plant topology, the process itself with its states, the installed equipment with its operational boundaries, or the available sensors, is needed to determine which data should be used as input for the model. To gather this knowledge in combination with data preparation makes the identification of such data-driven modeling approaches still a time-consuming task [47]. Thus, the mentioned domain knowledge should be formalized to support the data-driven model development for DTs and partly automate the process. Therefore, the following research question is stated:

RQ4 - How can an ontological knowledge representation of a Digital Twin in combination with semantic run-time data be used to semi-automatically generate data-driven simulation models?

Services are another important part of the DT concept as highlighted in [53]. Services encapsulate the functionality of a DT. In recent years, the microservice architectural style has become more relevant for building distributed software applications with improved scalability, and maintainability [13]. The idea is to create service-based applications by composing small, loosely coupled software services [15]. The size of services varies depending on the application, but the attribute "small" refers to their functionality rather than the code size. Services are important in the development of DTs, but they are only mentioned or described at a high level of abstraction in the literature,

missing implementation details and their fundamental requirements. Because most DT implementations are driven by a specific goal rather than an architectural template [27], the same can be said for the services that are implemented. There is still a significant gap in DT research on how to offer a higher number of services in the same environment to support complex decision making [11]. Thus, research should focus on the DT's services infrastructure, specifically addressing requirements and a service framework architecture that can later be applied to a variety of DT applications. This problem is addressed in the following research question:

RQ5 - What are the requirements and a resulting appropriate architecture for a Digital Twin service framework facilitating semantic interoperability between services?

The research questions mentioned above showed that there are still missing methods and concepts that facilitate digital twinning in industrial energy systems. Therefore, architectural guidelines, data integration methods, and information and knowledge management methods for digital twinning have to be developed. Semantic Web technology, with its capabilities and extensive toolset, can help to provide solutions. However, it is not an established technology for industrial applications, yet, and the combination with industrial standards like OPC UA is important to gain acceptance in that area. For this reason, this thesis shall also investigate how Semantic Web technology can leverage the implementation of DTs in the domain of industrial energy systems.

1.2 Aim of the Thesis

The overall aim of the thesis is to provide concepts and methods for building a DT in the domain of industrial energy systems. Therefore, the following five goals are defined and put into relation with the research questions mentioned above.

G1: Develop a generic Digital Twin architecture for industrial energy systems
Architectural templates and guidelines are missing for implementing DTs in the domain of industrial energy systems. Thus, a generic DT architecture for industrial energy systems shall be developed. As RAMI 4.0 provides a general solution space in the context of Industry 4.0, the developed DT architecture should also be aligned with this reference model. This goal is targeted by research question RQ1.

G2: Reusing existing OPC UA information models to provide semantic context information for the Digital Twin
Next to the specified communication protocols utilized by OPC UA, it also provides information modeling capabilities. Those information models hold structure and context to the OPC UA data. Usually, these information models are built during the engineering phase and contain relevant information useful for a DT during operation. Thus, research question RQ2 is targeting the extraction of this information from existing OPC UA information models and providing this information to the services of a DT.

G3: Provide a semantic integration method for OPC UA run-time data of a Digital Twin
A DT has to handle a large amount of time-series data, which are mainly produced by sensors. The DT has to provide semantics and context to make those data reusable for various services. In Industry 4.0, OPC UA is currently a recommended standard within the RAMI 4.0 communication and information layer. Thus, a method for efficiently integrating OPC UA run-time data and interlinking it with other information has to be provided. This goal is targeted by research questions RQ3a and RQ3b.

G4: Providing a method for supporting semi-automatic data-driven model identification
Dynamic models of the physical behavior are important for the operation of a DT. They are used to monitor or predict the condition of the DT and to optimize the plant's performance. As the models have to adapt to changes in the environment or the DT itself, data-driven models are more suitable for that kind of task. Thus, a method shall be provided which allows a DT of an industrial energy system to identify such data-driven models based on engineering information and historical run-time data of the plant. This goal is addressed by research question RQ4.

G5: Developing an architectural concept for a Digital Twin service framework
Services play a key role for DTs as they structure their functionality in a modular way. Thus, a DT service framework shall be provided, which is based on fundamental requirements found in the literature. This goal is tackled by research question RQ5.

1.3 Methodology

This section describes the methodology which is applied to answer the above-stated research questions and reach the goals of the thesis. As the research questions are interrelated, an agile, iterative, and incremental research design approach is applied. Figure 1.1 gives an overview of the steps in the applied research process. In the initial planning step, the research questions are formulated and the applied corresponding research methods are determined. Each step in this iterative process is related to a certain research question. The dissemination is conducted out if final results from a process step are available. Thus, the chronology of the publications has not the same order as the stated research questions.

Except for RQ3b, the explorative research is qualitatively evaluated by developing proof-of-concept implementations, which are applied to certain case studies. For RQ3b, a descriptive, qualitative research approach is used to evaluate the performance of the proposed data integration method. In addition, it is compared with other methods found in the literature. In the following, a more detailed description of the applied methods for answering each research question is given:

Applied Methods for RQ1 – Generic Digital Twin Architecture: Common services, concepts, architectures, and frameworks in the context of industrial DTs, which

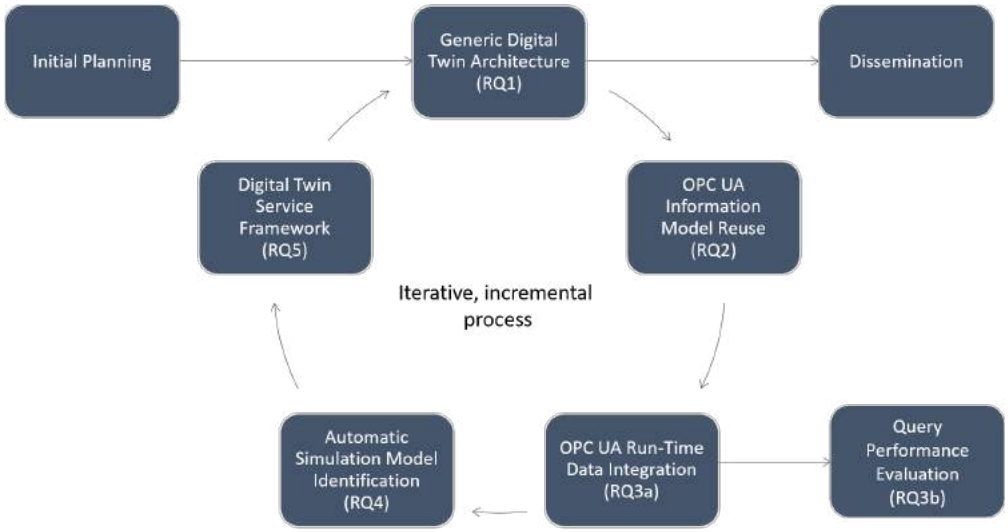


Figure 1.1: Iterative, incremental research process.

are applied to the operational phase of the life-cycle, were identified and analyzed based on a literature review. The acquired information was used to develop a technology-independent generic DT architecture. A proof-of-concept implementation was used for a use case of a thermal energy storage system, called Packed-Bed Thermal Energy Storage (PBTES), which is located at the Institute for Energy Systems and Thermodynamics (IET) at TU Wien. Therefore, an ontology was designed following the METHONTOLOGY approach [18]. The implemented ontology has been evaluated using common tools like the OOPS! Pitfall Scanner [39] and the HermiT reasoner [20].

Applied Methods for RQ2 – OPC UA information model reuse: Various transformation approaches between OPC UA information models and OWL ontologies found in the literature were analyzed. Based on the review, a two-step, rule-based transformation process was defined and evaluated with a proof-of-concept for the use case of a thermal heating process. Therefore, an OPC UA server was implemented, exposing the instantiated information model. The correctness of the transformed model was evaluated by applying SPARQL queries and compared the retrieved information with the expected results.

Applied Methods for RQ3a – OPC UA run-time data integration: Again, a literature review was conducted to identify relevant approaches to combine Semantic Web technology and OPC UA. Based on an analysis of the OPC UA meta information model and OWL profiles, a mapping between OPC UA and OWL full was defined. Afterward, a concept for OBDA of OPC UA run-time data, based on Custom Property Function (CPF) was developed. The feasibility of the proposed OBDA approach was evaluated by a proof-of-concept implementation for a use case of the PBTES.

Applied Methods for RQ3b – Query performance evaluation of semantic

sensor data integration methods: Three common semantic sensor data integration methods were identified in the literature. The scalability of these three methods was evaluated by applying a quantitative analysis of their query execution time. Therefore, artificial sensor data were generated, and eight different SPARQL queries were formulated to test various search dimensions in the time-series data. The number of sensors was varied between 10 and 500, and the produced amount of data per sensor was varied between 100 and 100.000 observations. The query execution time was measured for every combination of these two values for each integration method. Every measurement was repeated 20 times to calculate the mean and median execution time, which served for analyzing the results of the defined eight SPARQL queries.

Applied Methods for RQ4 – Semi-Automatic simulation model identification: A literature review for automatic model identification was carried out. To capture the essential plant information, an ontology was designed based on the Ontology 101 engineering method [37]. Relevant physical principles, e.g., the conservation of mass and energy, were identified and formulated as SPARQL rules to capture causal relations between sensor data. A proof-of-concept implementation for a thermal heating process was used to evaluate the feasibility of the proposed approach. Therefore, an Open Modelica simulation was developed to generate sensor data which were used to evaluate the semi-automatic model identification process.

Applied Methods for RQ5 – Digital Twin service framework: Functional and non-functional requirements were identified based on a literature review of DT frameworks, architectures and applications. The identified requirements are the foundation for an architecture of a technology-independent microservice-based framework. An implementation of this framework is provided as proof-of-concept and was evaluated against the identified requirements.

1.4 Summary of the Published Articles

Before a summary of the work is given, the interrelation of the published articles is described. Therefore, RAMI 4.0, Smart Data in Industry 4.0, and the concept of knowledge graphs are briefly introduced. Afterward, the publications are set into the proper context using these concepts. An understanding of these concepts is required to grasp Figure 1.3 in which the overview of the published articles and their contextual embedding into the thesis are visualized.

1.4.1 Interrelation of the Published Articles

The work in this thesis is tightly coupled to the concept of RAMI 4.0 which is defined in [2] and shown in Figure 1.2. In general, RAMI 4.0 is used to achieve a common understanding of standards, tasks, and use cases. Therefore, three different aspects or dimensions are used by RAMI 4.0: It expands the hierarchy levels of IEC 62264 by "Product" and "Connected World", defines six layers for an Information Technology

(IT) representation of an Industry 4.0 component and considers the life cycle of the product or system according to IEC 62890. For Industry 4.0, the product, as well as the collaboration with external entities, plays a vital role. Thus, these levels are also considered in RAMI 4.0. The life cycle is divided into a "type" and "instance" phase. The "type" phase is part of the engineering phase, which ends when a prototype is available. An "instance" is the system or product when it reaches the operational phase in the life cycle. Additionally, six RAMI 4.0 layers are defined to manage complexity, namely the Asset, Integration, Communication, Information, Functional, and Business Layer.

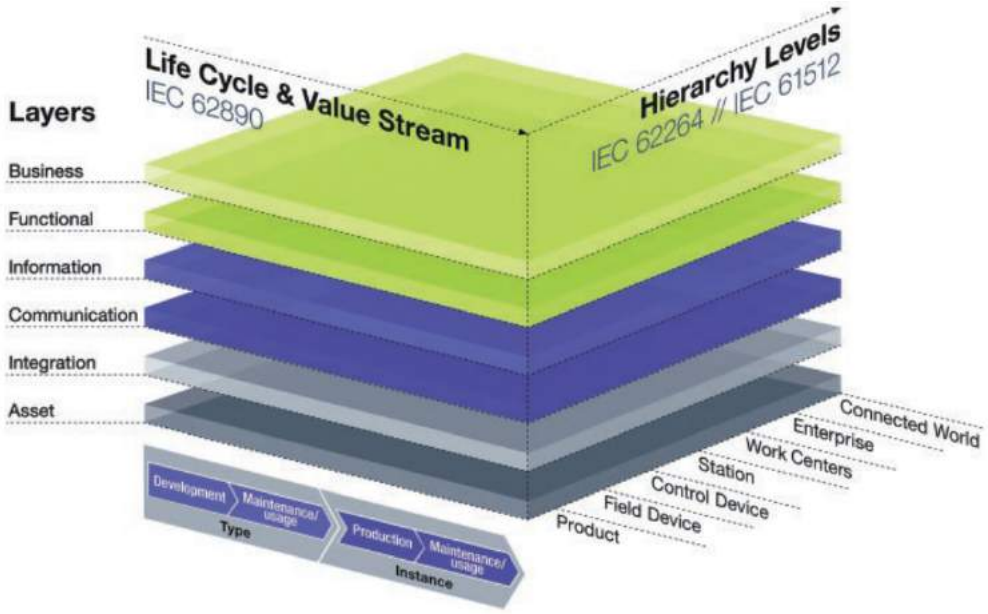


Figure 1.2: Reference architecture model for Industry 4.0 (RAMI4.0) [2]

With the help of RAMI 4.0, tasks and workflows can be broken down into manageable pieces and use cases can be provided with context. Even if RAMI 4.0 was originally designed for the production industry, it is also applicable in the domain of industrial energy systems. Thus, it is used in this work to investigate certain aspects of a DT in that domain.

The functional distribution of an industrial energy system can vary over various hierarchy levels, especially considering the cloud, fog, and edge computing paradigm in modern automation systems [58]. This aspect is not further investigated in this work. Also, only the operational and maintenance phase of the energy system is targeted. Thus, these two dimensions (i.e., life cycle & value stream, hierarchy level) are neglected in Figure 1.3 and only the RAMI 4.0 layers are taken into account. These layers are now shortly described, based on the information found in [2]:

Asset Layer – Physical components as well as human beings can be part of the Asset

Layer.

Integration Layer – This layer represents the interface between the real and the digital world. It provides information about the underlying assets. Computer-aided control of the technical process, sensors, as well as interactions with humans over Human-Machine-Interfaces (HMI) are located on this layer.

Communication Layer – This layer is concerned with data communication and data models. Currently, OPC UA is a recommended standard on this layer.

Information Layer – This layer provides a run time environment for (pre-)processing of events, execution of event-related rules, the persistence of data, ensuring data integrity, and consistent integration of different data for obtaining a higher data quality. This is achieved via service interfaces and by receiving and transforming events which are processed by the Functional layer.

Functional Layer – This layer provides a platform for horizontal integration of various functions. It also provides a run time and modeling environment for services that support business processes and a run time environment for applications and technical functionality.

Business Layer – Some aspects of this layer include the mapping between business models and the resulting processes, orchestration of services of the Functional layer and receiving events for advancing the business process.

Figure 1.3 provides an overview of the main DT components within the RAMI 4.0 layers. They are only depicted on a very abstract level. The Physical Entity at the Asset layer represents the industrial energy system, e.g., a heating process or PBTES.

At the Integration layer, run-time data and engineering data are made available to the DT. Run-time data is mainly produced by sensors whereas engineering data comes mainly from Piping and Instrumentation (P&I) diagrams in industrial energy systems. P&I diagrams contain information about the plant's equipment, its topology, and the instrumentation.

In this thesis, OPC UA is assumed as used industrial communication system, as it is also a recommended standard by the Industry 4.0 initiative for the communication and information layer. It provides unified transport mechanisms and offers information modeling capabilities. Such information models are standardized for certain domains in so-called Companion Specifications. This approach is suitable to build modularized information models [26], which can be reused by a DT. Currently, there is no suitable Companion Specification for industrial energy systems available, but the Data Exchange in the Process Industry (DEXPI)[3] initiative collaborates with the OPC Foundation to establish such a specification. This Companion Specification will provide an OPC UA information model for P&I diagrams. However, it has not been released, yet.

At the information layer, pure process information is not sufficient to achieve the optimum operation of an ICPS [9]. The needed additional machine-interpretable semantics and the

[3] https://dexpi.org/

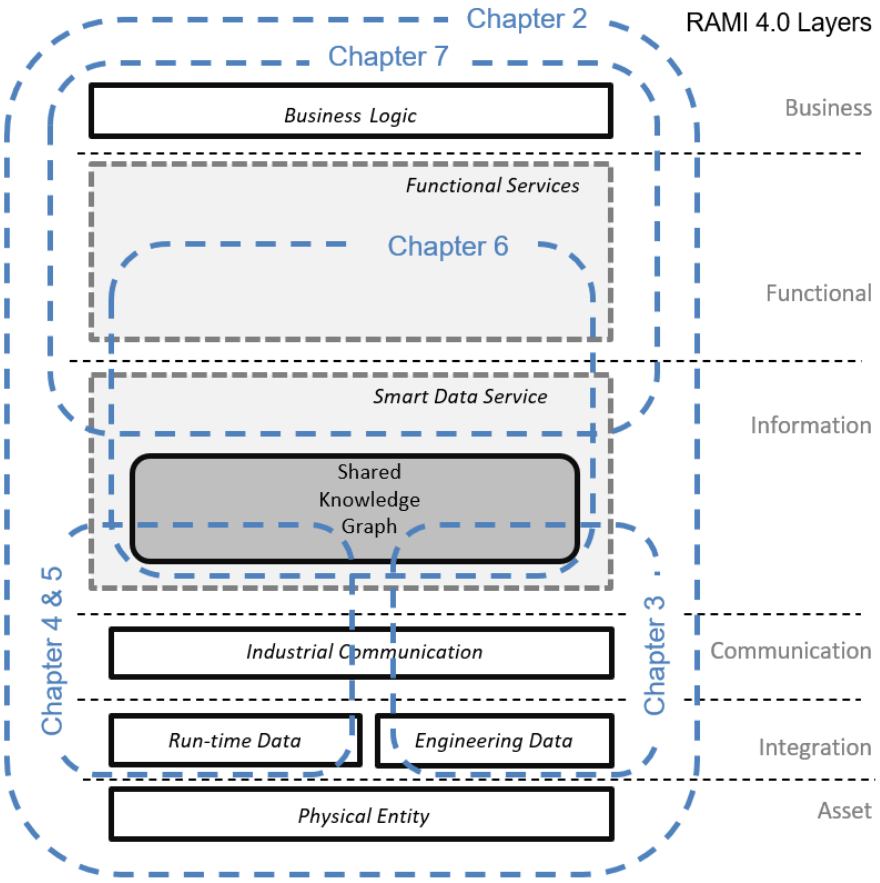


Figure 1.3: Overview of published papers and their contextual embedding into the thesis.

interlinking of various data can be realized by utilizing a so-called knowledge graph based on Semantic Web technology. The terms knowledge graph, knowledge-based system, and ontology are sometimes used interchangeably in the literature. With respect to [3], a knowledge-based system consists of two parts: a knowledge base and an inference engine. In this context, an ontology is the knowledge base in a knowledge-based system on which the reasoner can infer new knowledge. Based on the definition found in [17], a knowledge graph is a knowledge-based system (ontology and reasoner) that is also able to integrate information from external sources. This integration of external sources, like relational databases, is quite important for realizing DTs. However, some of the publications which are presented in this thesis are focusing more on the ontological aspect of such a knowledge graph. Thus, both terms are used in the following chapters.

A "Shared Knowledge Graph" is established within the DT. Access to the information stored in this knowledge graph, which combines various engineering information and run-time data, is granted via a so-called "Smart Data Service". The Smart Data Service is located at the Information layer of RAMI 4.0, as depicted in Figure 1.3.

Other functional services can be built on top of this Smart Data Service in the functional layer of RAMI 4.0. In the context of Industry 4.0, "Smart Services" are established, which are traditional industrial services combined with Internet-based services to create new innovative services [29]. According to [50], Smart Services are digital services that act on collected and analyze data from networked, intelligent technical systems and platforms. Thus, Smart Services respond to analyzed data from cross-functional areas. A premise for Smart Services is the availability of smart data [50]. The term smart data is closely related to big data, as it refers to structured, semi-structured, and unstructured data, which are acquired from a variety of heterogeneous sources [25]. This smart data is provided by the already mentioned Smart Data Service.

Details about certain aspects of the components can be found in the publications. The dashed lines in Figure 1.3 mark the area within the DT which a certain publication addresses. Also, a reference to the chapter in which the publication can be found is given. In Chapter 2, a general DT architecture is presented, which provides more details about the components in the various RAMI 4.0 layers. In Chapter 3, a concept for reusing existing OPC UA information models, which incorporate engineering information, e.g. information about the plant topology, is presented. This method can be used to populate the shared knowledge graph with engineering information. Chapter 4 presents an OBDA method for OPC UA. It shows how run-time data can be integrated into the shared knowledge graph. This method is compared with other semantic sensor data integration methods found in the literature in Chapter 5, by evaluating their query performance. A use case for utilizing the available information from the Shared Knowledge Graph to semi-automatically identify simulation models of the underlying process is presented in Chapter 6. Chapter 7 presents a concept for a microservice-based framework, which shows how functional services can be composed and integrated into business processes utilizing a workflow engine and a federated query engine.

1.4.2 Requirements and Industrial Use Cases

For a DT, created for the use case of an industrial energy system, four main requirements are identified for this thesis which are the foundation of the Generic Digital Twin Architecture (GDTA) presented in Chapter 2 and listed below:

GRQ1 A DT shall be able to handle equipment for energy production, storage, distribution, and consumption which is tightly coupled with the production process and therefore constitutes a valuable business asset.

GRQ2 A DT should be able to incorporate and interlink various data sources to provide the needed information about the physical asset's state to cope with certain tasks.

GRQ3 A DT should provide machine-readable semantics to its gathered data to facilitate (semi-)automatic information processing.

GRQ4 A DT should provide a flexible way of adding or adopting its functionality to be able to deal with changing demands from production.

Nevertheless, some topics are not targeted and left out of scope. These include:
Maximum response time - In some applications, the maximum response time is an essential factor to maintain operations. In this thesis, the DT is implemented based on IT systems, which can not guarantee a maximum response time. Thus, real-time requirements have to be handled by the underlying Operation Technology (OT) systems.
Security - Security in the context of a DT is a very important topic. Especially, the convergence of OT and IT make security considerations more important. However, this opens up a whole new research direction, not considered in this thesis.
Life-cycle Management - Even small changes in the production process, like replacing some of the equipment, have to be handled by the DT. This means it has to be aware of such replacements and also be aware that historical data maybe be deprecated. Internal simulation models of the DT have to adapt to the changes in the environment, and the deprecated data cannot be used anymore, e.g., for model identification or fault detection. Thus, the management of internal simulation models and data over the whole life-cycle has to be performed within a DT.

In the following chapters, two distinct use cases in the domain of industrial energy systems are used to evaluate the concepts and methods presented in this thesis. These two use cases represent the physical asset of the DT.

The first use case (A) is a PBTES, used in industrial processes with high operating temperatures up to 800 °. Typical industrial applications are in the steel, glass, and cement industry or in solar power plants. A test rig of such an energy storage device is located at the Institute of Energy Systems and Thermodynamics laboratory at the TU Wien. It consists of a conic bulk container filled with gravel as the storage medium and surrounded by insulation. Figure 1.4 shows a schematic diagram of the PBTES. The ambient temperature T_i, the temperature after the heater T_H, the outlet temperature T_o, and the mass flow m are measured. Inside the bulk container, temperature sensors at four different heights are installed ($T_{L1} - T_{L4}$) to keep track of the state of charge. The ventilation unit and a heater are used for loading the PBTES by blowing hot air into the bulk. The unloading is performed using cold air from the surrounding, which is blown through the hot bulk to increase the air's temperature. This hot air can now be used in the production process.

The second use case (B) is a thermal process, which is depicted in Figure 1.5. Air is heated up by an electrical heater $H1$ and blown into a chamber $SiPro$ by the ventilation unit $F1$. The temperature inside the chamber is controlled to hold certain set points. The hot air is retrieved by the ventilation unit $F2$ from the chamber and transported to a heat exchanger $HE1$ for heat recovery. Various temperature sensors are available to measure the inlet temperature T_{in}, the temperature after the heat exchanger T_{hx}, the supply temperature for the process T_{sup}, the process temperature T_p, the extracted temperature T_{ext}, and the outlet temperature T_{out}. Also, the mass flow in the inlet m_{in} as well as in the outlet m_{out} is measured.

Use case (A) was specified within the doctoral college SIC! and was investigated by several students regarding various research topics, e.g., in [23]. It was chosen because

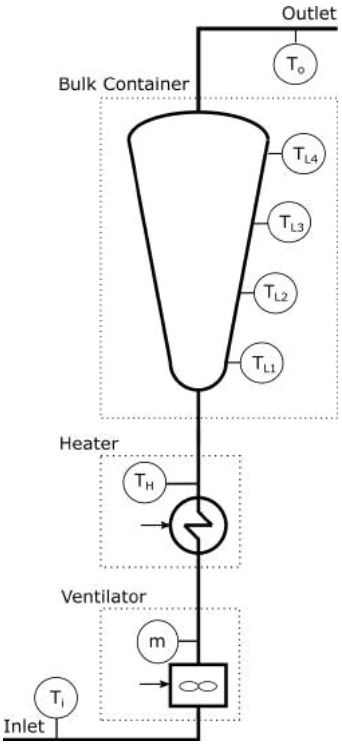


Figure 1.4: Use case (A) - Schematic diagram of the Packed Bed Thermal Energy Storage (PBTES).

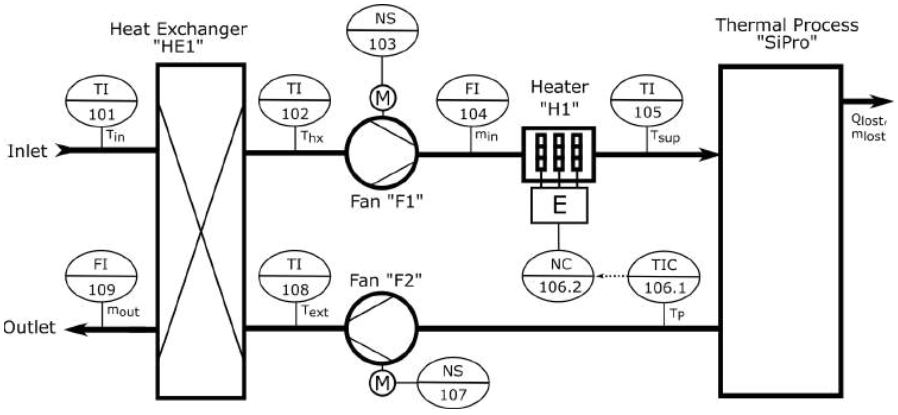


Figure 1.5: Use case (B) - P&I diagram of the heating process.

the test rig in the laboratory can be used for experiments without interfering with a real production process. In this thesis, use case (A) is applied in Chapter 2 and 4.

As use case (A) is just a thermal energy storage, a more complex use case (B) was developed for this thesis. This use case (B) provides a slightly more complex system, including energy production, distribution, and consumption. This introduces more freedom to investigate certain aspects of this thesis, like modeling causalities. Therefore, a simulation model was developed and used for investigating various scenarios. This use case is applied in Chapter 3, 6, and 7.

1.4.3 Summary

In the following, a summary of the publications is given. Details can be found in the referenced chapters.

Generic Digital Twin Architecture for industrial energy systems (Chapter 2) The concept and capabilities of a DT are not clearly defined in the literature because the domains, e.g manufacturing, smart grid, building automation, and also their applications are varying. Thus, most implementations are targeting specific use cases and applications. This is the reason why realizations follow a certain goal without any architectural template [27]. Even if some concepts, architectures, and frameworks for DT already exist, they are not based on a common architecture and name similar things differently. To overcome this problem, a technology-independent GDTA is proposed, which is aligned with the RAMI 4.0 layers. With the help of the GDTA, existing frameworks and technologies can be positioned in this architecture, facilitating the development of DTs.

The GDTA is based on the fundamental Five-Dimensional Digital Twin (5D-DT) concept presented in [53]. The 5D-DT consist of five main components or dimensions, namely the Physical Entity, the Virtual Entity, Connections, the Data Model, and Services. The Physical Entity comprises many subsystems that conduct specific functions and are supported by numerous sensors that collect state and operational data. The Virtual Entity tries to model the physical entity with great precision by combining many types of models such as geometry, physical, behavior, and rule models. Services for the Physical Entity and the Virtual Entity are included in the Service Model. Data from the Physical Entity, the Virtual Entity, the Services, as well as domain knowledge, and the fusion of those data make up the Data Model. The Connection Model describes each link between the DT's components. This 5D-DT concept with its five dimensions serves as the cornerstone for the propose GDTA.

The proposed GDTA is developed based on a literature review of concepts, architectures, and frameworks for DTs. A proof-of-concept implementation is used to evaluate the architecture. Therefore, ontology design methods are applied to build a shared knowledge graph based on Semantic Web technology. A use case of the PBTES, is used to evaluate a simulation service of the DT.

The GDTA introduces the concept of a "Smart Data Service" which is located at the Information Layer of RAMI 4.0. It provides access to archived run-time data, like sensor

data and context information mostly gathered from engineering data. The service provides endpoints for inserting new information as well as for information retrieval. Information ownership and access have also to be managed, including data access mechanisms. On the Functional Layer of the GDTA several service groups are identified, like Simulation Services, Monitoring Services, Diagnostic Services, Prediction Services, Control Services, and Reconfiguration Services. The Simulation Service is important as it is the foundation for other services. It provides models simulating the physical behavior of the Physical Entity of the DT. The DT can consist of more than one model, which are developed for a special purpose. Thus, a model has to be connected with context information by the Shared Knowledge Base to create a certain View on the Virtual Entity. The functional services have to be managed to include new ones easily and make them available for other services. Thus service composition and inclusion to business logic have to be performed.

The proof-of-concept implementation is focusing on the Virtual Entity and the simulation service. Therefore, a hierarchical ontology structure, with a top-level ontology and domain ontologies for every service class, is presented. Existing ontologies are reused and extended, like Ontology Web Language for Service (OWL-S) [33] for the service description and ML-schema [40] for describing the available simulation models. To test the implemented simulation service, a Representational State Transfer (REST) Application Programming Interface (API) is defined.

The proof-of-concept implementation showed the instantiation of the technology-independent GDTA based on Semantic Web technology and their applicability for handling context information about resources and services within a DT.

Reusing existing OPC UA Information Model to provide semantic context information for the Digital Twin (Chapter 3)
To expand the capabilities of Cyber-Physical Systems (CPSs), the semantics of data and context information are crucial. In addition to its communication capabilities, the OPC UA standard provides concepts for information modeling to give structure and meaning to OPC UA data. Thus, information about equipment and the plant topology may be available via OPC UA and used to contextualize data for further processing. However, the absence of formal semantics, the lack of browsing capabilities [43], and the low semantic expressiveness compared to more sophisticated knowledge representation languages [8], are disadvantages of the OPC UA information model. In most cases, a direct OWL representation of the OPC UA information model is not adequate because information models are not designed in the same way as ontologies. Thus, for certain applications, only parts of the OPC UA information model are needed to instantiate a domain-specific ontology.

It is shown how domain-specific ontologies can be instantiated using existing OPC UA information models while considering modifications to the source information model and the target ontology. The proposed method is intended for a use case in which the information model is already instantiated, and the address space is exposed by an OPC UA server.

The transformation process of OPC UA information model into a domain-specific ontology takes place in two steps. In the first step, the OPC UA information model is converted to OWL Full. While using OWL Full can result in undecidable reasoning problems, the mapping is simple, as most concepts can be mapped with a one-to-one relation. In the second step, the OWL Full graph is transformed into a domain-specific ontology using SPARQL rules. This domain-specific ontology may be built using OWL DL, which allows for reasoning if needed.

The rules are formulated as SPARQL construct clauses. They are searching the ontology graph for patterns and create new concepts and instances based on these patterns in another graph. To apply this method to different domain ontologies or the OPC UA information models, only these SPARQL rules must be altered. As a result, a flexible transformation method is developed.

A simple heating process is chosen as a use case to apply the proposed transformation process for evaluation. An OPC UA information model is developed for the heating process use case and instantiated in an OPC UA server. The information model captures the structure of the heating process, including its components, topology, sensors, and actuators. The information model is designed following current industry standards in the domain of piping and instrumentation. The Pipe, Equipment, Topology, and Instrumentation Ontology (PETIont) is utilized as the target ontology for the proof-of-concept.

In this use case, nine different SPARQL rules are formulated for the transformation between the given OPC UA information model and PETIont. When the transformation procedure is completed, the transformed information is accessible via a SPARQL endpoint. Because the target ontology is written in OWL DL, reasoning may be applied if necessary. SPARQL queries are used to retrieve information from the endpoint and compare it to the expected results in order to evaluate the transformation process.

The information stored in the target ontology can be used by a DT to facilitate plant monitoring, diagnosis, prediction, and control. The full potential of the suggested transformation approach is reached once this information is linked to additional domain-specific ontologies.

Semantic integration method for OPC UA run-time data of a Digital Twin (Chapter 4)
Semantic Web technology may be used to semantically enrich industrial data, integrate data from various sources, and improve interoperability. The ontological representation of the plant structure, together with its run-time data, enables the information processing within such a CPS, to act on the physical process. Therefore, engineering and run-time data have to be interlinked to form a so-called knowledge graph. The main objective in this data integration and the interlinking process is to preserve data in its original format and storage, which can be accomplished via OBDA approaches. Such an approach is presented in this work for OPC UA as it is a popular industrial communication standard.

A transformation of the OPC UA information model into an OWL ontology is performed

to provide access to run-time data via a SPARQL endpoint. Usually, data inside an ontology is stored in a graph-based database called triplestore and does not often change over time. In industrial operations, run-time data, such as sensor data or state conditions, are generated at high volume and velocity. Thus, triplestores are not well suited for huge time-series data since the enlargement of the graph results in a deterioration of the SPARQL query performance [34]. As a result, an OBDA technique based on CPF is proposed. With this method, run-time and historical time-series data may be retrieved on-demand and without the need for persistent storage in the triplestore.

A transformation between OPC UA meta-model and OWL Full is specified, to automatically turn OPC UA information models into OWL ontologies and demonstrate the application of CPF for OBDA. Therefore, the Apache Jena Framework[4] is utilized in the proof-of-concept because it supports OWL data and can be enhanced with CPF using its SPARQL query engine ARQ. To access the OPC UA data within the ARQ extension, the Eclipse Milo framework[5] is used for the OPC UA communication.

To gather the information model from the OPC UA server, an OPC UA client connects to the server and begins analyzing the address space by reading the NamespaceArray from the OPC UA server. Afterward, a transformation into RDF is carried out. In a second post-processing step, the OWL Full vocabulary is added. OWL Full is utilized because it allows using object properties for relating individuals and classes like it is done for instance-declarations in the OPC UA information model. Such constructs are not permitted in OWL DL.

The run-time data is accessible via the OPC UA Read service or the OPC UA HistoryRead service. To prevent a periodic or event-based updating of the data inside the ontology and to keep the amount of data in the triplestore as small as possible, the OBDA method loads the OPC UA data only on-demand.

As a use case for evaluating the ontology-based OPC UA data access, the PBTES is chosen. It consists of a ventilator, a heater, and a bulk container. It is also equipped with a number of temperature sensors as well as mass flow sensors. The PBR's OPC UA information model is instantiated in an OPC UA server and automatically converted to OWL. The current sensor data, as well as historical sensor data, can be fetched from the OPC UA server using SPARQL queries invoking the CPF.

It is demonstrated how such CPF and and the automated transformation of the OPC UA information models into OWL may be used to integrate OPC UA data into a knowledge graph, which can be utilized by a DT.

Query performance evaluation of semantic sensor data integration methods (Chapter 5)
Heterogeneous data sources are often distributed in ICPS, resulting in isolated data silos. Getting access to these various data sources might be problematic because a lot

[4] https://jena.apache.org/
[5] https://github.com/eclipse/milo

of implicit knowledge is required. For example, in order to get data from a relational database, the user must first understand its schema, which may consist of tables and columns with meaningless names and without providing semantics. Thus, the data integration procedure becomes an extremely time-consuming task. Some of these issues could be addressed with the use of a knowledge graph that provides semantic to the data and interlinks existing data sources. Aside from sensor metadata, such as engineering units, sensor location, and manufacturer information, time-series data must also be accessible via the knowledge graph to fully develop its potential. Different integration approaches have been proposed in the literature, but their scalability in terms of data access performance has yet not be compared. Thus, the question arises, what is the scalability of different sensor data integration approaches for knowledge graphs regarding their query execution time?

Three different methods were examined, which were discovered in related work and applied to an industrial use case. These three approaches are namely, "Ontology Storage", "Custom Property Function", and the "Ontop Framework[6]".

Ontology Storage – Storing sensor data inside an ontology is one technique of integrating sensor data from sensor networks into a knowledge graph. A observations is represented by a timestamp-value pair. The Sensor, Observation, Sample, and Actuator (SOSA) ontology [59] is a World Wide Web Consortium (W3C) recommendation and is also used for evaluating this approach. As the ontology is structured as a graph, every new observation expands the graph and thus the search space for data retrieval.

Custom Property Function – SPARQL's functionality can be extended by implementing a so called "Custom Property Function" (CPF). A property with a specified Uniform Resource Identifier (URI) determines a CPF, which executes custom code during the triple matching process. For the evaluation, a CPF is implemented for the Apache Jena Fuseki Framework to compare this storage strategy with the other two approaches.

Ontop – Ontop is an OBDA framework that is free and open-source. The idea behind OBDA is to utilize ontologies as an interface for data access. In this context, relational databases are often used as data sources. However, mappings between an ontology-based conceptual domain view and the database schema have to be defined.

To evaluate the three approaches, artificial sensor data are generated and stored in a Structured Query Language (SQL) database and a triplestore. The test data consists of random floating-point values with a corresponding timestamp. Eight different SPARQL queries are formulated, which perform a search over three different properties of an observation, namely the sensor-ID, the timestamp, and the values. The number of sensors used in the experiment and the number of values per sensor was varied from 10 to 500 and 100 to 100.000, respectively. Because of the page limitation, only two corner cases are presented. *Test Case A* has 10 sensors with 100.000 observations per sensor, and *Test Case B* has 500 sensors with 2.000 observations per sensor. The two cases are

[6]https://ontop-vkg.org/

comparable because they have the same amount of observations in total and demonstrate scalability problems with the number of sensors and the amount of observation over time, respectively. The execution time for all eight SPARQL queries is measured for several combinations of these two values. This procedure is carried out for each of the three sensor data integration methods. In addition, every measurement is also carried out 20 times to calculate the statistical parameters, like the mean execution time, which are used for further investigations.

The performance evaluation reveals that storing time-series data in an ontology causes performance issues if the time-series grows. The CPF is a very adaptable solution for incorporating many data sources, such as OPC UA or NoSQL databases. Nevertheless, the implementation of CPF has its limitations. The Ontop framework's query re-writing mechanism results in extremely fast data access. Ontop has the shortest query execution time in the majority of the test cases. Ontop is also able to handle a large amount of data and scales nicely. One disadvantage of the Ontop framework is that it can only access a single relational database. Furthermore, this database must be SQL compliant.

Method for supporting semi-automatic data-driven model identification (Chapter 6)
To optimize plant operations and enable production to be tied to the energy market, novel predictive control strategies have to be applied to ICPS. Accurate models of the plant's components are required for such control strategies to forecast their dynamic behavior and meet set points or detect malfunctions of the plant immediately. However, the overall model creation process requires knowledge, which is frequently only implicit, scattered, and rarely adequately documented. As a result, model identification is often the most time-consuming step within the process of developing model-based control strategies [28]. Thus, a semi-automatic identification process helps to reduce effort and could also be performed by software agents within a ICPS.

The cornerstone for such a semi-automatic model identification of components in industrial energy systems is a formal, machine-readable knowledge representation, as provided by OWL ontologies. Thus, in this work, information about plant components and equipment, as well as their causal relationships, are modeled in such an ontology. Also, a mapping of available sensor data to the appropriate virtual entity is performed. This enables direct sensor data access via the ontology, as well as the discovery of missing or redundant sensors within the plant's topology.

The ontology's primary information source is the P&I diagram, which is extensively used in plant engineering. PETIont represents the most important information from this P&I diagram, like equipment, topology, and instrumentation. This part of the ontology is used in combination with expert knowledge to create a virtual entity in the ontology automatically, representing the behavior of the physical plant equipment. The additional expert knowledge required to generate this virtual entity with its dynamic equipment models is expressed as SPARQL rules, which are implemented in three steps: The expert knowledge about plant equipment is encoded by rules, which specify the input and output characteristics of the Dynamic Model of equipment. These in and outputs

are related to the Virtual Properties in the PETIont. After each component's Dynamic Model and Virtual Properties have been constructed, equivalent properties are identified within the ontology. Equivalent refers to the fact that they represent the same physical characteristic. This can be done because the flow information is already available in the PETIont and knowledge about basic physics principles, like conservation of mass and energy, are encoded as SPARQL rules. Lastly, the relationship between available and associated sensors and their data is established. This enables OBDA for the available sensor data, which is needed for the identification process.

The proposed ontology-based model identification method is evaluated using a thermal heating process as a use case. Based on the input and output information for the components and the linked sensor data, a dynamic model is identified using the information stored in the ontology. To simulate the dynamic behavior of the components, in the described use case, a linear autoregressive model with exogenous input (ARX) was chosen. However, any regression model can be used. The model was trained using Python's scikitlearn framework, and predictions were made to compare the results with the expected output.

It is shown how a data-driven model development process can be partially automated, using ontological knowledge representation of plant equipment, topology, and instrumentation together with expert knowledge encoded in SPARQL rules. This method reduces the amount of time it takes to create data-driven models for ICPS or can be used within a DT simulation service.

Architectural concept for a Digital Twin service framework (Chapter 7)
Services are an important part for the development of DT. However, in most published DT frameworks or architecture, they are only addressed or described at a high level of abstraction. As most DT implementations are driven by a specific goal rather than an architectural template [27], the same can be said for the services that are deployed. In recent years, the microservice architectural style has gained popularity in ICT for developing distributed software applications with increased scalability and maintainability [13]. Orchestration or choreography is used to perform the service composition. Choreography follows a decentralized approach, whereas orchestration uses a centralized component.

In general, the design and implementation of services should be based on specific requirements guiding architectural principles. Due to a lack of such fundamental requirements and principles, application-specific solutions are produced, which are rarely reusable, which increases development time and costs. Thus, the DT's services infrastructure is investigated, specifically addressing requirements and a service framework architecture that can later be used for a variety of DT applications. Therefore, functional and non-functional requirements for a DT service framework are identified via a literature review. The requirements are grouped into three RAMI 4.0 layers (Information Layer, Functional Layer, and Business Layer) that are relevant to the proposed service framework.

The conceptual microservice framework for DT is designed based on the concepts of the GDTA and the identified requirements. In the proposed architecture, a knowledge

graph based on Semantic Web technologies is used to interlink information and provide semantics to the exchanged data. By facilitating a federated SPARQL query engine, services can add relevant information to the shared knowledge graph. Thus, information can either remain private or become part of the collective knowledge. OBDA is used to enable access to run-time and historical data through the knowledge graph while keeping the data in its original storage.

A Message-oriented Middleware (MOM) with a message broker is used to provide various communication patterns for inter-service communication. At the Business Layer, service composition is crucial for providing certain DT functionality. Choreography or orchestration can be used to provide this functionality. Choreography may provide advantages in some circumstances because it is a more decentralized approach. However, orchestration has advantages if complex control flows across several microservices occur or states have to be handled, like for error handling or user interaction. Workflow engines can not only be used at the enterprise level to automate business workflows, they can also be used to orchestrate microservices [22]. Usually, Business Process Model and Notation (BPMN) is used to describe these workflows. Thus, control flows across several services can be visualized, and long-lived transactions can be handled. Because workflow engines support BPMN, it may be used to communicate with non-software developers and seamlessly add DT capabilities into existing enterprise business processes.

A proof-of-concept for a DT of a thermal heating process is implemented to evaluate the suggested service architecture. A composite sensor data evaluation service, which analyzes sensor data from the plant and discovers anomalies, is used to showcase the service interaction. An anomaly is defined as a sensor fault or abnormal plant behavior induced by a malfunction of equipment. Three separate services are created for the described use case, which are orchestrated by the workflow engine Zeebe[7]. The inter-services communication is realized using the stream-based MOM Apache Kafka [8]. Every service has its own database or triple store, which is connected to a distributed knowledge graph through the federated query engine FedX [45]. JSON for Linking Data (JSON-LD) is used to exchange data across the services and provide semantics. Each service and its infrastructure (database, triplestore, etc.) is virtualized in Docker[9] containers.

With the presented proof-of-concept implementation of the DT service framework architecture, the feasibility has been shown and the design artifacts have been evaluated against the identified requirements.

1.5 Scientific Contribution to the State of the Art

This work contributes to the state of the art in different areas regarding the stated problems and research questions presented in Section 1.1. In the following, the main contributions of the published articles are summarized.

[7] https://github.com/camunda-cloud/zeebe
[8] https://kafka.apache.org/
[9] https://www.docker.com/

Generic Digital Twin architecture for industrial energy systems:
One of the first DT concepts was introduced in [21] and defines three main components: the physical space, the virtual space, and the connection between them to exchange data and information. Later, the so-called 5D-DT was presented in [53] as an extension of this concept, which additionally introduces the data and service aspect. More detailed architectures and frameworks evolved in the context of DT, like the "Intelligent DT" [7], the "Reference Framework for DT" [27], "Cognitive Twin Toolbox" [1], and the "Conceptual Digital Twin Model" [5], which all have different levels of abstraction and naming conventions.

To partly overcome this naming convention problem and provide an architectural template, the thesis contributes a technology-agnostic generic DT architecture that is compatible with the RAMI 4.0 layers. Even if OPC UA is used at the Communication Layer for the proof of concept, other communication protocols would also be possible. Applying the RAMI 4.0 layers to the GDTA could support the creation of a DT by applying common concepts and technologies known from this area. In addition, the concept of a context-dependent view of the Virtual Entity within the DT, which combines special-purpose simulation models with context information, is introduced. The provided prototypical implementation as a proof-of-concept is using Semantic Web technologies and focusing on the simulation service within the DT. It shows how to manage the context information of resources and services inside a DT and how a simulation service can create a view of the Virtual Entity.

Reusing existing OCP UA Information Model to provide semantic context information for the Digital Twin:
An information model of a digital process twin is presented in [9], which can also be converted into OPC UA. Similar information models have to be defined for industrial energy systems to facilitate interoperability. In this domain, P&I diagrams are an important source for generating DT, as shown in [46]. Thus, information models describing the plant topology and instrumentation seem to be useful. Approaches for transforming such OPC UA information models into OWL have been investigated. In [44], a formal mapping between OPC UA and OWL DL is presented. It is demonstrated why there is no simple mapping between OPC UA and OWL DL. One major issue is the use of Instance-Nodes in the OPC UA type definitions, which is not permitted in OWL DL. As a result, mappings can be specified in a variety of ways depending on design choices. Another mapping between OPC UA and OWL DL was presented in [8]. They discovered that axioms and properties in OWL are not available in OPC UA. To avoid the mapping problem OWL Full can be used instead of OWL DL [49]. Nevertheless, this approach is only appropriate if no reasoning is required.

In most cases, only parts of the OPC UA information model are needed and domain ontologies already exist and should be reused to provide interoperability. Thus, the thesis contributes a method for a flexible model transformation process between OPC UA information models and domain-specific ontologies. With the proposed transformation process, domain-specific ontologies can be instantiated based on information available in

already existing OPC UA information models. This approach targets especially existing legacy systems with OPC UA information models in place. The method is flexible regarding the exchange of the source information model and the domain-specific target ontology. Additionally, an OPC UA information model for the process industry based on well-established standards is contributed.

Semantic integration method for OPC UA run-time data of a Digital Twin & query performance evaluation:
The importance of automation models in Industry 4.0 and the Internet of Things (IoT) is described in [26]. The authors stated that next to a unified access method and services, the assignment of machine-interpretable semantics is essential for the digital representation of such models. They also considered linked data concepts and Semantic Web technologies as a suitable technology for such a task. On the other hand, OPC UA is a recommended standard in the context of the RAMI 4.0 Communication and Information layer. Thus, the semantic integration of OPC UA run-time data, which manifests as time-series data, is important for providing this information to DT services. In [30], it is used for sensor discovery To fully exploit the potential of ontologies, a semantic access layer is implemented that does not interfere with the existing OPC UA standard. In [57], OPC UA data access via an ontology is shown by mirroring OPC UA data into a relational database and mapping this data into the ontology.

In both cases, data has to be duplicated, which can lead to inconsistencies and performance issues. Thus, the thesis contributes a mapping between the OPC UA meta model and OWL full as well as an OBDA method for OPC UA run-time data using CPF in SPARQL. With this approach, data is further stored at the OPC UA server and only retrieved by the SPARQL engine when needed. The thesis also contributes an overview of common data integration methods for knowledge graphs in industrial use cases found in literature. Their scalability regarding the query performance of sensor data was evaluated. Additionally, a guideline for choosing the proper integration method for specific use cases is provided.

Method for supporting semi-automatic data-driven model identification:
Semantic data has already been used to identify models automatically. An automatic thermal model creation approach for thermal building models is presented in [31]. Data from an existing Building Information Model (BIM) is retrieved to create a simplified thermal RC-model. In [48], the idea of using OWL ontologies to describe the structure of an energy system is presented. The ontological domain models were built, and object dependencies were modeled. This was combined with statistical anomaly detection algorithms to improve data quality by cleaning sensor data from a power generation facility. The authors of [14] used an ontology-based BIM and the BASont ontology [38]. They presented a qualitative, symbolic, knowledge-based fault propagation approach for building automation systems.

Based on the ideas presented in these publications, the thesis contributes an ontology-based method for deriving causal relations used for data-driven model identification in industrial energy systems. The expert knowledge which is needed to derive these causalities is encoded as SPARQL rules and uses knowledge available in the shared

knowledge graph of the GDTA. Further, an ontology for plant equipment, topology, and instrumentation is developed, based on well-established standards, which is used to describe the physical entity of a DT.

Architectural concept for a Digital Twin service framework:
Services are an important part of ICPS in general. Thus, many frameworks exist proposing services and their composition. In [12], the design and implementation of a DT in smart manufacturing is discussed. A more concrete example of a service framework and service interaction within a DT is provided in [4]. In [55], a service-oriented and event-driven manufacturing information system architecture was proposed, similar to the previously presented architecture. A microservice architecture is also proposed in [6]. The authors present a framework for predictive analytics of IoT applications rather than targeting a DT architecture. The application of the microservice architecture for advanced manufacturing systems is investigated in [24]. The authors identified the suitability but also remarked that pitfalls and challenges have to be tackled when applying this architecture. A comparison between orchestration and choreography for automation systems is presented in [52]. As a decentralized approach, choreography is suitable for less complex functional associations, whereas orchestration should be preferred for more procedural associations. The authors have shown a more practical evaluation of these concepts for automation systems in [51]. Other articles are focusing more on the conceptual aspect of a DT and describe services and their interaction on a more abstract level, not providing any implementation details.

Microservices combined with event-based messaging are used in the majority of the frameworks presented because it can provide benefits such as separation of concerns, technology flexibility, scalability, and so on. Their underlying requirements, on the other hand, are rarely stated. Thus, one of main contribution of this thesis is the specification of functional and non-functional requirements for a DT service framework derived from a literature review. These requirements were grouped into the Information, Functional, and Business layer of RAMI 4.0 relevant to the proposed service framework. Also, a novel microservice architecture for DTs is proposed based on the identified requirements. This architecture combines event-based messaging with a workflow engine for service orchestration and federated knowledge graph to provide semantic interoperability between services.

References

[1] Sailesh Abburu et al. „COGNITWIN – Hybrid and Cognitive Digital Twins for the Process Industry“. In: *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. July. IEEE, June 2020, pp. 1–8. ISBN: 978-1-7281-7037-4. DOI: 10.1109/ICE/ITMC49519.2020.9198403. URL: https://ieeexplore.ieee.org/document/9198403/.

[2] Peter Adolphs et al. *Reference Architecture Model Industrie 4.0 (RAMI4.0)*. Tech. rep. July. VDI/VDE-Gesellschaft, 2015.

[3] R. Akerkar and P. Sajja. *Knowledge-Based Systems*. Jones & Bartlett Learning, 2010.

[4] Ameer B.A. Alaasam, Gleb Radchenko, and Andrey Tchernykh. „Stateful stream processing for digital twins: Microservice-based kafka stream dsl“. In: *SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings* (2019), pp. 804–809. DOI: 10.1109/SIBIRCON48586.2019.8958367.

[5] A. R. Al-Ali et al. „Digital Twin Conceptual Model within the Context of Internet of Things“. In: *Future Internet* 12.10 (2020), p. 163. DOI: 10.3390/fi12100163.

[6] Sajjad Ali, Muhammad Aslam Jarwar, and Ilyoung Chong. „Design methodology of microservices to support predictive analytics for IoT applications“. In: *Sensors (Switzerland)* 18.12 (2018). ISSN: 14248220. DOI: 10.3390/s18124226.

[7] Behrang Ashtari Talkhestani et al. „An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System“. In: *At-Automatisierungstechnik* 67.9 (2019), pp. 762–782. ISSN: 2196677X. DOI: 10.1515/auto-2019-0039.

[8] J. Bakakeu et al. „Automated reasoning and knowledge inference on OPC UA information models“. In: *Proceedings - 2019 IEEE International Conference on Industrial Cyber Physical Systems, ICPS 2019* (2019), pp. 53–60. DOI: 10.1109/ICPHYS.2019.8780114.

[9] Birte Caesar et al. „Information Model of a Digital Process Twin for Machining Processes“. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2020-September (2020), pp. 1765–1772. ISSN: 19460759. DOI: 10.1109/ETFA46521.2020.9212085.

[10] Diego Calvanese et al. „Ontop: Answering SPARQL Queries over Relational Databases“. In: *Semantic Web Journal* (2017). URL: https://github.com/ontop/ontop-examples/.

[11] Chiara Cimino, Elisa Negri, and Luca Fumagalli. „Review of digital twin applications in manufacturing“. In: *Computers in Industry* 113 (2019), p. 103130. ISSN: 01663615. DOI: 10.1016/j.compind.2019.103130. URL: https://doi.org/10.1016/j.compind.2019.103130.

[12] Violeta Damjanovic-Behrendt and Wernher Behrendt. „An open source approach to the design and implementation of Digital Twins for Smart Manufacturing“. In: *International Journal of Computer Integrated Manufacturing* 32.4-5 (2019), pp. 366–384. ISSN: 13623052. DOI: 10.1080/0951192X.2019.1599436. URL: https://doi.org/10.1080/0951192X.2019.1599436.

[13] Lorenzo De Lauretis. „From monolithic architecture to microservices architecture“. In: *Proceedings - 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019* (2019), pp. 93–96. DOI: 10.1109/ISSREW.2019.00050.

[14] Henrik Dibowski, Ondřej Holub, and Jiří Rojíček. „Ontology-based fault propagation in building automation systems“. In: *International Journal of Simulation: Systems, Science and Technology* 18.3 (2017), pp. 1.1–1.14. ISSN: 1473804X. DOI: 10.5013/IJSSST.a.18.03.01.

[15] Nicola Dragoni et al. „Microservices: Yesterday, Today, and Tomorrow“. In: *Present and Ulterior Software Engineering*. Cham: Springer International Publishing, 2017, pp. 195–216. ISBN: 978-3-319-67425-4. DOI: 10.1007/978-3-319-67425-4_12. URL: https://doi.org/10.1007/978-3-319-67425-4_12.

[16] Peter Drahos et al. „Trends in industrial communication and OPC UA“. In: *Proceedings of the 29th International Conference on Cybernetics and Informatics, K and I 2018* 2018-Janua (2018), pp. 1–5. DOI: 10.1109/CYBERI.2018.8337560.

[17] Lisa Ehrlinger and Wolfram Wöß. „Towards a definition of knowledge graphs“. In: *CEUR Workshop Proceedings* 1695 (2016). ISSN: 16130073.

[18] M Fernández-López, A Gómez-Pérez, and Natalia Juristo. „METHONTOLOGY: From Ontological Art Towards Ontological Engineering“. In: *Spring Symposium on Ontological Engineering of AAAI* SS-97-06 (1997), pp. 33–40. ISSN: 0769530303. DOI: 10.1109/AXMEDIS.2007.19. URL: http://oa.upm.es/5484/.

[19] Andreas Fernbach, Igor Pelesic, and Wolfgang Kastner. „Linked Data for Building Management“. In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. Florence, Italy: IEEE, 2016, pp. 6943–6945. ISBN: 9781509034741. DOI: 10.1109/IECON.2016.7793781.

[20] B Glimm et al. „HermiT: reasoning with large ontologies“. In: *Computing Laboratory, Oxford University* (2009), p. 64.

[21] Michael Grieves. „Digital Twin : Manufacturing Excellence through Virtual Factory Replication This paper introduces the concept of a A Whitepaper by Dr . Michael Grieves“. In: *White Paper* March (2014). URL: https://www.researchgate.net/publication/275211047%7B%5C_%7DDigital%7B%5C_%7DTwin%7B%5C_%7DManufacturing%7B%5C_%7DExcellence%7B%5C_%7Dthrough%7B%5C_%7DVirtual%7B%5C_%7DFactory%7B%5C_%7DReplication.

[22] Antonio Manuel Gutiérrez–Fernández, Manuel Resinas, and Antonio Ruiz–Cortés. „Redefining a process engine as a microservice platform“. In: *Lecture Notes in Business Information Processing* 281 (2017), pp. 252–263. ISSN: 18651348. DOI: 10.1007/978-3-319-58457-7_19.

[23] René Hofmann et al. „Comparison of a physical and a data-driven model of a Packed Bed Regenerator for industrial applications“. In: *Journal of Energy Storage* 23 (2019), pp. 558–578. ISSN: 2352-152X. DOI: 10.1016/j.est.2019.04.015.

[24] Aydin Homay et al. „A survey: Microservices architecture in advanced manufacturing systems“. In: *IEEE International Conference on Industrial Informatics (INDIN)* 2019-July (2019), pp. 1165–1168. ISSN: 19354576. DOI: 10.1109/INDIN41052.2019.8972079.

[25] Ahmed Ismail, Hong-Linh Truong, and Wolfgang Kastner. „Manufacturing process data analysis pipelines: a requirements analysis and survey“. In: *Journal of Big Data* 6.1 (2019), p. 1. ISSN: 2196-1115. DOI: 10.1186/s40537-018-0162-3. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-018-0162-3.

[26] Jürgen Jasperneite, Thilo Sauter, and Martin Wollschlaeger. „Why We Need Automation Models“. In: *Ieee Industrial Electronics Magazine* 14.1 (2020), pp. 29–40.

[27] Klementina Josifovska, Enes Yigitbas, and Gregor Engels. „Reference Framework for Digital Twins within Cyber-Physical Systems“. In: *Proceedings - 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems, SEsCPS 2019* (2019), pp. 25–31. DOI: 10.1109/SEsCPS.2019.00012.

[28] M. Killian and M. Kozek. „Ten questions concerning model predictive control for energy efficient buildings“. In: *Building and Environment* (2016). ISSN: 03601323. DOI: 10.1016/j.buildenv.2016.05.034.

[29] Gunther Koschnick. *Industrie 4.0: Smart services*. Tech. rep. July. Frankfurt am Main, Germany: German Electrical and Electronic Manufacturers' Association, 2016, pp. 13–14.

[30] Christoph Legat, Christian Seitz, and Birgit Vogel-Heuser. „Unified sensor data provisioning with semantic technologies“. In: *ETFA2011*. Toulouse, France: IEEE, Sept. 2011, pp. 1–8. ISBN: 978-1-4577-0017-0. DOI: 10.1109/ETFA.2011.6058989. URL: http://ieeexplore.ieee.org/document/6058989/.

[31] G. N. Lilis et al. „Semi-automatic thermal simulation model generation from IFC data“. In: *eWork and eBusiness in Architecture, Engineering and Construction - Proceedings of the 10th European Conference on Product and Process Modelling, ECPPM 2014* Eastman 2011 (2015), pp. 503–510. DOI: 10.1201/b17396-83.

[32] Somayeh Malakuti et al. „Digital Twins for Industrial Applications. Definition, Business Values, Design Aspects, Standards and Use Cases“. In: *White Paper* (2020), pp. 1–19.

[33] David Martin et al. „OWL-S: Semantic markup for web services“. In: *W3C member submission* 22.4 (2004). URL: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[34] Michael Martin, Jörg Unbehauen, and Sören Auer. „Improving the performance of semantic web applications with SPARQL query caching“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6089 LNCS.PART 2 (2010), pp. 304–318. ISSN: 03029743. DOI: 10.1007/978-3-642-13489-0_21.

[35] Zoran K. Morvay and Duan D. Gvozdenac. *Applied Industrial Energy and Environmental Management*. Chichester, UK: John Wiley & Sons, Ltd, Oct. 2008. ISBN: 9780470714379. DOI: 10.1002/9780470714379. URL: http://doi.wiley.com/10.1002/9780470714379.

[36] Benjamin Mörzinger et al. „Improving industrial optimizationwith Semantic Web technologies“. In: *SEMANTiCS 2018*. Vol. 2198. 2018.

[37] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Tech. rep. Stanford Knowledge Systems Laboratory, 2001, p. 25. DOI: 10.1016/j.artmed.2004.01.014. arXiv: 1304.1186.

[38] Joern Ploennigs et al. „BASont - A modular, adaptive building automation system ontology“. In: *IECON Proceedings (Industrial Electronics Conference)* (2012), pp. 4827–4833. DOI: 10.1109/IECON.2012.6389583.

[39] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. „OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation“. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), pp. 7–34.

[40] Gustavo Correa Publio et al. „ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies“. In: *CoRR* abs/1807.05351 (2018). arXiv: 1807.05351. URL: http://arxiv.org/abs/1807.05351.

[41] Vasja Roblek, Maja Meško, and Alojz Krapež. „A Complex View of Industry 4.0“. In: *SAGE Open* 6.2 (Apr. 2016). ISSN: 2158-2440. DOI: 10.1177/2158244016653987. URL: http://journals.sagepub.com/doi/10.1177/2158244016653987.

[42] Daniel Schachinger, Wolfgang Kastner, and Stefan Gaida. „Ontology-based abstraction layer for smart grid interaction in building energy management systems“. In: *2016 IEEE International Energy Conference, ENERGYCON 2016* (2016). DOI: 10.1109/ENERGYCON.2016.7513991.

[43] Rainer Schiekofer and Michael Weyrich. „Querying OPC UA information models with SPARQL“. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2019-Septe (2019), pp. 208–215. ISSN: 19460759. DOI: 10.1109/ETFA.2019.8868246.

[44] Rainer Schiekofer et al. „A formal mapping between OPC UA and the semantic web“. In: *IEEE International Conference on Industrial Informatics (INDIN)* 2019-July (2019), pp. 33–40. ISSN: 19354576. DOI: 10.1109/INDIN41052.2019.8972102.

[45] Andreas Schwarte et al. „FedX: Optimization techniques for federated query processing on linked data“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7031 LNCS.PART 1 (2011), pp. 601–616. ISSN: 03029743. DOI: 10.1007/978-3-642-25073-6_38.

[46] Seppo Sierla et al. „Integrating 2D and 3D Digital Plant Information Towards Automatic Generation of Digital Twins“. In: *IEEE International Symposium on Industrial Electronics* 2020-June (2020), pp. 460–467. DOI: 10.1109/ISIE45063.2020.9152371. arXiv: 2104.01854.

[47] Gregory A. Silver, Osama Al-Haj Hassan, and John A. Miller. „From domain ontologies to modeling ontologies to executable simulation models“. In: *Proceedings - Winter Simulation Conference* (2007), pp. 1108–1117. ISSN: 08917736. DOI: 10.1109/WSC.2007.4419710.

[48] Nina Solomakhina et al. „Extending statistical data quality improvement with explicit domain models“. In: *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, July 2014, pp. 720–725. ISBN: 978-1-4799-4905-2. DOI: 10.1109/INDIN.2014.6945602. URL: http://ieeexplore.ieee.org/document/6945602/.

[49] Gernot Steindl, Thomas Früwirth, and Wolfgang Kastner. „Ontology-Based OPC UA Data Access via Custom Property Functions“. In: *24th Interantional Conference on Emerging Technologies and Factory Automation*. Zaragoza, Spain, 2019.

[50] Carsten Stöhr et al. „Smart Services“. In: *Procedia - Social and Behavioral Sciences* 238 (2018), pp. 192–198. ISSN: 18770428. DOI: 10.1016/j.sbspro.2018.03.023. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877042818300235.

[51] Andreas Stutz et al. „Choreographies in Microservice-Based Automation Architectures: Next Level of Flexibility for Industrial Cyber-Physical Systems“. In: *Proceedings - 2020 IEEE Conference on Industrial Cyberphysical Systems, ICPS 2020* (2020), pp. 411–416. DOI: 10.1109/ICPS48405.2020.9274719.

[52] Andreas Stutz et al. „Orchestration vs. Choreography Functional Association for Future Automation Systems“. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 8268–8275. ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.12.1961. URL: https://doi.org/10.1016/j.ifacol.2020.12.1961.

[53] Fei Tao, Meng Zhang, and A.Y.C. Nee. „Five-Dimension Digital Twin Modeling and Its Key Technologies“. In: *Digital Twin Driven Smart Manufacturing* (2019), pp. 63–81. DOI: 10.1016/b978-0-12-817630-6.00003-5.

[54] Fei Tao et al. „Digital Twin in Industry: State-of-the-Art“. In: *IEEE Transactions on Industrial Informatics* 15.4 (2019), pp. 2405–2415. ISSN: 15513203. DOI: 10.1109/TII.2018.2873186.

[55] Alfred Theorin et al. „An event-driven manufacturing information system architecture for Industry 4.0“. In: *International Journal of Production Research* 55.5 (2017), pp. 1297–1311. ISSN: 1366588X. DOI: 10.1080/00207543.2016.1201604.

[56] VDI/VDE. *VDI/VDE 2193 - Language for I4.0 components*. Tech. rep. Verein Deutscher Ingenieure & Verband der Elektrotechnik Elektronik Informationstechnik, 2020.

[57] Shiyong Wang et al. „Knowledge reasoning with semantic data for real-time data processing in smart factory“. In: *Sensors (Switzerland)* 18.2 (2018), pp. 1–10. ISSN: 14248220. DOI: 10.3390/s18020471.

[58] Martin Wollschlaeger, Thilo Sauter, and Juergen Jasperneite. „The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0“. In: *IEEE Industrial Electronics Magazine* 11.1 (2017), pp. 17–27. ISSN: 19324529. DOI: 10.1109/MIE.2017.2649104.

[59] World Wide Web Consortium. *Semantic Sensor Network Ontology. W3C Recommendation*. https://www.w3.org/TR/vocab-ssn/. (accessed: 19.05.2021).

CHAPTER 2

Generic Digital Twin Architecture for Industrial Energy Systems

Publication: *G. Steindl, M. Stagl, L. Kasper, W. Kastner, and R. Hofmann, "Generic Digital Twin Architecture for Industrial Energy Systems," Applied Sciences, vol. 10, no. 24, p. 8903, Dec. 2020, doi: 10.3390/app10248903.*

Abstract: Digital Twins have been in the focus of research in recent years, trying to achieve the vision of Industry 4.0. In the domain of industrial energy systems, they are applied to facilitate a flexible and optimized operation. With the help of Digital Twins, the industry can participate even stronger in the ongoing renewable energy transition. Current Digital Twin implementations are often application-specific solutions without general architectural concepts and their structures and namings differ, although the basic concepts are quite similar. For this reason, we analyzed concepts, architectures, and frameworks for Digital Twins in the literature to develop a technology-independent Generic Digital Twin Architecture (GDTA), which is aligned with the information technology layers of the Reference Architecture Model Industry 4.0 (RAMI4.0). This alignment facilitates a common naming and understanding of the proposed architectural structure. A proof-of-concept shows the application of Semantic Web technologies for instantiating the proposed GDTA for a use case of a Packed-Bed Thermal Energy Storage (PBTES).

2.1 Introduction

2.1.1 Motivation

Digitalization is changing the way business is conducted within industrial value chains, facilitated by the rapid development of communication and information technology [33].

This process is also referred to as the fourth industrial revolution or Industry 4.0. The goal is a highly optimized and customized production, as well as enhanced automation and adaption capabilities [37]. To realize these visions of Industry 4.0, the DT is one of the most promising enabling technologies [39].

Industry 4.0 and the sustainable energy transition share important characteristics and can mutually benefit from each other [41]. Information and communication technology helps to increase energy efficiency and the interaction of industry with smart grids which facilitates the integration of renewable energy sources. DTs are also the key enabler for such applications, as their common functionality includes monitoring, diagnostic, prediction, and control [26].

The concepts and capabilities of DTs are not clearly defined and sometimes hard to grasp. This is caused by the fact that DTs can be applied for various tasks in different life-cycle phases and industrial domains. Thus, different interpretations of a DT exist, driven by specific use cases. This leads to the problem that concrete realizations of DTs follow a concrete goal without any architectural template [22]. For this reason, we present a novel generic architecture for a DT, which is in line with the RAMI 4.0, called GDTA. The GDTA facilitates a technology independent implementation of DTs and gives orientation for locating existing frameworks and technologies inside this architectural model. We also introduce a context-dependent View on the Virtual Entity inside the DT, consisting of special-purpose simulation models in combination with context information. Additionally, a prototypical implementation of the proposed GDTA is presented as a proof-of-concept, using Semantic Web technologies. It demonstrates how the context information of resources and services can be managed inside a DT, and how a View of the Virtual Entity can be based on a Simulation Service.

The remainder of the paper is structured as follows: Section 2.1.2 gives a short overview of related work in the area of DT concepts, frameworks, and reference architectures. Section 2.2 describes the methods which are applied for carrying out the presented work. Afterward, the GDTA is introduced in Section 2.3. The proposed architecture is used for our proof-of-concept implementation, based on Semantic Web technologies and applied for a use case of a thermal energy storage system in Section 2.4. In the end, the presented GDTA is discussed with respect to other DT architectural frameworks, and an outlook on our future work is given.

2.1.2 Related Work

In general, a DT can be defined as “a formal digital representation of some asset, process or system that captures attributes and behaviors of that entity suitable for communication, storage, interpretation or processing within a certain context” [28].

A very basic concept for structuring a DT defines three different aspects: the physical space, the virtual space, and the connection between them to exchange data and information [14]. A similar concept is known from the industrial domain as CPS or more specifically as Cyber-Physical Production System (CPPS). In [32], CPSs are described as

autonomous and cooperative elements and sub-systems across all levels of production, able to communicate with each other in situation-dependent ways. The goal of CPSs is to have elements that can acquire and process data, allowing them to self-control certain tasks and interact with humans. To reach that goal, a certain kind of virtual representation of the production system has to be available. Therefore, a CPS can be characterized by a physical asset and its cyber counterpart, which means that a DT can be seen as only the digital model inside a CPS [26]. Conversely, this also implies that a DT is the prerequisite for a CPS [40].

For CPSs, a five layer architecture (5C architecture) was proposed in [25], defining a "Smart Connection Level", "Data-to-Information Conversion Level", "Cyber Level", "Cognition Level", and "Configuration Level". These layers should help to develop and implement CPSs at a certain layer of this 5C architecture. In this context, the Smart Connection Level has to deal with acquiring accurate and reliable data from the physical entity and is the first step to create a CPS. Afterward, meaningful information is inferred from the data at the Data-to-Information Conversion Level. This level brings self-awareness to the machines. The Cyber Level acts as an information hub inside the 5C architecture, which also introduces the possibility of self-comparison of the performance of machines. In-depth knowledge of the monitored system is created at the Cognition Level. Expert users will be supported by this information to make the correct decision. At the Configuration Level, feedback is given from the cyber-space to the physical space. Here, the supervisory control resides and makes machines self-configurable and self-adaptive. This functional view of a CPS can also be beneficial for designing and implementing a DT for certain applications. Typical applications have been identified by a literature review in [21] and can be clustered in the following categories: simulation and optimization, monitoring, diagnosis, and prediction.

Another important concept for describing DTs is the so-called 5D-DT [38]. It is an evolution of the previously mentioned DT concept, which extends the three dimensions ("Physical Entity", "Virtual Entity", "Connection") by the data aspect as well as the service aspect. These five dimensions and their relations are shown in Figure 2.1. The "Physical Entity" consists of various subsystems that perform specific tasks, facilitated with different sensors that collect the states and working parameters. The "Virtual Entity" aims to model the physical entity with high precision by the integration of multiple different types of models such as geometry models, physical models, behavior models, and rule models. The "Service Model" includes services for the "Physical Entity" and the "Virtual Entity". It optimizes the operations of the "Physical Entity" and ensures the high fidelity of the "Virtual Entity" through calibration of the "Virtual Entity" parameters during run-time. The "Data Model" consists of five parts: data from the physical entity, data from the virtual entity, data from the services, domain knowledge, and the fusion of those data. The "Connection Model" describes each connection between the components of the DT. The ideas of the 5D-DT concept are the foundation for our proposed GDTA.

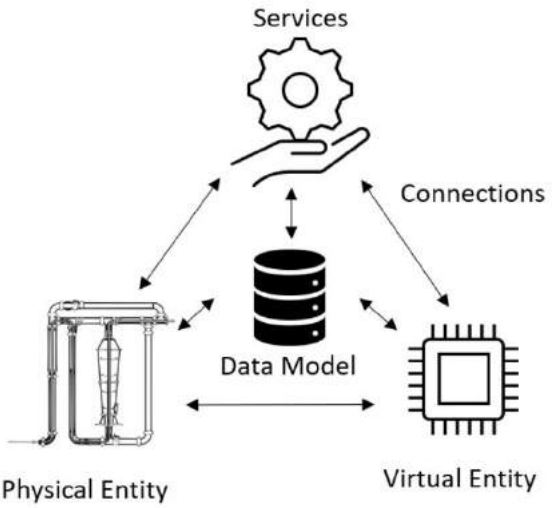


Figure 2.1: Five dimensions of the Five-Dimensional Digital Twin (5D-DT) concept adapted from [38].

In addition to these very general concepts with a high level of abstraction, more detailed architectural concepts and frameworks have been proposed for the implementation of DTs. In the following paragraph, an overview of these concepts is given and summarized in Table 2.1. Table 2.1 also gives a classification based on their level of abstraction. A high level of abstraction means that a more general concept is presented, whereas a low level indicates a more concrete architecture or framework, targeting the implementation of a DT. They are also used in the discussion (Section 2.5) to set our proposed GDTA into perspective by comparing it with these concepts.

"Intelligent Digital Twin"—In [5], an architecture for an "Intelligent Digital Twin" was proposed by applying algorithms known from Artificial Intelligence (AI). Next to the data acquisition interface, a synchronization interface is introduced to keep the simulation models of the DT in line with the physical asset, as they can otherwise differ over its life-cyle. Also, a co-simulation interface is described as a component of the architecture to enable communication with other DTs and to facilitate multidisciplinary co-simulation.

"Reference Framework for Digital Twins"—A reference framework for DTs in the context of the 5C architecture of CPS is presented in [22]. The main building blocks of DTs, including their properties (structure and interrelation), were specified for the proposed framework. Therefore, a systematic literature review was conducted followed by a framework analysis using grounded theory. They identified four main building blocks of a DT which are: a "Physical Entity Platform", a "Virtual Entity Platform", a "Data Management Platform", and a "Service Platform". They also identified three different types of physical entities, namely the "Physical Objects", "Physical Nodes", and "Humans". The "Virtual Entity Platform" consists of many virtual entity models, including information to mirror a certain aspect of the physical entity. It is responsible for the generation

and maintenance of "Semantic Models" (geometric models, physical models, behavioral models, rule models, process models) of the physical entity to create its virtual representation. The "Data Management Platform" performs data acquisition, management (collection, transmission, storage, integration, processing, cleaning, analysis, data mining and information extraction), and storage. The "Service Platform" consists of service models and a service management layer to organize services for concrete applications.

"Cognitive Twin Toolbox"—In [1], a conceptual architecture of the Cognitive Twin Toolbox (COGNITWIN) is presented with a special focus on the process industry. Three levels of twins were defined: A "Digital Twin" which only uses isolated models of the physical system, a "Hybrid Twin" which is also able to interconnect its models, and a "Cognitive Twin" which uses extended models that include expert knowledge for problem-solving and to handle unknown situations. The toolbox proposes five layers: "Model Management Layer", "Data Ingestion and Preparation Layer", "Service Management Layer", "Twin Management Layer", and a "User Interaction Layer". The specified model types are quite similar to the defined semantic models in [22] and consist of first-order principle models based on the underlying physics, empirical models, e.g., AI algorithms, and knowledge-driven models based on domain experts. The "Service Management Layer" is responsible for handling services, like registration and orchestration. Two types of services are distinguished. Data-driven and model-based driven services. The "Twin Management Layer" manages the structure of the DT. Especially, the synchronization problem caused by changes in the behavior of the physical system is handled here. The toolbox also introduces a "User Interaction Layer" where users can explore the COGNITWIN.

"Conceptual Digital Twin Model"—A conceptual model for a DT in the context of IoT is presented in [3]. The model is structured into five layers: "Physical Space", "Communication Network", "Virtual Space", "Data Analytics and Visualization", and "Application". Security aspects are also covered explicitly by a vertical "Security Layer" that overlaps with all other layers. The tasks of the various layers are quite similar to the already mentioned layers in other frameworks. Two conceptual use cases in the automotive domain and smart health care area are described, but no real implementation of the proposed model is presented.

"Asset Administration Shell"—In the context of the Industry 4.0 initiative, the AAS is introduced as a standardized digital representation of an asset. It is used to uniquely identify and describe the functionality of asset as well as the AAS. It also holds various models of certain aspects. Details of how the information of the AAS can be exchanged in a meaningful way between partners along a value chain can be found in [6], where a meta-model for the AAS is defined. As resource description, discovery, and access are the basic functionality of DTs [1], the current state of the AAS is only a first part of the solution. The discovery and the definition of how operations are provided and described by standardized interfaces is ongoing work for the AAS.

To structure the functionality of the presented architectures and frameworks, some kind of layered architecture is used to handle the complexity. As presented, layers are defined with different names, which often have similar functionality. This prevents the

establishment of a common view and terminology in the context of DTs. In order to solve this problem in the context of the Industry 4.0, RAMI 4.0 was introduced [2]. RAMI 4.0 is used to achieve a common understanding of standards, tasks, and use cases. Therefore, three different aspects or dimensions are used by RAMI 4.0: It expands the hierarchy levels of IEC 62264 [18] by "Product" and "Connected World", defines six layers for an IT representation of an Industry 4.0 component and considers the life cycle of the product or system according to IEC 62890. The life cycle is divided into a "type" and "instance" phase. The "type" phase is part of the engineering phase, which ends when a prototype is available. An "instance" is the system or product, when it reaches the operational phase in the life cycle. We used the mentioned six IT layers of the RAMI 4.0 (Business, Functional, Information, Communication, Integration and Asset Level) to structure our proposed GDTA (Section 2.3) to achieve a consistent naming and understanding of the used layers.

Table 2.1: Overview of concepts, architectures, and frameworks for Digital Twins (DT).

Name	**Target Domain**	**Structure**	**Main Parts**	**Level of Abstraction**
3D-DT [14]	Life-cycle Management	component-based	3 components	high
5D-DT [38]	Manufacturing	component-based	5 components	high
5C Architecture [25]	CPS in manufacturing	layer-based	5 layers	high
Intelligent DT [5]	Production Systems	component-based	4 interfaces & 9 components	low
Ref. Framework for DT [22]	CPS in general	component-based	4 main components	low
COGNITWIN [1]	Process Industry	components & layers	5 layers & 19 components	low
Conceptual DT Model [3]	CPS in general	layer-based	6 layers	medium
ASS [6]	Manufacturing	only meta-model	ongoing work	—

2.2 Materials and Methods

A literature review was conducted to identify common services, concepts, architectures, and frameworks in the context of industrial DTs applied to the operational phase of the life-cycle. These concepts were analyzed and used to develop a technology-independent generic architecture GDTA in line with RAMI 4.0.

Afterward, a proof-of-concept was implemented, based on Semantic Web technologies and ontologies. Therefore, a DT was implemented based on the GDTA as an exploratory prototype. The focus of the presented proof-of-concept is put on the Simulation Service and the modeling of context information, based on ontologies, to create a certain View on the Virtual Entity. Also, the Simulation Service was identified as a base service inside

the DT.

The ontologies are developed following the METHONTOLOGY approach [11]. The implemented ontologies have been evaluated with the help of the OOPS! Pitfall Scanner [35], which detects common inadequacies made during the ontology development process. The logical consistency of our ontology has been evaluated with the HermiT 1.4.3.456 reasoner [12].

To evaluate the proof-of-concept implementation, a thermal energy storage system was chosen as use case. With the help of the implemented DT for the PBTES, the basic simulation functionality is evaluated.

2.3 Proposed Generic Digital Twin Architecture

In this section, we present our proposed GDTA which targets the applications of DTs during the operational phase of an asset. The architectural model of the GDTA is depicted in Figure 2.2. The architecture is aligned with the IT layers of RAMI 4.0 to structure the specified components. The hierarchy levels of RAMI 4.0 are not taken into account because a DT can be located at various levels, depending on its application or the physical entity for which it is designed for. Thus, a DT can potentially cover all hierarchy levels of RAMI 4.0. As mentioned before, our DT architecture targets the "instance-phase" of the RAMI 4.0 life-cycle. Thus, only the IT layer dimension of RAMI 4.0 is shown in Figure 2.2 and the architecture is structured based on this dimension.

The GDTA is based on the introduced 5D-DT concept and inspired by the 5C architecture for CPSs, because both concepts have a sufficient level of abstraction, which means they provide a more conceptual view than a concrete architecture. However, they are still useful to identify key components and functionality for DT.

The proposed GDTA defines the basic structure and components of a DT without specifying or binding it to certain technologies. For realizing a DT based on the GDTA, various existing technologies and frameworks can be used to implement its functionality on different layers. The specified components of the DT are explained in more detail regarding their associated RAMI 4.0 layer in the following paragraph.

Asset Layer—The physical representation of the DT is located at the Asset Layer and corresponds to the Physical Entity of the 5D-DT concept. It holds the physical parts of the CPS.

Integration Layer—At the integration layer, Run-time Data as well as Engineering Data can be distinguished. Run-time Data are generated by sensors or events and represent the current state of the physical entity. They are usually time-series data. These data are very dynamic, so the underlying infrastructure has to follow application-specific requirements, like big data processing or real-time reaction. Archives, e.g., special time-series databases, can be used to store these data for diagnosis and model identification

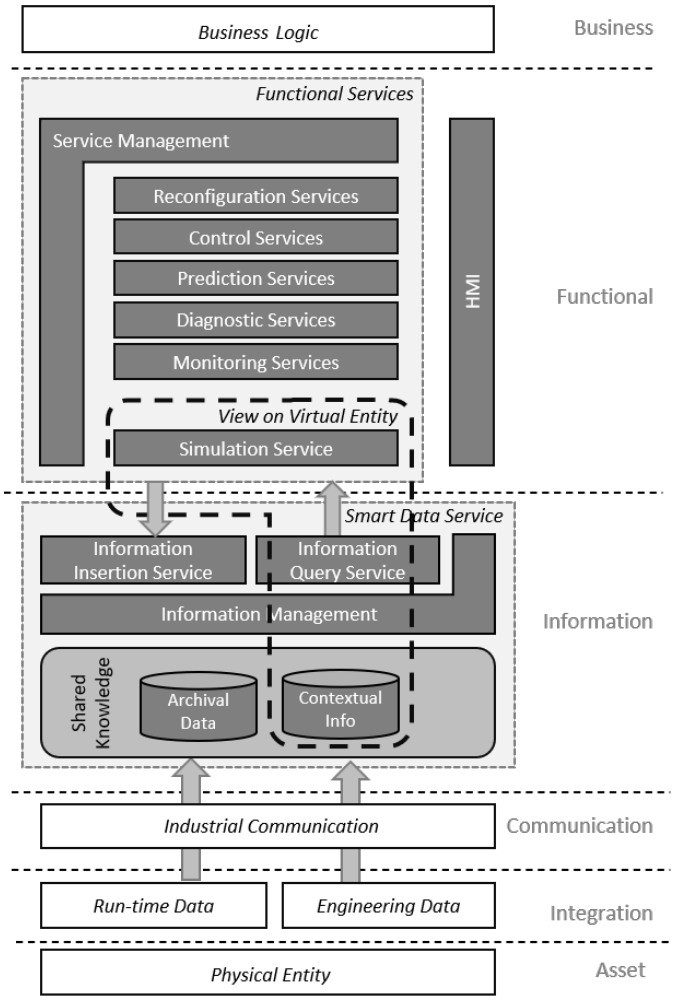


Figure 2.2: Generic Digital Twin Architecture (GDTA) model.

purposes. Engineering Data are usually static data, which means they will not often change over time. Examples are the plant topology or information about a physical component inside a plant. This information is mostly available in analog formats, like drawings of pipe and instrumentation diagrams, and has to be digitized. Even if the information is already available in machine-readable form, the information has to be transformed and integrated into the DT as contextual information.

Communication Layer—Industrial communication or IIoT protocols can be applied within the Communication Layer of the DT. The concrete protocol depends on the requirements of the application, e.g., real-time capabilities or publication/subscription support. Thus, a combination of various protocols can be expected, which are made transparent to the upper layers by the Smart Data Service. Here, OPC UA is a representative in the context of Industry 4.0. Next to its communication capabilities, information modeling

also enables the representation and access to context information through OPC UA.

Information Layer—Within the Information Layer of the DT architecture, data is gathered and enriched with semantics and related with other context information. Thus, the data dimension of the 5D-DT becomes an information dimension at this stage. The central component inside the Information Layer is a Shared Knowledge base, which stores contextual information about resources and services. Data are linked with this information to add semantics to it. Different levels of semantic expressivity can be achieved by applying different information modeling technologies. Ideally, data are stored in their original formats in proper databases. The Shared Knowledge base acts as a semantic integration layer for run-time data and engineering data by providing access to historical data (archives) and holding contextual information inside the Shared Knowledge. Access to the information inside the Shared Knowledge is granted by an Information Query Service. This service can retrieve information from the knowledge base and provide it to other services. The Information Insertion Service is used to add or change information from the upper layers of the DT architecture inside the Shared Knowledge base. Thus, information ownership and access have to be managed including data access mechanisms. This is performed by an Information Management component inside the DT architecture, which handles the information access of the Query and Insertion Service.

The above-mentioned components and services form the so-called Smart Data Service [24], which builds a central point of information inside the architecture. It provides information about resources and services through the Shared Knowledge base and makes it accessible for other services in the Functional Layer.

Functional Layer—In the Functional Layer, the service dimension of the 5D-DT concept is realized. A service-oriented architecture is applied to enable loose coupling and cohesion of certain functionality [20]. The services of the DT can be grouped by their functionality and can build on each other. Five groups are identified: Simulation Services, Monitoring Services, Diagnosis Services, Prediction Services, Control Services, and Reconfiguration Services. Next to these functional services, a Service Management component takes care of service registration, discovery, and obtaining status information about certain services. This information about a service is part of the Shared Knowledge in the Information Layer and will be inserted and queried through the offered Smart Data Service interface. The resource management is not handled by a central component, as the services have to handle their resources by themselves, facilitating the Shared Knowledge base.

Simulation Services are the core services, as they are part of the Virtual Entity of the 5D-DT concept. Typically, there exists not only one model of the physical entity in a DT, but a set of executable models that are specific for the intended purpose and also evolve over time [8]. Thus, different models for various domains, like the mechanical structure, thermal behavior, etc. can be hosted and used inside the Simulation Service of a DT. As for the other services, resources have to be managed by the service itself, and the related information about the models has to be made available through the Shared Knowledge base. A simulation model hosted by the Simulation Service in combination

with related context information from the Shared Knowledge base generates a certain View of the Virtual Entity inside the DT.

Monitoring Services are elementary services to acquire data from the physical entity and observe its current state. An example could be a fault detection service, which can be implemented based on simple statistical models and indicate abnormal operating conditions of a plant.

Diagnostic Services are services supporting, for instance, condition monitoring or root cause analysis of faults. They can build upon underlying Monitoring Services in combination with Simulation Services to gain more insight into the current state of the Physical Entity.

Prediction Services are important for the DT to make decisions based on information about future events. Such services can be used, for instance, for predictive maintenance or the prediction of energy consumption. Also, external variables, like renewable energy production or prices at the energy market, can be predicted. Additionally, external prediction services can be integrated. The prediction results can be used by Control Services for realizing an optimized control or used to generate recommendations for the operating staff.

Control Services have an influence on the operation of the plant via recommendations over Human-Machine Interface (HMI) or direct access to the process control. Control Services with direct access can bypass the Smart Data Service in the Information Layer to change the state of the physical entity without additional delay. Usually, Control Services make use of monitoring, diagnosis, and prediction services to achieve optimized operation. Thereby, the control strategy can change over time, caused by a reconfiguration of the Physical Entity or new objectives specified by the business logic.

Reconfiguration Service. Reconfiguration means a rather static change of the fundamental properties of the Physical Entity by the DT itself. This has a significant influence on the context information inside the Shared Knowledge. Reconfiguration can be initialized through events or changed objectives inside the business logic, which resides inside the business layer. The Reconfiguration Service takes care of such changes inside the DT.

Humans have to be informed about the current state of the Physical Entity as well as the DT itself to interact with it. Thus, an appropriate HMI is very important for almost every service of a DT.

Business Layer—In this layer, the business logic resides, which can also orchestrate a large amount of DT. It defines the overall objectives (e.g., to reduce the risk of downtime or cost), which should be reached with the help of the DT. As this highly depends on the business strategy, it is not relevant for the design of the GDTA but only relevant via specific inputs.

2.4 Proof-of-Concept: Digital Twin Instantiation

The following section describes a prototypical implementation of the proposed GDTA as it is shown in Figure 2.2. The focus of the implementation is the Virtual Entity consisting of a Simulation Service in combination with context information, as it is a fundamental service inside the DT. The code of the prototype is publicly available in a GitLab repository (`https://gitlab.tuwien.ac.at/iet/public/GDTA_Prototype`).

To realize the Shared Knowledge base, Semantic Web technology is used, based on RDF and OWL to describe the resources and services. Already existing ontologies, like OWL-S, ML-Schema [29] and OWL-Time [16] are reused and extended with new concepts. The ontologies are created using the tool Protégé [23]. The ontologies are loaded into Apache Jena Fuseki Server v.3.16.0 running in a Docker container. Jena Fuseki provides a SPARQL endpoint to access the information of the Shared Knowledge base. The service invocation is performed via Hypertext Transfer Protocol (HTTP) in combination with a REST API as suggested in [19].

2.4.1 Knowledge Representation Inside the Shared Knowledge Base

Semantic Web technologies are used to build the Shared Knowledge base as a central part of our Smart Data Service. Ontologies are used to hold information about resources and services in a formal and machine-readable way. A hierarchical design approach is used, consisting of a top-level-ontology and a domain ontology, as shown in Figure 2.3. Further hierarchy levels would be possible. The top-level ontology defines general terms that are common across all sub ontologies. Terms in the domain ontology are ranked under the terms of the top-level ontology. For the presented implementation the Base Service Ontology is defined as a top-level ontology holding general information about the available services of the DT. Other services implement their information inside the domain ontologies. For the current prototype, only the Simulations Service Ontology is implemented, but other domain ontologies can be developed quite similar.

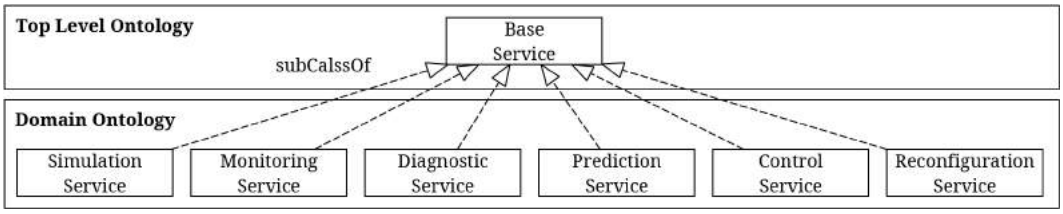


Figure 2.3: Top Level and Domain Ontology structure.

Base Service Ontology

The Base Service Ontology holds the necessary information for a general service description, which helps to discover and access the service. The implementation of the Base Service Ontology consists of a service description based on OWL-S profile classes and properties, additionally defined Quality of Service (QoS) metrics for the services, and a

specification of a service endpoint. OWL-S provides a set of vocabulary and semantic rules for formal description of Web Services. The information about a service is described using the OWL-S Profile, a subclass of ServiceProfile. More Information about OWL-S can be found in [29]. The main concepts and relations of the ontology are depicted in Figure 2.4. The grey concepts are part of OWL-S, whereas the white concepts are extensions and explained in more detail.

BaseService—The service is a subclass of the owl-s:Service class and described by the owl-s:Profile. The BaseService concept aggregates the essential information about the service, like the service endpoint, the current status, and quality metrics. A service is always related to a PhysicalEntity inside the ontology (not depicted in Figure 2.4). Thus, the services of a component or a whole asset can be retrieved from the Shared Knowledge base for the purpose of service discovery.

ServiceCategory—In order to classify a BaseService and define it as a certain functional service, the abstract class ServiceCategory is used. The categories correspond to the functional service types defined in the GDTA and also used for the grouping of the domain ontologies in Figure 2.3.

Quality Metrics—OWL-S has no capabilities for describing QoS for a certain service. To counteract, eleven additional properties are added to describe certain QoS metrics for a service, like Accessibility, ResponseTime, Availability, etc. This information can be used during service discovery to choose the best service if more than one is available.

Status—To gain information about the current service status, the last time of invocation, the invoked method, that invoked the service, and a service status message (“OK”, “In Use”,“Warning”,“Error”) is related with the service entity.

ServiceEndpoint—Each service has at least one service endpoint which enables service invocation or subscription in order to exchange data with a service. The endpoint categories can be flexible in nature and comprise of different protocols. In the prototypical implementation, a REST endpoint is specified in the ontology. However, endpoints are not restricted to that type, e.g., SOAP could also be used.

Simulation Service—Domain Ontology

A View on the Virtual Entity of the DT consists of a Simulation Service in combination with context information, stored in the Shared Knowledge base. The information about the simulation models used by the Simulation Services is captured by the Simulation Service—Domain Ontology. This ontology can be seen as a reference implementation for other domain ontologies. The main concepts of this ontology and their relations are shown in Figure 2.5.

The domain service ontology inherits all classes and properties from the Base Service Ontology and adds domain-specific knowledge. For the implementation of the Simulation Service—Domain Ontology, ML-Schema is used to describe data-driven and physical simulation models within a Simulation Service. ML-Schema is developed to represent and

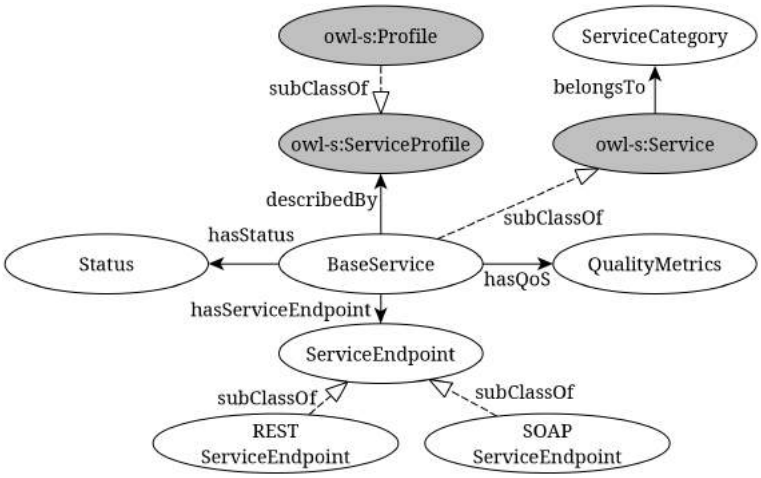


Figure 2.4: Part of the Base Service Ontology concepts and their relations.

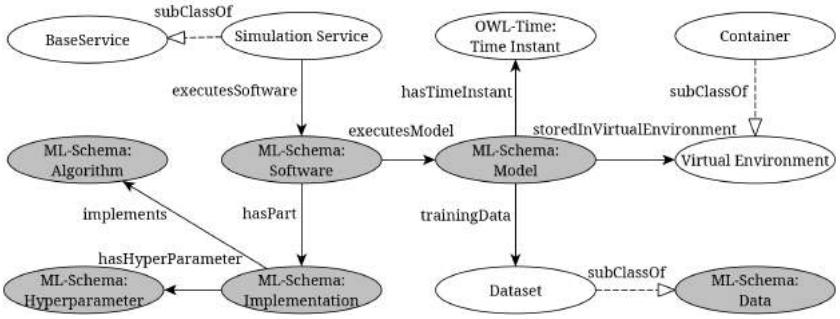


Figure 2.5: Main Simulation Service—Domain Ontology concepts and their relations.

interchange information on machine learning algorithms, datasets, and experiments [36]. The concepts in the ontology which are reused from ML-Schema are depicted with grey background in Figure 2.5. More Information about ML-Schema can be found in [10]. The additional concepts of the domain ontology are depicted with a white background.

The SimulationService is a special type of service for describing simulation models inside the DT. It inherits all properties from the BaseService concept in the top-level ontology. The Simulation Service executes a certain ML-Schema:Software which consists of an ML-Schema:Implementation and ML-Schema:Model. The model is trained on a certain Dataset, which has properties to describe how the data can be used (dataset location, dataset format, feature names, etc.).

The Virtual Environment provides a description of the environment in which a model is executed. A Container is one possibility of such a concrete realization of a Virtual Environment. The presented implementation uses a Docker container for providing such an environment. Each ML-Schema:Model is related with a OWL-Time:TimeInstant to capture the date and time of its training or parameter identification. This information is

used to handle various versions of a model.

2.4.2 Simulation Service API

The Simulation Service is realized as a microservice that communicates over REST-ful service endpoints with other services or agents. The stored information inside the Shared Knowledge base is made available by calling HTTP methods on the endpoints. In the same way, information can be inserted into the knowledge base, and the Simulation Service can be invoked. Currently, information about the simulation model and the service status can be retrieved, models and data can be uploaded, and the model can be trained. Also, predictions can be made by the Simulation Service, calling an HTTP method on a special endpoint. A full list of the implemented service endpoints, as well as the required parameters, can be found in Table 2.2. The Simulation Service adds information about a new model instance to the Shared Knowledge base automatically, whenever a new model is created or successfully trained, or a new hyper-parameter for the model is set. In the current implementation, only Matlab simulation models are supported by the Simulation Service.

2.4.3 Use Case: Packed-Bed Thermal Energy Storage

The PBTES is a reliable high-temperature thermal energy storage device with low investment costs. It is capable of operating at temperatures of above 800 °C [4] and thus applicable in variable industrial energy systems, as, for example, in the steel, glass, and cement industry or in solar power plants. Thermal energy storage solutions are required to match heat supply with demand and, thus, can contribute significantly to meeting society's desire for more efficient, environmentally friendly energy use [9]. Increasingly complex energy systems, induced by the transition to renewable energy sources [27], feature high flexibility that requires adequate control and optimization concepts. For fast analysis of various operating conditions and different parameters by simulation, detailed, but efficient models of such systems are needed. A PBTES is therefore considered as an ideal use case for a DT implementation.

Packed-Bed Thermal Energy Storage Test Rig

The PBTES represents the Physical Entity for the instantiated DT in this use case. It is located at the laboratory of the Institute for Energy Systems and Thermodynamics (IET) at TU Wien. A schematic illustration of the PBTES for loading and unloading is shown in Figure 2.6. It consists of an insulated vessel filled with gravel, an electric heater, and a ventilation unit. For charging the PBTES, hot air is ventilated through the gravel, which increases its temperature and stores sensible thermal energy. For discharging, cold air is ventilated through the hot tank and the heated air leaves the storage. To assess the thermodynamic conditions in the test rig's storage vessel, it is equipped with a total of 18 calibrated thermocouples, as well as mass flow and pressure measurement sensors

Table 2.2: Hypertext Transfer Protocol (HTTP) endpoint description for the implemented Simulation Service Application Programming Interface (API).

Endpoint	Description	HTTP	URL	Parameters
service	Returns information about the simulation service as JSON	'GET'	/servicestate	none
model	Returns information about the current model within the simulation service as JSON	'GET'	/model/	none
train	Trains the current model and returns information of the current model together with a summary of the model performance.	'PUT'	/train	data_path = path to the training data; model_params = dictionary with model parameters
predict	Returns prediction of the input data from the selected model instance as JSON	'GET'	/predict	model = modelLocation returned by calling model-endpoint or after a training invocation; data_path = path to the input data
uploaddata	Uploads new data to the simulation service which is used for training or prediction	'POST'	/upload/data/	file = data stored in a file in arbitrary format
uploadmodel	Uploads a new model for the Matlab based Simulation Service instance.	'POST'	/upload/model/	file = code stored in a Matlab file

at the inlet and outlet. For a detailed description of the test rig and it's measurement instrumentation, please refer to [17, 31].

Various models were developed to simulate the thermodynamic behavior of the PBTES, i.e., the measured values for intrinsic and outlet temperatures given the input values for temperature and mass flow. A physical model based on a 1D finite-difference approach was presented in [30]. A grey box model using recurrent neural networks was published in [15]. Furthermore, physical and data-driven modeling approaches for PBTES were compared and evaluated regarding prediction accuracy and modeling, as well as the computational effort [17]. In general, each of these evaluated simulation models can be reused in the Simulation Service of the DT. For the presented use case the mentioned grey box model was chosen, which was developed in [15].

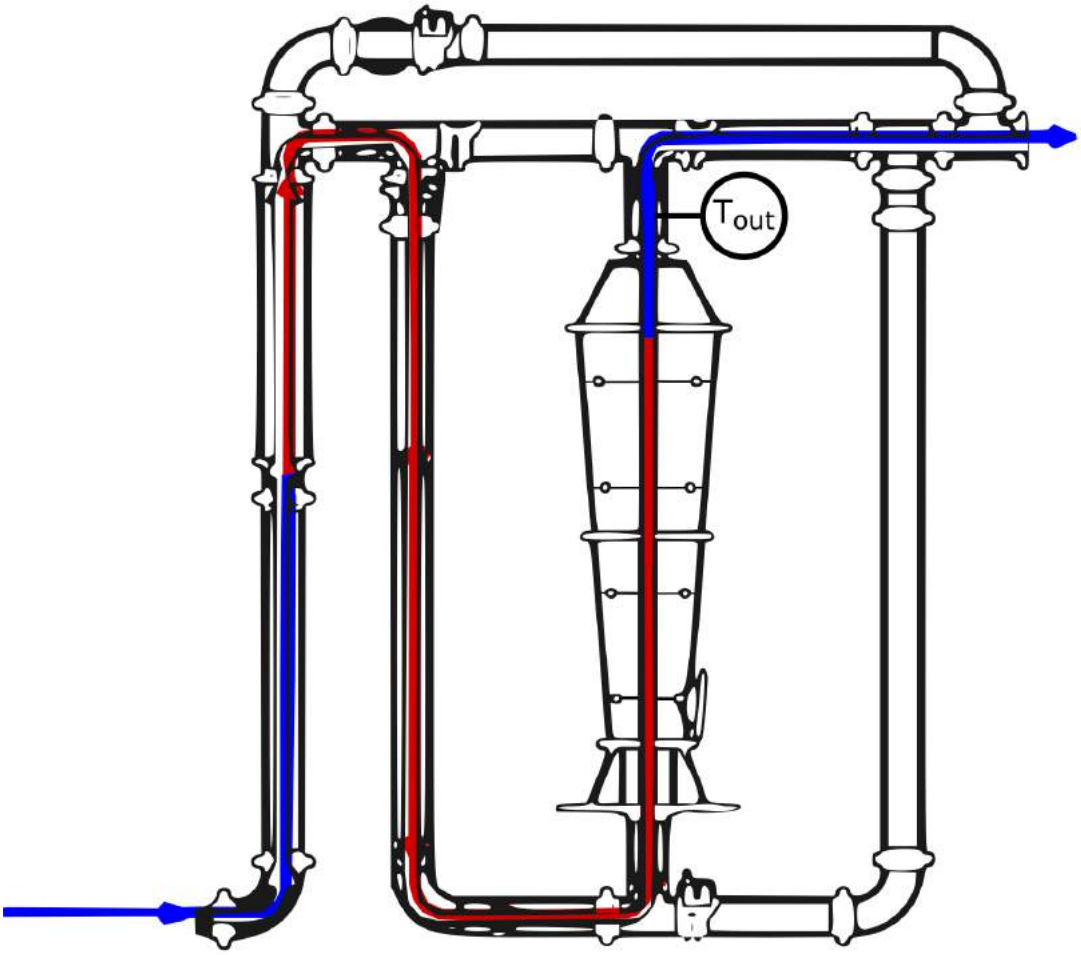


Figure 2.6: Bulk container of the Packed Bed Thermal Energy Storage (PBTES) installed at the laboratory in load and unload state.

Simulation Service Invocation

To provide some insights into the capability of our prototypical DT implementation, we show the procedure of loading a dynamic thermal model of the PBTES into the DT, train the model with available data, and use the model for a time-series prediction of the outlet temperature of the PBTES. The previously explained Simulation Service API is used to perform these tasks. The Simulation Service interacts with the Shared Knowledge base in the background to store and retrieve context information. This information is organized using the explained Base Service Ontology and the Simulation Service—Domain Ontology.

Uploading a Simulation Model—The implemented Simulation Service allows for uploading new simulation models. As mentioned before, a "Neural Net" grey box model, which is

implemented in Matlab is used. More details about the simulation model can be found in [15]. Figure 2.7 shows the sequence diagram of the upload process. For uploading a new model, the HTTP POST method is invoked on the endpoint with the URL /upload/model. As a parameter, the Matlab file is included in the HTTP body. The endpoint returns the internal path where the simulation model is stored. This path is needed afterward to access the specific model for training or prediction.

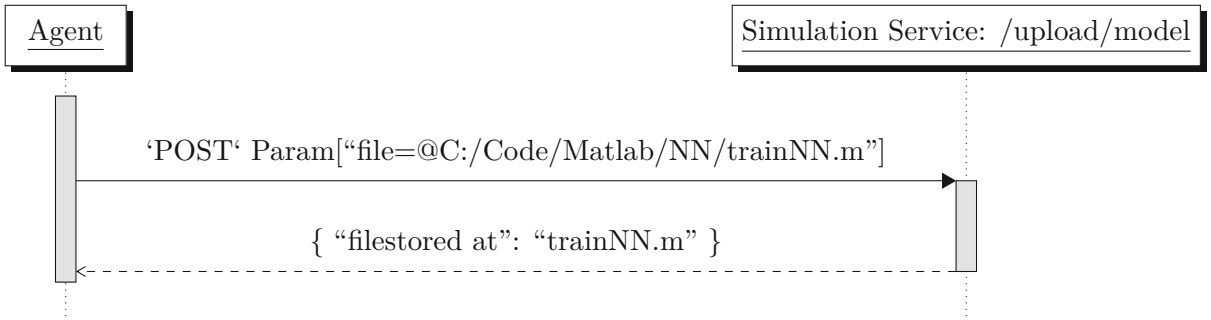


Figure 2.7: Sequence diagram for uploading a dataset at the service endpoint.

Uploading a Dataset—The HTTP POST method is used to upload a new dataset by invoking the simulation service's upload/data endpoint. The internal file path is returned in the HTTP response. The returned path is used in order to train a model or to make predictions with the uploaded dataset. The procedure is shown in the sequence diagram in Figure 2.8.

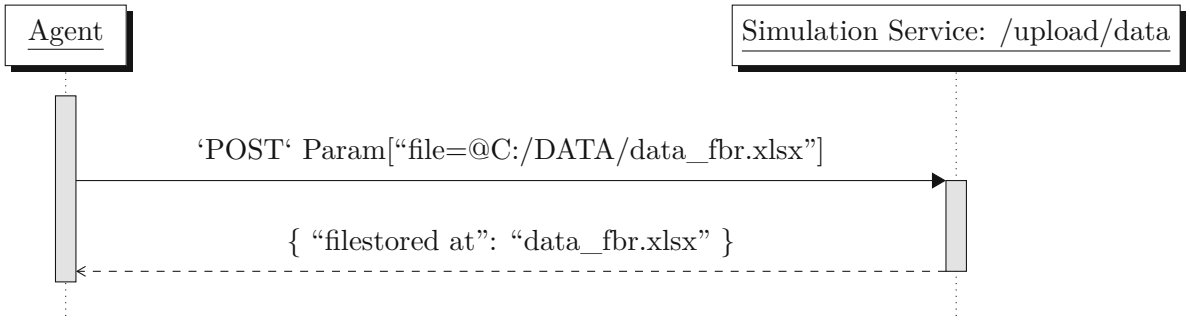


Figure 2.8: Sequence diagram for uploading a model at the service endpoint.

Train a Simulation Model—In order to train an uploaded simulation model with an uploaded dataset, the HTTP PUT method is invoked at the Simulation Service endpoint with the URL /train. The HTTP request holds the path to the uploaded training data and the model path as arguments. Additionally, simulation model parameters can also be forwarded as a dictionary. As a result of the training process, a new model description is returned in the JSON format and the information is also stored in the Shared Knowledge base. The old model is archived but can be used for predictions afterwards as well. The sequence is depicted in Figure 2.9, where just a small fraction of the returned information, formatted in JSON, is shown.

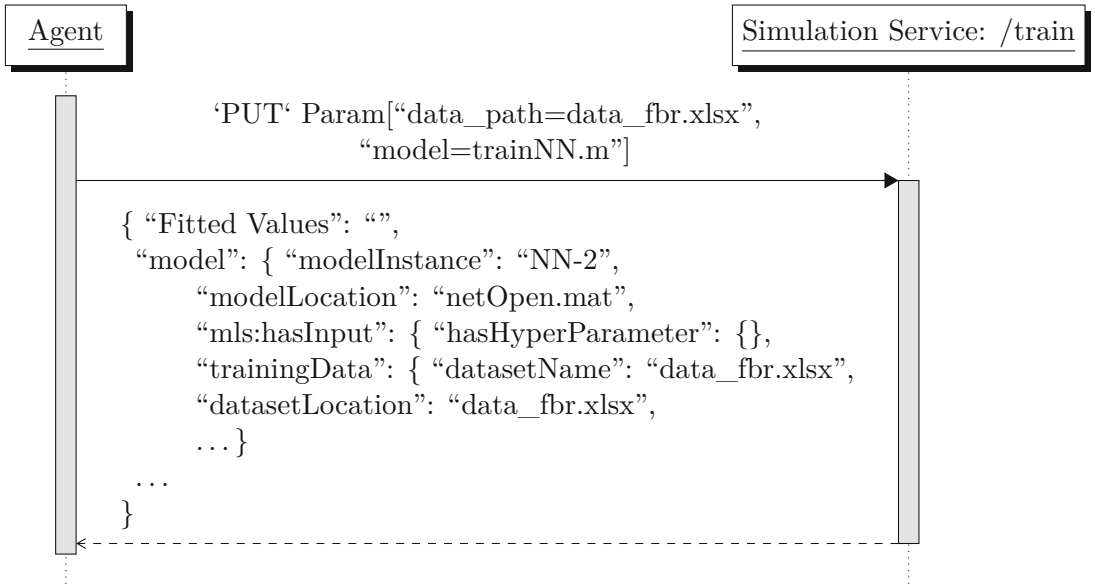


Figure 2.9: Sequence diagram for training a simulation model at the service endpoint.

Make a Prediction—In order to use the Simulation Service for predictions, the endpoint with the URL /predict has to be called using the HTTP GET method. Its arguments are the path to an input dataset and optionally a path to a model. If no path to a model is specified the service uses per default the most recently trained model to make predictions. The service endpoint returns the predictions in a JSON array as shown in Figure 2.10.

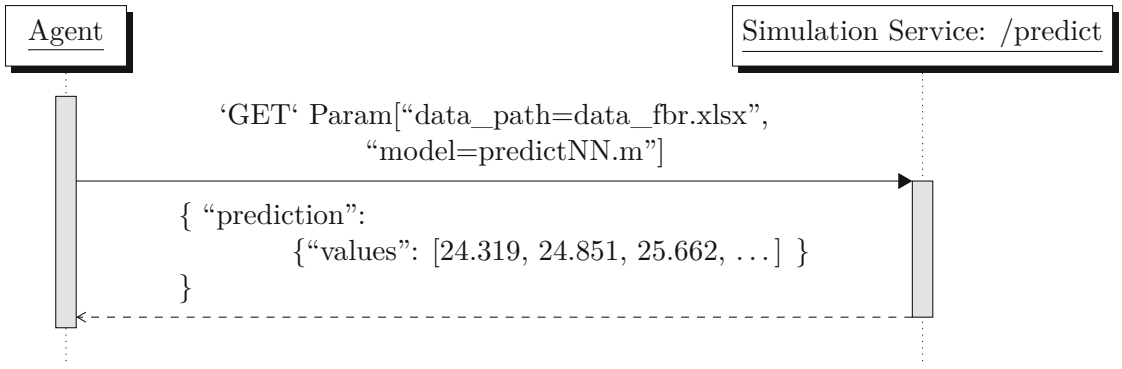


Figure 2.10: Sequence diagram for using a model to predict a time series at simulation endpoint.

In the presented use case of the PBTES, the output temperature is predicted by the Simulation Service for a specified input trajectory. The predicted loading cycle of the PBTES over time is shown in Figure 2.11. These prediction results could further be used for an optimized control strategy performed by the DT.

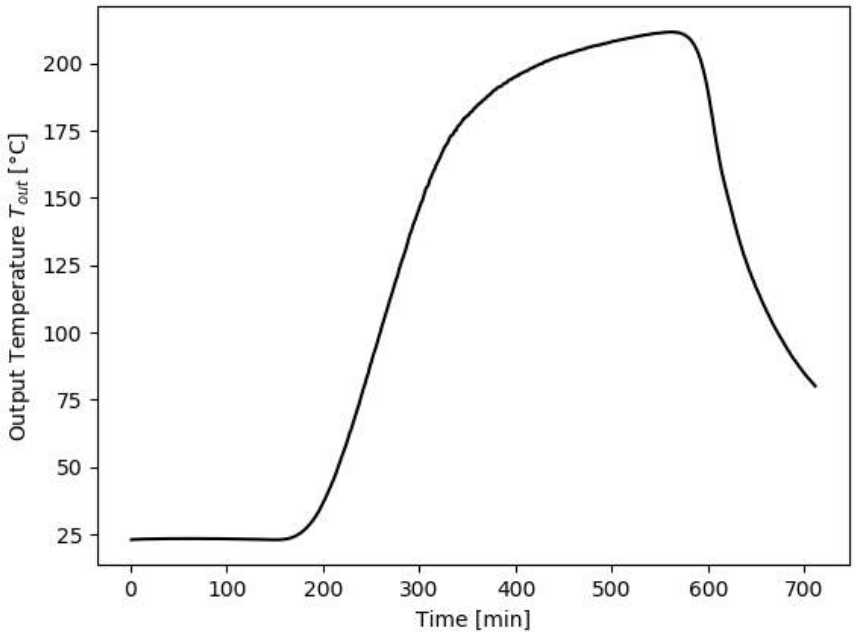


Figure 2.11: Results of the Simulation Service for the output temperature T_{out} of the Packed-Bed Thermal Energy Storage (PBTES) for a specific load cycle.

2.5 Discussion

The aim of our proposed GDTA is a simple architectural model that captures the essential components of a DT in a technology-independent way. We presented various concepts and architectures for DTs in Section 2.1.2. It becomes clear that DTs have to be able to handle various types of models for different applications. Also, a structuring of different functionality by using services as well as information and knowledge management are essential to gain a certain level of cognition.

The user interaction with a DT is a very important aspect, as in Industry 4.0 the workers or operators will be the most flexible part of the CPS. They have the role of a strategical decisions-maker and problem solver [13] so they have to interact closely with a DT. This is not reflected by most frameworks or concepts.

Almost all presented frameworks or architectures use a layered structure to specify some functionality. Various layers are introduced with different names, which often perform similar tasks. This leads to confusion. The alignment of our architecture with RAMI 4.0 facilitates a standardized wording and common understanding of the layers in a DT. This helps to classify, combine, and re-use already existing frameworks and technologies in the area of Industry 4.0 and the IIoT for designing and implementing a DT.

In the following paragraph, the concepts, architectures, and frameworks for DTs, which have been introduced in Section 2.1.2 are re-visited and set into relation with our

proposed GDTA.

The foundation of our GDTA is the 5D-DT concept [38]. The data and service dimension of this concept is refined in our architecture by identifying service clusters and introducing a Shared Knowledge base, which semantically enriches available data and stores information about resources and services. This Shared Knowledge base is the main component inside the Smart Data Service in our proposed architecture and enhances the data dimension of the 5D-DT.

The suggested architecture for an "Intelligent Digital Twin" in [5] emphasizes the usage of AI inside a DT to evaluate "what-if" scenarios. Therefore, certain components in the architecture are defined, like an "Intelligent Algorithm" module, or a "Co-simulation Interface". In our opinion, these are quite specific modules, which can be implemented by distinct services of the proposed GDTA (e.g., a simulation service can support co-simulation), but do not always have to be present in a DT. Also, the separation of "intelligent" algorithms from the services into dedicated components seems not always beneficial. Nevertheless, important aspects and problems are addressed in this work, like user feedback, which is also considered in our architecture.

A reference framework for DTs within CPSs is presented in [22] where four main building blocks are identified. These blocks can be mapped to certain layers of our proposed architecture, which is aligned with RAMI 4.0: The key concepts of the "Service Platform" are found in the Functional Layer in our architecture. The "Physical Entity Platform" is equivalent to the Asset and Integration Layer. In addition, our architecture captures the aspect of the communication layer. The main aspects of the "Data Management Platform" can be located on the Information Layer. As we propose a service-oriented architecture for the GDTA, some of the mentioned data management methods, like data processing, data cleaning, data analysis, data mining and information extraction can be carried out by specific services of the Functional Layer. The "Virtual Entity Platform" includes various "Semantic Models". This corresponds to our idea of a View on the Virtual Entity in the GDTA, consisting of various semantic simulation models in combination with context information, stored in the Shared Knowledge base. The proposed reference framework and the identified structural properties of the DT are very generic, but some additional aspects can be found in our GDTA. So we introduced a Service Management component and defined groups of common service types inside a DT, which help to structure and implement certain services. We also see the HMI interaction as an important part of our GDTA which almost every service has to implement. Thus, we include a HMI component in our GDTA.

The conceptual architecture of the Cognitive Twin Toolbox presented in [1] also introduces a "Knowledge Repository" in combination with "Cognitive Services". This is only applied for a "cognitive Twin". The basic DT concept implements only a so-called "Metadata Repository". A separation into two distinguishable components, as presented in [1] seems neither beneficial nor has practical advantages. Therefore, we introduced the Shared Knowledge base, which acts as a central point of information in our GDTA. Also a "Model Management Layer" is not explicitly stated in our architecture as resource management

has to be done by the service implementations themselves. The information about the resources has to be made available through the Shared Knowledge base. The functions described for the "Twin Management Layer" can be located in a Reconfiguration Service in the GDTA. The COGNITIVE Twin toolbox is a solution with various components that seems to be specific for their definition of the so-called "Hybrid" or "Cognitive Twin". Thus, we tried to reduce the components of the proposed GDTA to be as simple and generic as possible.

The conceptual model of a DT in the context of IoT, as presented in [3] introduces five layers which are quite similar to the layers of RAMI 4.0. The "Physical Space" corresponds to the Asset and Integration Layer; the "Virtual Layer", to the Information Layer; the "Data Analytics and Visualization" and "Application", to the Functional Layer. In our architecture, we chose a more service-oriented view of the various applications of the DT. Providing semantics of the data and a shared knowledge base is not explicitly handled in this conceptual model, in contrast to the suggested GDTA. The conceptual DT model emphasizes the importance of security by defining a vertical module overall specified layers. In our generic architecture, such a module is not explicitly depicted, as it applies to all levels of RAMI 4.0 implicitly and must be considered for the asset as a whole [34].

The AAS is a promising way of standardization as shown in [6] but still ongoing work. In its current state, the concept of the AAS meta-model can be applied and implemented in line with our proposed architecture. It was already shown, that the defined data model of the AAS can be semantically lifted to a knowledge representation based on RDF [7]. In the context of our proposed architecture, this enables the representation of the AAS inside the Shared Knowledge base.

The presented proof-of-concept implementation is the basis for further development of other services inside the DT. Additionally, the knowledge representation of the simulation models can be extended with application-specific domain ontologies. This would facilitate the integration of other semantic models of the DT as described in [22]. The hierarchical and modular ontology design, using a top-level ontology as well as various domain ontologies, facilitates the integration of new concepts and ontologies. These hierarchies can be further increased by introducing additional levels, like application or even task ontologies. The alignment with already existing or emerging standards, represented in OWL, can be performed by having similar concepts in the upper layers of the ontology hierarchy and using the owl:equivalentClass property defined in OWL 2.

The presented use case was chosen in the domain of industrial energy system, but the proposed GDTA can also be applied for a broader spectrum of applications. For this reason, we will investigate other domains and apply the proposed GDTA also for other use cases.

2.6 Conclusions

A novel GDTA based on the 5D-DT concept is presented and evaluated based on a prototypical proof-of-concept implementation. Other concepts, architectures, or frameworks in literature often use a layered structure with similar functionality but different names. To overcome this problem, we aligned our GDTA with the IT dimension of RAMI 4.0. This helps to have a common naming and understanding of the layers inside the GDTA, which facilitates the development of a DT.

The presented GDTA is technology-independent. We instantiated it based on Semantic Web technology and showed the suitability for handling context information about resources and services in combination with simulation models. This enables application dependable Views on the Virtual Entity of the DT. In our proof-of-concept, ontologies build the foundation for the Shared Knowledge base of the Smart Data Service. Existing ontologies like OWL-S or ML-Schema are reused to describe resources and services and facilitate interoperability.

Future work will further extend the service infrastructure inside the DT, including service management. Other functional services of a DT will be implemented and the domain-specific ontologies will be extended. For this goal, an advanced version management system has to be developed to keep track of changes made on the DT during its life-cycle.

Another research direction will investigate the applicability of already available IoT standards and frameworks at the Communication and Information Layer of our GDTA. It will investigate how those can be organized in the hierarchical dimension of the RAMI 4.0 using edge and cloud computing.

References

[1] Sailesh Abburu et al. „COGNITWIN – Hybrid and Cognitive Digital Twins for the Process Industry“. In: *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. July. IEEE, June 2020, pp. 1–8. ISBN: 978-1-7281-7037-4. DOI: 10.1109/ICE/ITMC49519.2020.9198403. URL: https://ieeexplore.ieee.org/document/9198403/.

[2] Peter Adolphs et al. *Reference Architecture Model Industrie 4.0 (RAMI4.0)*. Tech. rep. July. VDI/VDE-Gesellschaft, 2015.

[3] A. R. Al-Ali et al. „Digital Twin Conceptual Model within the Context of Internet of Things“. In: *Future Internet* 12.10 (2020), p. 163. DOI: 10.3390/fi12100163.

[4] A. Andreozzi et al. „Numerical investigation of sensible thermal energy storage in high temperature solar systems“. In: vol. 48. cited By 3. 2009, pp. 461–472. DOI: 10.2495/CMEM090421.

[5] Behrang Ashtari Talkhestani et al. „An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System“. In: *At-Automatisierungstechnik* 67.9 (2019), pp. 762–782. ISSN: 2196677X. DOI: 10.1515/auto-2019-0039.

[6] Sebastian Bader et al. „Details of the Asset Administration Shell“. In: *Plattform Industrie 4.0* 1 (2019), p. 473. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html.

[7] Sebastian R. Bader and Maria Maleshkova. „The Semantic Asset Administration Shell“. In: vol. 1. Springer International Publishing, 2019, pp. 159–174. DOI: 10.1007/978-3-030-33220-4_12. URL: http://link.springer.com/10.1007/978-3-030-33220-4%7B%5C_%7D12.

[8] Stefan Boschert and Roland Rosen. „Digital Twin—The Simulation Aspect“. In: *Mechatronic Futures*. Ed. by Peter Hehenberger and David Bradley. Cham: Springer International Publishing, 2016, pp. 59–74. ISBN: 978-3-319-32154-7. DOI: 10.1007/978-3-319-32156-1_5. URL: http://link.springer.com/10.1007/978-3-319-32156-1%20http://link.springer.com/10.1007/978-3-319-32156-1%7B%5C_%7D5.

[9] Ibrahim Dincer and Marc A. Rosen. „Chapter 6 - Heat Storage Systems“. In: *Exergy Analysis of Heating, Refrigerating and Air Conditioning*. Ed. by Ibrahim Dincer and Marc A. Rosen. Boston: Elsevier, 2015, pp. 221–278. DOI: 10.1016/B978-0-12-417203-6.00006-5.

[10] Diego Esteves et al. *ML Schema Core Specification*. Tech. rep. Cambridge: World Wide Web Consortium (W3C) - Machine Learning Schema Community Group, 2016.

[11] M Fernández-López, A Gómez-Pérez, and Natalia Juristo. „METHONTOLOGY: From Ontological Art Towards Ontological Engineering“. In: *Spring Symposium on Ontological Engineering of AAAI* SS-97-06 (1997), pp. 33–40. ISSN: 0769530303. DOI: 10.1109/AXMEDIS.2007.19. URL: http://oa.upm.es/5484/.

[12] B Glimm et al. „HermiT: reasoning with large ontologies“. In: *Computing Laboratory, Oxford University* (2009), p. 64.

[13] Dominic Gorecky et al. „Human-machine-interaction in the industry 4.0 era“. In: *Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014* (2014), pp. 289–294. DOI: 10.1109/INDIN.2014.6945523.

[14] Michael Grieves. „Digital Twin : Manufacturing Excellence through Virtual Factory Replication This paper introduces the concept of a A Whitepaper by Dr . Michael Grieves“. In: *White Paper* March (2014). URL: https://www.researchgate.net/publication/275211047%7B%5C_%7DDigital%7B%5C_%7DTwin%7B%5C_%7DManufacturing%7B%5C_%7DExcellence%7B%5C_%7Dthrough%7B%5C_%7DVirtual%7B%5C_%7DFactory%7B%5C_%7DReplication.

[15] V Halmschlager et al. „Grey Box Modeling of a Packed-Bed Regenerator Using Recurrent Neural Networks“. In: *IFAC-PapersOnLine* 52.16 (2019), pp. 765–770.

[16] Jerry R. Hobbs and Chris Little. *Time Ontology in OWL. W3C Candidate Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2020. URL: https://www.w3.org/TR/owl-time/.

[17] René Hofmann et al. „Comparison of a physical and a data-driven model of a Packed Bed Regenerator for industrial applications“. In: *Journal of Energy Storage* 23 (2019), pp. 558–578. ISSN: 2352-152X. DOI: 10.1016/j.est.2019.04.015.

[18] IEC. *IEC 62264 Enterprise-control system integration*. Tech. rep. Geneva: International Electrotechnical Commission, 2016.

[19] Michael Jacoby and Thomas Usländer. „Digital Twin and Internet of Things—Current Standards Landscape“. In: *Applied Sciences* 10.18 (2020), p. 6519. DOI: 10.3390/app10186519.

[20] François Jammes and Harm Smit. „Service-oriented paradigms in industrial automation“. In: *IEEE Transactions on Industrial Informatics* 1.1 (2005), pp. 62–70. ISSN: 15513203. DOI: 10.1109/TII.2005.844419.

[21] David Jones et al. „Characterising the Digital Twin: A systematic literature review“. In: *CIRP Journal of Manufacturing Science and Technology* 29 (2020), pp. 36–52. ISSN: 17555817. DOI: 10.1016/j.cirpj.2020.02.002. URL: https://doi.org/10.1016/j.cirpj.2020.02.002.

[22] Klementina Josifovska, Enes Yigitbas, and Gregor Engels. „Reference Framework for Digital Twins within Cyber-Physical Systems“. In: *Proceedings - 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems, SEsCPS 2019* (2019), pp. 25–31. DOI: 10.1109/SEsCPS.2019.00012.

[23] Holger Knublauch et al. „The Protégé OWL plugin: An open development environment for semantic web applications“. In: *International semantic web conference*. Springer. 2004, pp. 229–243.

[24] Gunther Koschnick. *Industrie 4.0: Smart services*. Tech. rep. July. Frankfurt am Main, Germany: German Electrical and Electronic Manufacturers' Association, 2016, pp. 13–14.

[25] Jay Lee, Behrad Bagheri, and Hung An Kao. „A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems“. In: *Manufacturing Letters* 3 (2015), pp. 18–23. ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001. arXiv: 1503.07717. URL: http://dx.doi.org/10.1016/j.mfglet.2014.12.001.

[26] Yuqian Lu et al. „Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues“. In: *Robotics and Computer-Integrated Manufacturing* 61.July 2019 (2020), p. 101837. ISSN: 07365845. DOI: 10.1016/j.rcim.2019.101837. URL: https://doi.org/10.1016/j.rcim.2019.101837.

[27] H. Lund et al. „Energy storage and smart energy systems“. In: *International Journal of Sustainable Energy Planning and Management* 11 (2016). cited By 139, pp. 3–14. DOI: 10.5278/ijsepm.2016.11.2.

[28] Somayeh Malakuti et al. „Digital Twins for Industrial Applications. Definition, Business Values, Design Aspects, Standards and Use Cases“. In: *White Paper* (2020), pp. 1–19.

[29] David Martin et al. „OWL-S: Semantic markup for web services“. In: *W3C member submission* 22.4 (2004). URL: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[30] F Mayrhuber, H Walter, and M Hameter. „Experimental and numerical investigation on a fixed bed regenerator“. In: *Proceedings of the 10th International Conference on Sustainable Energy and Environmental Protection", University of Maribor Press.* 2017, pp. 2017–30.

[31] A Michalka. „Experimentelle Untersuchungen eines Festbettregenerators mit feinem Kies als Speichermaterial“. PhD thesis. Technicacl University of Vienna, 2018.

[32] László Monostori. „Cyber-physical production systems: Roots, expectations and R&D challenges“. In: *Procedia CIRP* 17 (2014), pp. 9–13. ISSN: 22128271. DOI: 10.1016/j.procir.2014.03.115. URL: http://dx.doi.org/10.1016/j.procir.2014.03.115.

[33] Vinit Parida, David Sjödin, and Wiebke Reim. „Reviewing literature on digitalization, business model innovation, and sustainable industry: Past achievements and future promises“. In: *Sustainability (Switzerland)* 11.2 (2019). ISSN: 20711050. DOI: 10.3390/su11020391.

[34] PLATFORM INDUSTRIE 4.0. *Security in RAMI4.0.* Tech. rep. 2016, pp. 1–3. URL: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/security-rami40-en.pdf?%7B%5C_%7D%7B%5C_%7Dblob=publicationFile%7B%5C&%7Dv=5.

[35] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. „OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation“. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), pp. 7–34.

[36] Gustavo Correa Publio et al. „ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies“. In: *CoRR* abs/1807.05351 (2018). arXiv: 1807.05351. URL: http://arxiv.org/abs/1807.05351.

[37] Vasja Roblek, Maja Meško, and Alojz Krapež. „A Complex View of Industry 4.0“. In: *SAGE Open* 6.2 (Apr. 2016). ISSN: 2158-2440. DOI: 10.1177/2158244016653987. URL: http://journals.sagepub.com/doi/10.1177/2158244016653987.

[38] Fei Tao, Meng Zhang, and A.Y.C. Nee. „Five-Dimension Digital Twin Modeling and Its Key Technologies“. In: *Digital Twin Driven Smart Manufacturing* (2019), pp. 63–81. DOI: 10.1016/b978-0-12-817630-6.00003-5.

[39] Fei Tao et al. „Digital Twin in Industry: State-of-the-Art“. In: *IEEE Transactions on Industrial Informatics* 15.4 (2019), pp. 2405–2415. ISSN: 15513203. DOI: 10.1109/TII.2018.2873186.

[40] Thomas H.J. Uhlemann, Christian Lehmann, and Rolf Steinhilper. „The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0“. In: *Procedia CIRP* 61 (2017), pp. 335–340. ISSN: 22128271. DOI: 10.1016/j.procir.2016.11.152. URL: http://dx.doi.org/10.1016/j.procir.2016.11.152.

[41] UNIDO. *Accelerating clean energy through Industry 4.0: Manufacturing the next Revolution*. Tech. rep. Vienna, Austria: United Nations Industrial Development Organization, 2017, p. 56. URL: https://www.unido.org/sites/default/files/2017-08/REPORT%7B%5C_%7DAccelerating%7B%5C_%7Dclean%7B%5C_%7Denergy%7B%5C_%7Dthrough%7B%5C_%7DIndustry%7B%5C_%7D4.0.Final%7B%5C_%7D0.pdf.

CHAPTER 3

Transforming OPC UA Information Models into Domain-Specific Ontologies

Publication: *G. Steindl and W. Kastner, "Transforming OPC UA Information Models into Domain-Specific Ontologies," 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), 2021.*

Abstract: Semantics interoperability is important for cyber-physical systems to enable complex data processing and facilitating interworking. OPC Unified Architecture (OPC UA) provides an extensible information model but lacks formal semantics. Ontologies based on the Web Ontology Language (OWL) can provide such formal semantics. Thus, we present a transformation approach that converts OPC UA information models (or only parts of them) into domain-specific ontologies. The transformation process consists of two steps. The first step is a mapping from OPC UA to OWL Full. The second step performs a graph transformation, based on SPARQL rules, into the domain-specified target ontology. The adaption of this transformation to the source information model and the target ontology can be accomplished by only adapting these transformation rules. The presented approach is evaluated for a use case of an industrial heating process to show its flexibility.

3.1 Introduction

Data play a key role in the ongoing transition to Industry 4.0, trying to reach the goals of operational efficiency and productivity and a higher level of automation [12]. CPSs are applied in the industrial domain, which gather these data and process them. The

semantics of data and context information is important to further improve the abilities of the systems based on the underlying CPSs. The OPC UA standard provides, next to its communication specification, a standardized information model, providing additional structure and semantics to the data. Thus, information about equipment and the plant's topology can be made available in OPC UA and later be used to set data into context for further processing.

Because of its capabilities, OPC UA is a suitable technology for Industry 4.0 applications. A disadvantage of the OPC UA information model is its lack of formal semantics, the missing browsing capability [16] and a limited semantic expressiveness compared to more advanced knowledge representations [1]. However, these features are important to develop more sophisticated CPSs, like presented in [11], which are able to get an in-depth knowledge of the monitored system or make them self-configurable and self-adaptive.

A more formal, machine-readable knowledge representation is provided by the Semantic Web with its technologies, like RDF, OWL and SPARQL. Thus, transformations between the OPC UA information model and OWL have been proposed to use various features of OWL [17], [1],[18]. However, a simple OWL representation of the OPC UA information model is not always appropriate. For some applications, the information from the OPC UA model or only parts of it shall be used to instantiate a domain-specific ontology. Thus, this paper provides an answer to the question, how domain-specific ontologies can be instantiated based on already available OPC UA information models, considering changes of the source information model and the target ontology.

The remainder of the paper is structured as follows: Sec. 3.2 gives a short overview of related work in the area of OPC UA in combination with the Semantic Web. Sec. 3.3 explains the proposed process of transforming OPC UA information models into domain-specific ontologies. In Sec. 3.4, a proof of concept implementation is presented to evaluate the proposed transformation process for a use case of a thermal heating process. Finally, the proposed transformation process is discussed, and an outlook on our future work is given.

3.2 Related Work

Ontologies are used for knowledge representation and can be defined as a formal, explicit specification of a shared conceptualization [19]. Technologies and standards form the so-called Semantic Web Stack like RDF [8], RDFS [3], OWL [21], and SPARQL [6] support the development of ontologies. RDF defines triples to formulate statements, consisting of a subject, a predicate, and an object. RDFS introduces new concepts based on RDF to increase its semantic expressiveness. OWL further enhances this expressiveness by introducing concepts of formal logic. There exists different OWL levels, like OWL Full and OWL DL with its profiles EL, RL, QL. It is important to know that OWL Full has the least restrictions for modeling, which leads to undecidability when it comes to reasoning. More details can be found in [9].

OPC UA is an industrial communication standard that aims to solve interoperability problems at the transport and semantic layer in Industry 4.0 applications. Thus, next to the data communication, OPC UA facilitates the semantic description of data by defining an OPC UA information model. Everything can be described with this information model, from simple devices to very complex components in an object-oriented and semantically meaningful way. Therefore, OPC UA defines an address space model, which is the meta-model of the information model [14]. The address space model uses *Nodes* and *References* to form a graph structure. Based on this meta-model, the OPC UA information model defines the address space of the OPC UA server [15]. This address space is the information that is exposed by a specific OPC UA server. A detailed description of these concepts and their application can be found in [13].

A formal mapping between OPC UA and OWL DL is presented in [17]. It is shown, why a trivial mapping between OPC UA and OWL DL is not possible. One major issue is the usage of *Instance-Nodes* at the type definitions in OPC UA, which is not allowed in OWL DL. Thus, mappings have to be defined for such OPC UA design patterns, which can be specified in various ways, based on certain design choices. Another mapping between OPC UA and OWL DL has been presented in [1]. They identified axioms and properties which are available in OWL but not in OPC UA. Thus, they conclude that OWL is semantically more expressive than OPC UA and also specified a mapping between them.

The mapping problem can be partly avoided by using OWL Full instead of OWL DL, which allows one-to-one mappings for most of the OPC UA concepts, as shown in [18]. But such an approach is only applicable in cases where reasoning is not required.

However, all these approaches only allow a direct translation of the used OPC UA information model into an OWL representation, whereas our proposed approach also makes it possible to define mapping rules to extract the needed information from the OPC UA model and use it in a domain-specific target ontology. This makes our approach even more flexible.

3.3 Transformation Process

The proposed approach targets the situation where the information model is already deployed at a running OPC UA server, and the address space can be read by an OPC UA client.

For the model transformations, different levels of abstraction are of interest. A three-tier architecture with M2 as meta-model, M1 as model, and M0 as instance level can be used to describe these levels of abstraction, as shown in Fig. 3.1. The OPC UA address space model is located on Level M2 as the meta-model of the OPC UA information model [10]. For the ontology, the ontology definition is located on level M1 and the used ontology language, in this case OWL, on M2 [20].

The transformation of the OPC UA information model into a domain-specific ontology is done in two steps. The first step is transforming the OPC UA information model into

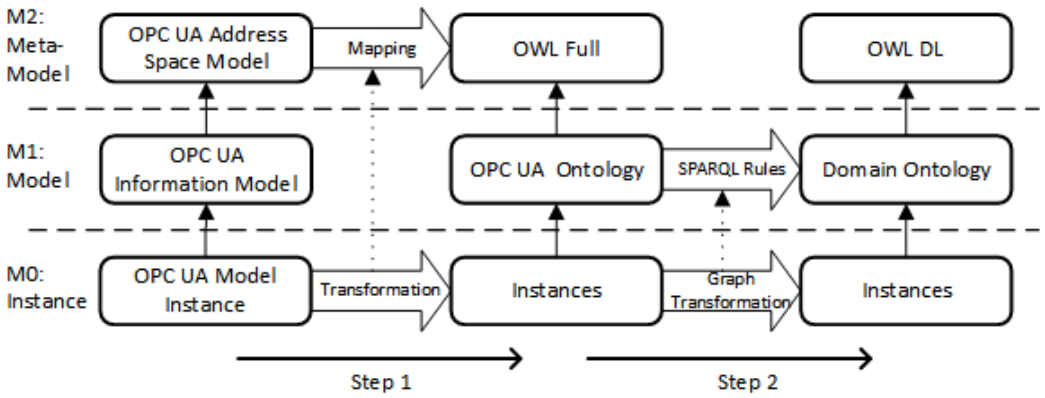


Figure 3.1: Model transformation on different levels of abstraction.

OWL Full. Using OWL Full makes reasoning impossible, but the mapping is straight forward, by a one-to-one mapping of most concepts. In the second step, the OWL Full graph is transformed with the help of SPARQL rules into a domain-specific ontology. This domain-specific ontology can be based on OWL DL, which re-enables reasoning again.

The second transformation step is performed at a lower level of abstraction, namely on M1. SPARQL rules are used to define how concepts are transformed from one ontology into the other. Therefore, SPARQL construct clauses are applied to search for patterns in the ontology graph and create new concepts and instances based on these patterns. To adapt this approach to other domain ontologies or the OPC UA information models, only these SPARQL rules have to be changed. This leads to a flexible transformation approach.

A more detailed description of the whole transformation process, as well as the involved components, is depicted in Fig. 3.2 and described in the following:

1. The OPC UA information model is designed, and an XML-Nodeset file is created.
2. The information model is instantiated in an OPC UA server, which exposes the information via its address space.
3. The whole OPC UA address space is read by a client from the server and a OWL Full representation is generated, based on defined mapping rules.
4. The OWL Full representation is stored in a so-called "Quad Store". A quad store is a specific database for RDF triples (triple store), which additionally provides the feature of named graphs.
5. The SPARQL transformation rules are defined. These rules specify the transformation between the OWL Full representation of the OPC UA information model and the domain-specific target ontology.

6. The specified SPARQL rules are executed at the SPARQL endpoint of the quad store. These rules create a new ontology stored in a new named graph to separate it from the source ontology.

7. The created ontology can now be accessed. New information can be inferred by reasoners and retrieved with SPARQL queries.

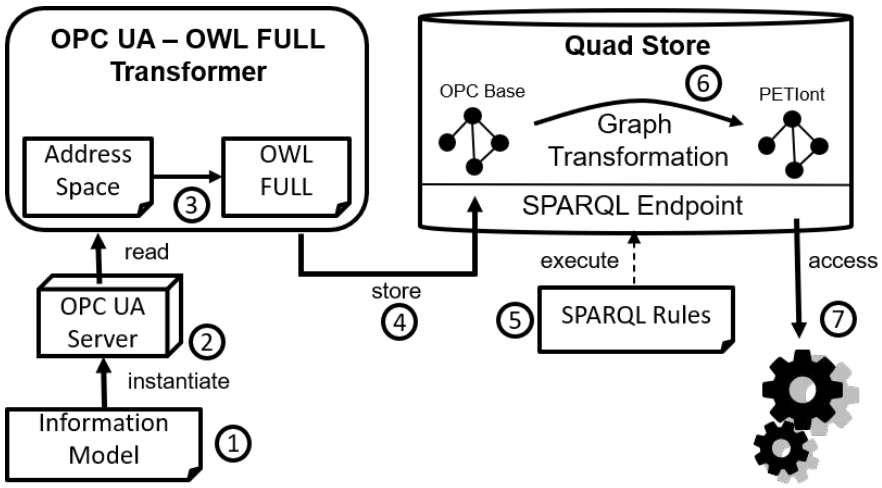


Figure 3.2: Model transformation process steps and involved components.

3.4 Use Case - Heating Process

To evaluate our proposed transformation approach, we selected a simple heating process as use case. The pipe and instrumentation (P&I) diagram of the heating process is shown in Fig. 3.3. The main components are an electric heater "H1", two fans "F1" and "F2", a heat exchanger "HE1", and an open vessel named "SiPro". Various temperature sensors (TI) and two flow sensors (FI) are installed at the inlet and outlet pipes. The fans can only be switched to discrete speeds, while the heater's power can be controlled continuously. Fresh air is heated by the electric heater "H1" and ventilated through the inlet pipe to the vessel. The heater is controlled by the temperature of the chamber, which is measured at the chamber outlet. The exhaust air of the process is used by the heat exchanger "HE1" for heat recovery.

3.4.1 OPC UA Information Model

For the described use case, shown in Fig. 3.3, an OPC UA information model is designed and initiated in an OPC UA server. The information model captures the structure of the heating process, including its components, their topology, sensors, and actuators. The information model builds on existing industrial standards. To structure the plant, elements defined by ISO 10209 [4] are used. The plant equipment is named and structured

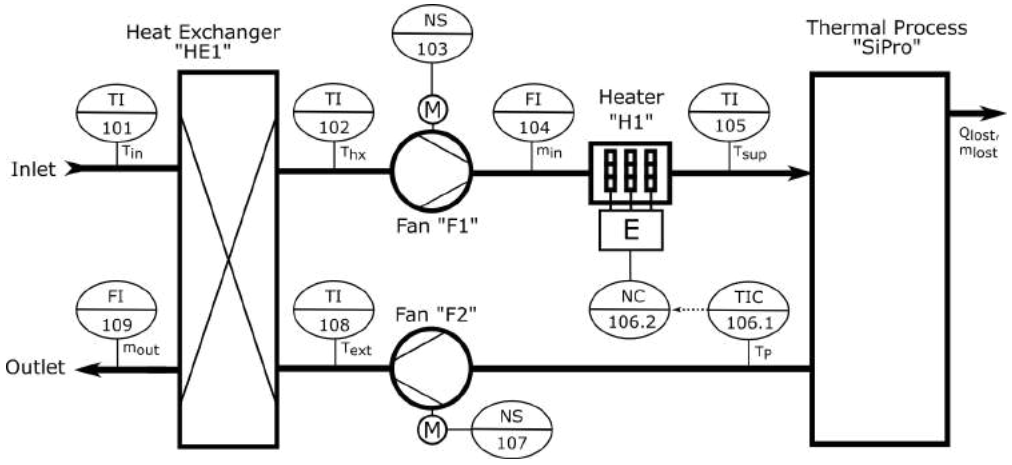


Figure 3.3: P&I diagram of the heating process.

based on EN ISO 10628 [5], which defines equipment for P&I diagrams, such as heat exchangers, blower fans, vessels, etc. A small excerpt from the resulting type hierarchy in the OPC UA information model is depicted in Fig. 3.4, limited to the exchanger type. The complete information model is available on GitHub[1].

As shown in Fig. 3.4, it can be specified whether the pressure or temperature change for a certain equipment should be neglected or not by two variables, called *NeglectPressureChange* and *NeglectTemperatureChange*. This information can be used later on for sensor data evaluation, which is out of scope for this paper. Nevertheless, this information is also used in the transformation step. Next to these variables, the *HeatExchangerType* defines additional objects for flow ports and methods for controlling the equipment.

The standard IEC 62424 [7] defines "process control engineering (PCE) requests", representing sensors, actuators, and control functions. These elements are used in the OPC UA information model, as depicted in Fig. 3.5. The designation of a PCE request consists of its category and its function. Also, a unique reference number is assigned. A standardized character string specifies the category and function. For example, the PCE request with the reference number 106.1 in Fig. 3.3 has the designation "TIC", where category "T" means temperature, the letter "I" stands for indication and "C" for a control function. More Information about the application of IEC 62424 can be found in [2].

Non-hierarchical references are also specified and used to describe the topology of the pant. They are shown in Fig. 3.6. Equipment can be connected via a mass transport (*hasMassFlowTo*) or via an information flow (*hasSignalTo*). The *hasPCE_Request* and *hasProcessConnection* references are used to connect sensors and actuators with their corresponding plant equipment.

For the use case of the thermal heating process, depicted in Fig. 3.3, an information model is loaded into an OPC UA server. This model instance is partly shown in Fig. 3.7. The

[1] https://github.com/Smart-Industrial-Concept/HeatingProcess_OPC_UA_Model

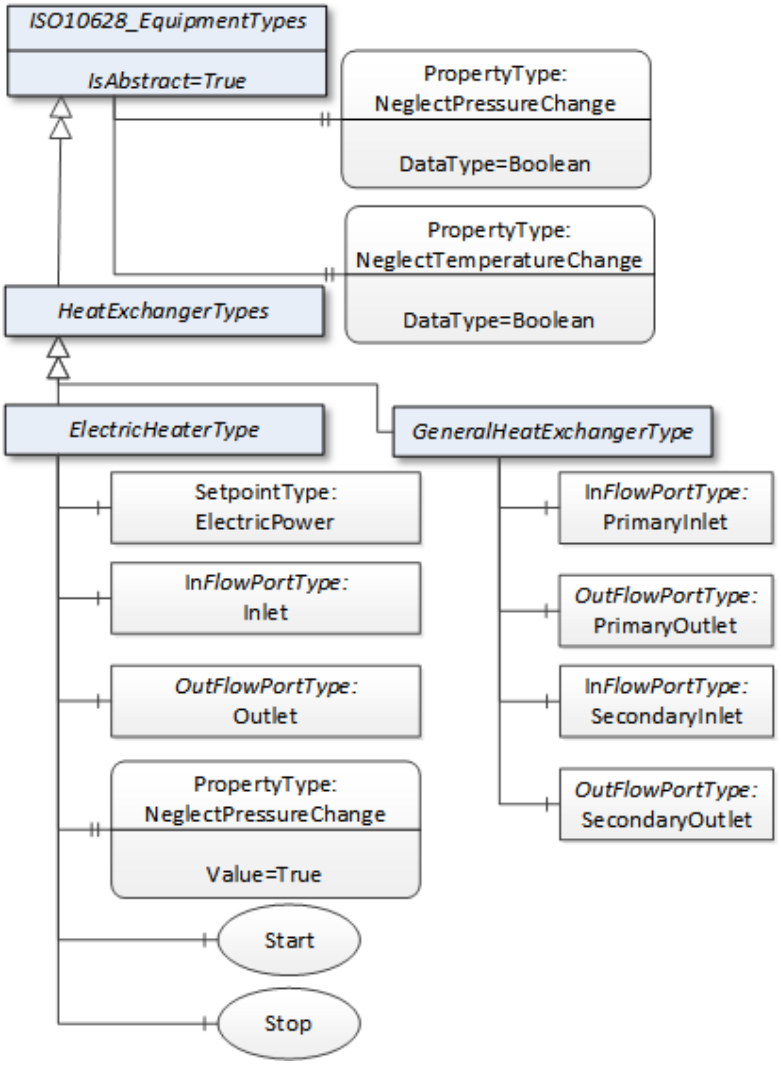


Figure 3.4: OPC UA information model - excerpt of the specified equipment.

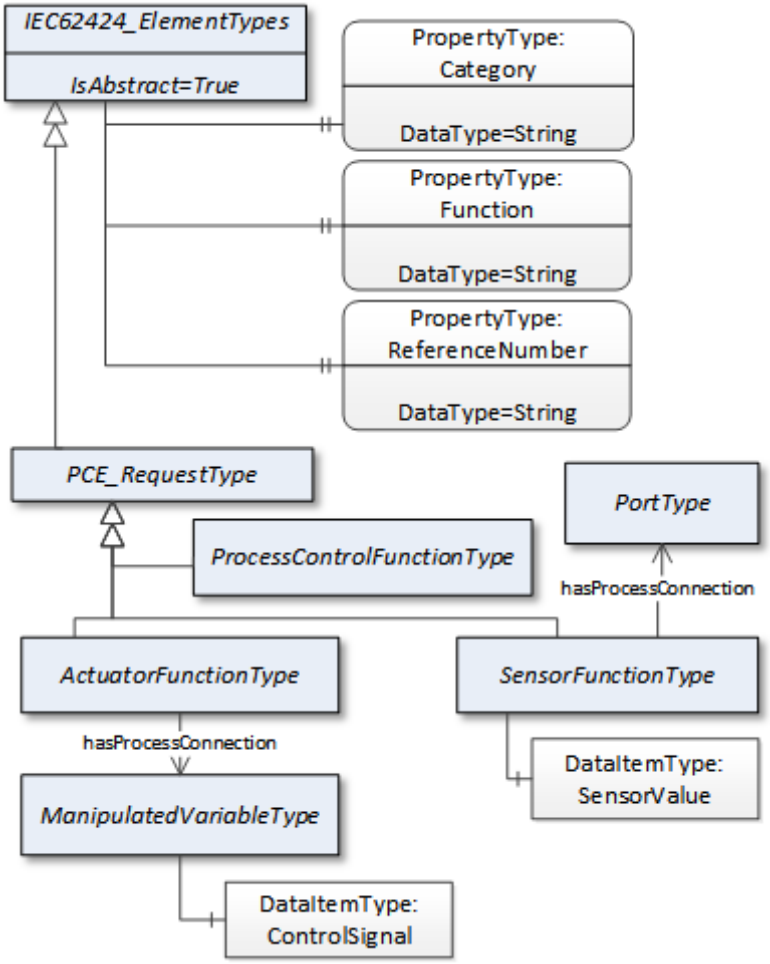


Figure 3.5: OPC UA information model - PCE request type

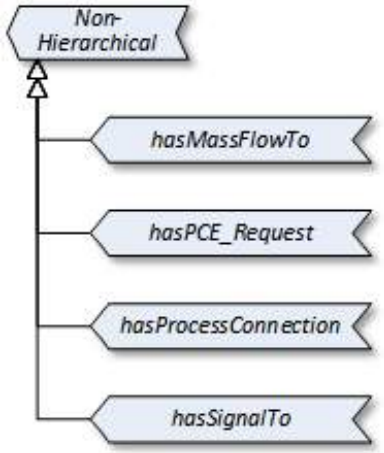


Figure 3.6: OPC UA information model - reference hierarchy

server exposes its address space, where a client can access it to perform the transformation process.

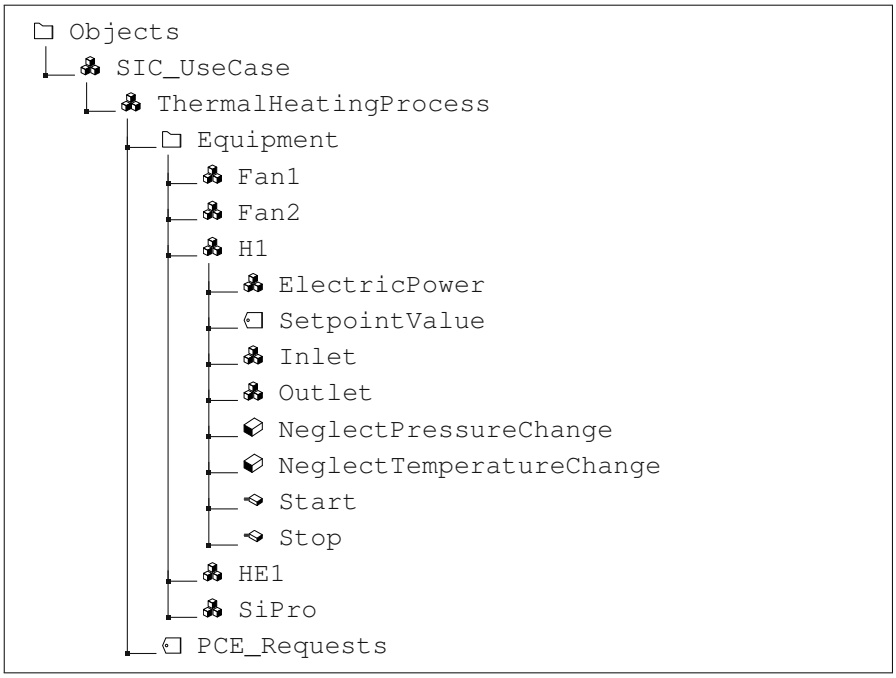


Figure 3.7: Instantiating the OPC UA information model for the simple thermal heating process.

3.4.2 Domain Ontology - PETIont

As target ontology for the presented proof of concept, the Pipe, Equipment, Topology, and Instrumentation Ontology (PETIont) is used. It has similar concepts as the OPC UA information model as it is partly based on the same standards, like IEC 62424 with its PCE requests. The ontology is implemented in OWL DL, and its main class concepts and relations are depicted in Fig. 3.8. The ontology distinguishes between hydraulic and thermal equipment. Thermal equipment changes the temperature between its input port and output port, whereas hydraulic equipment changes the pressure. The complete ontology can be found on GitHub[2].

3.4.3 Transformation Rules

The first step of the transformation process is reading the address space form the OPC UA server and creating the OWL Full representation of it. Basically, an OPC UA client connects to the server and extracts the address space from it. The mapping creates an *owl:Class* for the OPC UA node classes *ObjectType*, *VariableType* and *DataType*.

[2] https://github.com/Smart-Industrial-Concept/PETIont

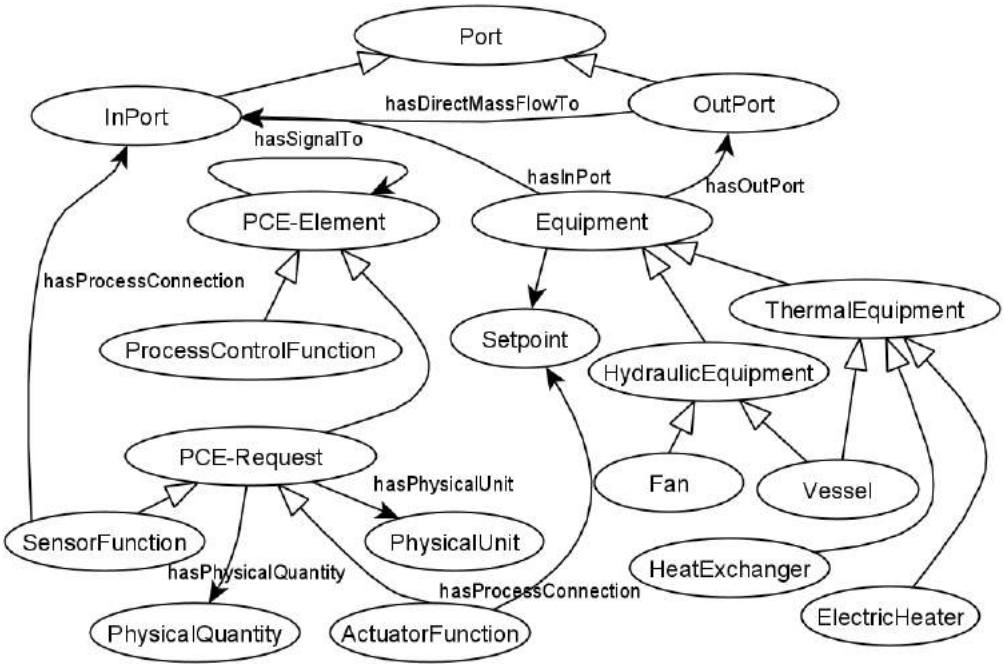


Figure 3.8: PETIont - main concepts and their relations.

Nodes of node class *ReferenceType* are mapped to *owl:ObjectProperty*. A more detailed description of all these transformation rules can be found in [18].

The second step in the transformation process is performing a graph transformation on the created OWL Full representation of the OPC UA information model. This is achieved with the help of the SPARQL construct feature, which creates RDF triples based on a SPARQL query. As the quad store supports named graphs, the results are stored in a new graph inside the quad store. Thus, the new graph can be built step-wise by executing rules which search for certain patterns inside the source graph. For our proof of concept, we used Apache Jena Fuseki version 3.17.0 as the quad store.

For the transformation in our use case between the presented OPC UA information model and PETIont, we specify nine different SPARQL rules. To make the rules more understandable, not all created relations between the concepts are explicitly mentioned in the following rule description. The transformation output is indicated with a "→":

1. Equipment where the pressure change is not neglected → *peti:HydraulicEquipment*
2. Equipment where the temperature change is not neglected → *peti:ThermalEquipment*
3. A *hasMassFlowto* reference between two ports belonging to different components → *peti:hasDirectMassFlowTo*
4. A *hasMassFlowTo* reference between two ports belonging to the same component → *peti:hasInternalMassFlowTo*

5. *ActuatorFunctionType* → *peti:PCEActuatorFunction*
6. *SensorFunctionType* → *peti:PCESensorFunction*
7. *SensorFunctionType* including a control function (letter "C") → *peti:ProcessControlFunction* and a *peti:Setpoint*
8. A "HasProcessConnection" between a PCE request and a component → *peti:hasProcessConnection*
9. A *HasSignalto* reference between two PCE requests → *peti:hasSignalTo*

To give an impression how these rules are implemented in SPARQL, Listing 3.1 shows rule number 3 as an example. This rule is chosen because of its traceability to understand the general principles without being too complex.

```
1  PREFIX uaBase: <http://auto.tuwien.ac.at/~ontologies/opcua#>
2  PREFIX : <http://auto.tuwien.ac.at/~ontologies/useCaseOPCUA#>
3  PREFIX peti: <http://auto.tuwien.ac.at/sic/PETIont#>
4
5  INSERT{
6   #create instances of the mass flow connection between the
7   #ports of connected component
8   GRAPH <http://auto.tuwien.ac.at/petiGraph> {
9     ?startPortURI peti:hasDirectMassFlowTo ?endPortURI.
10    ?endPortURI peti:hasDirectMassFlowFrom ?startPortURI.
11    }
12 }
13 WHERE{
14    #retrieve the mass flow connection between a outport of a component
15    #and the inport of another component
16    ?startPort :hasMassFlowTo ?endPort.
17    ?startPort a :outFlowPortType.
18    ?endPort a :inFlowPortType.
19
20    #retrieve the names of the connected components
21    ?startPort uaBase:BrowseName ?startPortName.
22    ?endPort uaBase:BrowseName ?endPortName.
23    ?startPort uaBase:ComponentOf ?startComp.
24    ?startComp uaBase:BrowseName ?startCompName.
25    ?endPort uaBase:ComponentOf ?endComp.
26    ?endComp uaBase:BrowseName ?endCompName.
27
28    #create the URIs of the instances in the target ontology
29    BIND("http://auto.tuwien.ac.at/sic/PETIont#" as ?petiURI)
30    BIND(uri(?petiURI + str(?startCompName) +"_" + str(?startPortName)) as ?
        startPortURI)
31    BIND(uri(?petiURI + str(?endCompName) +"_" + str(?endPortName)) as ?endPortURI
        )
32 }
```

Listing 3.1: SPARQL rule 3 - Transforming a mass flow connecting two different components.

Line 1 to 3 are just defining abbreviations for the namespaces. The "WHERE" clause is used to find patterns inside the OWL representation of the OPC UA information model. In the first block (lines 14 to 18), all *hasMassFlowTo* connections between an

outport and an inport are retrieved. In the second block (lines 20 to 26), the associated components' names are queried. These names are used to define the URIs used in the target ontology, stored in the variables *?startPortURI* and *?endPortURI*. This definition happens in the last block, starting in line 28. In the "INSERT" clause, the object property *peti:hasDirectMassFlowto* and its inverse property *peti:hasDirectMassFlowto* are instantiated in the target ontology (lines 9 and 10). The resulting target ontology is created in a named graph, specified in line 8, which separates it from the source graph.

3.4.4 Information Retrieval

After the transformation process is performed, the information is accessible via the SPARQL endpoint of the Fuseki Server. As the target ontology used OWL DL, reasoning can also be applied if necessary. The full potential of the approach will be achieved if the created domain-specific ontology will be interlinked with other domain ontologies to build a large knowledge graph. This knowledge graph would be accessible for various applications or services within Cyber-Physical Systems (CPSs).

To prove that our transformation was successful, a simple example is given in Listing 3.2. It shows how information can be retrieved from the instantiated target ontology with the help of SPARQL. Therefore, the plant topology information, represented in the domain-specific ontology, is accessed. The SPARQL query retrieves the installed equipment. Additionally, directly connected components located before and after the equipment are also retrieved. The answer to that query is also depicted at the end of Listing 3.2. To reduce the query's complexity, property paths are used in lines 8 and 11, supported in SPARQL version 1.1.

```
1  PREFIX : <http://auto.tuwien.ac.at/sic/PETIont#>
2
3  SELECT *
4  WHERE {
5    ?equipment a :Equipment
6
7    optional{
8     ?equipment :hasOutPort/:hasDirectMassFlowTo/^:hasInPort ?nextComp
9    }
10   optional{
11    ?equipment :hasInPort/^:hasDirectMassFlowTo/^:hasOutPort ?prevComp
12   }
13 }
```

```
| ?equipment | ?nextComp | ?prevComp |
|""""""""""""""""""""""""""""""""""""|
| :H1        | :SiPro    | :Fan1     |
| :Fan2      | :HE1      | :SiPro    |
| :HE1       | :Fan1     | :Fan2     |
| :Fan1      | :H1       | :HE1      |
| :SiPro     | :Fan2     | :H1       |
```

Listing 3.2: SPARQL query to retrieve the topology information with its corresponding result.

3.5 Discussion and Future Work

We evaluated the proposed approach for transforming OPC UA information models into domain-specific ontologies with a proof of concept. The adaption of the transformation process to other OPC UA information models or domain ontologies can be achieved by only adapting the SPARQL rules. The remaining steps in the process are executed automatically, which makes it flexible to an adaption of the source model or target ontology. An additional benefit of our approach is that it is also applicable for already existing OPC UA servers, as the information model is directly retrieved from the address space of a running OPC UA server.

We also briefly introduced the information retrieval by applying a SPARQL query to the endpoint. Having all this information about the plant topology, the available equipment, and instrumentation in a machine-readable, formal representation is very beneficial for a broad spectrum of applications or services. It can be used to facilitate the monitoring, diagnosis, prediction, and control of a plant. The interlinking of this information with other domain-specific ontologies will enable the proposed transformation approach's full potential.

A pitfall of applying SPARQL rules for the graph transformation could be, that dependencies between rules can easily be introduced. These dependencies would require a certain execution order for the transformation. This should be avoided by design or clearly documented if it cannot be avoided.

In the future, we want to use the available topology information from the OPC UA information model to evaluate sensor data in CPS, based on the information in the created domain-specific ontology and other interlinked domain ontologies, capturing further knowledge about the plant. Various services can make use of and contribute to such an interlinked, shared knowledge graph, which will enable new capabilities within CPSs.

References

[1] J. Bakakeu et al. „Automated reasoning and knowledge inference on OPC UA information models“. In: *Proceedings - 2019 IEEE International Conference on Industrial Cyber Physical Systems, ICPS 2019* (2019), pp. 53–60. DOI: 10.1109/ICPHYS.2019.8780114.

[2] Thomas Bindel and Dieter Hofmann. *R&I-Fließschema*. essentials. Wiesbaden: Springer Fachmedien Wiesbaden, 2016. ISBN: 978-3-658-15558-2. DOI: 10.1007/978-3-658-15559-9. URL: http://link.springer.com/10.1007/978-3-658-15559-9.

[3] Dan Brickley and R.V. Guha. *RDF Schema 1.1. W3C Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/rdf-schema/.

[4] EN ISO 10209:2012. *Technical product documentation - Vocabulary - Terms relating to technical drawings, product definition and related documentation*. Tech. rep. 2012.

[5] EN ISO 10628. *Diagrams for the chemical and petrochemical industry*. Tech. rep. European Standards, 2012.

[6] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language. W3C Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2013. URL: https://www.w3.org/TR/sparql11-query/.

[7] IEC-62424. *Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*. Tech. rep. International Electrotechnical Commission, 2016.

[8] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation*. Tech. rep. World Wide Web Consortium, 2004. URL: https://www.w3.org/TR/rdf-concepts/.

[9] Markus Krötzsch. „OWL 2 Profiles: An Introduction to Lightweight Ontology Languages“. In: *Reasoning Web. Semantic Technologies for Advanced Query Answering: 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings*. Ed. by Thomas Eiter and Thomas Krennwallner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 112–183. ISBN: 978-3-642-33158-9. DOI: 10.1007\/978-3-642-33158-9_4.

[10] Byunghun Lee et al. „Model transformation between OPC UA and UML“. In: *Computer Standards and Interfaces* 50 (2017), pp. 236–250. ISSN: 09205489. DOI: 10.1016/j.csi.2016.09.004. URL: http://dx.doi.org/10.1016/j.csi.2016.09.004.

[11] Jay Lee, Behrad Bagheri, and Hung An Kao. „A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems“. In: *Manufacturing Letters* 3 (2015), pp. 18–23. ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001. arXiv: 1503.07717. URL: http://dx.doi.org/10.1016/j.mfglet.2014.12.001.

[12] Yang Lu. „Industry 4.0: A survey on technologies, applications and open research issues“. In: *Journal of Industrial Information Integration* 6 (2017), pp. 1–10. ISSN: 2452414X. DOI: 10.1016/j.jii.2017.04.005. URL: http://dx.doi.org/10.1016/j.jii.2017.04.005.

[13] Wolfgang Mahnke, Stefan-Helmut Leitner, and Matthias Damm. *OPC Unified Architecture*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 978-3-540-68898-3. DOI: 10.1007/978-3-540-68899-0. URL: http://link.springer.com/10.1007/978-3-540-68899-0.

[14] OPC Foundation. *OPC Unified Architecture Specification. Part 3: Address Space Model. Release 1.04*. Tech. rep. OPC Foundation, 2017.

[15] OPC Foundation. *OPC Unified Architecture Specification. Part 5: Inforamion Model. Release 1.04*. Tech. rep. 2017.

[16] Rainer Schiekofer and Michael Weyrich. „Querying OPC UA information models with SPARQL“. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2019-Septe (2019), pp. 208–215. ISSN: 19460759. DOI: 10.1109/ETFA.2019.8868246.

[17] Rainer Schiekofer et al. „A formal mapping between OPC UA and the semantic web“. In: *IEEE International Conference on Industrial Informatics (INDIN)* 2019-July (2019), pp. 33–40. ISSN: 19354576. DOI: 10.1109/INDIN41052.2019.8972102.

[18] Gernot Steindl, Thomas Früwirth, and Wolfgang Kastner. „Ontology-Based OPC UA Data Access via Custom Property Functions“. In: *24th Interantional Conference on Emerging Technologies and Factory Automation*. Zaragoza, Spain, 2019.

[19] Rudi Studer, V Richard Benjamins, and Dieter Fensel. „Knowledge Engineering: Principles and methods“. In: *Data & Knowledge Engineering* 25 (1998), pp. 161–197.

[20] Koen H. Van Dam and James Keirstead. „Re-use of an ontology for modelling urban energy systems“. In: *3rd International Conference on Next Generation Infrastructure Systems for Eco-Cities, INFRA 2010 - Conference Proceedings* (2010). DOI: 10.1109/INFRA.2010.5679232.

[21] W3C OWL Working Group. *OWL 2 Web Ontology Language. Document Overview*. Tech. rep. World Wide Web Consortium (W3C), 2012. URL: https://www.w3.org/TR/owl2-overview.

CHAPTER 4

Ontology-Based OPC UA Data Access via Custom Property Functions

Publication: *G. Steindl, T. Frühwirth and W. Kastner, "Ontology-Based OPC UA Data Access via Custom Property Functions," 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019, pp. 95-101, doi: 10.1109/ETFA.2019.8869436.*

Abstract: Cyber Physical Production Systems have a need of sharing and interlinking information and knowledge over different domains. In the area of industrial automation, OPC Unified Architecture (OPC UA) is a widely used and established standard for communication and information modeling. We propose an ontology-based OPC UA data access method utilizing custom property functions, which enables interlinking between OPC UA information and other factory data. To avoid duplicated data and to reduce the communication overhead in the proposed method, the OPC UA run-time data are loaded on-demand and are not persistently stored in the triplestore. To enable fast and easy ontology-based access and interlinking of the OPC UA information, the needed ontology is automatically generated from the OPC UA information model. A proof of concept demonstrates the application of our approach for a laboratory use-case of a Packed-Bed Regenerator.

4.1 Introduction

The process of digitalization in industry is an ongoing challenge for industry itself as well as for academia. This transformation is often called the fourth industrial revolution

or Industry 4.0. The main goals of Industry 4.0 are optimization and customization of production as well as enhanced automation and adaption [21]. This claims benefits for increasing flexibility of the whole production system, which is a composition of human resources, production equipment, and aggregated products that interact over cyber-physical interfaces [20]. Such production systems are referred to as CPPSs.

CPPSs can be structured into five levels, which is called the 5C architecture [13] and depicted in Figure 4.1. This architecture consists of a connection, conversion, cyber, cognition, and configuration level. The Semantic Web Stack, including the RDF, the RDFS and the OWL as well as SPARQL can be applied on the upper levels of this 5C architecture to implement and enhance its functionality. Semantic Web technologies can be used to semantically enrich industry data, integrate data from heterogeneous sources, and increase interoperability. With the help of Semantic Web technologies engineering and run-time information can be interlinked to form a so-called ontology-based Linked Factory Data [23]. A key aspect of Linked Factory Data is to keep the data in their natural format and storage, which can be achieved with the help of various types of ontology-based data integration methods [6].

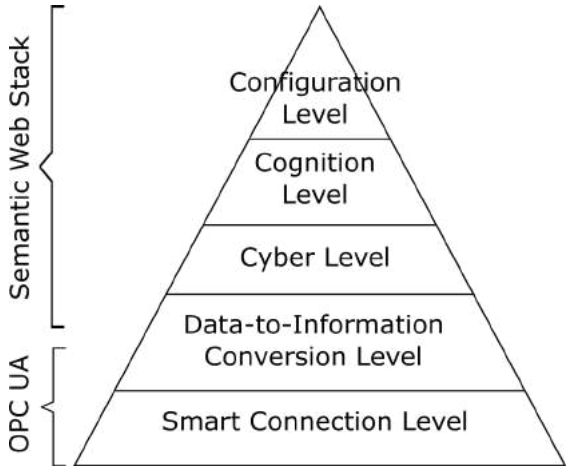


Figure 4.1: Levels of the 5C architecture for CPPS [13] and the application of OPC UA and Semantic Web technologies

In the area of industrial automation, OPC UA is a well-established standard for industrial communication. It specifies the data exchange and facilitates semantic interoperability by use of an extensible information model. Thus, OPC UA is a framework for object-oriented data and information representation and exchange [4]. As it is widely used and also recommended by the Industry 4.0 initiative [1], the integration into Linked Factory Data is addressed in this paper. Therefore, the transformation of the OPC UA information model into an OWL ontology as well as an ontology-based data access method is presented, which both are prerequisite for OPC UA integration into Linked Factory Data.

Usually, the information in an ontology is stored in a graph-based triplestore and is not changing frequently over time. In industrial processes, run-time data, like sensor data

or state conditions, are generated in high volume and velocity. Triplestores are not well suited for large time series data as it will reduce the performance of SPARQL queries [16], because the query engine has to perform a pattern matching over the whole graph in the triplestore. Therefore, we propose an ontology-based data access method, which uses custom property functions in SPARQL to retrieve run-time data and time series data on-demand and without a persistent storage of the data in the triplestore. To reduce the engineering effort and to enable an easy ontology-based OPC UA data access, an automatic transformation of the OPC UA information model into an OWL ontology is performed.

The remainder of this paper is structured as follows: Section 4.2 is giving a short introduction into the OPC UA information model and Semantic Web technologies. Also, a literature review of Semantic Web applications in combination with OPC UA is presented. Section 4.3 explains the proposed ontology-based OPC UA data access method as well as the automatic ontology generation process. Section 4.4 presents a laboratory use case of a Packed-Bed Regenerator, which is used to demonstrate our approach. In Section 4.5, a conclusion is drawn and ideas for future work are discussed.

4.2 State of the art

This section gives a short overview of OPC UA and its information modeling as well as some introduction to the Semantic Web technologies used in this work. Also previous work is reviewed which already applies Semantic Web technologies in combination with OPC UA.

4.2.1 OPC UA Information Model

A key feature of OPC UA is its information modelling capability. It allows to describe everything from simple devices (sensors, actuators, etc.) to very complex components (production machinery, energy storage devices, etc.) in an object-oriented and semantically meaningful way [15]. The information model is thereby built from *nodes*, representing objects, variables, etc. and *references*, representing relations between nodes. OPC UA defines eight different types of nodes, so-called *node classes*. They are briefly summerized in Table 4.1.

Depending on its node class, each node has a set of *attributes*. Most importantly, each node has a unique *NodeId*, which consists of a *NamespaceIndex* and an *Identifier*, e.g. *"id=2;s=Heater"*. Other attributes are the *NodeClass* (one of the entries of Table 4.1), *DisplayName* (a human-readable name for the node), *Description* (a human-readable description of the nodes's purpose), possibly a *Value*, and many more. The information model exposed by a specific OPC UA server is called the server's *address space*. OPC UA clients use *services* to interact with the server, e.g. the *Browse* service to navigate through the address space, the *Read/Write* service to read/write variable values, the *HistoryRead/HistoryUpdate* service to read/update historic values of variables, the *Call*

Table 4.1: OPC UA node classes

Node class	**Comment**	**Example**
Object Type	The ObjectType can be used to model complex objects. Typically, these objects expose some internal structure.	HeaterType
Object	An Object is an instance of the corresponding ObjectType, just like objects in programming are instances of their corresponding class.	Heater
DataType	DataTypes are typically simple types such as String, Boolean, Float, Int32, etc. However, they can also have a more complex internal structure if needed.	Float
Variable Type	VariableTypes are used to model the value and structure of data. Additional information, e.g. the EngineeringUnit, can be added. The value is of a specific DataType.	AnalogItemType
Variable	A Variable is an instance of the corresponding VariableType.	Temperature
Reference Type	The ReferenceType node class is used to define the references between nodes and their semantics.	HasComponent
Method	Methods can be called by an OPC UA client. Input arguments, as well as output arguments, are supported.	SetSetpoint()
View	Views can be used to structure and filter the information in a user-group-specific way.	OperatorView

service to invoke methods, etc. The historical values themselves are not visible in the OPC UA address space of the server.

4.2.2 Semantic Web Technologies

In the context of Semantic Web, various technologies are standardized under the lead of the W3C[1]. One of the base technologies is called RDF [12], which can be used to model information by creating statements about resources. These statements consist of a subject, a predicate and an object, and, therefore, are called triples. As the object of such a statement can be used as a subject in another statement, the information is represented as a graph with interlinked resources. Special databases, called triplestores, are used for this kind of data.

RDFS [2] and OWL [26] are both formal knowledge representation languages. RDFS is an extension of the basic RDF and provides vocabulary to create hierarchies of classes and properties. OWL extends RDFS with further language constructs, like cardinalities, value restrictions, characteristics of properties, and complex class construction and is based on description logic. Thus, RDFS and OWL enables reasoning for the Semantic Web.

With the help of these technologies and standards, so-called ontologies can be created. In computer science, ontologies are an explicit specification of a conceptualization, where a conceptualization is a simplified abstract model of a certain domain [8]. In addition, ontologies in computer science should be machine readable. Thus, an ontology can be defined as *"... a formal, explicit specification of a shared conceptualization"* [24].

To retrieve information from such an ontology, SPARQL [9] was designed to query RDF data including RDFS and OWL constructs. It supports a *SELECT* statement to perform a graph-based pattern matching over the RDF graph and to retrieve data in a table-based fashion. It also supports the *CONSTRUCT* clause, which allows to extract information from the ontology and to create new RDF graphs based on these data.

Ontologies can be used in different fields of application, like communication, interoperability, as well as information sharing and reuse [25]. In the context of CPPS, data and information integration is one of the key applications, where ontologies are applied. OBDA, as the foundation of Ontology-Based Data Integration (ODBI), enables semantic enrichment of available data from heterogeneous sources to create a semantic integration layer, which is able to provide a higher level of abstraction [22].

Since in the manufacturing domain, data is hardly available in the RDF format, mappings and transformations from existing data sources have to be performed [17]. For relational databases, frameworks for OBDA, like Ontop exist [3] and have already been applied for industrial use cases [18]. Similar concepts have to be investigated for OPC UA, which is a common data source in an industrial environment [4].

[1] https://www.w3.org

4.2.3 Combining Semantic Web Technology and OPC UA

Semantic Web technology in combination with OPC UA can be used in CPPSs for various applications, like creating flexible orchestration plans in manufacturing [11] or semantic data integration and analysis like conceptually shown in [19].

Semantic Web technologies are also used to harmonize the access and utilization of industrial devices. Therefore, a conceptional architecture with an OPC UA adaption layer is presented in [10], to perform a transformation of data exchange structures into RDF. The transformation itself is not described in detail in this work, because it primarily focuses on the creation of a plant model ontology.

A more concrete implementation strategy of combining OPC UA and Semantic Web for an ontology-based OPC UA data access can be found in [14]. Thereby, Semantic Web technology is used in combination with OPC UA for sensor discovery. A semantic access layer is implemented to tap the full potential of ontologies while not affecting the existing OPC UA standard. The data are retrieved by a subscription handler if available. Otherwise, a sensor data request is triggered frequently to map the OPC UA data to a sensor ontology. In [27], the OPC UA ontology-based data access is enabled by mirroring the OPC UA data into a relational database and to map this data into an ontology. This architecture is chosen to enable real-time processing of a large amount of industrial data. The stored data in this database are used in combination with the ontology to perform reasoning periodically to generate new knowledge dynamically. Both of these approaches to ontology-based OPC UA data access have the disadvantages of duplicating data or communication overhead, if the information is frequently polled.

To directly access the OPC UA data, a so-called linked data adapter is implemented in [7]. This adapter provides data from an OPC UA server as RDF via a REST interface. The base functionality of OPC UA is mapped to HTTP GET, POST, PUT and DELETE requests. As the adapter returns RDF data, URI dereferencing can be used in SPARQL to access these data on-demand.

Our proposed approach of ontology-based OPC UA data access is a direct extension of the SPARQL query engine, by implementing so-called custom property functions. This extension implements an efficient data extraction mechanism, as it only retrieves the necessary data from an OPC UA server, if a SPARQL query is invoked, which needs these specific data. This enables data access on-demand, but waives an additional REST mapping.

4.3 Proposed Ontology-based OPC UA Data Access Method

To enable ontology-based OPC UA data access, an OWL model has to be created and mapped to the OPC UA information model. To reduce engineering effort, the OPC UA information model, which is stored in the OPC UA server, is extracted by a software

module and automatically transformed into OWL. This needs only to be performed once or if the OPC UA information model of the server has changed.

To implement the proposed ontology-based OPC UA data access method, a Semantic Web framework has to be used which supports custom property functions. Inside these functions, the OPC UA server is accessed and its run-time data is retrieved. For our proof of concept, the Apache Jena Framework[2] is used. It is able to handle the OWL data and its SPARQL query engine ARQ can be extended with custom property functions. For the OPC UA communication the Eclipse Milo framework[3] is chosen to implement an OPC UA client inside the ARQ extension. To test the implementation, an Apache Jena Fuseki server is configured to load the ARQ extension and to provide a SPARQL web interface for exploring the data in the triplestore. Figure 4.2 gives an overview of the used software modules of our proof of concept. The OPC UA ontology extraction as well as the custom property function implementation are described in more detail in the following subsections.

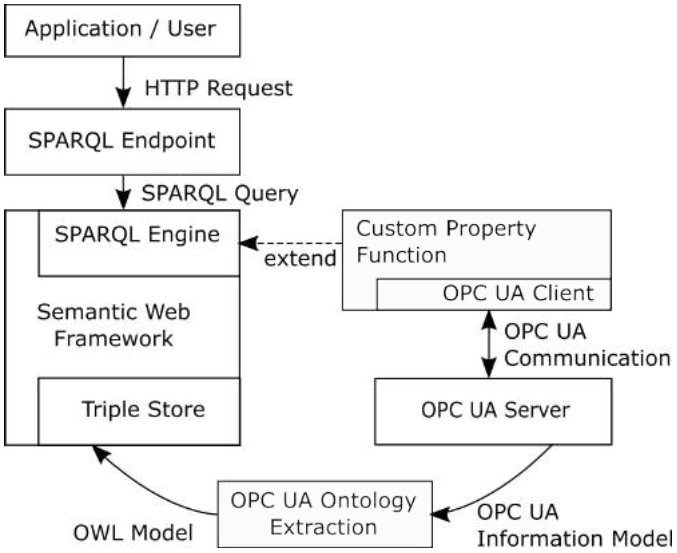


Figure 4.2: Software Modules of the proof of concept

4.3.1 OPC UA Ontology Extraction

The components and steps involved in generating OWL models from the address space of an existing OPC UA server are illustrated in Figure 4.3. First, the OPC UA client connects to the OPC UA server and starts analyzing the address space by reading the OPC UA server's *NamespaceArray*, which associates each *NamespaceIndex* to its corresponding *NamespaceUri*. This is required because each *NodeId* only contains the *NamespaceIndex*, while in Semantic Web technologies each resource is identified by

[2]https://jena.apache.org
[3]https://projects.eclipse.org/proposals/milo

⟨*Namespace*⟩#⟨*Identifier*⟩. Next, the OPC UA client recursively browses the address space of the OPC UA server and creates an RDF model for each *Namespace*. It thereby maps OPC UA nodes to RDF resources and OPC UA references to RDF properties. Additionally, for each reference between two OPC UA nodes, an RDF statement is created in the model. If such a statement involves resources of different namespaces, and thus different models, it is added to the model corresponding to the namespace with the higher index, according to the *NamespaceArray*. This way, the model corresponding to namespace with *NamespaceIndex* 0 only contains statements about namespace 0, namespace 1 contains statements about namespace 1 and 0, etc. Finally, attributes in OPC UA (such as *NodeId*, *BrowseName*, etc.) are added to the corresponding resources by means of literals.

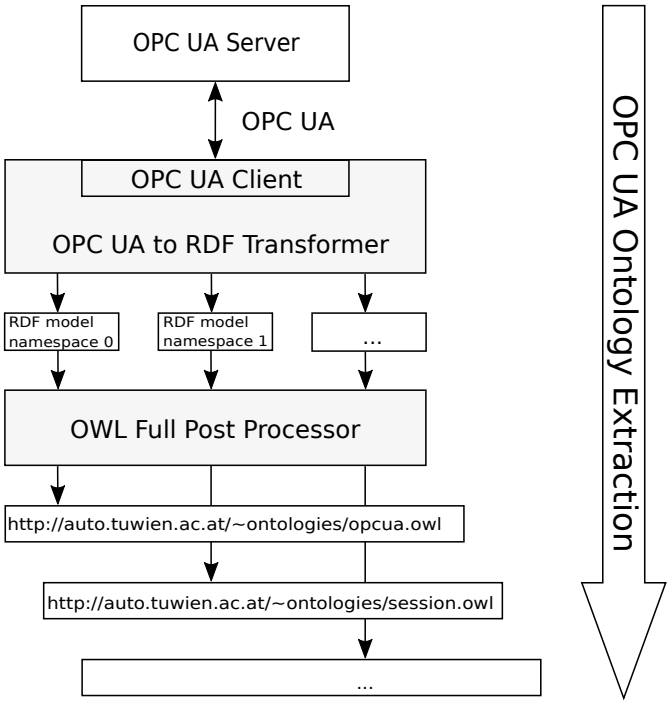


Figure 4.3: OPC UA ontology extraction process

The result of this transformation step is a set of models, which only use the vocabulary provided by RDF. Additional vocabulary of a more specific ontology language, e.g. RDFS, OWL DL or OWL Full, is then added in the post processing step. As different ontology languages provide different modelling concepts, the resulting RDFS/OWL models may deviate quite substantially from the initial OPC UA address space. A very prominent challenge in this regards is the capability of OPC UA to arbitrarily create references between objects (corresponding to individuals in most ontology languages) and types (corresponding to classes in most ontology languages). Only OWL Full allows to use object properties for relating individuals and classes with similar flexibility. This short discussion already indicates that the post processing step requires many design decisions

and compromises to be made. As a more expressive ontology language enables the creation of models that better represent the OPC UA address space, a post processor for OWL Full has been developed in a first attempt. The post processor currently implements the following transformation rules and can easily be extended:

- An rdfs:label is created for each resource from the OPC UA node's *DisplayName*.
- All resources corresponding to an OPC UA node of node class *ObjectType*, *VariableType*, or *DataType* are declared as owl:Class.
- All resources corresponding to an OPC UA node of node class *ReferenceType* are declared as owl:ObjectProperty.
- All resources corresponding to an OPC UA node being the source of a *HasTypeDefinition* reference are declared as individual of the corresponding class.
- All resources corresponding to an OPC UA node being the source of a *HasSubtype* reference are declared as superclass of the corresponding class.
- All object properties, for which the corresponding OPC UA *ReferenceType*'s attribute *Symmetric* is set, are declared as symmetric property.
- For all object properties, for which the corresponding OPC UA *ReferenceType*'s attribute *InverseName* is set, an inverse object property is created.
- All properties relating OPC UA nodes to their OPC UA attributes are declared as annotation properties.

While not always being respected, it is best practice in ontology engineering to use namespaces that actually correspond to ontology documents accessible via the web. This idea is not present in OPC UA, and, thus, the namespaces extracted from the OPC UA server's *NamespaceArray* have no meaning except for being unique. For this reason, the post processor allows to substitute namespaces with user-defined replacements, e.g. `http://opcfoundation.org/UA/` is substituted with `http://auto.tuwien.ac.at/~ontologies/opcua.owl`.[4] This completes the OPC UA ontology extraction process.

[4]The resulting ontology documents are available in RDF/XML format at `http://auto.tuwien.ac.at/~ontologies/opcua.owl`, `http://auto.tuwien.ac.at/~ontologies/session.owl`, and `http://auto.tuwien.ac.at/~ontologies/packed-bed-regenerator.owl`.

4.3.2 Ontology-based OPC UA data access method

As most of the run-time data that are accessible through the *Read* or *HistoryRead* service are changing over time, they should not be statically stored in the ontology. An ontology-based data access method has to be provided which loads the OPC UA data on-demand to avoid a periodical or event-based update of the ontology and to keep the triplestore as small as possible.

We have implemented such an ontology-based data access for the OPC UA *Read* and *HistoryRead* service as an extension to a SPARQL query engine. SPARQL is a very flexible language which allows to implement custom property functions to add functionality. Two custom property functions were implemented, namely *value* and *histValue*. These functions use the information stored in the ontology to retrieve the OPC UA endpoint url and the OPC UA node ID to connect to the OPC UA server and request the required data. These two custom property functions are registered within the `http://auto.tuwien.ac.at/~ontologies/opcua.owl` namespace.

The *value* property function can be applied on every OPC UA node in the ontology. If for that node the *Read* service is available, the data are retrieved during the SPARQL query and the value is assigned to the SPARQL variable. Listing 4.1 shows a SPARQL query which uses the custom property function *value*.

The *histValues* property function returns the timestamp and its related value and binds them to the specified SPARQL variables. As additional parameter, the start and end time of the *HistoryRead* service can be defined. If no end time is defined, the current time is used as default value. Table 4.2 illustrates the implemented function overloadings of *histValues*. The usage of the custom property function *histValues* is shown in Listing 4.2

Table 4.2: Overloading of the Custom Property Function *histVlaues*

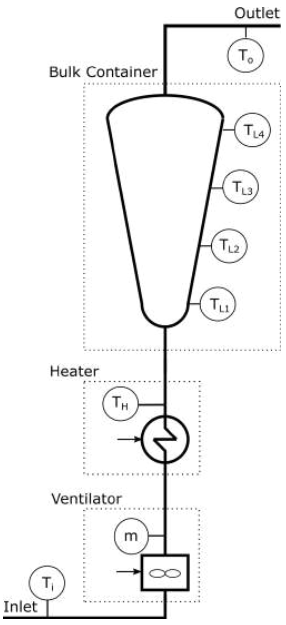
Function Signature	**Behaviour**
histRead(?timestamp ?value)	Retrieves all available data
histRead(?timestamp ?value int n)	Retrieves the n last values till now
histRead(?timestamp ?value "YYYY-MM-DD hh:mm:ss")	Retrieves all data since the specified date
histRead(?timestamp ?value "YYYY-MM-DD hh:mm:ss" "YYYY-MM-DD hh:mm:ss")	Retrieves all data between the first and second date-string

SPARQL can not only be used for querying, but also for updating or inserting new data. For example, if certain OPC UA run-time data should be stored persistently in the

ontology, the SPARQL *INSERT* command can be utilized.

4.4 Use Case - Packed-Bed Regenerator

As use-case for the ontology-based OPC UA data access, a model of a PBTES test rig is chosen, which is located at the laboratory of the Institute for Energy Systems and Thermodynamics (IET) at TU Wien (Figure 4.4b). The PBTES is a thermal energy storage, which has a conic steel container filled with gravel as storage medium and surrounded by an insulation. Ambient air is used as a heat transfer fluid. It gets heated by a electric heater and transported through the tank during the charging phase. The hot air heats up the gravel inside the tank which stores the energy. For discharging, cold air is ventilated through the hot gravel. The schematic of the PBTES and its components are depicted in (Figure 4.4a)



(a) Schematic diagram of the Packed-bed Regenerator

(b) Test rig at the laboratory of TU Wien [5]

Figure 4.4: Packed-bed Regenerator

The PBTES consists of a ventilator, a heater and the bulk container. It has various temperature sensors installed as well as a mass flow sensor. The bulk container has four temperature sensors at different levels to track its state of charge. An OPC UA information model has been created which is partly shown in Figure 4.5. This model is automatically transformed into OWL, as explained in previous Section. With this OWL model and the implemented custom property function *value* and *histValues*, ontology-

based OPC UA data access is performed, by sending SPARQL queries to the Apache Jena Fuseki Server.

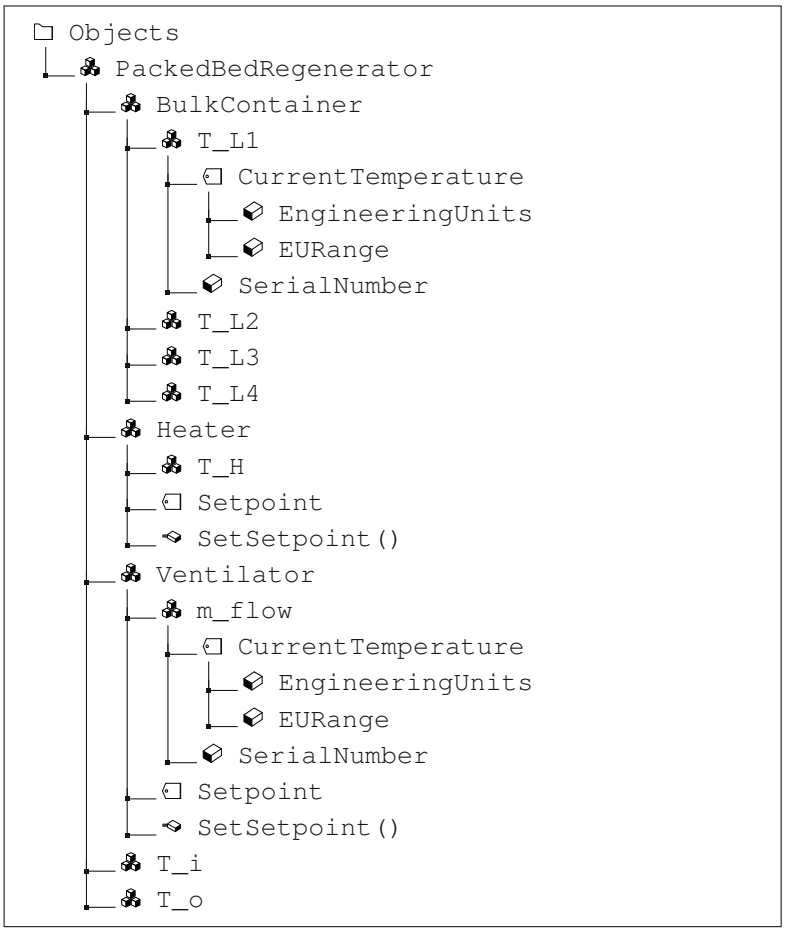


Figure 4.5: OPC UA Information Model of the Packed-Bed Regenerator

Listing 4.1 shows a SPARQL query which retrieves the current value of the temperature sensor T_{L1} in the bulk container of the PBTES. The SPARQL query engine connects to the OPC UA server in the background, requests the data, and assigns it to the SPARQL variable *?temp*. The answer to this query is also shown at the end of Listing 4.1.

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX uaBase: <http://auto.tuwien.ac.at/opcua.owl#>
3  PREFIX j.0: <http://auto.tuwien.ac.at/packed-bed-regenerator.owl#>
4
5  SELECT ?displayName ?temp
6  WHERE {
7      ?sensor rdfs:label "T_L1".
8      ?sensor rdfs:label ?displayName.
9      ?sensor uaBase:HasComponent/uaBase:value ?temp
10 }
```

```
|?displayName|?temp |
```

```
|"T_L1"       | 81.00|
```

Listing 4.1: SPARQL query using custom property function *value* and its corresponding answer

Listing 4.2 shows a SPARQL query which retrieves the historical data (timestamp and values) of the temperature sensor T_{L1}, starting at 2019-03-21 10:00:00. As the retrieved data is assigned to a SPARQL variable, the values can be processed by the SPARQL engine. In the case of this query, a filtering of the values greater than "41.0" is applied. The corresponding answer to this query is also partly shown in Listing 4.2.

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX uaBase: <http://auto.tuwien.ac.at/opcua.owl#>
3  PREFIX j.0: <http://auto.tuwien.ac.at/packed-bed-regenerator.owl#>
4
5  SELECT ?displayName ?time ?temp
6  WHERE {
7      ?sensor rdfs:label "T_L1".
8      ?sensor rdfs:label ?displayName.
9      ?sensor uaBase:HasComponent/uaBase:histValues (?time ?temp "2019-03-21
       10:00:00").
10     FILTER(?value > "41.0")
11 }
```

```
|?displayName|?time                                  |?temp |
|"T_L1"       |"Thu Mar 21 10:00:00 CET 2019"|  81.0|
|"T_L1"       |"Thu Mar 21 10:00:01 CET 2019"|  80.0|
|  ...        |  ...                                 |  ... |
|"T_L1"       |"Thu Mar 21 10:25:52 CET 2019"| 104.0|
```

Listing 4.2: SPARQL query using custom property function *histValue* and its corresponding answer

In Listing 4.3 a SPARQL query is shown, which retrieves all temperature sensor values from the bulk container and calculates the average temperature. As the OPC UA type definitions are present in the *Objects* and the *Types* folder of the OPC UA information model, the query has to state that the *BulkContainer* is a component of the *PackedBedRegenerator*. Afterwards, all components of the *BulkContainer* which are from type *TemperatureSensorTypes* are retrieved and their current values are accessed by invoking the custom property function *uaBase:value*.

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX uaBase: <http://auto.tuwien.ac.at/opcua.owl#>
3  PREFIX j.0: <http://auto.tuwien.ac.at/packed-bed-regenerator.owl#>
4
5  SELECT  (avg(?values) as ?averageTemp)
6  WHERE {
7      ?regenerator rdfs:label "PackedBedRegenerator".
8      ?regenerator uaBase:HasComponent+ ?bulkCont.
9      ?bulkCont uaBase:HasTypeDefinition  j.0:BulkContainerType.
10     ?bulkCont uaBase:HasComponent+ ?tempSensors.
11     ?tempSensors uaBase:HasTypeDefinition  j.0:TemperatureSensorType.
12     ?tempSensors uaBase:HasComponent/uaBase:value ?values
13     }
```

```
|?averageTemp|
|"""""""""""""|
|        195.0|
```

Listing 4.3: SPARQL query to retrieve the average temperature of the bulk container and its corresponding answer

The temperatures in the bulk container are $T_{L1} = 180\,°C$, $T_{L2} = 190\,°C$, $T_{L3} = 200\,°C$ and $T_{L4} = 210\,°C$. The answer to this query is an average temperature of $T_A = 195\,°C$, which is also shown as a result at the end of Listing 4.3. The calculation of the average temperature is performed by the SPARQL engine, as the retrieved data are assigned to the SPARQL variable *values*.

4.5 Conclusion and future work

We have shown how ontology-based OPC UA data access can be implemented with the help of custom property functions and an automatic transformation of the OPC UA information model into OWL. The combination of Semantic Web technologies and OPC UA enables new possibilities in the context of CPPS, which could lead to higher flexibility in the production system. We have shown the application of the SPARQL query language to retrieve OPC UA data from a specified information model.

In this paper, we did not explore the reasoning capabilities of OWL, which can be beneficial for certain applications in the context of CPPS. Further research will be carried out to enable reasoning for the OPC UA ontology and combining it with logical rules. Also, a performance evaluation of our approach, regarding to query execution time and memory space, is planned as future work.

The presented proof of concept showed only a simple use case in an isolated environment. We believe that the full potential of the ontology-based OPC UA data access is only achieved, if other information, like environmental conditions, production plans, device information, etc. are interlinked to build up Linked Factory Data. This will lead to new insights into the data and the production system itself.

Acknowledgements

This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) for the „PoSyCo - Power System Cognification“ project under the contract number 867276, as well as the „SIC! - Smart Industrial Concept!“ doctoral program.

References

[1] Simon Baier et al. *Guidlines. Industry Services - Technical services in the lifecycle of machines and plants.* Tech. rep. Frankfurt am Main, Germany: German Electrical

and Electronic Manufacturers' Association, 2015. URL: http://www.industry.siemens.com/services/global/en/Pages/home.aspx.

[2] Dan Brickley and R.V. Guha. *RDF Schema 1.1. W3C Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/rdf-schema/.

[3] Diego Calvanese et al. „Ontop: Answering SPARQL Queries over Relational Databases". In: *Semantic Web Journal* (2017). URL: https://github.com/ontop/ontop-examples/.

[4] Peter Drahos et al. „Trends in industrial communication and OPC UA". In: *Proceedings of the 29th International Conference on Cybernetics and Informatics, K and I 2018* 2018-Janua (2018), pp. 1–5. DOI: 10.1109/CYBERI.2018.8337560.

[5] Philipp Drochter. „Auslegung, Konstruktion und Errichtung eines Festbettregenerators". MA thesis. TU Wien, Faculty of Mechanical, Industrial Engineering, Institute for Energy Systems, and Thermodynamics, 2016.

[6] Fajar J Ekaputra et al. „Ontology-Based Data Integration in Multi-Disciplinary Engineering Environments: A Review". In: *Open Journal of Information Systms* 4.1 (2017), pp. 1–26. ISSN: 2198-9281.

[7] Markus Graube, Leon Urbas, and Jan Hladik. „Integrating industrial middleware in Linked Data collaboration networks". In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 2016-Novem. November 2017. IEEE, Sept. 2016, pp. 1–8. ISBN: 978-1-5090-1314-2. DOI: 10.1109/ETFA.2016.7733710. URL: http://ieeexplore.ieee.org/document/7733710/.

[8] Thomas Gruber. „Toward Principles for the Design of Ontologies Used for Knowledge Sharing". In: *International Journal Human-Computer Studies* 43 (1995), pp. 907–928.

[9] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language. W3C Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2013. URL: https://www.w3.org/TR/sparql11-query/.

[10] David Hastbacka and Alois Zoitl. „Towards semantic self-description of industrial devices and control system interfaces". In: *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, Mar. 2016, pp. 879–884. ISBN: 978-1-4673-8075-1. DOI: 10.1109/ICIT.2016.7474867. URL: http://ieeexplore.ieee.org/document/7474867/.

[11] Badarinath Katti, Christiane Plociennik, and Michael Schweitzer. „SemOPC-UA: Introducing Semantics to OPC-UA Application Specific Methods". In: *IFAC-PapersOnLine* 51.11 (2018), pp. 1230–1236. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.08.422.

[12] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation.* Tech. rep. World Wide Web Consortium, 2004. URL: https://www.w3.org/TR/rdf-concepts/.

[13] Jay Lee, Behrad Bagheri, and Hung An Kao. „A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems". In: *Manufacturing Letters* 3 (2015), pp. 18–23. ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001. arXiv: 1503.07717. URL: http://dx.doi.org/10.1016/j.mfglet.2014.12.001.

[14] Christoph Legat, Christian Seitz, and Birgit Vogel-Heuser. „Unified sensor data provisioning with semantic technologies". In: *ETFA2011*. Toulouse, France: IEEE, Sept. 2011, pp. 1–8. ISBN: 978-1-4577-0017-0. DOI: 10.1109/ETFA.2011.6058989. URL: http://ieeexplore.ieee.org/document/6058989/.

[15] Wolfgang Mahnke and Stefan-Helmut Leitner. „OPC Unified Architecture - The future standard for communication and information modeling in automation". In: *ABB Review* 3 (2009), p. 2009.

[16] Michael Martin, Jörg Unbehauen, and Sören Auer. „Improving the performance of semantic web applications with SPARQL query caching". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6089 LNCS.PART 2 (2010), pp. 304–318. ISSN: 03029743. DOI: 10.1007/978-3-642-13489-0_21.

[17] Benjamin Mörzinger et al. „A large-scale framework for storage, access and analysis of time series data in the manufacturing domain". In: *Procedia CIRP* 67.January (2018), pp. 595–600. ISSN: 2212-8271. DOI: 10.1016/j.procir.2017.12.267. URL: http://dx.doi.org/10.1016/j.procir.2017.12.267.

[18] Benjamin Mörzinger et al. „Improving industrial optimizationwith Semantic Web technologies". In: *SEMANTiCS 2018*. Vol. 2198. 2018.

[19] Marek Obitko and Václav Jirkovský. „Big Data Semantics in Industry 4.0". In: *HoloMAS*. Vol. 1. 2015, pp. 217–229. ISBN: 978-3-642-40089-6. DOI: 10.1007/978-3-319-22867-9_19. arXiv: arXiv:1710.04131v1. URL: http://www.springer.com/series/1244%20http://link.springer.com/10.1007/978-3-319-22867-9%7B%5C_%7D19.

[20] Luis Ribeiro and Mats Bjorkman. „Transitioning from Standard Automation Solutions to Cyber-Physical Production Systems: An Assessment of Critical Conceptual and Technical Challenges". In: *IEEE Systems Journal* 12.4 (2018), pp. 3816–3827. ISSN: 1932-8184. DOI: 10.1109/JSYST.2017.2771139.

[21] Vasja Roblek, Maja Meško, and Alojz Krapež. „A Complex View of Industry 4.0". In: *SAGE Open* 6.2 (Apr. 2016). ISSN: 2158-2440. DOI: 10.1177/2158244016653987. URL: http://journals.sagepub.com/doi/10.1177/2158244016653987.

[22] Daniel Schachinger, Wolfgang Kastner, and Stefan Gaida. „Ontology-based abstraction layer for smart grid interaction in building energy management systems“. In: *2016 IEEE International Energy Conference (ENERGYCON)* (2016), pp. 1–6. DOI: 10.1109/ENERGYCON.2016.7513991. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7513991.

[23] Gernot Steindl, Bernhard Heinzl, and Wolfgang Kastner. „A Novel Ontology-Based Smart Service Architecture for Data-Driven Model Development“. In: *eKNOW 2019 - The Eleventh International Conference on Information, Process, and Knowledge Management*. Ed. by Jaime Lloret Mauri. Athens, Greece, 2019, pp. 26–27.

[24] Rudi Studer, V Richard Benjamins, and Dieter Fensel. „Knowledge Engineering: Principles and methods“. In: *Data & Knowledge Engineering* 25 (1998), pp. 161–197.

[25] Mike Uschold and Michael Gruninger. „Ontologies : Principles , Methods and Applications“. In: *The Knowledge Engineering Revie* 11 (1996), pp. 93–136. DOI: 10.1017/S0269888900007797.

[26] W3C OWL Working Group. *OWL 2 Web Ontology Language. Document Overview*. Tech. rep. World Wide Web Consortium (W3C), 2012. URL: https://www.w3.org/TR/owl2-overview.

[27] Shiyong Wang et al. „Knowledge reasoning with semantic data for real-time data processing in smart factory“. In: *Sensors (Switzerland)* 18.2 (2018), pp. 1–10. ISSN: 14248220. DOI: 10.3390/s18020471.

CHAPTER 5

Query Performance Evaluation of Sensor Data Integration Methods for Knowledge Graphs

Publication: *G. Steindl and W. Kastner, "Query Performance Evaluation of Sensor Data Integration Methods for Knowledge Graphs," 2019 IEEE Big Data, Knowledge and Control Systems Engineering (BdKCSE), 2019, pp. 1-8, doi: 10.1109/BdKCSE48644.2019.9010668.*

Abstract: In this paper, a Smart Data Service, based on Semantic Web technology is introduced, which supports the control engineer during the data-driven model development process by enabling enhanced data analysis.

As a perquisite for such a service, sensor data consisting of semantic meta data as well as time series data have to be integrated into a so-called knowledge graph. Therefore, three different integration approaches, found in the literature, were evaluated and compared regarding their query execution performance. The characteristics and limitations of these three methods are discussed to specify the conditions for their specific utilization.

5.1 Introduction

For the development of advanced control strategies for industrial as well as building energy management systems, accurate system models are required. For retrofitting of existing systems with already available monitored data, data-driven model development can be applied to create such models. Nevertheless, the model identification process is time consuming task and requires additional domain knowledge [14]. Unfortunately, this knowledge is often scattered and not well documented.

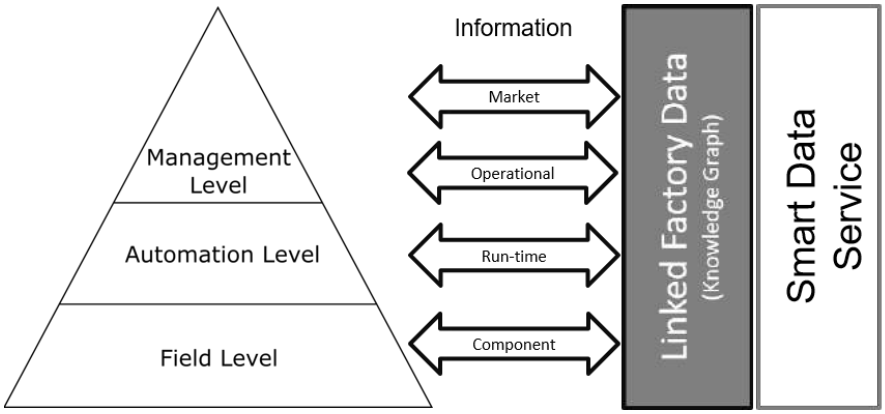


Figure 5.1: Automation Pyramid

From a control engineer's perspective, various data and information sources are available on different levels of the automation pyramid. A generic version of this pyramid – covering factory automation systems as well as building automation systems – is shown in Fig. 5.1. For developing novel control strategies, information out of various layers of the automation pyramid has to be taken into account. Such an information is, for instance, the price from an energy market, the capabilities of installed components, operational data, as well as run-time information, like plant conditions or sensor data.

At the so-called management level, data is usually kept in (relational) database management systems, CSV files, with proprietary formats or sometimes accessible via Web interfaces. Below at the automation level, other data sources are common and often bundled in supervisory control and data acquisition systems based on, e.g. Open Process Control (OPC) or Building Automation and Control Network (BACnet) servers. These heterogeneous data sources are often distributed with their underlying data stored in isolated silos. In any case, access to these different data sources can be difficult, since a lot of implicit knowledge is needed. To retrieve information from a relational database, for example, the user has to know its schema, including tables and columns, which can have meaningless names without any semantics. This makes a data integration process a very laborious task.

A so-called Linked Factory Data [16], where the implicit knowledge is captured in an ontology and is interlinked with existing information sources, could help to overcome some of these problems. Based on Semantic Web technology, the Linked Factory Data forms a so-called knowledge graph. This graph can be used to build a Smart Data Service [10], which will act as a single point of interaction for control engineers. This service provides the interlinked, structured and partly labeled data, to reduce the effort of information gathering during the data-driven model development process. It will also facilitate an enhanced semi-automatic data analysis process.

In the context of industrial energy systems, the integration of sensor data into the Linked Factory Data is important. Next to meta data of sensors, e.g. engineering units,

localization of the sensors, manufacturer information, also the time series data have to be accessible over the knowledge graph. Various integration methods are proposed in literature, but their scalability, regarding their data access performance has not been compared, yet. Therefore, the following two research questions are addressed within this paper:

- RQ1: What is the scalability of different sensor data integration approaches for knowledge graphs, regarding their query execution time?
- RQ2: What are the limitations and conditions for the utilization of these approaches?

The remainder of the paper is structured as follows: Sec. 5.2 gives an overview of the fundamental Semantic Web technologies and describes the three investigated sensor data integration approaches found in literature. Sec. 5.3 explains the applied performance evaluation method. The results of this evaluation are presented in Sec. 5.4, followed by a discussion of the results in Sec. 5.5. Finally, a short conclusion is drawn in Sec. 5.6.

5.2 Sensor Data Integration Methods

Semantic Web technologies are the foundation of the Linked Factory Data and the investigated sensor data integration methods. Three different approaches were evaluated which were found in related work and applied to an (industrial) use-case. The approaches are called "Ontology Storage", "Custom Property Function", and "Ontop Framework". They are described in more detail in this section. First, the fundamentals of the Semantic Web technologies are explained.

5.2.1 Semantic Web Technologies

The vision of the Semantic Web was to annotate Web content to add semantics to it. This allows making the content more accessible to humans and computers [2]. Ontologies are used as tool to specify the meaning. An ontology can be defined as *"... a formal, explicit specification of a shared conceptualization"* [17], usually applied for a certain domain of interest. Today, various Semantic Web technologies are standardized and maintained by the W3C[1]. These technologies are also known as the Semantic Web Stack. The main technologies in place are the RDF, RDFS, OWL, and the SPARQL.

RDF [9] is used to formulate statements about resources with the help of so-called triples. A triple consists of a subject, a predicate, and an object. It represents some kind of knowledge. As an object can be used as a subject for another statement, the information is forming a graph and, thus, interlinks various resources. RDFS [3] is adding semantics by introducing class and property hierarchies, as well as the domain and range

[1] https://www.w3.org

constraints for properties. OWL [19] provides additional language constructs, which allow to define, for instance, cardinalities, value restrictions and complex class constructs. These three technologies enable the formulation of a formal knowledge base or ontology. New knowledge can be inferred from the ontology with the help of reasoning techniques.

The ontology, based on triple-statements is usually stored in a specific database, which is called triple-store. To retrieve the stored information, a query language has been standardized called SPARQL [8]. SPARQL performs pattern matching over the triple-graph and is comparable to the SQL for relational databases.

Ontologies can be applied for communication tasks, interoperability problems, as well as information sharing and re-use [18]. In particular in the engineering domain, in which various engineering disciplines have to work together using different terminologies accompanied by various data formats and schema, ODBI can be very useful. ODBI captures the implicit knowledge across the these heterogeneous data sources in an ontology to create semantic interoperability [6].

As in the manufacturing domain data is hardly available in the RDF format, mappings and transformations from existing data sources into RDF have to be performed to integrate these data into an ontology [12]. The concept of mapping data source into an ontology is named OBDA. One of such an OBDA framework is called Ontop[2] and explained in some more detail in one of the following sections.

Sometimes the terms ontology, knowledge graph and knowledge-based system are used in an intermingled way. With respect to [1], a knowledge-based system consists of two parts: a knowledge base and an inference engine. The ontology works as the knowledge base where a reasoner can be applied to infer new knowledge. Based on this definition, a knowledge graph can be defined as a knowledge-based system, with a knowledge base (ontology) and reasoning engine, as well as the integration of information from external sources [5]. As our proposed Linked Factory Data includes such external sources, we use the term knowledge graph when we refer to it.

5.2.2 Ontology Storage

One way of integrating sensor data from sensor networks into a knowledge graph is to store the sensor data inside the ontology. The sensor observations are then represented by a timestamp-value pair. The data inside an ontology is organized as a graph. This way, every new observation is enlarging the graph and also the search space for data retrieval. Such an approaches has been used, for example, to create a Linked Data for building management systems [7]. The data update and synchronization between the data stored in the triplestore and the plant has to be handled by an additional client software.

[2] https://ontop.inf.unibz.it/

Even if this approach seems reasonable and easy to implement, the scalability of this approach seems limited as the ontology is growing with every new observation. This can reduce the performance when accessing data in some cases.

5.2.3 Custom Property Function

The functionality of SPARQL is extensible, by implementing a so-called CPF. A CPF allows to execute some piece of code during the process of triple matching, which is determined by the property URI. Most SPARQL engines of Semantic Web frameworks, like Eclipse RDF4J[3] or Apache Jena[4] are supporting the implementation of such custom functions.

A CPF has also been used to integrate sensor data from an OPC UA server into an ontology [15]. Inside this CPF, an OPC UA client was implemented, which was able to connect to an OPC UA server. During a SPARQL query with a CPF, the data is retrieved from the OPC UA server. The OPC UA data stays in its normal storage and the triplestore is not growing by adding new observations over time.

The CPF approach seems to be very flexible supporting the integratation of different kinds of data sources which provide some API. Thus, a CPF allows to integrate not only OPC UA, but also e.g special time series databases or connections via Web interfaces.

To compare this approach with the other two methods, a CPF was implemented for the Apache Jena Fuseki Framework. A common relational database was used as data source. The implementation uses an RDF mapping file to specify the database connection, the related sensor data, as well as a query string. The query string is sent to the specified connection endpoint. In the case of a relational database, the query string is a SQL string. This could also be changed to access NoSQL databases. Listing 5.1 shows such an exemplified mapping for a sensor with the ID *S0* and a database connection *Connection1*.

```
@prefix ns1: <http://sic.auto.tuwien.ac.at/mappings#> .
@prefix sosa: <http://www.w3.org/ns/sosa/>.

sosa:S0 ns1:hasMapping ns1:MappingS0 .

ns1:MappingS0 ns1:hasDBConnection ns1:Connection1;
              ns1:hasSQLString "SELECT 'time', 'value'
                                FROM SensorData
                                WHERE (sensorID = 'S0')
                                ORDER BY 'time' DESC ".

ns1:Connection1 ns1:db "Sensors" ;
                ns1:passw "password" ;
                ns1:url "localhost:5432" ;
                ns1:user "username" .
```

Listing 5.1: Mapping file for CPF

[3] https://rdf4j.eclipse.org
[4] https://jena.apache.org/

It is assumed that every sensor can store its data in another database. Thus, an arbitrary amount of database connections can be defined in the mapping file, which can be associated with the mapping of a sensor.

The CPF is registered under the standard namespace and the property name *getHistValues* is assigned. As parameters for the CPF, two SPARQL variables are mandatory to store the timestamp and the corresponding value. Two additional parameters can be specified, to constrain the time interval of data retrieval. These two parameters are optional and are used to reduce the amount of data which have to be retrieved, by filtering the timestamps directly at the data source level. This improves the performance for certain types of queries. The usage of the CPF inside a SPARQL query is shown in Tab. 5.2.

5.2.4 Ontop Framework

Ontop is an open-source OBDA framework. The concept of OBDA is to use ontologies as interface for data access [13]. OBDA allows to abstract the data sources schema details and makes information access easier for users. Usually the data source are relational databases which are queried over SQL.

However, mappings between an conceptual domain view in terms of an ontology and the database schema have to be defined. A formal definition of OBDA is given in [20], by defining a relational database source $\mathcal{D}$ which conforms to the data source schema $\mathcal{S}$, an ontology $\mathcal{O}$ and a mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{O}$. The OBDA specification is defined as $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$. The ontology $\mathcal{O}$ provides the conceptual view of the data.

Ontop enables the virtual integration of a data source into an RDF graph without transforming and materializing the relational data. This is performed by query re-writing techniques [4]. Therefore, mappings between the database and the ontology have to be defined. The mapping file for the presented use case can be found in Listing 5.2. Ontop is also compliant to W3C recommendations, like SPARQL, but can only connect to one SQL compliant database.

5.3 Evaluation Method

The performance evaluation was carried out on a Virtual Machine (VM) which has four 64-bit Intel Cores @ 2.6 GHz and uses CentOS Linux 7.6.1810 as operating system.

To provide a SPARQL endpoint and persistent RDF storage, an open-source Semantic Web framework was used. As the CPF was already implemented for Apache Jena Fuseki v3.12.0, it was also used as SPARQL endpoint for the ontology storage.

Ontop v1.18.1 was chosen for OBDA because it is a very popular open-source framework, well documented, features active development within the last year and provides the possibility to access external relational databases. Alternatives, which were presented in [11] could not fulfill all of these requirements. Additionally, Ontop was already proposed in other publications as a framework for accessing time series data (cf. [12]).

To store the test sensor data, PostgreSQL v11.5 was installed as relational database management system, because it is supported by Ontop. For our test case, we used a very simple database schema with only one table. The database table is shown in Table 5.1. A sensor observation is represented by a single row in this table. Every observation has a unique index and consists of a sensor ID, timestamp and the corresponding value.

Table 5.1: Sensor Data Table in relational database

Index	**SensorID**	**Time**	**Value**
0	S1	2019-08-01 00:00:00	5.56
1	S2	2019-08-01 00:00:00	25.36
...	...	...	...
10	S1	2019-08-01 00:15:00	78.36
11	S2	2019-08-01 00:15:00	68.75
...	...	...	...

As test data, random floating point values were generated and assigned with a timestamp. The timestamps start with 2019-08-01 00:00 and have a resolution of 15 minutes. The generated random floating point values are in a range between 0 and 100.

To evaluate the data access performance of all three sensor data integration approaches, eight SPARQL queries were specified (Table 5.2) and executed. The query Q7* is a modification of Q7 to investigate a special behavior of the CPF approach in more detail. A query can search over three different properties of the observation, which are the *SensorID*, the *Time* and the *Value*. These three search dimensions are also indicated in the query list, shown in Table 5.2.

For the test procedure the amount of sensors as well as the amount of values per sensor were varied. The amount of sensors were set to 10, 100, 250, and 500. The amount of values per sensor were set to 100, 1000, 2500, 10.000, and 100.000. For every combination of these two values, the execution time for all eight SPARQL queries is evaluated. This procedure was repeated for all three presented sensor data integration approaches.

To represent the sensor data inside the knowledge graph, the Semantic Sensor Network (SSN) ontology was used, which is a recommendation of the W3C. For the evaluation, only view concepts from the core part of the SSN, the SOSA ontology, are used depicted in Fig. 5.2.

To evaluate the ontology storage approach, an RDF file with individuals was created. Every observation was connected with the corresponding sensor via the SOSA properties (*madeBySensor* and *madeObservation*). For every test case, the Fuseki server was restarted and the RDF data was loaded. Afterwards, the queries were sent to the SPARQL endpoint and the execution time for every SPARQL query was measured.

Table 5.2: Queries

	SPARQL Query		Search Dimension		
Nb	**Standard**	**Custom Property Function**	**SensorID**	**Time**	**Value**

Q1	SELECT ?time ?value
	WHERE {
	sosa:S1 sosa:madeObservation ?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime
	?time}
		SELECT ?time ?value
		WHERE {
		sosa:S1 :getHistValues(?time ?value)
		}			

Q2	SELECT ?time ?value
	WHERE {
	sosa:S2 sosa:madeObservation ?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime ?time.
	Filter(?time >= "2019-08-01T07:00:00"^^xsd:dateTime
	&& ?time <= "2019-08-01T10:00:00"^^xsd:dateTime)}
		SELECT ?time ?value
		WHERE {
		sosa:S2 :getHistValues(?time ?value
		"2019-08-01T07:00:00" "2019-08-01T10:00:00")
		}
				X	

Q3	SELECT ?time ?value
	WHERE {
	sosa:S3 sosa:madeObservation ?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime ?time.
	Filter(?value >96.1)}
		SELECT ?sensor ?time ?value
		WHERE {
		sosa:S3 :getHistValues(?time?value)
		Filter(?value >96.1)
		}
					X

Q4	SELECT ?sensor ?time ?value
	WHERE {
	?sensora sosa:Sensor.
	?sensor sosa:madeObservation ?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime?time.
	Filter(?value >99.5)}
		SELECT ?sensor ?time ?value
		WHERE {
		?sensor a sosa:Sensor.
		?sensor :getHistValues(?time ?value)
		Filter(?value >99.5)
		}
			X		X

Q5	SELECT ?time ?value WHERE { sosa:S5 sosa:madeObservation ?obs. ?obs sosa:hasSimpleResult ?value. ?obs sosa:resultTime?time Filter(?value >= 50 && ?time >= "2019-08-01T07:00:00"^^xsd:dateTime && ?time <= "2019-08-01T10:00:00"^^xsd:dateTime) } OrderBy ?time	SELECT?time ?value WHERE { sosa:S5 :getHistValues(?time ?value "2019-08-01T07:00:00" "2019-08-01T10:00:00") Filter(?value >= 50) }OrderBy ?time		X	X

Q6	SELECT ?sensor ?time ?value
	WHERE {
	?sensor a sosa:Sensor.
	?sensor sosa:madeObservation ?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime ?time.
	Filter(?time >= "2019-08-01T10:00:00"^^xsd:dateTime
	&& ?time <= "2019-08-01T11:00:00"^^xsd:dateTime)}
		SELECT ?sensor ?time ?value
		WHERE {
		?sensor a sosa:Sensor.
		?sensor :getHistValues(?time ?value
		"2019-08-01T10:00:00"
		"2019-08-01T11:00:00")
		}
			X	X	

Q7	SELECT ?sensor ?time ?value
	WHERE {
	?sensor a sosa:Sensor.
	?sensor sosa:madeObservation?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime ?time.
	Filter(?value >99.5 &&
	?time >="2019-08-01T07:00:00"^^xsd:dateTime)}
		SELECT ?sensor ?time ?value
		WHERE {
		?sensor a sosa:Sensor.
		?sensor :getHistValues(?time ?value
		"2019-08-01T07:00:00") Filter(?value >99.5)
		}
			X	X	X

Q7*	SELECT ?sensor ?time ?value
	WHERE {
	?sensor a sosa:Sensor.
	?sensor sosa:madeObservation?obs.
	?obs sosa:hasSimpleResult ?value.
	?obs sosa:resultTime ?time.
	Filter(?value >99.5 &&
	?time >="2019-08-01T07:00:00"^^xsd:dateTime &&
	?time <="2019-08-01T07:30:00"^^xsd:dateTime)}
		SELECT ?sensor ?time ?value
		WHERE {
		?sensor a sosa:Sensor.
		?sensor :getHistValues(?time ?value
		"2019-08-01T07:00:00" "2019-08-01T07:30:00")
		Filter(?value >99.5)
		}
			X	X	X

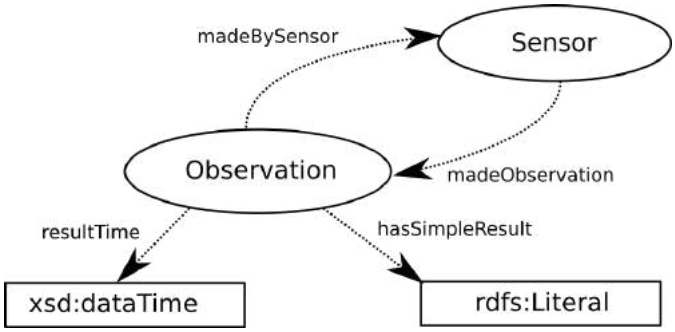


Figure 5.2: Overview of used SOSA classes and properties

To access the data through CPF, an RDF mapping file was created for every test case. The mapping file defines a database connection for every sensor. An example of such an CPF mapping file is shown in Listing 5.1.

Additionally to the CPF mapping file, an individual was created in the ontology for every sensor. To test the CPF approach, the Fuseki server and the database server holding the test data were started. Only the SOSA ontology and the individuals were loaded into the triplestore. Then all eight SPARQL queries, as shown in Table 5.2, were executed. As the CPF defines a special property, the CPF queries are slightly different than the standard SPARQL queries.

To test the Ontop approach, a proper mapping file was created, which is partly shown in Listing 5.2. It describes how to map the data from the database (source) into certain concepts of the ontology (target). Listing 5.2 illustrates the mapping definition for the *Observation* and the *Sensor* class including their properties. This file is loaded at the startup of Ontop. The database connection details are also configured in a separate file. Afterwards, only the data in the relational database were altered and all eight SPARQL queries were executed.

```
[MappingDeclaration] @collection [[
  mappingId   urn:Observation
  target      :Observation_{index} a :Observation;
                        :hasSimpleResult {value};
                        :resultTime {time};
                        :madeBySensor :{sensorID}.

      :{sensorID} :madeObservation :Observation_{index}.
  source      SELECT "index", sensorID, time, value
              FROM SensorData

  mappingId   urn:SensorID
  target      :{sensorID} a :Sensor.
  source      SELECT Distinct sensorID
              FROM SensorData
]]
```

Listing 5.2: Ontop mapping file

As caching is performed in the relational databases as well as the triplestore, the execution measurement was repeated several times and also the execution order of the queries was randomly changed. Thus, every measurement was repeated 20 times. Afterwards, the distribution of the execution time was analyzed. For further investigations, the mean execution time was calculated and used for the analysis.

To verify that no server error occurred during the query execution, which would lead to wrong measurements of the data retrieval process, all query results were automatically checked, if the retrieved data were as expected.

5.4 Query Performance Evaluation

In this section, the results of two expressive and complementary test cases are presented. Both cases have the same amount of observations (1.000.000) which are stored in the relational database or triplestore. They only differ by the amount of sensors and observations per sensor:

- *Test Case A*: 10 sensors with 100.000 observations per sensor
- *Test Case B*: 500 sensors with 2000 observations per sensor

Fig. 5.3 shows the mean execution time for the queries Q1, Q2, Q3, Q5 for *Test Case A*. Q1 represents the base line, because all generated test data are loaded. For the queries Q2, Q3 and Q5, the sensor name is specified in the SPARQL query, which means that a search must only be performed over the timestamps and values.

As shown in Fig. 5.3, the CPF mean execution time of Q3 is much higher than for the query Q2 and Q5. The reason for this behavior is the additional filtering of the sensor values in Q3. This filter function is performed by the SPARQL engine over all returned sensor values. Thus, the CPF returns the whole data stored in the database for the sensor *S3* to the SPARQL endpoint. After that, the filtering is done by the SPARQL engine. Therefore, it takes about the same amount of time as to retrieve and filter the data directly from the triplestore.

Ontop performs query re-writing, which means that the value filter is already applied at the relational database in SQL. Thus, the filtering is performed much faster and has no big influence on the query execution time at all.

In query Q5, only data between a certain time interval is retrieved from the relational database. In case of CPF, filtering of timestamps is performed at the database and the amount of returned is much smaller than in Q3. Thus, the value filtering is performed much faster by the SPARQL engine and the execution time of the CPF for Q5 is smaller.

The test case shows, that value filtering can have a significant impact on the CPF performance, if large amount of data is returned. Thus, an additional query Q7* was introduced to investigate this behavior in more detail.

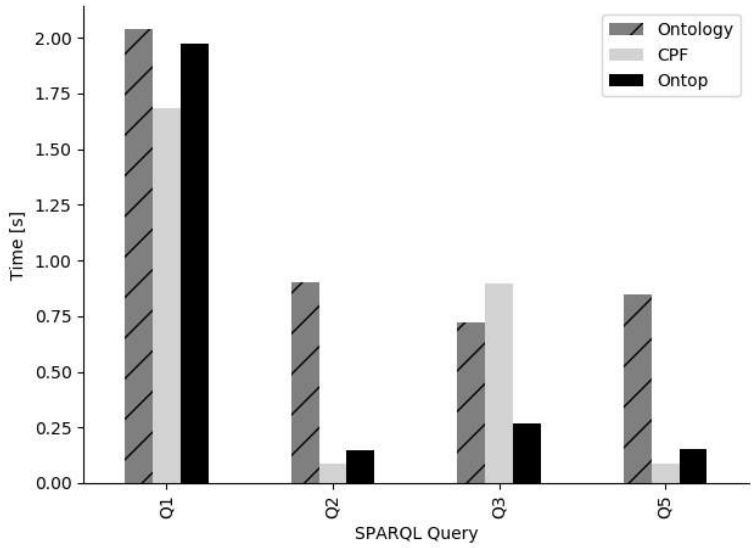


Figure 5.3: *Test Case A*: Mean query execution time for Q1, Q2, Q3, and Q5

The queries Q4, Q6 and Q7 do not specify a particular sensor in their query. This means, a search has to be performed over all available sensors. Fig. 5.4 shows the mean execution times for these queries. The performance of the CPF for Q4 an Q7 is worse than for Q6, because of the already mentioned additional value filtering. This is demonstrated by the execution of query Q7*. This query is a modification of query Q7, which adds an additional time constraint (time $\leq$ 2019-08-01T07:30:00). As the CPF performs filtering of the timestamps directly at the database, the amount of retrieved data is reduced. This constraint reduces the number of returned values from 999.720 to 1.500. Therefore, the value filtering for the CPF at the SPARQL engine is performed much faster. This reduces the execution time for CPF significantly.

It is also shown in Fig. 5.4, that the additional time constraint in Q7* increases the mean execution time for the ontology storage by about 4 seconds. To investigate this behavior and exclude any connection problems during the procedure, the distribution of the access time was examined shown in Fig. 5.5. The box-plot shows a symmetrical distribution around the mean value. This implies that no outliers caused that increase in time, but the additional filter parameter which has to be handled by the SPARQL engine.

The queries Q1, Q2, Q3 and Q5 only retrieve data for one particular sensor. Therefore, a variation in the amount of sensors – as performed for *Test Case B* – has no influence on the query execution time.

Fig. 5.6 shows the results for the queries Q4, Q6 and Q7 of *Test Case B*. For the ontology storage approach and Ontop, the variation has no influence, as the overall amount of observations stay the same. But, the performance of the CPF decreases significantly. This is caused by the CPF design approach. The CPF assumes that every sensor can be

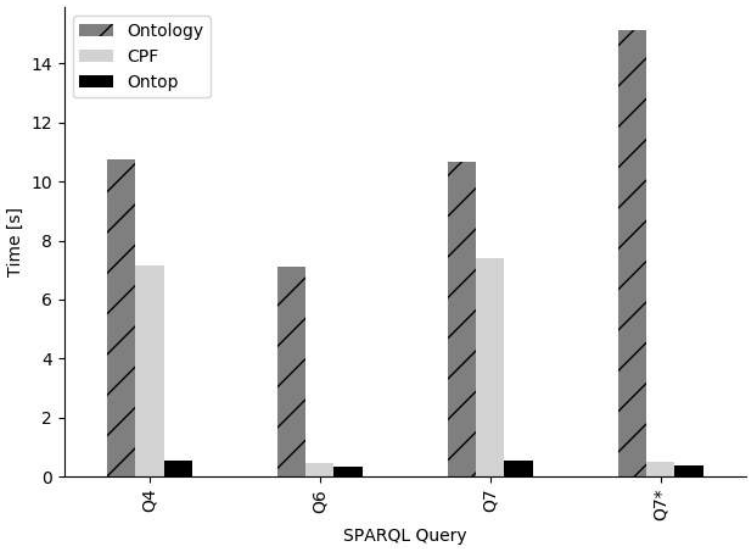


Figure 5.4: *Test Case A*: Mean query execution time for Q4, Q6,Q7, and Q7*

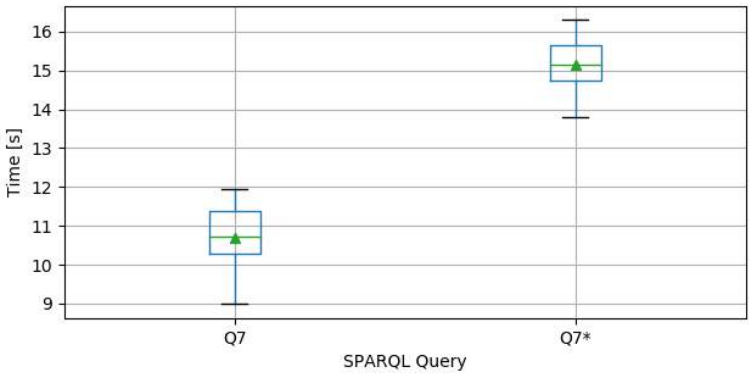


Figure 5.5: *Test Case A* - Ontology storage method: Execution Time

stored in another database. Therefore, the CPF has to iterate over the sensors in the SPARQL result set and performs separated SQL queries for every sensor. This reduces the query execution performance drastically.

Ontop supports only one database connection and applies an optimized query rewriting. Thus, only one SQL query is necessary facilitating very fast data retrieval.

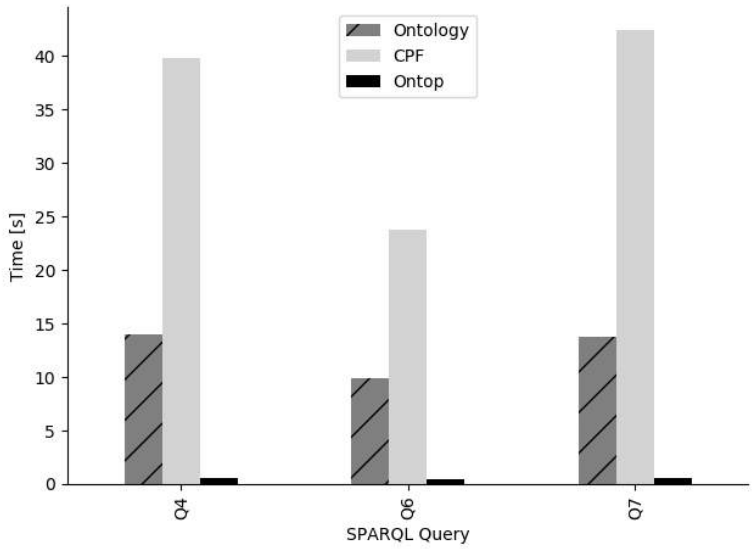


Figure 5.6: Test Case B: Mean query execution time

5.5 Discussion

The results of the performance evaluation show, that the storage of time series data in the ontology leads to performance issues, if the time series grows over time. The ontology storage method took much more time for the most queries (Q1, Q2, Q5, Q6, Q7, Q7*) than the other two approaches. Thus, these method can not be recommended for most use cases. Nevertheless, for implementation of prototypes or use cases with only a small amount of data, it seems to be a feasible solution as long as around 100.000 observations are taken into account under the outlined test conditions. If the amount of observations is constantly growing over time, other approaches should be preferred.

The CPF is a very flexible approach to include different kinds of data sources, e.g. OPC UA or NoSQL databases. Nevertheless, the CPF implementation has two limitations. One drawback is the filtering over sensor data values, as is not directly performed at relational database but at the SPARQL endpoint. This drastically increases the query execution time, if large amount of data is returned from the database. This can be avoided, if the CPF is extended with additional function parameters to perform filtering directly at the data source. This issue will be subject for further investigations and was not in the scope of the present work.

The second limitation of the CPF is caused by its flexible design approach allowing every sensor value to be stored in another relational database. This feature supports the integration of more than one relational database into a knowledge graph, but also effects that a SQL query is sent to the database for every sensor in the SPARQL result set. This iteration leads to a performance reduction for use cases where a query has to search over a large amount of sensors. If the sensor data are stored in the same database, the CPF performance could be improved by optimizing the SQL queries. This could be considered for future implementations.

The query re-writing process, performed by the Ontop framework, leads to very fast data access. In most of the test cases, Ontop has the shortest query execution time. Thus, Ontop is able to scale well for large amount of data. A drawback of the Ontop framework is its limitation of accessing only a single relational database. Additionally, this database has to be SQL compliant. Thus, a connection to NoSQL databases is not possible. The disadvantage of only one database connection could be solved by a workaround, using data virtualization systems like *Teiid*[5]. Such systems allow to connect multiple data sources to one virtual database. Also, this approach would demand for further tests out of scope for the present work.

5.6 Conclusion

To find answers for the stated RQ1, the performance evaluation and comparison of three different approaches suitable for sensor data integration into a knowledge graph was carried out.

To answer RQ2, the characteristics and their resulting limitations of all three methods as discussed in the previous section were addressed. To conclude the conditions for their application, it can be stated that, if the sensor data is already stored externally in an SQL compliant relational database, the Ontop framework should be preferred. Only if the data sources are not SQL-compliant, like NoSQL databases or data held in OPC UA servers, the CPF can be applied. Improvements of the CPF should be made to avoid performance reduction for certain types of queries.

If more than one SQL database should be integrated, CPF is a feasible solution or Ontop with additional third party data virtualization software could be used.

Storing time series data directly in the ontology should be avoided. Only for little data, up to about 100.000 observations, it is still quite efficient under the presented test conditions. This means, that all queries could be executed under one second.

Fig. 5.7 sums up the stated conditions by means of a design support decision tree.

As future work, the already suggested improvements for the CPF implementation will be made to overcome its discussed shortcomings. A CPF will be applied to a real industrial

[5] http://teiid.io/

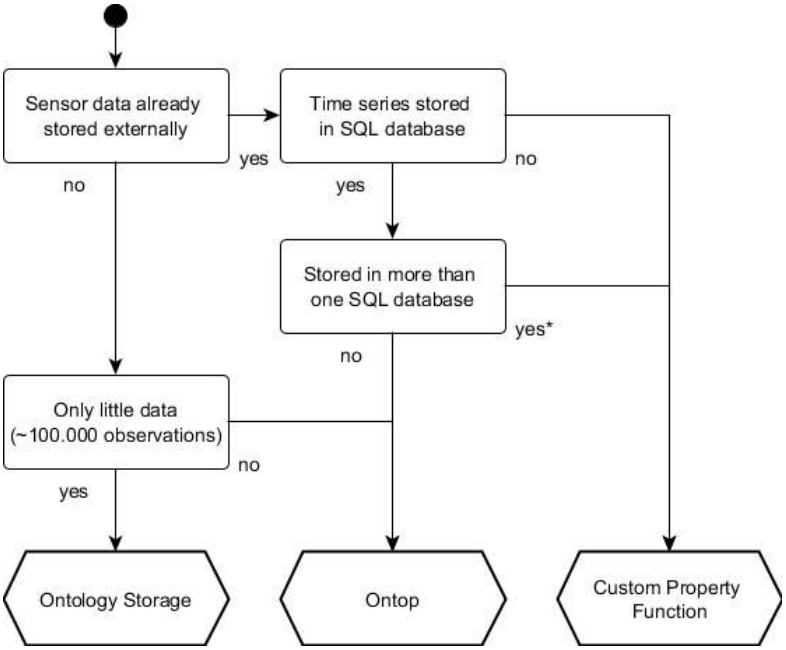


Figure 5.7: Design support decision tree

use case, to interlink non-SQL-compliant data sources with other information inside the Linked Factory Data.

Acknowledgment

This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) for the „BIM4BEMS - Building Information Modeling for Building Energy Management Systems“ project under the contract number 854677, as well as the cooperative doctoral school „SIC! - Smart Industrial Concept!“.

References

[1] R. Akerkar and P. Sajja. *Knowledge-Based Systems*. Jones & Bartlett Learning, 2010.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. „The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities“. In: *Scientific American* (2001), pp. 1–10.

[3] Dan Brickley and R.V. Guha. *RDF Schema 1.1. W3C Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/rdf-schema/.

[4] Diego Calvanese et al. „Ontop: Answering SPARQL Queries over Relational Databases“. In: *Semantic Web Journal* (2017). URL: https://github.com/ontop/ontop-examples/.

[5] Lisa Ehrlinger and Wolfram Wöß. „Towards a definition of knowledge graphs“. In: *CEUR Workshop Proceedings* 1695 (2016). ISSN: 16130073.

[6] Fajar J Ekaputra et al. „Ontology-Based Data Integration in Multi-Disciplinary Engineering Environments: A Review“. In: *Open Journal of Information Systms* 4.1 (2017), pp. 1–26. ISSN: 2198-9281.

[7] Andreas Fernbach, Igor Pelesic, and Wolfgang Kastner. „Linked Data for Building Management“. In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. Florence, Italy: IEEE, 2016, pp. 6943–6945. ISBN: 9781509034741. DOI: 10.1109/IECON.2016.7793781.

[8] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language. W3C Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2013. URL: https://www.w3.org/TR/sparql11-query/.

[9] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation*. Tech. rep. World Wide Web Consortium, 2004. URL: https://www.w3.org/TR/rdf-concepts/.

[10] Gunther Koschnick. *Industrie 4.0: Smart services*. Tech. rep. July. Frankfurt am Main, Germany: German Electrical and Electronic Manufacturers' Association, 2016, pp. 13–14.

[11] Franck Michel, Johan Montagnat, and Catherine Faron-Zucker. „A survey of RDB to RDF translation approaches and tools“. In: *Informatique, Signaux Et Systèmes* (2014), p. 23. URL: https://hal.archives-ouvertes.fr/hal-00903568/.

[12] Benjamin Mörzinger et al. „A large-scale framework for storage, access and analysis of time series data in the manufacturing domain“. In: *Procedia CIRP* 67.January (2018), pp. 595–600. ISSN: 2212-8271. DOI: 10.1016/j.procir.2017.12.267. URL: http://dx.doi.org/10.1016/j.procir.2017.12.267.

[13] Antonella Poggi et al. „Linking data to ontologies“. In: *Journal on Data Semantics X*. Ed. by Stefano Spaccapietra. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 133–173. ISBN: 3540776877. DOI: 10.1007/978-3-540-77688-8_5.

[14] Gregory A. Silver, Osama Al-Haj Hassan, and John A. Miller. „From domain ontologies to modeling ontologies to executable simulation models“. In: *Proceedings - Winter Simulation Conference* (2007), pp. 1108–1117. ISSN: 08917736. DOI: 10.1109/WSC.2007.4419710.

[15] Gernot Steindl, Thomas Früwirth, and Wolfgang Kastner. „Ontology-Based OPC UA Data Access via Custom Property Functions“. In: *24th Interantional Conference on Emerging Technologies and Factory Automation*. Zaragoza, Spain, 2019.

[16] Gernot Steindl, Bernhard Heinzl, and Wolfgang Kastner. „A Novel Ontology-Based Smart Service Architecture for Data-Driven Model Development“. In: *eKNOW 2019 - The Eleventh International Conference on Information, Process, and Knowledge Management*. Ed. by Jaime Lloret Mauri. Athens, Greece, 2019, pp. 26–27.

[17] Rudi Studer, V Richard Benjamins, and Dieter Fensel. „Knowledge Engineering: Principles and methods“. In: *Data & Knowledge Engineering* 25 (1998), pp. 161–197.

[18] Mike Uschold and Michael Gruninger. „Ontologies : Principles , Methods and Applications“. In: *The Knowledge Engineering Revie* 11 (1996), pp. 93–136. DOI: 10.1017/S0269888900007797.

[19] W3C OWL Working Group. *OWL 2 Web Ontology Language. Document Overview*. Tech. rep. World Wide Web Consortium (W3C), 2012. URL: https://www.w3.org/TR/owl2-overview.

[20] Guohui Xiao et al. „Ontology-based data access: A survey“. In: *IJCAI International Joint Conference on Artificial Intelligence* 2018-July.December (2018), pp. 5511–5519. ISSN: 10450823.

CHAPTER 6

Ontology-Based Model Identification of Industrial Energy Systems

Publication: *G. Steindl and W. Kastner, "Ontology-Based Model Identification of Industrial Energy Systems," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), 2020, pp. 1217-1223, doi: 10.1109/ISIE45063.2020.9152386.*

Abstract: The paper presents a novel approach for ontology-based model identification of industrial energy systems. Therefore, an ontology is designed which is extended by formal rules to automatically identify causal relations between the components' input and output quantities. Also, available sensor data are mapped with rules to these quantities to enable direct data access for model identification. The proposed approach is evaluated for a use case of an industrial heating process. The results show, that the required information is available in the extended ontology to automatically generate dynamic models of the plant equipment.

6.1 Introduction

With the ongoing fourth industrial revolution, inter-networked CPSs are now going to change the industrial environment. A key challenge in this context is resource efficiency [1]. Therefore, energy efficiency in the industry has to be addressed to reduce greenhouse gas emission and facilitate the sustainable energy transition. To reach these goals, retrofitting of existing plants is important and also challenging.

To avoid high investment costs for new plant components, novel predictive control strategies for existing equipment can be applied. These strategies can optimize the

plant's operation and enable coupling of the plant's production with the energy market. Prerequisites for such control strategies are accurate models of the plant's components to predict their dynamic behavior and meet given constrains or detect faulty behavior.

For retrofitting plants, process data from various sensors are usually already available. This facilitates a data-driven model development approach. But, data alone is not enough. Additionally, the overall model development process needs knowledge, which is often implicit, scattered, and unfortunately seldom well documented. This requires the model developer to rely on domain experts [14]. Thus, model identification is often the most time-consuming task in the process of model-based control development[10].

To reduce this effort and partly automate the model development step, knowledge about the plant and its components has to be available in a machine-readable format. This implies the usage of formal knowledge representation, which also facilitates the creation of so-called Smart Data.

Smart Data integrates various data sources which are correlated and analyzed to use them for decision making and action processing [8]. It serves as a foundation for Smart Services in the context of Industry 4.0 [18].

This leads to the idea of a Smart Data Service for data-driven model development, which is based on formal knowledge representation, like an ontology or knowledge graph. The Smart Data Service can integrate various data sources and once done acts as a single point of information for model developers as well as other (software) agents. Additional information (e.g., about data quality, operational anomalies, sensor faults) are also valuable for the model developer and later on the data analysts or operators of a plant.

This paper focuses on the ontological knowledge representation, which is the foundation for an automatic model identification of components in industrial energy systems. In such an ontology, information about the components and equipments of plants as well as their causal relations is modeled. Additionally, related sensor data are mapped to the component models to facilitate a semi-automatic model identification process.

The remainder of the paper is structured as follows: Section 6.2 gives a short introduction to ontologies and the Semantic Web Stack, which is heavily used in this work. Also, some literature in the context of automatic model identification is presented. Section 6.3 explains the applied method to develop a plant ontology and automatically generate causal relations inside this ontology. Afterward, the proposed method is applied and evaluated for an industrial use case of a thermal heating process in Section 6.4. In Section 6.5, conclusion are drawn and ideas for future work are briefly presented.

6.2 State-of-the-Art

This section gives a short introduction to some of the technologies from the Semantic Web Stack. Also, some literature in the context of automatic model identification is discussed, which the ideas of this paper are based on.

An ontology can be defined as "...a formal, explicit specification of a shared conceptualization" [19]. To specify such a formal ontology, a so-called Semantic Web Framework has been established, which consists of several technologies. These technologies are standardized by the W3C[1].

The RDF [11] is designed for modeling information by formulating statements, so-called triples. A triple consists of a subject, a predicate and an object, which are identified by their URI. An object can be used as a subject in another statement. This allows to interlink statements in a graph-based fashion. The RDFS [2] adds additional semantics to RDF, by introducing class and property hierarchies as well as domain and range constraints for properties. The OWL [20] provides even more language constructs, like cardinalities or value restrictions, to formulate more complex semantic expressions. These three standards are the base technologies for expressing an ontology, which is also sometimes called a knowledge base. With additional reasoning capability, new knowledge can be inferred from such an ontology. If also external information is integrated, it is called a knowledge graph [4].

To retrieve information from an ontology or knowledge graph, the SPARQL [7] were specified as W3C Recommendation. It is comparable to the SQL for relational database management systems. With SPARQL version 1.1 an update functionality was introduced, which enables creating, updating and removing RDF graphs in a graph store [6]. This enables the formulation of complex rules, which can create new triples inside an ontology.

Semantic information has already been used for automatic model identification. In [12], an automatic thermal model creation approach for thermal building models is presented. Therefore, information from an existing BIM is retrieved. With that information, a simplified thermal RC-model was generated. BIM already contains static building information like geometry and thermal properties of the model elements. Dependencies between quantities are only implicit modeled in this context.

The idea of using OWL ontologies to describe the structure of an energy system can be found in [15]. The ontological domain models were constructed and dependencies between objects modeled. This was combined with statistical anomaly detection algorithms, to improve data quality by cleaning sensor data of a power generation facility. Duplicated and neighboring sensors were identified by searching for equipment of the same type and deployed at the same location. The relations in the ontology were only statically modeled and could not be identified by rules.

The authors of [3] used an ontology-based BIM, facilitating the BASont ontology [13]. They presented a qualitative, symbolic, knowledge-based fault propagation approach for building automation systems. The scope of this work was fault detection and diagnosis, thus they also specified causal relations, based on SPARQL rules.

A similar approach is followed in this paper, but focusing on using that information to automatically create dynamic component models based on available sensor data. The

[1] https://www.w3.org

required information for such a process is retrieved from the created ontological knowledge base.

6.3 Methods

The following subsections describe the method which is followed in this paper. First, an ontology is designed to capture the essential plant information in a formal, machine-readable format. Second, based on the knowledge captured by this so-called Plant Equipment, Topology and Instrumentation Ontology (PETIont), formal rules are defined to automatically generate information about the virtual representation of the plant equipment and the causal relations between the components. The last step is the mapping of available sensor data to the related virtual entity. This enables direct data access of sensor data through the ontology and the detection of missing or redundant sensors inside the plant's topology. The procedure is described in more detail in the following two subsections.

6.3.1 PETIont - Plant Equipment, Topology and Instrumentation Ontology

A thermohydraulic plant ontology is designed to capture the required knowledge about the installed equipment, the plant's topology as well as the available instrumentation (sensors and actuators). The main concepts of the developed ontology are partly depicted in Fig. 6.1. The available sensor data can be directly accessed through OBDA. Further details on OBDA and a comparison of various sensor data integration methods into knowledge graphs can be found in [17].

The main information sources for the ontology are diagrams, commonly used in plant engineering, like a P&I diagram. The contained information in such diagrams and the various types of plant equipment are specified in EN ISO 10628 [5]. A simple P&I diagram is depicted in Fig. 6.2a, which also includes the instrumentation of the plant, based on IEC 62424 [9]. This standard defines so called Process Control Engineering (PCE) requests to represent sensor and actuator functionality. The upper part of the PETIont, which has a solid border in Fig. 6.1, captures the essential information from the P&I diagram and reuses the terminology from the above-mentioned standards.

Inside PETIont, the plant equipment is divided into two sub-classes. *Thermal Equipment* changes only thermal states inside the plant, whereas *Hydraulic Equipment* changes only hydraulic states. Usually, plant equipment will be both, but in some cases, one aspect can be neglected. As an example, the temperature increase of the air caused by a fan is usually ignored. Thus, a fan can be modeled most of the time as a hydraulic component. Subtypes are defined for fans, electric heaters, heat exchangers, vessels and can be extended in the future.

The ontology defines properties to describe the interconnection between the plant equipment. These connections can be a mass flow (*hasMassFlowTo*) or an information flow

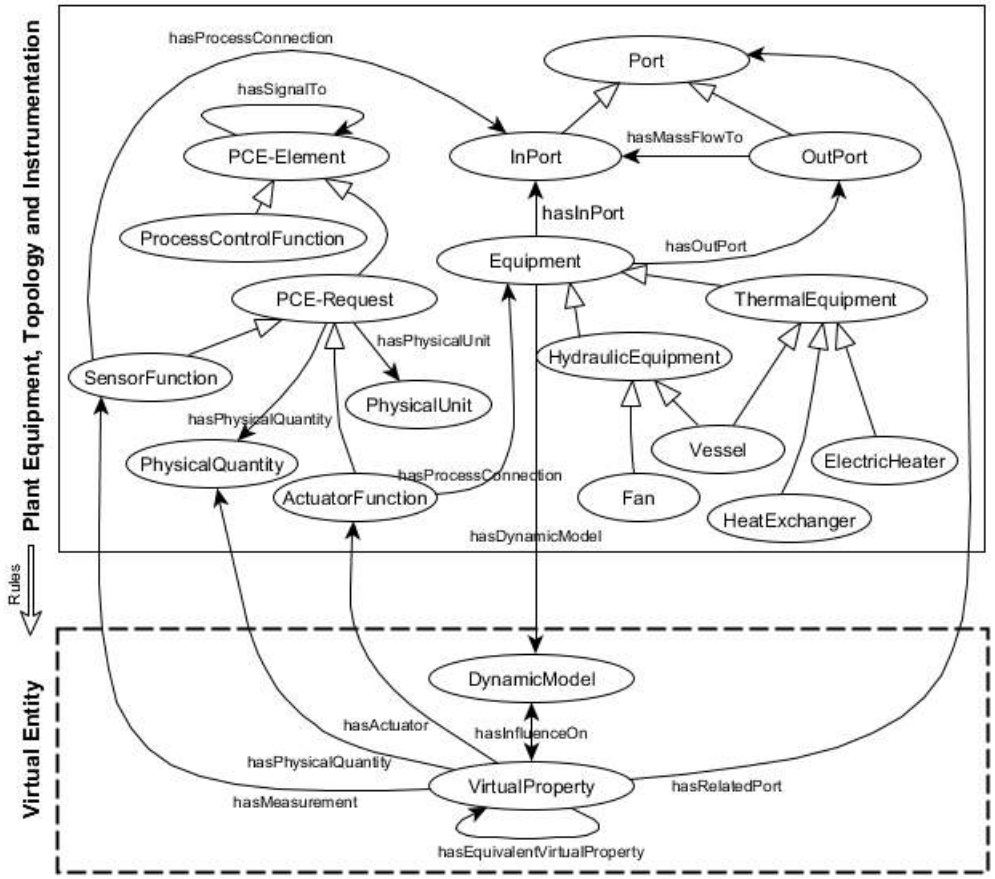


Figure 6.1: Main concepts of the *Plant Equipment, Topology and Instrumentation Ontology (PETIont)*.

(*hasSignalTo*). Mass flow is transported via pipes, whereas signals are typically used to connect process control functions with PCE requests.

The sensor functionality of PCE requests provide data for the identification of the equipment models. Additional semantic sensor information about the physical quantity and measurement unit is present in the ontology. This information is used for the mapping of sensors to their virtual representation and to automatically detected different measurement scales, e.g., Celsius and Kelvin.

The part of PETIont which is surrounded by the dashed rectangle in Fig. 6.1 is reasoned automatically based on the information from the upper part of the ontology and SPARQL rules (explained in the next subsection). This part of the ontology forms a virtual entity, representing the behavior of the physical plant equipment. The *Dynamic Model* concept represents some kind of mathematical model for plant equipment which will be identified from data. It is related to some *Virtual Properties*, which are quantities that have an influence on or are influenced by the dynamic behavior of the plant equipment. These

Virtual Properties represent the inputs and outputs of the model. They are quantities, like mass flow rates or temperatures which can correspond to real sensor data but can also be non-measurable, like a set point or the value of a manipulated variable of a controller. The term *"Virtual Property"* is used to indicate that the property belongs to the virtual entity of the plant equipment. As an example, consider a heater model, where the outlet temperature is influenced by the heater's set point, the inlet temperature, and the mass flow through the heater. All these quantities are represented as *Virtual Properties* in the ontology. The generated *Virtual Properties* are connected to PCE requests, if available. This enables direct data access for the model identification task later on.

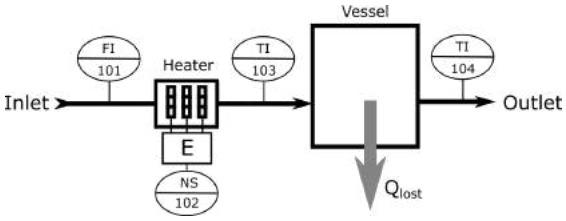
6.3.2 Rule-Based Virtual Entity Creation

The general idea of the rule-based virtual entity creation is depicted in Fig. 6.2. The information from the P&I diagram, which is already captured in the upper part of the ontology, is the starting point. The additional expert knowledge, which is needed to create the virtual entity with the dynamic models of the equipment, is formulated as SPARQL rules and applied in three steps:

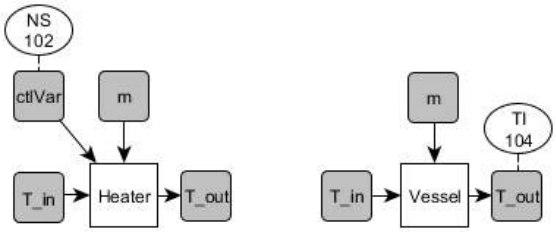
Step 1 - Creating Dynamic Model with Virtual Properties

The expert knowledge about the plant equipment is encoded in SPARQL rules, defining the input and output properties of the equipment's *Dynamic Model*. The related *Virtual Properties* are created and connections to PCE requests with actuator functionality are set based on these rules (Fig. 6.2b).

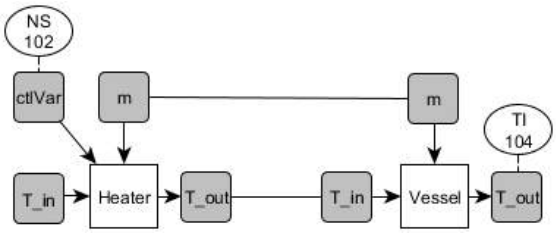
Rules for a fan, an electric heater, a heat exchanger as well as a vessel are defined. As an example, the rule for a *Vessel* type with one inlet and one outlet is presented in Listing 6.1. This rule creates a *Dynamic Model* concept and its related *Virtual Properties* for the input and output temperature as well as the mass flow. The names for the concepts are created from the URI of the *Vessel* instance. These *Virtual Properties* are connected to the equipment's *Dynamic Model*, with the information about the causal relation (*hasInfluenceOn* or *isInfluencedBy*). The *Virtual Properties* are then related to the input or output ports of the plant equipment.



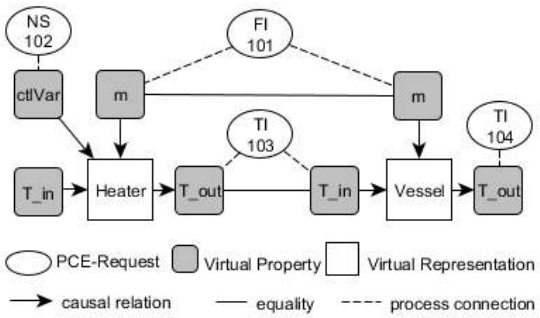
(a) Step 0 - Essential information about the plant in the P&I diagram



(b) Step 1 - Creating *Dynamic Models* with *Virtual Properties* and related PCE requests (actuators)



(c) Step 2 - Identify equivalent *Virtual Properties*



(d) Step 3 - Associate *Virtual Properties* with PCE requests (sensors)

Figure 6.2: Rule-based creation of the causal relations in the PETIont

```
PREFIX : <http://auto.tuwien.ac.at/sic/PETIont#>

INSERT{
#create dynamic model
  ?dynModelURI a :DynamicModel.
  ?comp :hasDynamicModel ?dynModelURI.

#create virtual properties
  ?TinURI a :VirtualProperty.
  ?TinURI :hasPhysicalQuantity :Temperature.
  ?ToutURI a :VirtualProperty.
  ?ToutURI :hasPhysicalQuantity :Temperature.
  ?mURI a :VirtualProperty.
  ?mURI :hasPhysicalQuantity :MassFlowRate.

#connect properties with dynamic model and related ports
  ?TinURI :hasInfluenceOn ?dynModelURI.
  ?dynModelURI :isInfluencedBy ?TinURI.
  ?TinURI :hasRelatedPort ?inPort.
  ?inPort :hasRelatedVirtualProperty ?TinURI.

  ?mURI :hasInfluenceOn ?dynModelURI.
  ?dynModelURI :isInfluencedBy ?mURI.
  ?mURI :hasRelatedPort ?inPort.
  ?inPort :hasRelatedVirtualProperty ?mURI.

  ?dynModelURI :hasInfluenceOn ?ToutURI.
  ?ToutURI :isInfluencedBy ?dynModelURI.
  ?ToutURI :hasRelatedPort ?outPort.
  ?outPort :hasRelatedVirtualProperty ?ToutURI.
} WHERE {
  ?comp a :Vessel;
  :hasInPort ?inPort.
  :hasOutPort ?outPort.

  BIND(uri(str(?comp)+"_Tin") AS ?TinURI).
  BIND(uri(str(?comp)+"_m") AS ?mURI).
  BIND(uri(str(?comp)+"_Tout") AS ?ToutURI).
  BIND(uri(str(?comp)+"_DynModel") AS ?dynModelURI)
}
```

Listing 6.1: SPARQL rule for constructing the *Dynamic Model* and its *Virtual Properties* for a *Vessel* type.

Step 2 - Identify equivalent *Virtual Properties*

After the *Dynamic Model* with its *Virtual Properties* is created for every component, they have to be connected inside the ontology in case they are equivalent. Equivalent means, that they represent the same property of the plant system (Fig. 6.2c).

Based on the flow information, captured in the upper part of the ontology and additional knowledge about basic physical principles, like conservation of mass and energy, SPARQL rules are formulated.

Like for bond graphs, the concept of flow and effort properties can be used to define and structure the SPARQL rules. For example, mass flow rate and the heat flow are flow

properties and the pressure and temperature are effort properties. With the help of this concept, the assignment of equivalent *Virtual Properties* can be formulated as follows:

- *Rule C1*: properties with the physical quantity of mass flow rate are equivalent if they have a mass flow connection without any branch between.
- *Rule C2a*: properties with the physical quantity of temperature are equivalent if they share a direct mass flow connection.
- *Rule C2b*: properties with the physical quantity of temperature are equivalent if they share a direct mass flow connection with only hydraulic components in between.
- *Rule C3a*: properties with the physical quantity of pressure are equivalent if they share a direct mass flow connection.
- *Rule C3b*: properties with the physical quantity of pressure are equivalent if they share a direct mass flow connection with only thermal components in between.

Some of these rules could have been encapsulated in another rule, but they are separated to reduce complexity. The rules for pressure and temperature are quite similar as both are effort properties. A rule for the heat flow is not formulated as for the current use case the energy transfer is always related to mass transportation.

Rules C1 to C3 are also formulated as SPARQL Update rules. An example of such a SPARQL rule is shown in Listing 6.2.

```
PREFIX : <http://auto.tuwien.ac.at/sic/PETIont#>

INSERT{
  ?allTempProperties :hasEquivalentVirtualProperty
      ?equivalentProperty.
  ?equivalentProperty :hasEquivalentVirtualProperty
      ?allTempProperties.

}WHERE {

  ?allTempProperties a :VirtualPropery;
                     :hasPhysicalQuantity :Temperature;
                     :hasRelatedPort ?startPort.
  ?startPort :hasDirectMassFlowTo | :hasDirectMassFlowFrom ?endPort.
  ?endPort :hasRelatedVirtualProperty ?equivalentProperty.
  ?equivalentProperty :hasPhysicalQuantity :Temperature.
}
```

Listing 6.2: SPARQL rule C2a - Connecting equivalent temperature properties

Step 3 - Associate sensors with Virtual Properties

After all *Virtual Properties* are created and connected with equivalent properties from other components, the relation to available and associated sensors and their data is created (Fig. 6.2d). This enables access to the available sensor data through the ontology. Also, redundant or missing sensors can be identified. The mapping between *Virtual Properties* and the sensors is again specified with the help of SPARQL rules.

- *Rule S1*: Mapping of mass flow rate sensors
- *Rule S2*: Mapping of temperature sensors
- *Rule S3*: Mapping of pressure sensors

The mapping *Rule S2* is presented in Listing 6.3. The rule searches for PCE requests (sensors) with a physical quantity of temperature and assigns these sensors to the *Virtual Properties* which are reachable through a direct mass flow connection. Also, equivalent *Virtual Properties* are connected with these senors.

```
PREFIX : <http://auto.tuwien.ac.at/sic/PETIont#>

INSERT{
  ?allProperties :hasMeasurement ?sensor
}WHERE {
  ?sensor a :SensorFunction;
          :hasPhysicalQuantity :Temperature;
          :hasProcessConnection ?port.

  ?port (:hasDirectMassFlowTo | :hasDirectMassFlowFrom)* ?connectedPorts.
  ?connectedPorts :hasRelatedVirtualProperty ?property.

  ?property :hasPhysicalQuantity :Temperature;
            :hasEquivalentVirtualProperty* ?allProperties.
}
```

Listing 6.3: SPARQL rule S2 - Mapping temperature sensors to their *Virtual Property*

6.4 Use Case - Heating Process

A thermal heating process is used as a use case to evaluate the proposed ontology-based model identification approach. The P&I diagram of this process is shown in Fig. 6.3. For referencing the instrumentation, labels are used instead of numbers.

An electric heater *H1* is used to warm up the air, which is transported to an open vessel *SiPro* which is not insulated, thus the temperature inside the vessel is influenced by a surrounding temperature which acts as a disturbance to the process. The temperature *T_p* inside this vessel is controlled by a PI-controller. The air is transported to the vessel by a simple fan *F1* which creates a fixed pressure increase. The extracted air is used by a heat exchanger *HE1* for heat recovery.

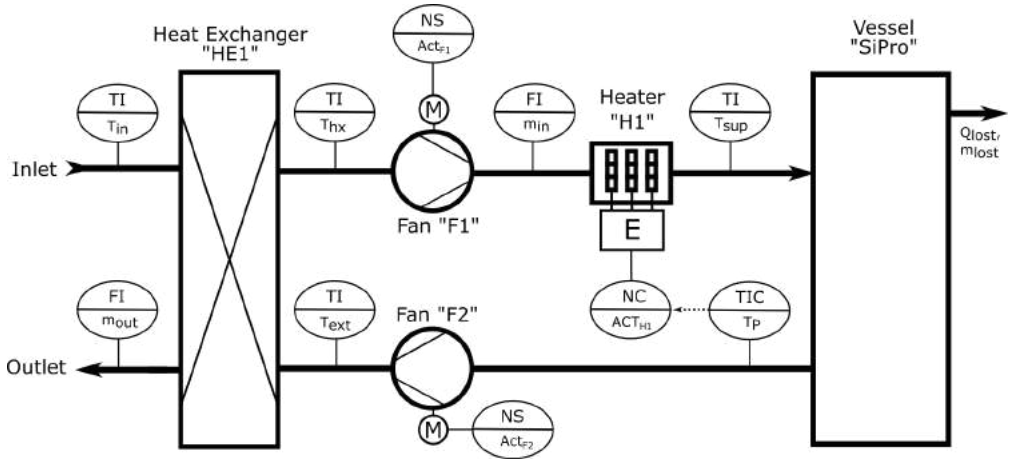


Figure 6.3: P&I diagram of the heating process.

An OpenModelica simulation model is utilized for evaluation and shown in Fig. 6.4. This simulation model is used to create synthetic sensor data for the data-driven model development process.

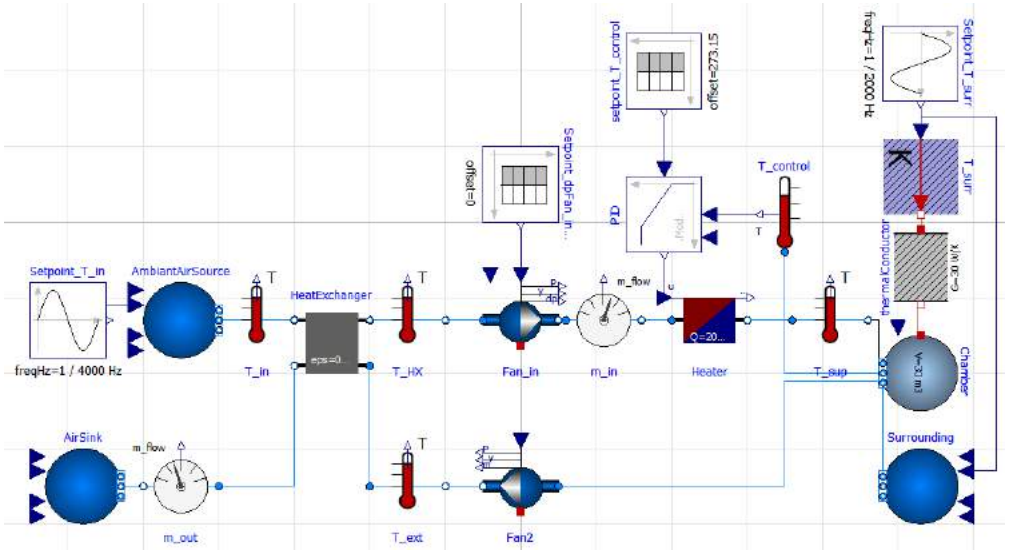


Figure 6.4: OpenModelica simulation model of the heating process use case.

For the model identification task, a pseudo-random binary signal is used for the input and disturbance variables. The data is applied to the model and the generated output data is used as synthetic sensor data for model identification, afterward.

To make the artificial sensor data more realistic, a random noise signal was added to it. This signal has a Gaussian distribution with a standard deviation of 0.1° C for the temperature sensors and 0.01 kg/s for the mass flow sensors. For evaluation purposes,

different set points were applied to the Modelica simulation model and the model's outputs are recorded.

The basic topology information from the P&I diagram is modeled in the ontology. Afterward, the presented SPARQL rules are applied to create the *Virtual Entity* inside the ontology, with its causal relations and sensor mappings. These causal relations are depicted in Fig. 6.5. With the information about the plant equipment in the ontology, in combination with the direct sensor data access, the model identification task can be performed automatically. The sensor data retrieval through the SPARQL endpoint is based on custom property functions and explained in detail in [16]. Apache Jena Fuseki is used as SPARQL endpoint, where the ontology is loaded and the queries are executed. Some SPARQL queries and their answers, which are used for the model identification process, are shown below.

Listing 6.4 retrieves all input *Virtual Properties* for the heater *H1* with their related sensors if they exist. The answer to that query shows, that for every *Virtual Properties* of heater *H1* an associated sensor exists, even if it was initially not directly associated with that component. An example of such newly inferred knowledge results in sensor *T_hx*. It can be seen, that the sensor *T_hx* is located directly after the heat exchanger *HE1* in the P&I diagram (Fig. 6.3). After applying the explained SPARQL rules, this sensor is also associated with the input temperature of the heater *H1*, because the fan

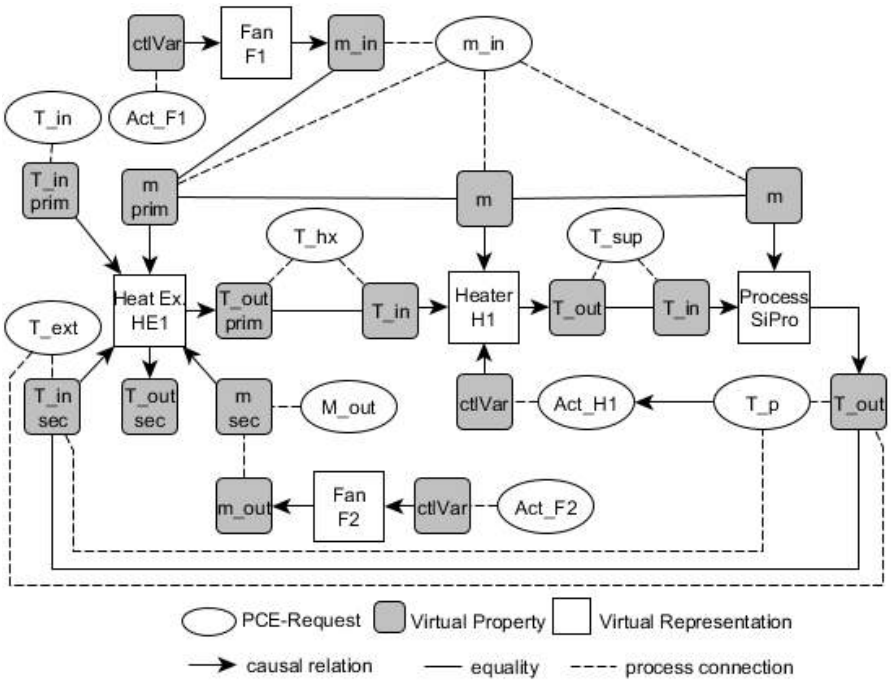


Figure 6.5: Automatically created causal relations and sensor mapping inside the ontology for the presented use case.

F1 was specified as a hydraulic equipment. The same applies to the other temperature and mass flow rate sensors. Thus, the sensor data is directly accessible for the model identification task, without any further human intervention.

```
PREFIX : <http://auto.tuwien.ac.at/sic/PETIont#>

SELECT ?property ?sensor ?actuator
WHERE {
   :H1 :hasDynamicModel ?dynModel.
   ?dynModel :isInfluencedBy ?property.
   optional{
     ?property :hasMeasurement ?sensor.
   }
   optional{
     ?property :hasActuator ?actuator
   }
}
```

```
|?property |?sensor |?actuator  |
|"""""""""""""""""""""""""""""""|
|:H1_ConVar |:        |:Act_H1    |
|:H1_Tin    |:T_hx    |           |
|:H1_m      |:m_in    |           |
```

Listing 6.4: SPARQL query to retrieve the input properties for the electric heater *H1* and their associated PCE requests.

Similar queries can be formulated to find all existing components with their input and output properties as well as related sensor data. With this information, models can be trained automatically.

It is also possible to search the ontology for redundant sensor information in the plant. The query in Listing 6.5 retrieves all *Virtual Properties* inside the plant which have redundant sensors installed (e.g., *T_ext* and *T_p*). The answer is in compliance with the P&I diagram of Fig. 6.3. In combination with additional semantic sensor information, like the physical unit of the measurement, this can be used for further sensor data evaluation.

```
PREFIX : <http://auto.tuwien.ac.at/sic/PETIont#>

SELECT ?property (count(?property) as ?sensorCount)
WHERE{
   ?property a :VirtualProperty.
   ?property :hasMeasurement ?sensor
}GROUP BY (?property)
HAVING (count(?property) > 1)
```

```
|?property    |?sensorCount |
|"""""""""""""""""""""""""""|
|:HE1_Tin_sec|2            |
|:SiPro_Tout |2            |
```

Listing 6.5: SPARQL query and answer to retrieve redundant sensors

With the information stored in the ontology, a dynamic model is identified based on the input and output information as well as the related sensor data. In principle, any regression model, like an artificial neural network or a support vector machine can be

applied. Such more sophisticated models can be useful for real plant data with high non-linearity. For our use case, a linear autoregressive with exogenous input (ARX) is chosen to represent the dynamic behavior of the components. The model was trained with the Python scikit-learn framework[2]. Prediction results of the mass flow *m_in* and the heat exchanger temperature *T_hx* are shown in Fig. 6.6. A sliding window serial-parallel model structure is applied to predict the ARX-model output with a window size of 15 seconds by a one second simulation step size. A qualitative analysis shows that the retrieved sensor data for these models are correct and the results of the ARX-models follow the synthetic sensor data.

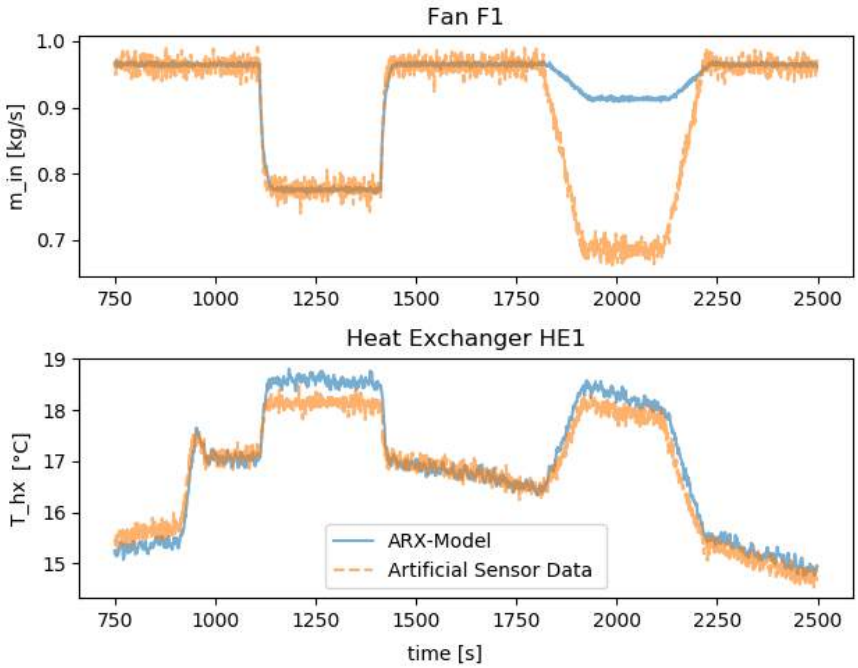


Figure 6.6: Prediction results of the linear ARX-models, which are created based on the knowledge from the ontology.

The available causal relations in the ontology could have also advantages for sensor data evaluation. As an example, Fig. 6.6 shows two decreases in the mass flow *m_in*, starting at about 1,100 and 1,800 seconds. The first one was produced by reducing the set point of fan *F1* and the second by introducing an increased flow resistance through the heat exchanger. The Model of *F1* follows the reduced set point but shows a mismatch at the second decrease because the predicted mass flow at this pressure set point should be much higher. This can be an indication for a sensor fault, detected by the ARX-model. To verify the measurement of sensor *m_in*, the causal relation in the ontology can be used (Fig. 6.5). An implicit sensor redundancy of *m_in* is found in the temperature sensor *T_hx*. As Fig. 6.6 shows, *T_hx* has no large discrepancies from the prediction,

[2]https://scikit-learn.org

which means that the measured mass flow *m_in* is correct. This implies an abnormal behavior inside the plant and, thus, further investigations for the usage of these causal relations for sensor data evaluation have to be carried out.

6.5 Conclusion & Future Work

The paper illustrates that – based on ontological knowledge of a plant equipment, its topology, and instrumentation – a data-driven model development process can be partly automated. This reduces the overall amount of time needed for developing model-based control strategies of industrial energy systems.

In the presented approach, additional work has to be carried out to initialize the plant's topology information in the ontology. Yet, it cannot be expected from engineers to be aware of ontology modeling. Therefore, future work will investigate, how already existing information sources can be utilized to create the initial ontology, taking into account that the process of information modeling should be based on well-known industrial standards or languages. In this context, OPC UA often named as a pathfinder for information modeling in the area of Industry 4.0 seems to be a promising candidate.

For real-world applications, related sensor data have to be pre-processed before they can be used for model identification. This was not necessary for our use case as we used synthetic sensor data. For this reason, future work will investigate how sensor data can be automatically pre-processed and evaluated based on the created causal relations in the ontology.

Acknowledgment

This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) for the „PoSyCo - Power System Cognification“ project under the contract number 867276, as well as the „SIC! - Smart Industrial Concept!“ doctoral program at the TU Wien.

References

[1] Klaus Bauer and Et Al. *Securing the future of German manufacturing industry. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Final report of the Industrie 4.0 Working Group.* Tech. rep. Federal Ministry of Education and Research, 2013.

[2] Dan Brickley and R.V. Guha. *RDF Schema 1.1. W3C Recommendation.* Tech. rep. World Wide Web Consortium (W3C), 2014. URL: https://www.w3.org/TR/rdf-schema/.

[3] Henrik Dibowski, Ondřej Holub, and Jiří Rojíček. „Ontology-based fault propagation in building automation systems“. In: *International Journal of Simulation: Systems, Science and Technology* 18.3 (2017), pp. 1.1–1.14. ISSN: 1473804X. DOI: 10.5013/IJSSST.a.18.03.01.

[4] Lisa Ehrlinger and Wolfram Wöß. „Towards a definition of knowledge graphs“. In: *CEUR Workshop Proceedings* 1695 (2016). ISSN: 16130073.

[5] EN ISO 10628. *Diagrams for the chemical and petrochemical industry.* Tech. rep. European Standards, 2012.

[6] Paula Gearon, Alexandre Passant, and Axel Polleres. *SPARQL 1.1 Update - W3C Recommendation.* Tech. rep. World Wide Web Consortium, 2013.

[7] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language. W3C Recommendation.* Tech. rep. World Wide Web Consortium (W3C), 2013. URL: https://www.w3.org/TR/sparql11-query/.

[8] Fernando Iafrate. *From Big Data to Smart Data.* Ed. by Camille Rosenthal-Sabroux. Advances i. ISTE Ltd and John Wiley & Sons, Inc., 2015. ISBN: 9781848217553.

[9] IEC-62424. *Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools.* Tech. rep. International Electrotechnical Commission, 2016.

[10] M. Killian and M. Kozek. „Ten questions concerning model predictive control for energy efficient buildings“. In: *Building and Environment* (2016). ISSN: 03601323. DOI: 10.1016/j.buildenv.2016.05.034.

[11] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation.* Tech. rep. World Wide Web Consortium, 2004. URL: https://www.w3.org/TR/rdf-concepts/.

[12] G. N. Lilis et al. „Semi-automatic thermal simulation model generation from IFC data“. In: *eWork and eBusiness in Architecture, Engineering and Construction - Proceedings of the 10th European Conference on Product and Process Modelling, ECPPM 2014* Eastman 2011 (2015), pp. 503–510. DOI: 10.1201/b17396-83.

[13] Joern Ploennigs et al. „BASont - A modular, adaptive building automation system ontology“. In: *IECON Proceedings (Industrial Electronics Conference)* (2012), pp. 4827–4833. DOI: 10.1109/IECON.2012.6389583.

[14] Gregory A. Silver, Osama Al-Haj Hassan, and John A. Miller. „From domain ontologies to modeling ontologies to executable simulation models“. In: *Proceedings - Winter Simulation Conference* (2007), pp. 1108–1117. ISSN: 08917736. DOI: 10.1109/WSC.2007.4419710.

[15] Nina Solomakhina et al. „Extending statistical data quality improvement with explicit domain models“. In: *2014 12th IEEE International Conference on Industrial Informatics (INDIN).* IEEE, July 2014, pp. 720–725. ISBN: 978-1-4799-4905-2. DOI: 10.1109/INDIN.2014.6945602. URL: http://ieeexplore.ieee.org/document/6945602/.

[16] Gernot Steindl, Thomas Früwirth, and Wolfgang Kastner. „Ontology-Based OPC UA Data Access via Custom Property Functions“. In: *24th Interantional Conference on Emerging Technologies and Factory Automation*. Zaragoza, Spain, 2019.

[17] Gernot Steindl and Wolfgang Kastner. „Query Performance Evaluation of Sensor Data Integration Methods for Knowledge Graphs“. In: *6th IEEE International Conference on Big Data, Knowledge and Control Systems Engineering*. Sofia, 2019.

[18] Carsten Stöhr et al. „Smart Services“. In: *Procedia - Social and Behavioral Sciences* 238 (2018), pp. 192–198. ISSN: 18770428. DOI: 10.1016/j.sbspro.2018.03.023. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877042818300235.

[19] Rudi Studer, V Richard Benjamins, and Dieter Fensel. „Knowledge Engineering: Principles and methods“. In: *Data & Knowledge Engineering* 25 (1998), pp. 161–197.

[20] W3C OWL Working Group. *OWL 2 Web Ontology Language. Document Overview*. Tech. rep. World Wide Web Consortium (W3C), 2012. URL: https://www.w3.org/TR/owl2-overview.

CHAPTER 7

Semantic Microservice Framework for Digital Twins

Publication: *G. Steindl and W. Kastner, "Semantic Microservice Framework for Digital Twins," Applied Sciences, vol. 11, no. 12, p. 5633, Jun. 2021, doi: 10.3390/app11125633.*

Abstract: Digital Twins (DT) in industrial cyber-physical systems are the key enabling technology for Industry 4.0. Services are an essential part of almost every DT concept, but their interaction is usually implementation-specific since no common guidelines are available. This work identifies some fundamental requirements for a DT service framework based on applications identified in corresponding literature. Based on these requirements, a service framework architecture is proposed. The architecture utilizes Semantic Web technology and a workflow engine for service orchestration to support the fulfilment of the identified requirements. As a case study for sensor data evaluation of an industrial process, a proof-of-concept implementation is presented, showing the feasibility and suitability of the proposed DT service framework architecture.

7.1 Introduction

The fourth industrial revolution or Industry 4.0 is changing business within the industry, caused by the rapid development of Information and Communication Technology (ICT) [37]. One of the most promising technologies that enables us to achieve the vision of Industry 4.0 is the DT [52]. A DT can be defined as "a formal digital representation of some asset, process or system that captures attributes and behaviors of that entity suitable for communication, storage, interpretation or processing within a certain context" [33]. DTs can be used within ICPS for monitoring, diagnostic, prediction, and control [32].

During the last few years, a number of research projects have addressed DT architectures and frameworks [25, 7, 28, 51, 1, 48]. A very interesting concept is the 5D-DT presented in [51], where five elements or dimensions for a DT are specified: "Physical Entity", "Virtual Entity", "Connection", "Data" and "Services". In this conceptual model, the aspect of services is emphasized as an important part of the DT. This 5D-DT approach is used in [48] as the basis for the GDTA, which is aligned with the six IT layers of the RAMI 4.0 [2] to structure the elements of the GDTA model. However, services are only considered in a more general and abstract way in the GDTA.

In ICT, the microservice architectural style has become more relevant for building distributed software applications with improved scalability and maintainability in recent years [12]. The idea is to build service-based applications by composing small, loosely coupled software services [15]. The actual size of services depends on the application, but the attribute "small" targets their functionality rather than the lines of code. The service composition can be performed via orchestration or choreography. Orchestration is a centralized approach, whereas choreography follows a decentralized method. Sometimes, it even can be beneficial to apply both of those two concepts, e.g., in automation systems [50]. Another design decision that has to be made when utilizing microservices is inter-service communication. The communication can be established using an asynchronous or synchronous request–reply pattern or event-driven asynchronous message exchange. In microservice architectures, event-driven communication is preferred as it supports decoupling of services from each other [35]. A common technology for synchronous request–reply communication is HTTP, which is often used in combination with REST. For event-driven asynchronous communication, many message-oriented middleware solutions exist. One such solution that is widely used is Apache Kafka, which provides fault-tolerant, scalable, and stream-based messaging [21].

As described before, services play a key role in building DTs, but in most DT frameworks or architectures, they are only mentioned or described at a high level of abstraction. As most DT implementations are realized following a specific goal without any architectural template [28], the same holds for the implemented services. There are many design choices for how these services can be composed for later interaction. Their final design and implementation should be based on distinct requirements. Still, there is a significant gap in DT research, regarding how to offer a higher number of services in the same environment to support complex decision making [10]. Missing architectural guidelines are resulting in application-specific solutions which are barely reusable, and increase development time and costs. Thus, our research focuses on the services infrastructure of a DT and explicitly addresses requirements and a service framework architecture, which can later be applied for various DT applications.

The main contributions of our work are the specification of functional and non-functional requirements for a DT service framework derived from a literature review. These requirements were clustered based on three RAMI 4.0 IT layers (Information Layer, Functional Layer, and Business Layer) which are relevant for the proposed service framework. Resting upon the identified requirements, a novel microservice architecture for

DTs is proposed. This architecture uses a federated knowledge graph to provide semantic interoperability between services and enables both choreography and orchestration by incorporating event-based messaging in combination with a workflow engine. A case study for a DT of a thermal heating process is used to investigate and evaluate the proposed service architecture. Therefore, microservices for an automatic sensor data evaluation were implemented and the design artifacts are checked against the identified requirements.

The remainder of the paper is structured as follows: Section 7.2 gives a short overview of related work in the area of DT service framework as well as DT architectures and applications which are used to derive requirements for a DT service framework from. Next, the identified requirements and the derived service framework are presented in Section 7.3. The proposed service framework is implemented as a proof-of-concept for a defined use-case in Section 7.4. In the end, the presented service framework and future research directions are discussed.

7.2 Related Work

A short overview of available DT service frameworks is given, and literature that is used to derive requirements for such a service framework is presented.

7.2.1 Digital Twin Service Frameworks

In this section, service framework architectures dedicated to DTs and related areas, such as IoT applications and smart manufacturing, are presented to find commonalities and shortcomings, which influence the design of our proposed DT service framework.

The design and implementation of a DT in smart manufacturing is presented in [11]. The authors investigated available open-source tools and technology for the implementation of the DT. In this context, they proposed a DT concept in alignment with the Industrial Internet Reference Architecture (IIRA) and the RAMI 4.0. They also introduced a microservice framework and defined 36 services, clustered in groups related to their components in their conceptual DT model, such as monitoring services, things and event management services, simulation management services, decision-making and control services. The import role of semantic interoperability and a life cycle-oriented knowledge base of DTs is also conceptually addressed by their architecture but not fully explored in their prototype. A Service Manager component is responsible for the composition and orchestration of the services, but further details of how this is performed are not given.

A more concrete example of a service framework and the service interaction inside a DT is provided in [3]. The authors proposed a microservice approach in combination with an event-based architecture. They argue that a DT should follow an event-driven architecture style to parallelize processing and provide near real-time capabilities. Their solution to the problem is the use of Apache Kafka for state tracking, since most of the stream processing in DT implies stateful operations, but microservices should be stateless

in principle. Thus, stateful stream processing can be supported in a DT. They carried out performance analysis and showed that Apache Kafka is suitable for managing the states with some restrictions. Such tracking of the current system state is relevant for stream data and human–machine interaction.

Similar to the previously presented architecture, a service-oriented and event-driven manufacturing information system architecture was proposed in [55]. The event-driven architecture is used to avoid point-to-point device and service integration and ensures loose coupling of the services. Apache ActiveMQ [6] is used as Enterprise Service Bus (ESB). The presented use-case is targeting discrete manufacturing, in which the authors showed the integration of devices and services on all hierarchy levels. The service composition in this system is performed using choreography to avoid a central orchestrator. The benefit they explored using their microservices in combination with event-based communication is that services can be developed and tested in isolation, as mock-ups may serve as temporary replacement for other applications.

A microservice architecture is also used in [5]. Rather than targeting a DT architecture, the authors present a framework for predictive analytics of IoT applications. However, some concepts are similar and helpful in the context of DTs. The authors used a containerized microservice architecture to build the data pipeline. Their implementation is based on Docker [14] and Apache Kafka. A central orchestrator is used to combine multiple operations provided by microservices. Additionally, data modeling is used, based on the Web of Objects framework [20] to achieve semantic interoperability.

In most of the presented frameworks, microservices combined with event-based messaging are used, as they can provide benefits regarding separation of concerns, flexibility in choosing technology, scalability, etc. However, their underlying requirements are mostly not stated. Therefore, the next section analyzes DT frameworks and applications to identify some fundamental requirements regarding a DT framework.

7.2.2 Requirements for a Digital Twin Service Framework

Requirements found in the literature primarily target the overall concept of a DT, rather than only focusing on the service infrastructure. Nevertheless, some general requirements are also relevant for the proposed service framework. Here, the main literature is presented, which is used to derive the requirements presented in Section 7.3.1.

A requirements-driven DT framework for smart manufacturing or Industry 4.0 is presented in [34]. Some requirements mentioned there are also specifically relevant for a service framework, e.g., a DT's capability should be modular with clear boundaries (RN2). Services should also provide some form of narrow intelligence to solve some special tasks in the application domain (RI3). Furthermore, expertise should be incorporated into services to realize intelligence in solutions (RI4). Another aspect is the interoperability with DT clients, which must be realized by providing an appropriate service interface (RI5). DT should also be extensible, which means incorporating data or services from outside (RN4). An important aspect is that a DT framework must support an evolution

rather than a revolution of capabilities, helping us to introduce and further develop the DT over its whole life-cycle (RN3). New services are supporting the evolution of existing capabilities. As DT should provide an added value over lifetime, the DT services have to be integrated into the business process at the enterprise level in some way (RB1).

In [53], a method for DT-driven product design, manufacturing, and service is introduced. In this context, nine service categories are presented. Some of these service categories demand some requirements regarding the service communication infrastructure. For instance, a service for real-time state monitoring requires a certain data acquisition infrastructure (RF1) and certain communication patterns (RF2). The same holds for a service within the category of energy consumption analysis and forecast, such as processing heterogeneous data from various sources (RI1, RF3), dealing with historical data (RI2) and supporting request–response communication with other services (RF4). These two examples show that a service framework has to be able to support different communication patterns.

A DT architecture reference model for the cloud-based CPS (C2PS) is presented in [4], in which the key properties of computation, control, and communication are described. The cloud-based approach seems reasonable, based on requirements such as computational power and scalability. On the other hand, this can cause a problem, as the communication over the Internet is critical, regarding availability [19] and the communication delay [13] for real-time control applications. Thus, a combination of an edge, fog, and cloud-based approach is feasible for a DT to provide services with their needed resources, as presented in [40]. However, in some use-cases with a high demand for reliability, security, and privacy, the hosting of the DT has to be done on-premise (RN1). To enable this flexibility, a containerized solution for DT services, as proposed in [8], seems to be suitable.

Human–machine interaction is also essential in the context of DTs, which includes social and technical aspects [24]. Maintenance scenarios are one example of such a human–machine interaction. These scenarios can be quite complex, and their states have to be stored during this process, as shown in [36] for gas turbine maintenance. Thus, the state must also be stored at the involved services in such a human–machine interaction (RB2).

Services of a DT often have to process a large amount of data. Such data-intensive applications have some basic non-functional requirements which should be met, such as reliability (RN5), scalability (RN3), and maintainability (RN2, RN4) [29]. Unfortunately, there are no easy solutions to meet these requirements, but specific architectural patterns and techniques can be applied during implementation to fulfil them.

7.3 Microservice Framework Architecture

In this section, the identified functional and non-functional requirements are presented, which are used to design the proposed DT service framework architecture. A detailed description of this service framework architecture can be found in Section 7.3.2.

7.3.1 Identified Requirements

Based on the literature review, some essential fundamental requirements are identified to implement a DT service framework. They are grouped into functional and non-functional requirements. The functional requirements are again grouped by the IT layers of the RAMI 4.0. The list is not comprehensive but gives a solid base for a service framework architecture and its implementation.

Non-Functional Requirements

Table 7.1 shows the identified non-functional requirements. The possibility to host the services on-premise is, in some use-cases, a prerequisite because of data ownership issues or response time (RN1). The other requirements are mainly concerned with reliability (RN5), scalability (RN3), and maintainability (RN2, RN4), which are relatively common for data-intensive systems.

Table 7.1: Non-functional requirements.

ID	Requirement	Origin
RN1	The DT and its services should be able to be hosted at the cloud as well as on-premises for data ownership and performance reasons.	[4, 8]
RN2	Services of a DT should be loosely coupled to add or remove new services without influencing each other.	[34, 53]
RN3	Services of a DT should be scalable to handle requests from a single machine up to a whole factory.	[34, 29]
RN4	Services of a DT should be maintainable by different development teams (third party integration).	[29]
RN5	The service infrastructure of a DT should tolerate short down times of single services to increase the reliability.	[29]

Functional Requirements

The following functional requirements are clustered by the Information Layer, Functional Layer, and Business Layer defined in RAMI 4.0.

The requirements for the Information Layer are shown in Table 7.2. They are mainly concerned with interoperability issues, e.g., how information is provided to other services (RI3, RI4) and exchanged between services and other systems (RI5). Furthermore, how heterogeneous data can be integrated (RI1) and how these data can be interlinked with context information (RI2) to provide further semantics is targeted.

Table 7.2: Functional requirements for the Information Layer.

ID	Requirement	Origin
RI1	The DT should be able to process heterogeneous data from different sources.	[53]
RI2	The DT should be able to interlink time series data with context information, to make it interpretable for other services.	[53]
RI3	Services of a DT should have control about the information they provide to other services.	[34]
RI4	The DT should have a service which provides access to the information provided by all services of the DT.	[34, 48]
RI5	Services of the DT should exchange information in a semantically meaningful way.	[34]

Requirements for the Functional Layer are shown in Table 7.3. They define how the continuous stream of data (e.g., sensor data) is handled (RF1) and which communication patterns between the services are needed (RF2, RF3, RF4). The communication patterns are manifold: providing sensor data to many services needs an event-based 1:n communication; a monitoring service receiving data from many sources needs an n:1 communication; prediction services would most likely need a request–reply communication pattern.

Table 7.3: Functional requirements for the Functional Layer.

ID	Requirement	Origin
RF1	Services shall be able to access a continuous stream of (sensor) data to monitor the system in real-time.	[53]
RF2	Data streams should be accessible by multiple services simultaneously to process it in parallel and reduce reaction time of the system.	[53]
RF3	Services of the DT should be able to receive data streams from multiple sources at the same time to fuse and process data.	[53]
RF4	Services of the DT should be able to respond to a specific service request to enable a one-to-one communication for information retrieval.	[53]

The requirements for the Business Layer are shown in Table 7.4. The integration of the DT capabilities and services into the business processes at the enterprise level is essential to support the value-added chain (RB1). Furthermore, human–machine interaction requires holding the state for a certain period (RB2), e.g., a maintenance assignment to a service technician triggered by a predictive maintenance service of the DT. The state of the involved services has to be stored till the service technician confirms some action.

Table 7.4: functional requirements for the Business Layer.

ID	Requirement	Origin
RB1	The functional services of the DT should be able to be integrated into the business processes at enterprise level to support the value-added chain.	[34]
RB2	Service interaction states should be traceable to facilitate human–machine interaction and the identification of service faults.	[24]

7.3.2 Proposed Service Framework Architecture

Based on the identified requirements presented in Section 7.3.1, design decisions for the service framework architecture are made and explained in this section. The principal architecture of the proposed service framework for a DT is shown in Figure 7.1. It is based on the concepts of the GDTA presented in [48], which is also aligned with the RAMI 4.0 IT layers. Relevant for the proposed service framework are the Information Layer, the Functional Layer, and the Business Layer, which are also depicted in Figure 7.1.

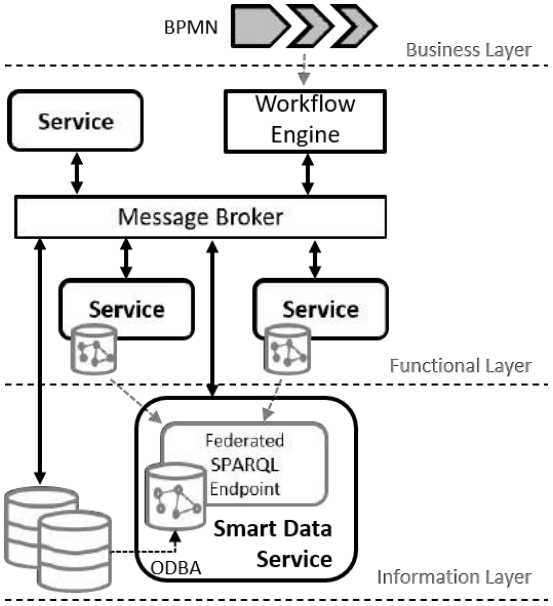


Figure 7.1: Proposed micorservice framework architecture, in alignment with the RAMI4.0 IT-layers.

One of the main components present in the GDTA is the shared knowledge as part of the Smart Data Service on the Information Layer. This Smart Data Service provides contextual information about data and resources, which can be facilitated by the services of the DT. Its shared knowledge can be realized using Semantic Web technologies to build a so-called knowledge graph. A knowledge graph is a knowledge-based system,

which consists of an ontology and reasoner, but also is capable of integrating external information sources [16]. In this context, an ontology means a formal, explicit specification of a shared conceptualization [49]. Ontologies, based on OWL can provide the semantics for the data processed by the DT and can be used to enable automatic reasoning if needed. The data can be accessed via SPARQL endpoints. A SPARQL endpoint is a web service that is capable of receiving and processing SPARQL protocol requests using HTTP. Fundamental information about the physical entity of the DT, such as plant equipment, topology, and the instrumentation, is directly stored in a triplestore, which is managed by the Smart Data Service itself.

Services can add relevant information to the knowledge graph by facilitating a federated SPARQL query engine, within the Smart Data Service. A federated query engine is able to integrate distributed data sources virtually. That means a query is sent to several SPARQL endpoints, and the results are merged. This process is fully transparent for the user, as it seems that only one triplestore is queried [42]. Services can include graphs from their local triplestore that are managed by the services themselves. Thus, information can stay private or can be included into the shared knowledge.

To provide access to historical data and add context and semantics to it, OBDA is used. Data from a relational database is mapped to ontological concepts and can be accessed through the knowledge graph. The loading from the data is only performed when the data is accessed. Thus, the data stay in their original database and do not have to be copied into the shared knowledge graph, which usually enhances the data access performance [47].

With the help of the presented concepts for building a Smart Data Service, existing and also newly created ontologies can be used within a DT to provide a description in terms of classes, properties, and their interrelation for a specific domain (TBox). Various services can instantiate the individuals (ABox). As they can refer to TBox concepts, clear semantics to the data is provided. This approach has the advantage that data integration can be performed with less effort, as the knowledge graph acts as an abstract semantic integration layer [41]. Thus, data from relational databases and also from other sources, e.g., OPC UA can be included in the knowledge graph and made accessible for all services. Furthermore, the interoperability between services is enhanced, as the exchanged data can be referred to concepts defined in the TBox, providing precise semantics to the data. This is additionally supported by formats such as JSON-LD, which allow stating such references. Another advantage of using a knowledge graph based on Semantic Web technology is the support for knowledge discovery. Data from various services can be connected, and new insights into the DT can be gained.

The Functional Layer contains the actual services of the DT. Thus, the requirements targeting inter-service communication are relevant, but also the non-functional requirements have influence on the design.

A microservice architecture facilitates maintainability because services can be realized and deployed by independent development teams. Another advantage of microservices

is that they can be containerized. Thus, they can be deployed and orchestrated in a virtualized environment hosted in the cloud or on-premise if needed.

For the inter-service communication, a Message-oriented Middleware (MOM) with a message broker is used. This MOM allows the realization of various communication patterns, such as a 1:n or an n:1 communication, which are useful for building data pipelines. A typical request–reply communication is also often required for certain services, which can be implemented on top of such an MOM. The use of an MOM can help to decouple services and support reliability and scalability by operating a cluster of brokers. Suppose the broker is able to store messages; the reliability can be further increased because, if a service is not available for a short period, it can retrieve the message from the broker once it is reconnected.

Service composition is fundamental at the Business Layer to provide certain functionality of the DT. This composition can be implemented by choreography or by orchestration. In some cases, choreography might have advantages because it is a more decentralized approach. However, if states have to be handled, e.g., for error handling or user interaction, this can be laborious [39]. Orchestration, on the other hand, uses a central component, where the composition logic is located. This can lead to a tight coupling of services and integrating service logic into the orchestrator [35]. The workflows in which a DT service is involved can be located on the enterprise or production level [30]. At the enterprise level, typically BPMN is used to describe these workflows, and sometimes workflow engines are used to automate them. Such workflow engines can also be used for microservice orchestration [26]. Using orchestration combined with a workflow engine to manage the flow between various microservices can help to visualize these flows and handle long-lived transactions. As workflow engines support BPMN, this notation can be used to communicate with a non-software developer and seamlessly integrate the DT capabilities into existing business processes on the enterprise level. In Figure 7.1, only a central workflow engine is depicted, but it is also possible to use multiple decentralized engines.

7.4 Proof-of-Concept: Automatic Sensor Data Evaluation

For evaluating the proposed service architecture, a proof-of-concept for a DT of a thermal heating process is implemented. The service interaction is demonstrated with a composite sensor data evaluation service, which analyzes sensor data from the plant and detects anomalies. The anomaly is classified as a sensor fault or abnormal behavior of the plant caused by a malfunction of the equipment.

For the presented use-case, three different services are implemented, which are orchestrated by the workflow engine Zeebe [9], as shown in Figure 7.2. The communication between the services takes place over the stream-based MOM Apache Kafka [21]. Every service holds its own database or triple store, which are connected by the federated query engine FedX [38] to build a distributed knowledge graph as a shared knowledge base for the services. The information exchange between the services is based on JSON-LD.

This format is used because it provides semantics by referencing specific contexts with URI. Thus, ontological concepts defined in the shared knowledge graph can be used to unambiguously specify the meaning of the data, which facilitates interoperability.

The services are implemented in Python 3.9.1 and use several libraries to communicate with Zeebe, Kafka, the triplestores, or the databases. Every service and its infrastructure (database, triplestore, etc.) are virtualized in Docker containers.

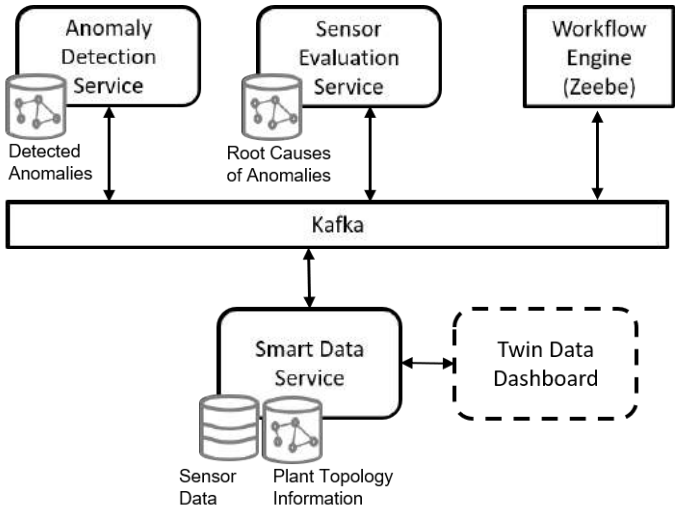


Figure 7.2: Overview about implemented services for sensor data evaluation.

After introducing the use-case, the implemented services are explained in more detail in the following subsections. The source code and a Docker-compose file to set up the service infrastructure can be found on GitHub [43].

7.4.1 Use-Case: Thermal Heating Process

As depicted in the pipe- and instrumentation diagram in Figure 7.3, a simple thermal heating process is used to evaluate the proposed service architecture. The ventilation unit "F1" blows ambient air through an electric heater "H1" into a vessel called "SiPro" where a thermal process takes place. The temperature of the process is controlled by modifying the power of the heater. The air is retrieved from the vessel by the ventilation unit "F2". The heat exchanger "HE1" is used for heat recovery of the exhaust air. Five temperature and two mass flow sensors are placed in the supply and return ducts. The ambient temperature surrounding the vessel causes heat loss of the vessel. The temperature after the heat exchanger T_{hx}, the supply temperature T_{sup}, and the process temperature T_p are measured by sensors with the IDs 102, 105, 106.1. The mass flow into the process is measured by the sensor m_{in}. Those are the most relevant data points for the presented sensor evaluation showcase.

The data for the service evaluation is generated by a simulation, and implemented in Open Modelica [22]. The simulation was used to produce data for a sensor fault as well

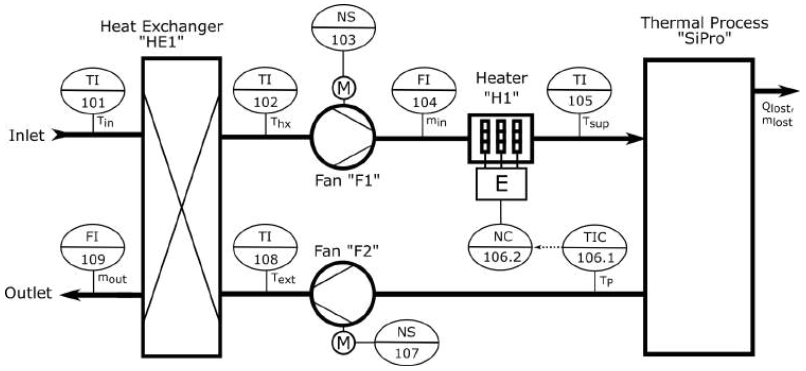


Figure 7.3: Use-case: Pipe and instrumentation diagram of the heating process.

as abnormal behavior of the plant caused by a clogged duct. During a regular operation, the temperature inside the process T_p is controlled and is following the setpoint trajectory shown in Figure 7.4. As shown, there are three operating points for T_p: 50 °C, 70 °C, and a standby mode, in which the temperature is held at 30 °C. During standby, the pressure setpoint of the ventilation unit "F1" is reduced. Thus, the mass flow m_{in} is also decreased.

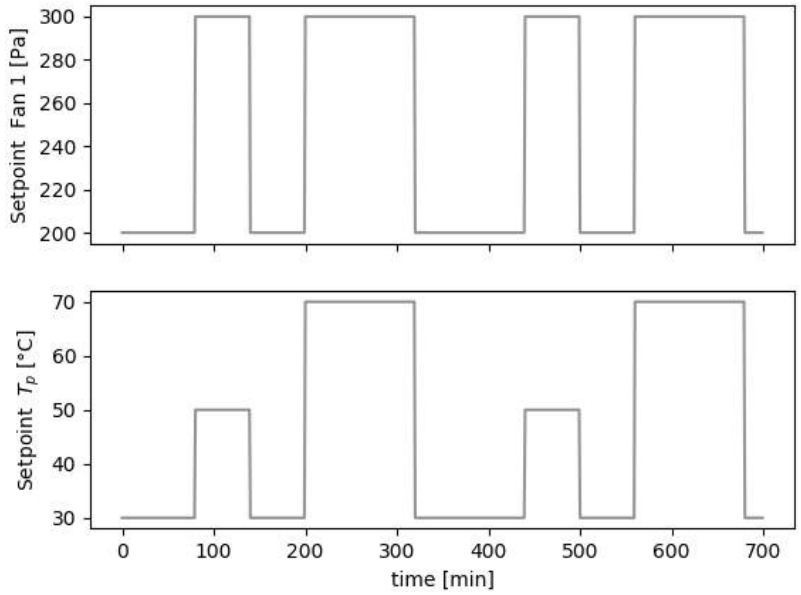


Figure 7.4: Setpoint for the ventilation unit "Fan 1" and the process temperature T_p.

To emulate the behaviour of real sensors, random noise is added to the simulated values, which is modeled as a normal distribution with zero mean and a standard deviation of 0.2 °C and 0.01 kg/s, respectively. The Open Modelica simulation model is available at the GitHub repository [44].

7.4.2 Smart Data Service and Communication Infrastructure

Figure 7.5 shows more details of the Smart Data Service as well as the needed communication infrastructure. The separated processes that build the Smart Data Service, their encapsulation into Docker containers, and the communication between them are shown.

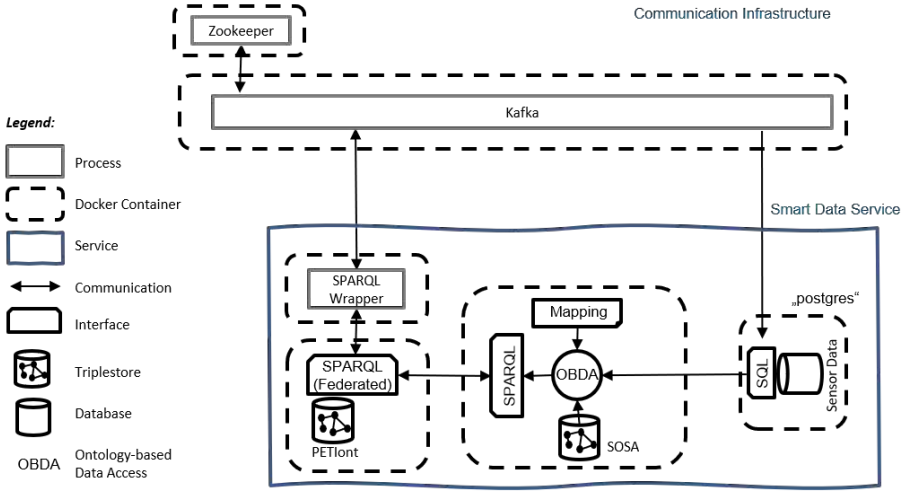


Figure 7.5: Smart Data Service and Communication Infrastructure.

The Smart Data Service provides access to the knowledge graph formed by interlinking various SPARQL endpoints. Essential plant information, such as equipment and topology information, is directly stored at the Smart Data Service. For the presented proof-of-concept, the Plant, Equipment, Topology and Instrumentation Ontology (PETIont) [46] was used to describe the heating process use-case. Information provided by other services is stored and managed by the services themselves but included in the knowledge graph by the federated SPARQL engine FedX. Thus, this information is interlinked and accessible through the Smart Data Service.

The time-series data from the sensors are stored in a PostgreSQL [54] database. To make these data accessible through the Smart Data Service and interlink them with context information, OBDA is utilized. The Ontop Framework [57] is used to map the data into the Sensor, Observation, Sample, and Actuator (SOSA) ontology [56]. Therefore, mappings are defined to populate the SOSA ontology with individuals based on the data stored in the PostgreSQL database. As an advantage of this approach, the data remain in the relational database and are only loaded in a virtual knowledge graph if needed by a SPARQL query request.

The communication between the services is handled by the stream-based MOM Apache Kafka. Kafka is designed as a distributed system, running on a server cluster. Apache Zookeeper provides a centralized service to manage the nodes inside this cluster. Kafka topics are used to send data to services inside the DT. The request–reply communication pattern is implemented on top of the Kafka infrastructure. The requesting service can

add a reply topic to its request, to which the reply will be sent. Such a pattern is implemented to communicate with the Smart Data Service.

As the Smart Data Service communicates with SPARQL internally, a wrapper is implemented to enable requests from other services over Kafka. Services can send queries to the Kafka topic, to which the Smart Data Service is listing to. The answer is sent to the reply-topic specified in the request formatted in JSON-LD

7.4.3 Anomaly Detection Service

The Anomaly Detection Service is used to find deviation in the data of single sensors. Therefore, a data-driven approach was chosen. A linear ARX model is trained based on data which is provided by the Smart Data Service. Details about how these models can be derived and identified based on the information stored in the knowledge graph can be found in [46].

The service infrastructure is shown in Figure 7.6. The "Data-driven Anomaly Detection Process" listens to a specified Kafka topic and starts the anomaly detection if a request arrives. The trained ARX models are executed in a serial–parallel fashion, which means the models are used to predict certain time-steps, and the results are compared with the actual measurements. If the error is larger than a certain threshold, which is three times the standard deviation of the model error, the time is marked as an anomaly. Then, the prediction window is shifted by one time step and actual measured values are used as input for the next prediction.

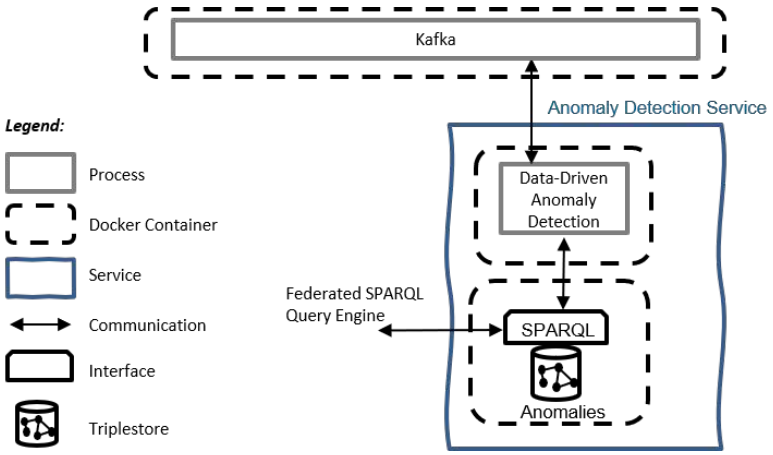


Figure 7.6: Anomaly Detection Service.

If an anomaly for a sensor value is detected, OWL-Time [27] is used to store that information in the knowledge graph. As depicted in Figure 7.7, an anomaly is defined in PETIont and described as OWL-Time "time:Interval". A relation is set between the sensor and the anomaly with the property "peti:hasAnomaly". This information is stored in the local triplestore of the service. As the triplestore of this service is connected to the

federated query engine, the information is also instantly accessible through the Smart Data Service.

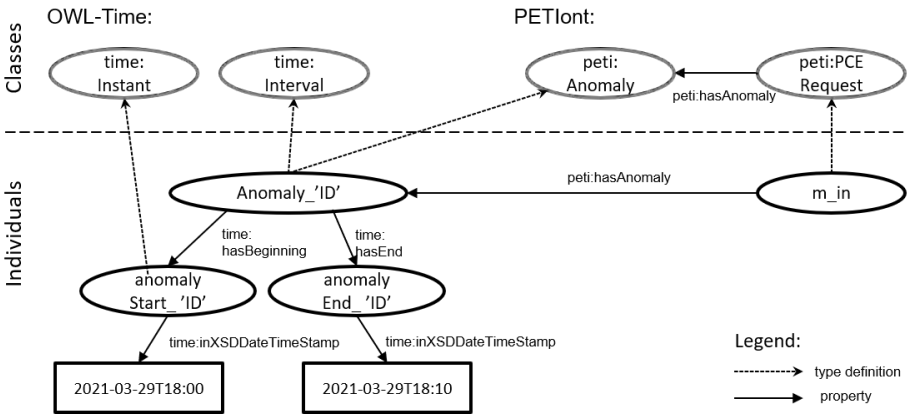


Figure 7.7: Anomaly modeled inside the knowledge graph.

7.4.4 Sensor Evaluation Service

The Sensor Evaluation Service receives detected anomalies and classifies these anomalies as a sensor fault or abnormal behavior of the plant. Therefore, the service has information about the causal relations between sensors and actuators stored in its triplestore, connected to the federated query engine (Figure 7.8). A detailed description of how such relations can be automatically derived based on topology and equipment information stored in the knowledge graph can be found in [46].

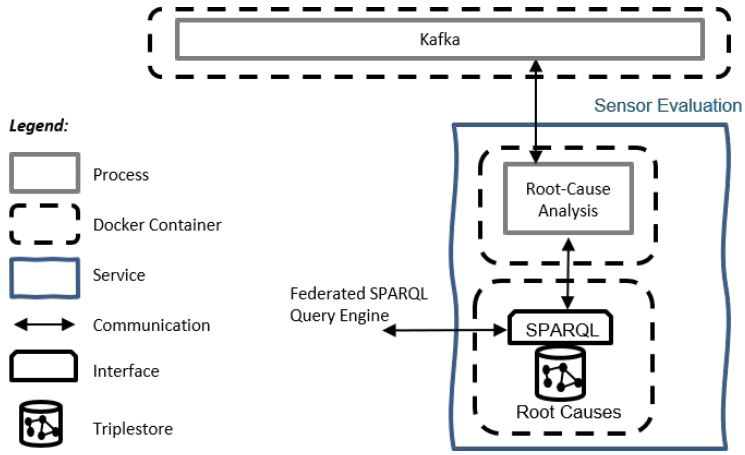


Figure 7.8: Smart Data Service and Communication Infrastructure.

The first step the service performs is clustering the anomalies based on their timely occurrence. For a cluster of anomalies, the root sensor is detected based on the causal relations retrieved from the knowledge graph. For every cluster of anomalies,

a “peti:Incident” is created in the knowledge graph related to the anomalies in the cluster via the “peti:hasRelatedAnomaly” property. Implicit redundant sensors are searched, starting at the identified root sensor. The implicit redundant sensors are checked if they are also detected with an anomaly. If so, a simple majority voting is performed to decide if it is a single sensor fault or an abnormal behavior of the plant is detected. The classification is performed by specifying the “peti:Incident” as a subclass of “peti:AbnormalBehavior” or “peti:SensorFault”, as shown in Figure 7.9. A sensor fault is also related to the identified faulty sensor, which is a subtype of “peti:PCE-Request” in PETIont. The result is stored at the local triplestore, which is connected to the federated query engine. Thus, this information is also accessible by the Smart Data Service and also sent to a Kafka reply-topic encoded as JSON-LD.

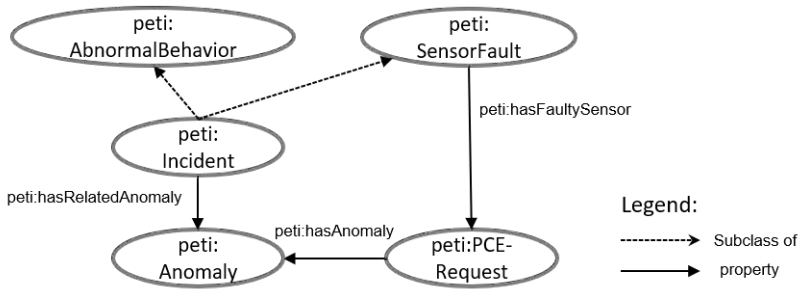


Figure 7.9: Incident classification.

7.4.5 Service Orchestration

The microservice orchestration is performed using the open-source workflow engine Zeebe. This workflow engine allows defining the processes visually in BPMN version 2.0. The communication between Zeebe and the services is based on gRPC Remote Procedure Calls. It is also possible to react to messages from Kafka or other MOMs. For the presented use-case, Kafka Connect is used to establish the connection between Zeebe and the services, as shown in Figure 7.10. Elasticsearch [18] is used by the Zebee workflow engine to store the execution states of the workflows with their internal messages. The Zebee Operate tool is for user interaction, such as visualizing the workflow state of the current execution.

The workflow in BPMN for the sensor evaluation of the presented use-case is depicted in Figure 7.11. The first “service task” is sending a Kafka message to the Anomaly Detection Service, described in Section 7.4.3. Message parameters, such as a correlation ID, or a reply topic, are defined. The “message catch event” is used afterwards to wait for the response of the service. The results of the Anomaly Detection Service are encapsulated in the Zebee message. If no anomalies are found, the process is finished. Otherwise, the results are handed over to the Sensor Evaluation Service, described in Section 7.4.4. Again, the catch message event is used to receive the response from Kafka. Depending on the result, a faulty behavior is logged for further analysis, or the maintenance of

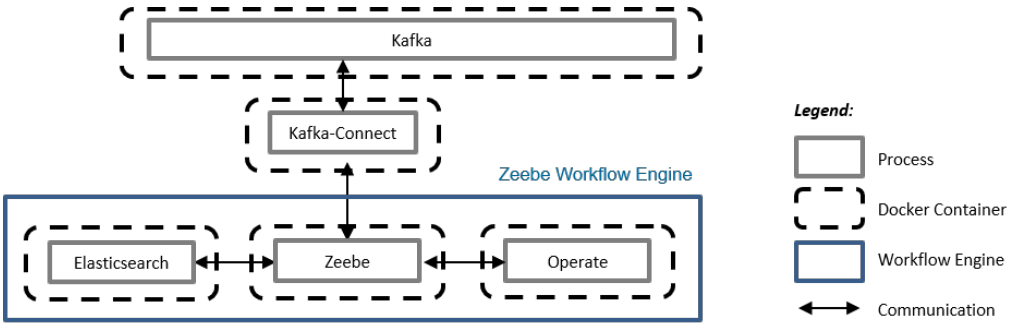


Figure 7.10: Zeebe workflow engine infrastructure.

the faulty sensor is commissioned. Therefore, a service notification can be sent to a service technician. The workflow engine holds the state for maintenance till the service technician confirms the replacement of the sensor.

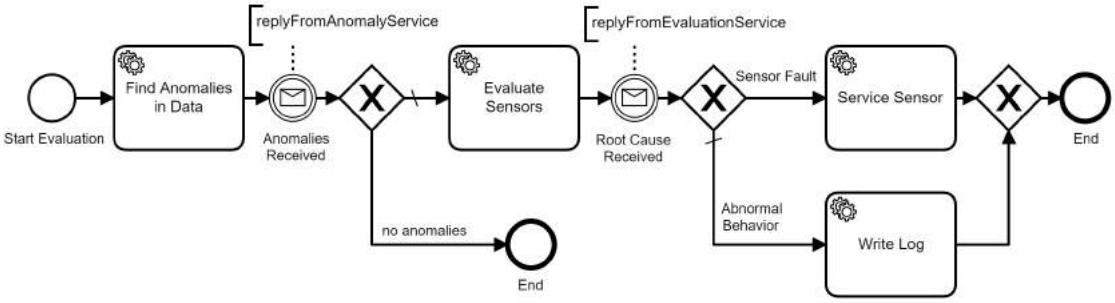


Figure 7.11: Service orchestration for sensor data evaluation.

For the implementation of the logging of abnormal behavior, a simple Zeebe Worker is used. The same applies to the Sensor Service task. The confirmation by a service technician is emulated via the command-line interface of Zeebe.

7.4.6 Results of the Sensor Evaluation Process

Two scenarios, “A” and “B”, are introduced for testing the sensor evaluation service. In both scenarios the mass flow, which is measured by the sensor m_{in}, shows an anomaly. The anomalies of m_{in} are depicted in Figure 7.12 and marked with “A” and “B”.

The first scenario (“A”) is caused by a clogged duct, which results in a reduced mass flow. Therefore, an additional flow resistor is used in the simulation, which is rapidly increased 220 min after the start. As the fan has no closed-loop control, the mass flow decreases rapidly. After 17 min, the resistance is removed so that the mass flow can increase to its normal value again.

The second scenario (“B”) is caused by a faulty mass flow sensor m_{in}, which delivers the wrong values. To make things more difficult for the sensor evaluation service and show

its capability, the faulty sensor values for the mass flow sensor m_{in} are exactly the same as for the anomaly "A", but in that case, the mass flow is not reduced in the simulation. Thus, the sensor evaluation service has to use context information to classify the occurred anomalies correctly. The faulty sensor data starts about 600 min after the start.

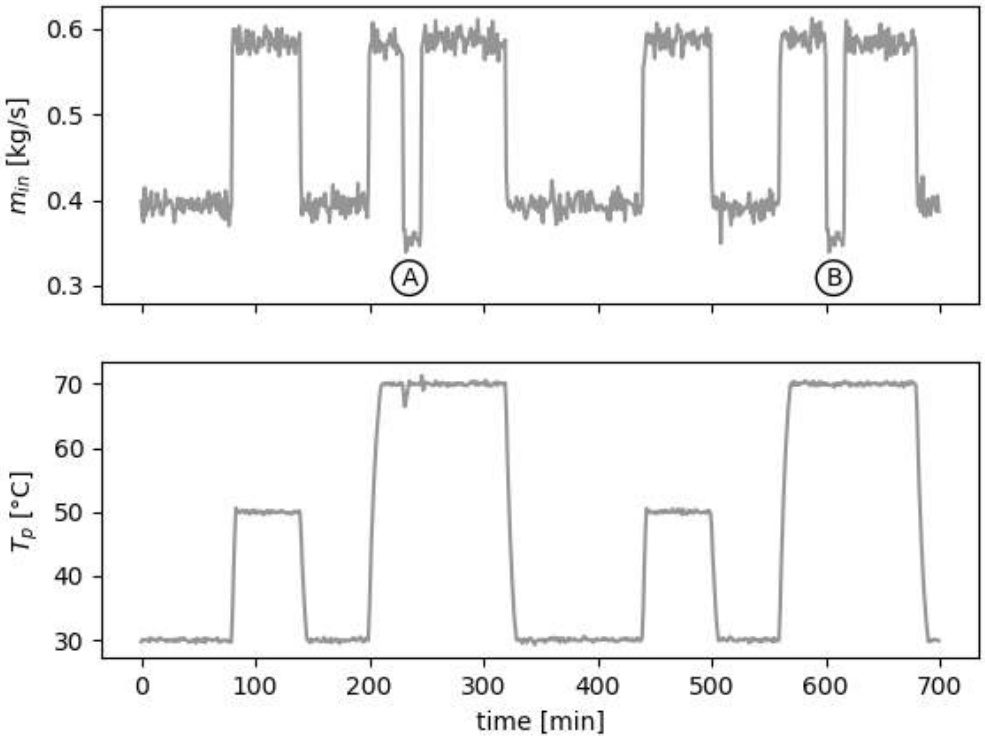


Figure 7.12: Scenarios: sensor fault ("A") and clogged duct ("B").

The workflow shown in Figure 7.10 is invoked with a unique correlation ID, to assign messages to the workflow instance. With that correlation ID, the Data Anomaly Detection service is started. The deviations between the internal ARX model results and the measured values are visualized in Figure 7.13. The periods in time where the deviation is exceeding the threshold are marked as red area. As the internal models are executed in a serial–parallel manner, the unexpected values are propagating through the various models, as shown for the anomaly "B". The simulated values are used as input values within the simulation window. Thus, a clogged duct, which results in a reduced mass flow, shown by the discrepancy between the measured and simulated values of m_{in}, has also influence on the simulated values of T_{hx}, T_{sup}, and T_p during the parallel operation.

The detected anomalies are encoded in JSON-LD, based on the parts of PETIont and OWL-Time, as shown in Figure 7.7, and sent as response. The response is handed over to the Sensor Evaluation Service by the workflow engine. The service analyses the detected anomalies based on causal relations derived from the topology information. The service clusters all identified anomalies into two groups, based on their timely occurrence.

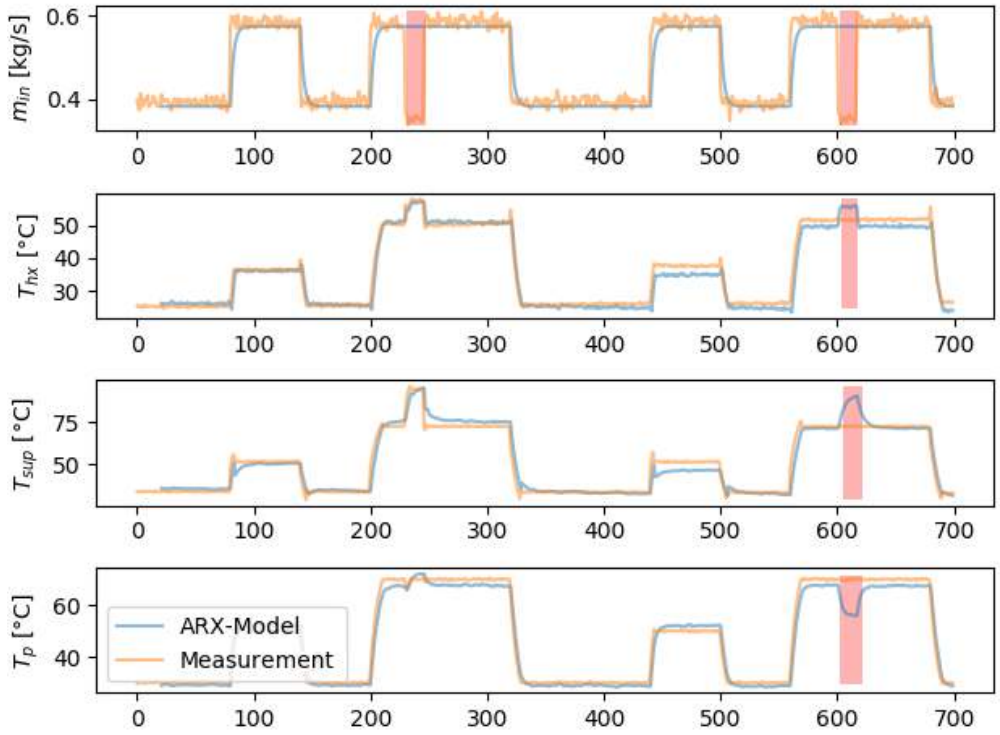


Figure 7.13: Anomalies detected by the Sensor Anomaly Detection Service.

The first cluster or incidence contains only one anomaly (Scenario A). The other sensor values which are influenced by m_{in} are in line with the simulated output of the ARX models. Thus, the measurements are correct and the incident is classified as abnormal behavior of the plant, causing a reduced mass flow m_{in}.

For scenario B, a deviation between the measurement and the simulation is visible also at sensors which are influenced by m_{in} (Figure 7.13). This means there is a discrepancy between the measurements and the simulated ARX model. The causality information from the knowledge graph is used to identify the root cause of the deviations—in this case, a faulty sensor value of m_{in}. A simplified visualization of the causal relations is shown in Figure 7.14. Based on this information, the root sensor m_{in} is identified and the incident is classified as a sensor fault.

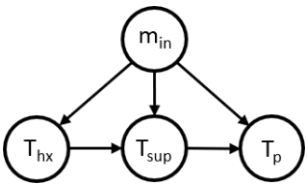


Figure 7.14: Causal relations between sensors.

The identified incidences are classified based on the parts of PETIont, which is shown in Figure 7.9. In the presented use-case scenario, "A" is correctly classified as abnormal behavior and "B" as sensor fault. The result is encoded as JSON-LD and returned as a reply to the workflow engine over Kafka.

As mentioned before, the other services for logging and interaction with a service technician are only implemented as mock-up.

7.5 Discussion

The proposed DT service framework architecture was evaluated with the proof-of-concept implementation for the presented use-case. The service interaction combined with the shared and federated knowledge graph and the service orchestration with the workflow engine has been shown.

The presented functional and non-functional requirements for a DT service framework do not form a comprehensive list, but they can be used as a starting point for the implementation. In particular, non-functional requirements depend on the concrete implementation and cannot be fulfilled by a general architecture. However, the architecture can support their achievement. An essential non-functional requirement was not targeted by the proposed architecture, as it has to be considered inherently—security. Nevertheless, security has to be considered in DT applications during the whole life-cycle.

The proposed DT service framework architecture can be extended for particular use-cases. The proof-of-concept implementation for the sensor data evaluation for a DT used open-source tools to fulfill the identified requirements. Table 7.5 shows design artifacts and the requirements which were supported to be met by the specific artifact.

The microservice architectural style supports the maintainability of the whole framework (RN2, RN4), as they can be extended and developed by individual teams. Furthermore, microservice can be easily containerized, which facilitates the hosting in a cloud infrastructure as well as on-premise (RN1).

Inter-service communication is a crucial part of the whole service framework. Using a MOM with event-based messaging is also beneficial for decoupling microservices (RN2) as well as enable 1:n and n:1 communication (RF2, RF3). Apache Kafka, used in the proof-of-concept implementation, further supports the data streams (RF1), which are stored persistently in Kafka (RN5). Thus, connecting services can also access previously sent data. Kafka topics are used to enable request–reply communication between services (RF4). Furthermore, it supports scalability by running multiple brokers on a server cluster (RN3).

Table 7.5: Requirements supported by design artifacts.

Design Artifact	Supported Requirements
Microservice Architecture	RN2, RN4
Containerization	RN2, RN4
MOM (Apache Kafka)	RN2, RN3, RN5, RF1, RF2, RF3, RF4
Shared Knowledge graph	RI1, RI5
Federated Query Engine	RI4, RI3
OBDA	RI2
Workflow Engine	RB1, RB2

Semantic Web technology can be utilized to perform data integration (RI1) and build up the shared knowledge graph. Based on the ontologies used in the knowledge graph, data can be exchanged between services and external clients using JSON-LD (RI5). Using standard ontological models is vital to provide interoperability. A hierarchical structure of the used ontology, with an upper, a domain, and task ontology, can help to provide a common understanding of concepts. Several ontology integration methods exist for such cases as described in [17]. As there will be hardly a consensus about the ontologies to be used, topics such as ontology alignment and mapping will become more important. Additionally, proper ontology design methods, such as those presented in [23], facilitate the reusability of these ontologies.

The federated query engine enables the distribution of the knowledge graph. Every service of the DT manages its information and can decide which part should be shared with other services by including it in the knowledge graph. The parts which are not included are only available locally by the service itself (RI3, RI4). However, reasoning on the shared knowledge graph can introduce problems because of inconsistency. As services can add relevant information by themselves to the shared knowledge graph, this problem becomes more likely. To avoid this problem, reasoning should only be applied to private parts of the knowledge graph, where consistency can be guaranteed to some extent. If reasoning should be performed on the whole graph, other techniques have to be applied which can deal with uncertainty, e.g., fuzzy reasoning methods. Even if reasoning capabilities can not be provided, the knowledge graph can still be used for data retrieval and knowledge discovery.

Using a workflow engine for service orchestration is beneficial for holding states during human–machine interaction. Using BPMN also facilitates the integration DT services into workflows at enterprises level. For the production level planning, BPMN is not always sufficient and other notations such as Coloured Petri Nets might be more suitable [31]. Extensions of the workflow engine could support this multi-level workflow execution in a more convenient way.

Future work will investigate the real-time performance of the proposed service architecture for various use-cases. In this context, the integration of OPC UA is planned. How OPC

UA information can be included in the shared knowledge graph has already been shown in [45]. Next, OPC UA integration into the service framework will be investigated. In this context, the execution time of SPARQL endpoints can become relevant, which will be further investigated.

References

[1] Sailesh Abburu et al. „COGNITWIN – Hybrid and Cognitive Digital Twins for the Process Industry“. In: *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. July. IEEE, June 2020, pp. 1–8. ISBN: 978-1-7281-7037-4. DOI: 10.1109/ICE/ITMC49519.2020.9198403. URL: https://ieeexplore.ieee.org/document/9198403/.

[2] Peter Adolphs et al. *Reference Architecture Model Industrie 4.0 (RAMI4.0)*. Tech. rep. July. VDI/VDE-Gesellschaft, 2015.

[3] Ameer B.A. Alaasam, Gleb Radchenko, and Andrey Tchernykh. „Stateful stream processing for digital twins: Microservice-based kafka stream dsl“. In: *SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings* (2019), pp. 804–809. DOI: 10.1109/SIBIRCON48586.2019.8958367.

[4] Kazi Masudul Alam and Abdulmotaleb El Saddik. „C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems“. In: *IEEE Access* 5 (2017), pp. 2050–2062. ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2657006.

[5] Sajjad Ali, Muhammad Aslam Jarwar, and Ilyoung Chong. „Design methodology of microservices to support predictive analytics for IoT applications“. In: *Sensors (Switzerland)* 18.12 (2018). ISSN: 14248220. DOI: 10.3390/s18124226.

[6] Apache Software Foundation. *Apache ActiveMQ - Flexible & Powerful Open Source Multi-Protocol Messaging*. https://activemq.apache.org/. (accessed: 19.05.2021).

[7] Behrang Ashtari Talkhestani et al. „An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System“. In: *At-Automatisierungstechnik* 67.9 (2019), pp. 762–782. ISSN: 2196677X. DOI: 10.1515/auto-2019-0039.

[8] Kirill Borodulin et al. „Towards digital twins cloud platform: Microservices and computational workflows to rule a smart factory“. In: *UCC 2017 - Proceedings of the10th International Conference on Utility and Cloud Computing* (2017), pp. 205–206. DOI: 10.1145/3147213.3149234.

[9] Camunda. *Zeebe Workflow Engine for Microservices Orchestration*. https://github.com/camunda-cloud/zeebe. (accessed: 12.05.2021).

[10] Chiara Cimino, Elisa Negri, and Luca Fumagalli. „Review of digital twin applications in manufacturing“. In: *Computers in Industry* 113 (2019), p. 103130. ISSN: 01663615. DOI: 10.1016/j.compind.2019.103130. URL: https://doi.org/10.1016/j.compind.2019.103130.

[11] Violeta Damjanovic-Behrendt and Wernher Behrendt. „An open source approach to the design and implementation of Digital Twins for Smart Manufacturing“. In: *International Journal of Computer Integrated Manufacturing* 32.4-5 (2019), pp. 366–384. ISSN: 13623052. DOI: 10.1080/0951192X.2019.1599436. URL: https://doi.org/10.1080/0951192X.2019.1599436.

[12] Lorenzo De Lauretis. „From monolithic architecture to microservices architecture“. In: *Proceedings - 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019* (2019), pp. 93–96. DOI: 10.1109/ISSREW.2019.00050.

[13] Kai Ding et al. „Defining a Digital Twin-based Cyber-Physical Production System for autonomous manufacturing in smart shop floors“. In: *International Journal of Production Research* 57.20 (2019), pp. 6315–6334. ISSN: 1366588X. DOI: 10.1080/00207543.2019.1566661. URL: https://doi.org/00207543.2019.1566661.

[14] Docker. *Docker - Accelerate how you build, share and run modern applications.* https://www.docker.com/. (accessed: 19.05.2021).

[15] Nicola Dragoni et al. „Microservices: Yesterday, Today, and Tomorrow“. In: *Present and Ulterior Software Engineering*. Cham: Springer International Publishing, 2017, pp. 195–216. ISBN: 978-3-319-67425-4. DOI: 10.1007/978-3-319-67425-4_12. URL: https://doi.org/10.1007/978-3-319-67425-4_12.

[16] Lisa Ehrlinger and Wolfram Wöß. „Towards a definition of knowledge graphs“. In: *CEUR Workshop Proceedings* 1695 (2016). ISSN: 16130073.

[17] Fajar J Ekaputra et al. „Ontology-Based Data Integration in Multi-Disciplinary Engineering Environments: A Review“. In: *Open Journal of Information Systms* 4.1 (2017), pp. 1–26. ISSN: 2198-9281.

[18] Elastic. *Elasticsearch - distributed, multitenant-capable full-text search engine.* https://www.elastic.co/elasticsearch/. (accessed: 19.05.2021).

[19] Maximilian Engelsberger and Thomas Greiner. „Software architecture for cyber-physical control systems with flexible application of the software-as-a-service and on-premises model“. In: *Proceedings of the IEEE International Conference on Industrial Technology* 2015-June.June (2015), pp. 1544–1549. DOI: 10.1109/ICIT.2015.7125316.

[20] Sheikmohammadmostakim Fattah et al. „Building IoT services for aging in place using standard-based IoT platforms and heterogeneous iot products“. In: *Sensors (Switzerland)* 17.10 (2017), pp. 1–29. ISSN: 14248220. DOI: 10.3390/s17102311.

[21] Apache Software Foundation. *Apache Kafka.* https://kafka.apache.org/. (accessed: 12.05.2021).

[22] Peter Fritzson et al. „The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development“. In: *Modeling, Identification and Control* 41.4 (2020), pp. 241–295. DOI: 10.4173/mic.2020.4.1.

[23] Thomas Frühwirth, Wolfgang Kastner, and Lukas Krammer. „A methodology for creating reusable ontologies“. In: *Proceedings - 2018 IEEE Industrial Cyber-Physical Systems, ICPS 2018* (2018), pp. 65–70. DOI: 10.1109/ICPHYS.2018.8387639.

[24] Dominic Gorecky et al. „Human-machine-interaction in the industry 4.0 era“. In: *Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014* (2014), pp. 289–294. DOI: 10.1109/INDIN.2014.6945523.

[25] Michael Grieves. „Digital Twin : Manufacturing Excellence through Virtual Factory Replication This paper introduces the concept of a A Whitepaper by Dr . Michael Grieves“. In: *White Paper* March (2014). URL: https://www.researchgate.net/publication/275211047%7B%5C_%7DDigital%7B%5C_%7DTwin%7B%5C_%7DManufacturing%7B%5C_%7DExcellence%7B%5C_%7Dthrough%7B%5C_%7DVirtual%7B%5C_%7DFactory%7B%5C_%7DReplication.

[26] Antonio Manuel Gutiérrez–Fernández, Manuel Resinas, and Antonio Ruiz–Cortés. „Redefining a process engine as a microservice platform“. In: *Lecture Notes in Business Information Processing* 281 (2017), pp. 252–263. ISSN: 18651348. DOI: 10.1007/978-3-319-58457-7_19.

[27] Jerry R. Hobbs and Chris Little. *Time Ontology in OWL. W3C Candidate Recommendation*. Tech. rep. World Wide Web Consortium (W3C), 2020. URL: https://www.w3.org/TR/owl-time/.

[28] Klementina Josifovska, Enes Yigitbas, and Gregor Engels. „Reference Framework for Digital Twins within Cyber-Physical Systems“. In: *Proceedings - 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems, SEsCPS 2019* (2019), pp. 25–31. DOI: 10.1109/SEsCPS.2019.00012.

[29] Martin Kleppmann. *Designing Data-Intensive Applications*. O'Reilly Media, Inc., 2017. ISBN: 9781449373320.

[30] Daniel Kozma, Pal Varga, and Felix Larrinaga. „Dynamic Multilevel Workflow Management Concept for Industrial IoT Systems“. In: *IEEE Transactions on Automation Science and Engineering* (2020), pp. 1–13. ISSN: 1545-5955. DOI: 10.1109/tase.2020.3004313.

[31] Dániel Kozma, Pál Varga, and Felix Larrinaga. „Data-driven Workflow Management by utilising BPMN and CPN in IIoT Systems with the Arrowhead Framework“. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2019-September (2019), pp. 385–392. ISSN: 19460759. DOI: 10.1109/ETFA.2019.8869501.

[32] Yuqian Lu et al. „Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues“. In: *Robotics and Computer-Integrated Manufacturing* 61.July 2019 (2020), p. 101837. ISSN: 07365845. DOI: 10.1016/j.rcim.2019.101837. URL: https://doi.org/10.1016/j.rcim.2019.101837.

[33] Somayeh Malakuti et al. „Digital Twins for Industrial Applications. Definition, Business Values, Design Aspects, Standards and Use Cases“. In: *White Paper* (2020), pp. 1–19.

[34] James Moyne et al. „A Requirements Driven Digital Twin Framework: Specification and Opportunities“. In: *IEEE Access* 8 (2020), pp. 107781–107801. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3000437.

[35] Sam Newman. *Building Microservices - Design Fine Grained Systems*. O'Reilly Media, Inc., 2021. ISBN: 9781492034025.

[36] Dmitri Panfilenko et al. „BPMN for knowledge acquisition and anomaly handling in CPS for smart factories“. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2016-November (2016), pp. 2–5. ISSN: 19460759. DOI: 10.1109/ETFA.2016.7733686.

[37] Vinit Parida, David Sjödin, and Wiebke Reim. „Reviewing literature on digitalization, business model innovation, and sustainable industry: Past achievements and future promises“. In: *Sustainability (Switzerland)* 11.2 (2019). ISSN: 20711050. DOI: 10.3390/su11020391.

[38] Eclipse RDF4J. *Federation With FedX*. https://rdf4j.org/documentation/programming/federation/. (accessed: 12.05.2021).

[39] Chris Richardson. *Microservices Patterns*. Manning Publications, 2018, p. 520. ISBN: 9781617294549.

[40] Ahmed Saad, Samy Faddel, and Osama Mohammed. „IoT-based digital twin for energy cyber-physical systems: design and implementation“. In: *Energies* 13.18 (2020). ISSN: 19961073. DOI: 10.3390/en13184762.

[41] Daniel Schachinger, Wolfgang Kastner, and Stefan Gaida. „Ontology-based abstraction layer for smart grid interaction in building energy management systems“. In: *2016 IEEE International Energy Conference, ENERGYCON 2016* (2016). DOI: 10.1109/ENERGYCON.2016.7513991.

[42] Andreas Schwarte et al. „FedX: Optimization techniques for federated query processing on linked data“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7031 LNCS.PART 1 (2011), pp. 601–616. ISSN: 03029743. DOI: 10.1007/978-3-642-25073-6_38.

[43] Gernot Steindl. *Digital Twin Service Framework*. https://github.com/Smart-Industrial-Concept/DigitalTwinServiceFramework. (accessed: 19.05.2021).

[44] Gernot Steindl. *Heating Process Simulation*. https://github.com/Smart-Industrial-Concept/HeatingProcessSimulation. (accessed: 19.05.2021).

[45] Gernot Steindl, Thomas Früwirth, and Wolfgang Kastner. „Ontology-Based OPC UA Data Access via Custom Property Functions“. In: *24th Interantional Conference on Emerging Technologies and Factory Automation*. Zaragoza, Spain, 2019.

[46] Gernot Steindl and Wolfgang Kastner. „Ontology-Based Model Identification of Industrial Energy Systems“. In: *IEEE International Symposium on Industrial Electronics* 2020-June (2020), pp. 1217–1223. DOI: 10.1109/ISIE45063.2020.9152386.

[47] Gernot Steindl and Wolfgang Kastner. „Query Performance Evaluation of Sensor Data Integration Methods for Knowledge Graphs“. In: *6th IEEE International Conference on Big Data, Knowledge and Control Systems Engineering*. Sofia, 2019.

[48] Gernot Steindl et al. „Generic digital twin architecture for industrial energy systems“. In: *Applied Sciences (Switzerland)* 10.24 (2020), pp. 1–20. ISSN: 20763417. DOI: 10.3390/app10248903.

[49] Rudi Studer, V Richard Benjamins, and Dieter Fensel. „Knowledge Engineering: Principles and methods“. In: *Data & Knowledge Engineering* 25 (1998), pp. 161–197.

[50] Andreas Stutz et al. „Orchestration vs. Choreography Functional Association for Future Automation Systems“. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 8268–8275. ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.12.1961. URL: https://doi.org/10.1016/j.ifacol.2020.12.1961.

[51] Fei Tao, Meng Zhang, and A.Y.C. Nee. „Five-Dimension Digital Twin Modeling and Its Key Technologies“. In: *Digital Twin Driven Smart Manufacturing* (2019), pp. 63–81. DOI: 10.1016/b978-0-12-817630-6.00003-5.

[52] Fei Tao et al. „Digital Twin in Industry: State-of-the-Art“. In: *IEEE Transactions on Industrial Informatics* 15.4 (2019), pp. 2405–2415. ISSN: 15513203. DOI: 10.1109/TII.2018.2873186.

[53] Fei Tao et al. „Digital twin-driven product design, manufacturing and service with big data“. In: *International Journal of Advanced Manufacturing Technology* 94.9-12 (2018), pp. 3563–3576. ISSN: 14333015. DOI: 10.1007/s00170-017-0233-1.

[54] The PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. https://www.postgresql.org/. (accessed: 12.05.2021).

[55] Alfred Theorin et al. „An event-driven manufacturing information system architecture for Industry 4.0“. In: *International Journal of Production Research* 55.5 (2017), pp. 1297–1311. ISSN: 1366588X. DOI: 10.1080/00207543.2016.1201604.

[56] World Wide Web Consortium. *Semantic Sensor Network Ontology. W3C Recommendation*. https://www.w3.org/TR/vocab-ssn/. (accessed: 19.05.2021).

[57] Guohui Xiao et al. „The virtual knowledge graph system ontop“. In: *CEUR Workshop Proceedings* 2663 (2020), pp. 1–16. ISSN: 16130073.

CHAPTER 8

Conclusion & Future Work

In this thesis, methods and concepts for creating DTs in the domain of industrial energy systems were presented. The already specified goals of the thesis, defined in Chapter 1.2, are used to discuss the results and identify future research directions.

G1: Develop a generic Digital Twin architecture for industrial energy systems
The novel GDTA is introduced and evaluated based on a prototypical proof-of-concept implementation. The four general requirements for a DT in the industrial energy system domain, and relevant for this thesis are stated in Chapter 1.4.2. It has been demonstrated how the GDTA can be used to instantiate a DT for the use case of a PBTES, fulfilling requirement GRQ1. The concept of a shared knowledge, based on Semantic Web technologies, enables the integration and interlinking of external data sources (GRQ2) and also provides machine-readable semantics for semi-automatic information processing (GRQ3). The requirement for seamlessly adding or adopting the DT's functionality is accomplished by facilitating a service-based architecture (GRQ4).

It has been shown that in the related literature, layered structures with similar functionality but different names are frequently used in other DT concepts, architectures, or frameworks. The different names and their functionality are elaborated in Chapter 2.1.2. The proposed GDTA is aligned with the IT dimension of RAMI 4.0 to solve this problem. This facilitates the development of a DT by allowing for a common naming and understanding of the layers within the GDTA.

Simulation models are essential parts of the DT, but they heavily depend on the targeted application and task. Therefore, the DT has to be able to deal with various models, which also can change over time and provide context information with the models. For the proof-of-concept implementation of the GDTA, Semantic Web technology is used for combining such a simulation model with context information about resources and services. This combination enables application-dependable views on the DT's Virtual Entity. Ontologies serve as the foundation for the shared knowledge, which is the basis

of the Smart Data Service. In this shared knowledge, information from the whole DT life-cycle can be managed and interlinked. Even if the GDTA was designed based on a use case of an industrial energy system, it is likely that it also can be applied in other domains. Its suitability will be tested and evaluated in future research projects. Also, the evaluation of the GDTA, which is only based on one use case, might not be sufficient. Thus the application of the GDTA for other types of energy systems will also be elaborated. Therefore, an upcoming research project in the domain of wind power generation will be used.

G2: Reusing existing OPC UA information models to provide semantic context information for the Digital Twin
OPC UA is currently one of the main technologies used in IIoT applications. With a proof of concept, the proposed approach for transforming OPC UA information models into domain-specific ontologies is evaluated. These ontologies can be included in the shared knowledge graph of the GDTA to make the information available for services inside the DT. The transformation process can be adapted to other OPC UA information models or domain ontologies by simply changing the SPARQL rules. The remaining steps of the process can be automated, making it adaptable to changes in the source model or target ontology. Additionally, because the information model is directly retrieved from the address space of a running OPC UA server, the proposed approach is also applicable to already existing OPC UA servers. The presented transformation method has only be evaluated on a single proof-of-concept to show its applicability. Further evaluations will be done in the future by applying this method to other OPC UA information models as well as domain ontologies in ongoing research projects.

G3: Provide a semantic integration method for OPC UA run-time data of a Digital Twin
Next to engineering information provided by the OPC UA information models, it is also important for a DT to have access to run-time data from the underlying process. Thus, it was demonstrated how CPF can be utilized to create ontology-based OPC UA data access. This enables the integration of OPC UA run-time data into the shared knowledge graph as proposed in the GDTA. It has been demonstrated how these data can be retrieved from a knowledge graph using SPARQL.

Additionally, a performance evaluation and comparison of the CPF approach with two other methods for sensor data integration into a knowledge graph were carried out. All three approaches were evaluated for their characteristics and the resulting limitations were discussed. Therefore, two corner cases are presented which show the strengths and weaknesses of the individual approaches. It has been shown that the Ontop framework should be preferred if the sensor data is already stored externally in an SQL compliant relational database. The CPF can be used if the data sources are not SQL-compliant, such as special time-series databases or OPC UA servers. It should be avoided to store time-series data directly in an ontology, as the query performance degrades over time. However, it is still quite efficient for certain use cases or prototypes, where only a small amount of data is produced.

G4: Providing a method for supporting semi-automatic data-driven model identification

To show an application of how information and semantically enriched run-time data can be used by a DT, the simulation model identification process is partly automated, based on ontological knowledge of plant equipment, topology, instrumentation, and run-time data. SPARQL rules were used to infer new knowledge, like causality relations. In combination with simulation models, such causality information can be used to develop novel services for DTs. It can also enhance the explainability of ICPS, e.g., by applying it to fault detection and analysis scenarios. Further use cases will be explored in the future, extending this approach to electrical energy systems to evaluate the full potential of such an approach.

G5: Developing an architectural concept for a Digital Twin service framework
The presented service framework is based on the microservice architectural style. This facilitates maintainability and extendability. Microservices may also be containerized, making them easier to host in both cloud and on-premise environments. The shared knowledge graph of the GDTA can be distributed thanks to a federated query engine. Each DT service handles its own data and information. It can select which parts of it should be shared with other services by providing it to the knowledge graph. It has been shown that for holding states during long-lived transactions, like human-machine interactions, a workflow engine for service orchestration can be beneficial. Also, the usage of BPMN makes it easier to integrate DT services into enterprise workflows. It has to be mentioned that requirements regarding real-time capability for the service framework are not addressed, as the used communication network technologies and operating systems are not capable of that. Interaction between services and the plant only happens by sending set points to the underlying control system. The underlying control system has to be able to deal with delayed set points without reaching critical system states. If a real-time response is needed, this has to be implemented directly in the control system and using real-time OT communication.

Also other future research directions are manifold in the area of DTs. For the DT architecture, the other two dimensions of RAMI 4.0 should be further investigated. On the hierarchy dimension, the application of edge, fog, and cloud computing has to be analyzed. Also, managing information over the whole life-cycle of the DT and perform change management is an important topic, which needs further intensive research.

In the area of knowledge graphs suitable for industrial applications, machine learning approaches can be applied for analyzing the graph and to detect issues, like inconsistencies or missing links. A drawback when working with ontologies and knowledge graphs compared to machine learning methods is that they cannot deal with uncertainty. Thus, fuzzy reasoning might be an interesting line of attack in the area of knowledge graphs, introducing the concept of impreciseness into reasoning. Such approaches might be worth to be investigated in the near future.

Also, it is an important question, how concepts and methods presented in this thesis can be combined with the ongoing standardization activities regarding the AAS. In recent years, the concept of the AAS has matured. Next to its data model, the interface description of an AAS has also been defined. In this context, the concept of semantic

IDs, defined by the AAS data model, can be used to refer to ontological knowledge representations. Also, the standardized RDF export format of the AAS is useful to integrate the information available in an AAS into an knowledge graph. This integration enables the interlinking of information from the whole life-cycle and all hierarchy levels of the RAMI 4.0, to support knowledge-discovery with the help of Semantic Web technology.

List of Figures

List of Tables

Acronyms

5D-DT Five-Dimensional Digital Twin.

AAS Asset Administration Shell.

AI Artificial Intelligence.

API Application Programming Interface.

ARX autoregressive with exogenous input.

BACnet Building Automation and Control Network.

BIM Building Information Model.

BPMN Business Process Model and Notation.

CPF Custom Property Function.

CPPS Cyber-Physical Production System.

CPS Cyber-Physical System.

CPSs Cyber-Physical Systems.

DT Digital Twin.

ESB Enterprise Service Bus.

GDTA Generic Digital Twin Architecture.

HMI Human-Machine Interface.

HTTP Hypertext Transfer Protocol.

ICPS Industrial Cyber-Physical System.

ICT Information and Communication Technology.

IIoT Industrial Internet of Things.

IIRA Industrial Internet Reference Architecture.

IoT Internet of Things.

IT Information Technology.

JSON-LD JSON for Linking Data.

MOM Message-oriented Middleware.

OBDA Ontology-Based Data Access.

ODBI Ontology-Based Data Integration.

OPC Open Process Control.

OPC UA OPC Unified Architecture.

OWL Web Ontology Language.

OWL-S Ontology Web Language for Service.

P&I Piping and Instrumentation.

PBTES Packed-Bed Thermal Energy Storage.

PCE Process Control Engineering.

PETIont Plant Equipment, Topology and Instrumentation Ontology.

QoS Quality of Service.

RAMI 4.0 Reference Architecture Model Industry 4.0.

RDF Resource Description Framework.

RDFS Resource Description Framework Schema.

REST Representational State Transfer.

SOSA Sensor, Observation, Sample, and Actuator.

SPARQL SPARQL Protocol and RDF Query Language.

SQL Structured Query Language.

SSN Semantic Sensor Network.

URI Uniform Resource Identifier.

VM Virtual Machine.

W3C World Wide Web Consortium.