NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

DIN Spec 91345 RAMI 4.0 compliant data pipelining: An approach to support data understanding and data acquisition in smart manufacturing environments

Kevin Pierre Nagorny

Master in Industrial Informatics

DOCTORATE IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
November, 2021

DEPARTAMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

DIN Spec 91345 RAMI 4.0 compliant data pipelining:
An approach to support data understanding and data acquisition in smart manufacturing environments

Kevin Pierre Nagorny

Master of Engineering in Industrial Informatics

Adviser: José António Barata de Oliveira
Full Professor, NOVA University Lisbon

Co-adviser: Armando Walter Colombo
Full Professor, University of Applied Sciences Emden/Leer

Examination Committee:

Chair: Luís Manuel Camarinha de Matos,
Full Professor, FCT-NOVA

Opponents: Paulo Jorge Pinto Leitão,
Full Professor, Polytechnic Institute of Bragança
Luís Domingos Ferreira Ribeiro,
Assistant Professor, Linköping University

Adviser: José António Barata de Oliveira,
Full Professor, NOVA University Lisbon

Members: Paulo Jorge Freitas de Oliveira Novais,
Full Professor, University of Minho
Luís Manuel Camarinha de Matos,
Full Professor, FCT-NOVA

DOCTORATE IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
November, 2021

A DIN Spec 91345 RAMI 4.0 compliant data pipelining model: An approach to support data understanding and data acquisition in smart manufacturing environments

Acknowledgements

These are the last sentences I am writing for this dissertation. At the same time, it is the end of a journey, the closure of a chapter in my life and an opportunity for reflection and appreciation.

My last decade would have been completely different without my co-supervisor Professor Armando Walter Colombo, whom I met in Emden (Germany) as one of his master students. Through him I went to the Institute for Applied Systems Technology Bremen (ATB), came in contact with Professor José Barata and became a PhD student in Lisbon. What an honour that you made this journey possible! Thank you for your outstanding professional and emotional support, for putting so much effort into me, for being a role model with your passion and commitment, for helping to overcome obstacles and for encouraging me the whole time. I am immeasurably grateful.

A special thanks to Professor José Barata, my supervisor in Lisbon, who accepted me as his remote PhD student from Germany. Thank you for enabling this international PhD journey, for your guidance and wise recommendations along the whole road and for all your support in this work. What a beautiful story of my life, to do a PhD at such an internationally recognised university in beautiful Lisbon, Portugal. You made this possible and I will never forget that.

In this context, I would also like to express my appreciation to the FCT PhD program team. An excellent program, recommendable without any hesitation, offering a good mix of relevant hard and soft skills, taught by a passionate and highly regarded team. I learned a lot from you and feel very prepared for the coming challenges.

Many thanks to my colleagues in the ATB. Thank you for the provision of an excellent research and technological development environment and a huge international network of academic and industrial partners. You have already brought me into seven European projects and several project proposals. Thank you for all the things I learned so far and for letting me see Europe and its people on several business trips. You have made a valuable contribution to my personal development.

Also, many thanks to my proof-readers who have spent hours/days/weeks with me and my dissertation to give me tips for improvements. Hayat and Wladimir, Rebecca Siafaka, Susanne Lane and Ana Correia, thank you very much for doing this and helping me to enhance the quality of this work.

How worthwhile would life be without a good social fundament? Cheers to all my dear friends who supported me on this journey: Anna-Lena and Christian; Carina and Thais; Feetu Nyrhinen; Hayat and Wladimir; Ines Zimmermann; Iris-Maria Kersten; Janina and Matthias; Maria Sell; Nadja, Patrick and my godchild Jakob; Mounir, Nathalie and Louisa; Rebecca Siafaka; Susanne and Philipp; and my colleagues from Emden. Thank you all for the beautiful and joyful time we spent and also for the fruitful discussions that influenced the development of this work.

A big thanks to my buddy Claas Lühring, well knowing that I wouldn't have come this far without him. An honour to have such a good friend.

Last but not least, an appreciation to the strongest pillar in my life - my family. Thanks to my mother for leading me on this path, for building me up when I was down, for listening and understanding and for being a good heart. Thanks to my father for being a role model for ambition, discipline and commitment. Thanks to my sister for being a good friend. Thank you all for always encouraging me, for giving me guidance and support when needed, and for being a safe harbour, whatever comes.

Abstract

Today, data scientists in the manufacturing domain are confronted with a set of challenges associated to data acquisition as well as data processing including the extraction of valuable information to support both, the work of the manufacturing equipment as well as the manufacturing processes behind it.

One essential aspect related to data acquisition is the pipelining, including various communication standards, protocols and technologies to save and transfer heterogenous data. These circumstances make it hard to understand, find, access and extract data from the sources depending on use cases and applications.

In order to support this data pipelining process, this thesis proposes the use of the semantic model. The selected semantic model should be able to describe smart manufacturing assets themselves as well as to access their data along their life-cycle.

As a matter of fact, there are many research contributions in smart manufacturing, which already came out with reference architectures or standards for semantic-based meta data description or asset classification. This research builds upon these outcomes and introduces a novel semantic model-based data pipelining approach using as a basis the Reference Architecture Model for Industry 4.0 (RAMI 4.0).

Keywords: Data Pipeline, Industry 4.0, RAMI 4.0, Smart Manufacturing, Semantics

Sumário

Hoje em dia, os cientistas de dados no domínio da manufatura são confrontados com várias normas, protocolos e tecnologias de comunicação para gravar, processar e transferir vários tipos de dados. Estas circunstâncias tornam difícil compreender, encontrar, aceder e extrair dados necessários para aplicações dependentes de casos de utilização, desde os equipamentos aos respectivos processos de manufatura.

Um aspecto essencial poderia ser um processo de canalisação de dados incluindo vários normas de comunicação, protocolos e tecnologias para gravar e transferir dados. Uma solução para suporte deste processo, proposto por esta tese, é a aplicação de um modelo semântico que descreva os próprios recursos de manufactura inteligente e o acesso aos seus dados ao longo do seu ciclo de vida.

Muitas das contribuições de investigação em manufatura inteligente já produziram arquitecturas de referência como a RAMI 4.0 ou normas para a descrição semântica de meta dados ou classificação de recursos. Esta investigação baseia-se nestas fontes externas e introduz um novo modelo semântico baseado no Modelo de Arquitectura de Referência para Indústria 4.0 (RAMI 4.0), em conformidade com a abordagem de canalisação de dados no domínio da produção inteligente como caso exemplar de utilização para permitir uma fácil exploração, compreensão, descoberta, selecção e extracção de dados.

Palavras-chave: Pipeline de Dados, Indústria 4.0, RAMI 4.0, Manufatura Inteligente, Semântica

Summary

The approach presented addresses today's challenges of understanding and collecting data across the broad spectrum of available data in (smart) manufacturing environments. Various data types and data models, not always related to each other, produced in different volumes and velocities, saved and transferred in form of data buckets and continuous or event-based data streams, using different communication protocols, languages and interfaces are today normal conditions in manufacturing environments. This situation is one of the reasons why data understanding, selection and extraction is an increasing time intensive activity.

The presented approach does not propose a rebuilding of conventional manufacturing environments to force a harmonisation of data and data related infrastructures. The approach takes the current situation as a pre-condition which will exist probably for further decades and tries to deliver answers for an on top solution that can be applied in conventional but also in newer smart manufacturing systems.

The doctoral work describes a DIN Spec 91345 RAMI 4.0 compliant data pipelining approach to support data understanding and data extraction in (smart) manufacturing environments. The approach uses a semantic model that describes assets and data along their life-cycle to support their exploration in a smart manufacturing environment. Basis for the semantic model is the RAMI 4.0 as one of the most popular reference architectures and further standards used to describe smart manufacturing environments on a metadata level. An additional description describes the access to available data which enables the overall data pipelining approach to extract needed data by using traditional data pipelining solutions.

Presented are the concept and approach for DIN Spec 91345 RAMI 4.0 compliant data pipelining which is further divided into the semantic model, a software architecture and a validated TRL-6 prototype.

The results of this doctoral work contribute to research areas that deal with data management in smart manufacturing systems and delivers ideas to build, handle and exploit self-descriptive smart manufacturing environments based on semantic models and to link these models with available data along life-cycles of assets located in such environments.

Resumo

A abordagem apresentada aborda os desafios actuais de compreensão e recolha de dados através do vasto espectro de dados disponíveis no ambiente (inteligente) de manufatura. Vários tipos de dados e modelos de dados, nem sempre relacionados entre si, produzidos em diferentes volumes e velocidades, guardados e transferidos sob a forma de baldes de dados e fluxos de dados contíguos ou baseados em eventos, utilizando diferentes protocolos de comunicação, línguas e interfaces são hoje condições normais nos ambientes de manufatura. Esta situação é uma das razões pelas quais a compreensão, selecção e extracção de dados é uma actividade cada vez mais intensiva em termos de tempo.

A abordagem apresentada não propõe uma reconstrução dos ambientes de manufatura convencionais para forçar uma harmonização dos dados e das infraestruturas relacionadas com os dados. A abordagem toma a situação actual como uma condição prévia que irá existir provavelmente durante mais algumas décadas e tenta dar respostas para uma solução de topo que possa ser aplicada em sistemas de manufatura convencionais mas também em sistemas de fabrico inteligentes mais recentes.

O trabalho de doutoramento descreve uma abordagem de canalisação de dados em conformidade com DIN Spec 91345 RAMI 4.0 para apoiar a compreensão e extracção de dados em ambientes (inteligentes) de manufatura. A abordagem utiliza um modelo semântico que descreve recursos e dados ao longo do seu ciclo de vida para apoiar a sua exploração num ambiente de manufatura inteligente. A base para o modelo semântico é a RAMI 4.0, uma das arquitecturas de referência mais populares e outras normas utilizadas para descrever ambientes de manufatura inteligentes ao nível dos metadados. Uma descrição adicional descreve o acesso

aos dados disponíveis que permite a abordagem global de canalisação de dados para extrair os dados necessários ao utilizar soluções tradicionais de canalisação de dados.

Apresentam-se o conceito e a abordagem para a canalisação de dados em conformidade com a norma DIN Spec 91345 RAMI 4.0, que se divide ainda no modelo semântico, uma arquitectura lógica e um protótipo validado TRL-6.

Os resultados deste trabalho de doutoramento contribuem para áreas de investigação que lidam com a gestão de dados em sistemas de manufatura inteligentes e fornecem ideias para construir, manusear e explorar ambientes de manufatura inteligentes auto-descritivos baseados em modelos semânticos e ainda para ligar estes modelos com dados disponíveis ao longo de ciclos de vida de bens localizados em tais ambientes.

Table of Contents

Figures

Tables

Codes

List of Abbreviations

3D	3 Dimensional
AAS	Asset Administration Shell
ACL	Agent Communication Language
AG	Aktiengesellschaft
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BDV	Big Data Value
BDVA	Big Data Value Association
BSD	Berkeley Software Distribution
CAD	Computer-Aided Design
CEP	Complex Event Processing
CPS	Cyber-Physical System
CQL	Cassandra Query Language
CRISP-DM	Cross-Industry Standard Process for Data Mining
CRUD	Create, Read, Update, and Delete
CSI	Code Space Identifier
CSV	Comma-separated values
DAC	Data Access Catalogue
DIN	Deutsches Institut für Normung
DPWS	Devices Profile for Web Services
DS	Data Source
DSB	Data Source Bundle
e.g.	exempli gratia
ELT	Extract-Load-Transform
ERP	Enterprise resource planning
ETL	Extract Transform Load
EU	European Union
FTP	File Transfer Protocol
GPL	General Public License
GPS	Global Positioning System
GRAI	Graphs with Results and Actions Inter-related
GUI	Graphical User Interface
H2020	Horizon 2020
HDFS	Hadoop Distributed File System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
i.e.	in example
IAIS	Institute for Intelligent Analysis and Information Systems
IBM	International Business Machines Corporation
IC	Item Code
ICD	International Code Designation

ICT	Information and Communications Technology
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIC	Industrial Internet Consortium
IIoT	Industrial Internet of Things
IIRA	Industrial Internet Reference Architecture
IMSA	Intelligent Manufacturing System Architecture
IoT	Internet of Things
IP	Internet Protocol
IRDI	International Registration Data Identifier
IRI	Internationalized Resource Identifier
ISA	International Society of Automation
ISO	International Organization for Standardization
IT	Information Technology
JSF	Java Server Faces
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MIT	Massachusetts Institute of Technology
MQTT	Message Queuing Telemetry Transport
OPC UA	OPC Unified Architecture
OPI	Organization Part Identifier
OS	Operating System
OT	Operational Technology
OWL	Web Ontology Language
PB	Property Builder
PhD	Doctor of Philosophy
PLC	Programmable Logic Controller
PPP	Private Public Partnership
QR	Quick Response
RA	Reference Architecture
RAMI 4.0	Reference Architectural Model Industrie 4.0
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	REpresentational State Transfer
RFID	Radio-Frequency Identification
SFS	Search Filter and Select
SGAM	Smart Grid Architecture Model

SMPC	Sub Model Property Characterisation
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SotA	State of the Art
SPARQL	Simple Protocol and RDF Query Language
SPEC	Specification
SQL	Structured Query Language
TCP	Transmission Control Protocol
TDB	Triple Data Base
TRL	Technology Readiness Level
TSDB	Time Series Database
TSN	Time-Sensitive Networking
UDP	User Datagram Protocol
UI	User Interface
URI	Uniform Resource Identifier
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e. V.
VDI	Verein Deutscher Ingenieure
VW	Volkswagen
W3C	World Wide Web Consortium
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

1

INTRODUCTION

1.1 Motivation

The current situation in smart manufacturing

Keywords such as cloud of services, (Industrial) Internet of Things ((I)IoT), virtual world, systems of systems, Big Data analysis, autonomous units, adaptive and predictive control, computing (multicore), mobility, connectivity, complex event processing, or forecasting of complex scenarios are some bricks of industrial digitalisation and networking mostly recognized under the term smart manufacturing systems as an emerged outcome of the (often called) fourth industrial revolution (Armando Walter Colombo et al., 2021; Hermann, Pentek, & Otto, 2015; Jasperneite, 2012; Kagermann, Helbig, Hellinger, & Wahlster, 2013; Leitão, Pires, Karnouskos, & Colombo, 2020; Peres et al., 2020; Wermann, Kliesing, Colombo, & Moraes, 2015).

Market trends, social media feedback, rising individual customer demands associated to mass customisation as well as extreme customisation (los $\geq$ 1) for tangible and intangible products and services generate new Key Performance Indicators (KPIs) for the industrial environment (Armando Walter Colombo et al., 2019). Some of them include production flexibility, visibility, waste efficiency, supply chain adaption, etc. Moreover, CO2 footprint, eco-efficiency, ambient conditions, agile supply chain and other factors influence the number and characteristics of those KPIs (ECSEL PMB, 2016).

Manufacturing systems are no longer hierarchical physical and logical capsulated and structured systems but heterogeneous, loosely coupled, non-hierarchical structured, collaborating cyber-physical systems of systems with event-based communication in unified networks (Borangiu et al., 2020; Schwab, 2016).

Due to these trends, the manufacturing eco-system has to become smart, i.e., digitalisation and networking for allowing collaboration, intelligence for real-time decision-making processes vertical along the enterprise and system architecture and horizontal along the value stream (value chain) and supply chain (L. Camarinha-Matos, Afsarmanesh, & Ollus, 2008; Armando Walter Colombo et al., 2021).

The complexity of smart manufacturing systems creates the demand for new approaches and methodologies needed to handle such new systems in an effective, efficient and flexible way. This results in new challenges for research, innovation and development.

This doctoral work focuses on data understanding and data acquisition processes in such smart manufacturing systems coping with their increasing complexity.

The value of data

Many people say "data" is the new gold of our century (Forbes Africa, 2019; Kroes, 2011; O'Brien, 2018). Today, large amounts of data can be stored and shared around the globe, leading to novel and fundamental transformations of businesses, communication channels and decision-making. One reason for this is the great exploitation potential behind the data. A big exemplary exploitation field is the mining of hidden information in data. Descriptive, predictive or prescriptive analytics (Soltanpoor & Sellis, 2016) enables a precise living description of the as-is state, a prevention of risks and problems, and can provide advises for actions to change current progressions into a specific direction.

The fact that there is a great potential for use behind the data also applies to the area of smart manufacturing (Nagorny, Lima-Monteiro, Barata, & Colombo, 2017). Smart manufacturing systems are producing many different kinds of data; mixed and aggregated with data from interconnected systems potentially located in several domains and layers. Finding new associations, influencing factors, patterns and observing such findings through data value stream observation, is one of the main objectives of Big Data analysis in smart manufacturing. It enables the generation of knowledge in form of identified associations and patterns in huge amounts of continuously changing data (streams) and the observation of value streams based on such associations and patterns. Extracted knowledge increases the visibility of such systems, enables prediction of happenings, enables new kinds of diagnosis possibilities, enables decision support/making, supports KPI optimisation and is the base for several other use cases. The exploitation of these use cases will potentially innovate business fields through improved maintenance services (e.g. anomaly/failure detection/prediction, system observation); pattern observation e.g. for hacker detections; extended manufacturing system reports; KPI improvements/monitoring; customer demands identification based on data; or virtual model improvements of physical components for simulation. These were just a small range of potential possibilities for data exploitation in the smart manufacturing domain.

Multi characteristic data usually comes from a variety of sources in different volumes and in form of streams or data buckets (M. Chen, Mao, & Liu, 2014). By analysing latest reports focusing on predictions about the data generation-consuming-traffic, a doubling of data growth every two years is foreseen. It is assumed that a similar trend is also valid in the manufacturing domain (Brown, Chui, & Manyika, 2011) so that the industry will be confronted with the management of amounts of data in a range from gigabytes and terabytes, till dimensions of petabytes, exabytes and zettabytes.

Data Science includes many R&D challenges and is a comprehensive multidisciplinary field as indicated in Figure 1-1.

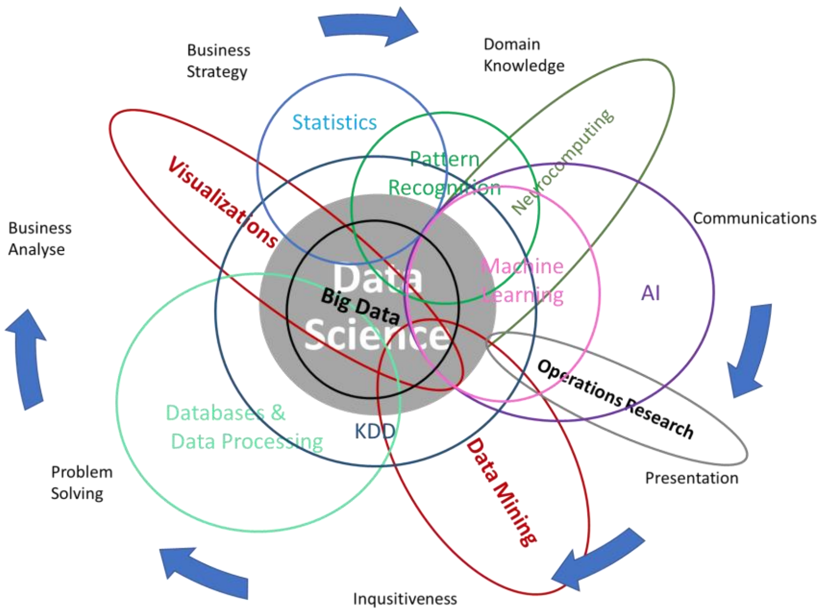


Figure 1-1: Data Science - A multidisciplinary field (compare *(Kelleher & Tierney, 2018)*)

The rising complexity of manufacturing systems, the rising amounts of available and accessible data and the huge exploitation field shows that data will get more and more into the focus of future smart manufacturing systems and with this focus also the comprehensive multidisciplinary field of data science (see Figure 1-1).

1.2 Scope, research challenges and boundaries

One of the most time consuming processes in data analytics is the data understanding and data preparation phase, according to the Cross Industry Standard Process for Data Mining (CRISP-DM) (Jones, 2011; Shearer, 2000).

The data understanding phase includes an initial data collection, their description and their exploration, while the data preparation phase includes - among others - the selection and extraction of data. Based on the experience of the author, this is also valid for a data analysis process in the (smart) manufacturing domain (Nagorny et al., 2017).

These phases can take more than 60 % of the time in an entire data analysis process (Cios, Pedrycz, Swiniarski, & Kurgan, 2007). Results of investigations, searching for reasons of this high time consumption in the (smart) manufacturing domain, have identified three main causes (Nagorny, Scholze, Colombo, & Oliveira, 2020):

Cause 1 – Data Chaos: Even in modern digitalised manufacturing environments, different kinds of data are stored unstructured in several kinds of data sources. Some data are single files while others are bundled into data buckets, and others are only available as a data stream or in a request-response mechanism using different kinds of communication protocols.

Cause 2 – Inefficient data understanding and selection: Efficient approaches to explore, search, filter, identify, understand and select required data are missing, and available data is often proprietary without applied classification and standardisation approaches.

Cause 3 - Time and resource consuming data extraction: Even if the data is known, extracting heterogeneous data (e.g. for a later data analysis) is still a challenge since multiple communication protocols and technologies need to be considered. In practice this often means that data has to be collected from different data sources (message brokers, databases, services, etc.) and that the correct location/reference within a data source has to be identified (for instance a column in a specific database table). This is often a time and resource intensive process in which several Information Technology (IT) experts with domain knowledge have to be involved.

These three causes led the author to the definition of three major research challenges, addressed in this doctoral work:

Challenge 1 – Structuring the data chaos: The growing volume, variety, velocity and the growing complexity of data in the smart manufacturing domain requires new approaches to manage the data chaos. It has to be answered how data can be classified, structured and described and how established and newer standards – emerging from the Industry 4.0 - can support this process. Very heterogeneous digital brownfields as well as modern green field ecosystems where any kind of data can be part of an asset life-cycle have to be considered for a comprehensive solution.

Challenge 2 – Ease data exploration in heterogeneous data environments: The exploration, searching, filtering, identification, understanding and selection of data in heterogeneous data environments is currently a complex and time-consuming task, made only possible by expert data scientists.

Challenge 3 – Simplify data extraction for heterogeneous data environments: The data extraction in a heterogeneous data environment should consider the path dependency in the industrial domain to assure an applicability in brownfields as well as in modern green fields. The extraction of the data should be independent from used communication technologies, data storage technologies, data formats or data schemas.

The boundaries of these research challenges in this doctoral work should also be described. This work elaborates an approach to support especially data understanding (in detail: data collection, exploration and description) and data acquisition (in detail: data selection and extraction). It does not cover data quality verification, data construction, integration or formatting. The validation of results happens in a discrete manufacturing environment. Considering main aspects mentioned by reference architectures and models for data management and smart manufacturing, this doctoral work does not consider data analytics, data protection (e.g., data anonymisation), comprehensive cybersecurity, trust or inter-enterprise data sharing aspects. Therefore, the results of this work are supposed to be used internally in an enterprise with an already secured IT infrastructure. It also does not consider the functionalities or business logic of assets in a smart manufacturing environment.

1.3 Research Questions and Hypotheses

Based on the three identified research challenges (described in section 1.2), the following three research questions and hypotheses have been identified.

1.3.1 First Research Question and Hypothesis

Question: *Is it possible to classify, to structure and to describe heterogenous brownfield and greenfield smart manufacturing data environments by considering latest standards coming from the Industrie 4.0 platform?*

Hypothesis: *If latest norms issued by the Industrie 4.0 platform, such as the DIN Spec 91345 RAMI 4.0, as well as other data classification standards, are used to elaborate a semantic data model, then it is possible to classify, structure and describe heterogeneous brownfield and greenfield smart manufacturing data environments.*

1.3.2 Second Research Question and Hypothesis

Question: *Can a concept and an approach be elaborated, and a software architecture be specified, which support the exploitation of a unified semantic data model to enable an easy exploration, search, filtering, identification, understanding, selection and extraction of data, supporting also data understanding and data acquisition, in smart manufacturing environments?*

Hypothesis: *If a knowledge graph that describes a smart manufacturing data environment based on a unified semantical data model is used as a basis, then it is possible to define a concept and approach, and to specify a software architecture, which support the exploitation of a unified semantic data model, to enable an easy exploration, search, filtering, identification, understanding, selection and extraction of data, supporting also data understanding and data acquisition, in smart manufacturing environments.*

1.3.3 Third Research Question and Hypothesis

Question: *Is it possible to implement and validate in a relevant industrial environment a semantic model-based data pipelining tool prototype, using current data pipelining and software engineering technologies?*

Hypothesis: *If a triple store, using SPARQL to navigate over a knowledge graph, is combined with software engineering frameworks and existing data pipelining technologies, then it is possible to implement and validate an industrial-compliant data pipeline tool prototype.*

1.4 Research Objectives and Research Requirements

This section defines six research objectives and associated research requirements.

The achievements of these objectives will allow to validate the hypotheses associated to the three research questions of this thesis.

1.4.1 Research Objectives

- **Research Objective 1** – Analysis of standards and reference architectures relevant for data representation and structuring in smart manufacturing environments considering proposals of Germanys strategic program under the name "Industry 4.0" (Schwab, 2016; VDI, April 2015).
- **Research Objective 2** – Development of a concept and approach to search, filter, select and extract data based on the semantic model, to realise comprehensive data search, filter, selection and extraction to support data understanding and data acquisition in smart manufacturing systems.
- **Research Objective 3** – Specification of a semantic model to standardise, categorise and structure data along asset life-cycles, including their linkages, compliant with DIN-SPEC-91345, which specifies the reference architecture model 4.0 (RAMI 4.0) and is open to add further standards.
- **Research Objective 4** – Specification of a software architecture that uses the semantic model to realise data search, filter, selection and extraction.
- **Research Objective 5** – Implementation of a generic prototype which supports data understanding and data acquisition in smart manufacturing environments.

Research Objective 6 – Integration, validation and demonstration of the prototype in a relevant industrial environment (Technology Readiness Level (TRL) 6[1]).

Table 1-1 shows a mapping of the Research Questions and Hypotheses (RQ&H_i) to defined Research Objectives (RO_i).

[1] See https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf

Table 1-1: Mapping: Research Questions and Hypotheses <-> Research Objectives

RO / RQ	RO_1	RO_2	RO_3	RO_4	RO_5	RO_6
RQ&H_1	X		X			
RQ&H_2		X	X	X		
RQ&H_3					X	X

1.4.2 Research Requirements

Under research requirements (RR) will be considered the set of expected results of the doctoral work. The research requirements are consequently mapped into the research questions and hypotheses.

The International Electrotechnical Commission (IEC) and International Organization for Standardization (ISO) defines a norm for requirements in ISO/IEC Directives, Part 2, 2021, 3.3.3. There a requirement is defined as an "*expression, in the content of a document, that conveys objectively verifiable criteria to be fulfilled and from which no deviation is permitted if conformance with the document is to be claimed.*" (ISO/IEC, 2018). According to this norm the requirements get classified as

- *"shall"*, which is equivalent to *"is to, is required to, it is required that, has to, only ... is permitted, it is necessary"* and indicates a mandatory requirement.
- *"should"*, which is equivalent to *"it is recommended that, ought to"* and indicates a recommendation.
- *"may"*, which is equivalent to *"is permitted, is allowed, is permissible"* and indicates a permitted requirement.

Each research requirement has an identifier (IDi_j), a description, and an indication to the associated research Questions and Hypotheses (RQ&H_i), as shown in Table 1-2.

Table 1-2: List of research requirements

ID	Research requirement description	Associated RQ & H
Addressing the first research objective		
RR_ID1_1	The report shall provide an analysis of **reference architectures** relevant for data representation and structuring in smart manufacturing environments, with emphasis in considering proposals of Germanys strategic program under the name Industry 4.0.	RQ&H_1
RR_ID1_2	The report shall provide an analysis of technology **standards** relevant for data representation and structuring in smart manufacturing environments, with emphasis in considering proposals of Germanys strategic program under the name Industry 4.0.	RQ&H_1
Addressing the second research objective		
RR_ID2_1	The concept and approach shall conceptually show that a knowledge graph which describes a smart manufacturing data environment based on a unified semantical data model can be used as basis to realise the features for **data exploration, search, filtering, identification, understanding, selection and extraction**.	RQ&H_2
RR_ID2_2	The concept and approach should show that it is suitable to elaborate a semantic model.	RQ&H_2
RR_ID2_3	The concept and approach should show that it is suitable to elaborate a software architecture that uses the semantic model to realize data search, filtering, selection and extraction.	RQ&H_2
Addressing the third research objective		
RR_ID3_1	The semantic data model shall be DIN-SPEC-91345 standard (RAMI 4.0) compliant.	RQ&H_1
RR_ID3_2	The semantic data model following the industrial digitalisation framework described by the DIN-SPEC-91345 standard (RAMI 4.0), should support the semantical unification in the smart manufacturing domain.	RQ&H_1
RR_ID3_3	The semantic data model shall show that available Industry 4.0-related classification standards, like e.g. eCl@ss, can be integrated.	RQ&H_1

RR_ID3_4	The semantic data model shall show that it is suitable to structure and to describe assets and associated digitalised data along the whole asset life-cycle.	RQ&H_1
RR_ID3_5	The semantic data model shall show that it is suitable for heterogeneous brownfield and greenfield smart manufacturing data environments.	RQ&H_1, RQ&H_2
RR_ID3_6	The semantic data model shall show that it can be one major key element to support data understanding and data acquisition in smart manufacturing.	RQ&H_1, RQ&H_2
Addressing the fourth research objective		
RR_ID4_1	The software architecture should enable the management of a semantic model.	RQ&H_2
RR_ID4_2	The software architecture shall enable the exploitation of a semantic model to realise data **exploration, search, filtering, identification, understanding and selection**.	RQ&H_2
RR_ID4_3	The software architecture shall enable the exploitation of a semantic model to realise data **extraction**.	RQ&H_2
Addressing the fifth research objective		
RR_ID5_1	The generic prototype should be suitable for heterogenous data environments.	RQ&H_3
RR_ID5_2	The generic prototype shall be suitable for industrial Industry 4.0 compliant greenfields.	RQ&H_3
RR_ID5_3	The generic prototype shall be suitable for industrial brownfields.	RQ&H_3
Addressing the sixth research objective		
RR_ID6_1	The generic prototype shall prove that the solution is suitable for **exploration, search, filtering, identification and understanding** of data in heterogenous data environments.	RQ&H_3
RR_ID6_2	The generic prototype shall prove that the solution is suitable for **selection** of data in heterogenous data environments.	RQ&H_3

RR_ID6_3	The generic prototype shall prove that the solution is suitable for **extraction** of data in heterogenous data environments (independently from used data storage or streaming technologies, communication technologies, protocols, data formats or data models).	RQ&H_3
RR_ID6_4	It may be proved that Graph Triple Stores can be used as basis to manage the contents of the knowledge graph based on the semantic data model.	RQ&H_3
RR_ID6_5	It shall be proved that existing data pipelining technologies can be extended to enable semantic model-based data pipelining.	RQ&H_3

1.5 Research and Innovation contributions

This doctoral work makes four major research and innovation contributions, as schematically described in Figure 1-2. These contributions are used to prove the achievements of the six research objectives, associated to the set of research requirements described in Table 1-2.

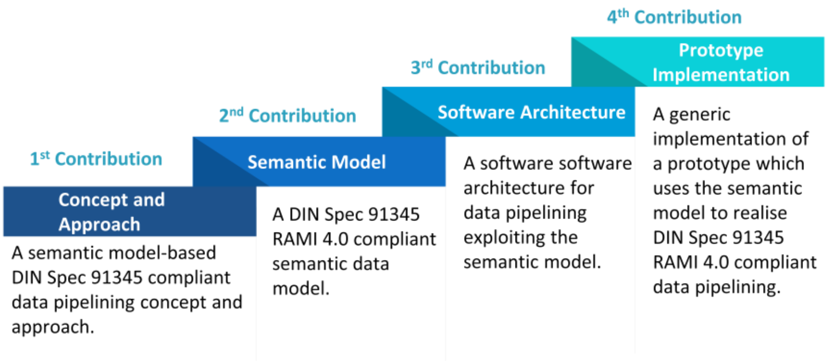


Figure 1-2: Research and Innovation Contributions

- **Concept and Approach (Research)** - A concept and approach for semantic model-based DIN Spec 91345 compliant data pipelining.
- **Semantic Model (Research)** - A DIN Spec 91345 RAMI 4.0 compliant semantic data model which enables a standardised description of smart manufacturing environments, available data and the access to them.
- **Software Architecture (Research & Innovation)** - A software architecture which exploits the semantic model for DIN Spec 91345 compliant data pipelining.

- **Prototype implementation (Innovation)** - A generic implemented semantic model-based DIN Spec 91345 compliant data pipelining prototype validated in an industrially relevant environment.

1.6 Doctoral Thesis Report Structure

Table 1-3: Structure of the Doctoral Thesis

Section	Contents	Addressed RO	Description
1	Introduction	-	Introduces this doctoral work and describes the motivation, focus and aimed contributions. It also describes the research questions and related hypotheses, and defines overall objectives and requirements
2	State of the Art review	RO1	Provides a comprehensive state of the art overview which includes among others an analysis of reference architectures and standards to address the first research objective. Further are identified relevant research gaps and describes how this work contributes to these gaps.
3	Concept and Approach	RO2	Describes the concept and approach of this doctoral work.
4	Detailed Approach and Specification	RO3-4	Presents the solution beyond the state of the art for DIN Spec 91345 RAMI 4.0 compliant data pipelining.
5	Implementation and Testing of the Approach	RO5-6	Describes the implementation of a generic prototype and the testing of the approach in an industrial relevant environment.
6	Validation and Assessment of results	-	Provides the overall assessment and interpretation of results achieved in sections 2-4.
7	Conclusions and Outlooks	-	Provides some concluding remarks and an outlook for future research and innovation opportunities.

2

STATE OF THE ART REVIEW

This section provides a state-of-the-art review related to the main areas addressed in this thesis.

2.1 Smart Manufacturing

Smart manufacturing environments tend to be complex infrastructures in which many aspects need to be considered. This doctoral work aims to generate a solution to support data understanding and data preparation for such systems and therefore gives an overview of such smart manufacturing environments.

2.1.1 Towards Smart Manufacturing

The evolution towards a smart factory came from three – so called – “industrial revolutions” and the currently running fourth one. The **first** industrial revolution brought the development of the steam machine, which produced mechanical energy usable for mechanical production machines. The **second** revolution brought electrical energy with possibilities for electrical measurements, signals, electrical motors and possibilities to build more complex manufacturing machines for mass production. The **third** digital industrial revolution brought the computer technology (electronics and IT) which brought automation technologies such as programmable logic controllers, computer-controlled scheduling systems, robot technologies, and several other technologies, which are nowadays often called "conventional" or "traditional" manufacturing technologies. Today we are experiencing the **fourth** industrial revolution which describes the exploitation of the trend that electronics are getting smaller, faster, cheaper and more mobile; that the IT infrastructure (Wireless Local Area Network (WLAN), Local Area Networks (LAN), Metropolitan Area Network (MAN), Wide Area Network (WAN), etc.) is expanding worldwide, becoming faster and enabling the connection of all types of electronics connected to this infrastructure.

Achievements of the third revolution such as Global Positioning System (GPS), high precise robots, new kinds of highly precise sensors and actuators, etc. are basis for the fourth industrial revolution. Some examples are e.g. emerging third platform technologies with inter-dependencies between social media, mobile and cloud computing and (big) data analysis (Golden, 2014) to unlock potentials of conventional IT technologies, or the deployment of innovation accelerators as IoT, additive manufacturing, and robotics, coupled with the integration of operations technologies (Y. Wang, 2015).

Future smart manufacturing infrastructures have to enable the exploitation of these new opportunities. Even today, people are surrounded by interconnected digital environments continuously generating more synergies with connected devices and software. Such an evolution also happens in the manufacturing domain. Future smart manufacturing infrastructures are confronted with the **digitalisation** and **virtualisation** of (physical) objects enhanced with sensors, processors, memory and communication devices able to communicate coactively and to exchange information independently through a reactive, predictive, social, self-aware and/or autonomous behaviour (A.W. Colombo, 2014; Lukac, 2015). A used term for such intelligent physical objects which are communicating in (I)IoT networks is ICPS.

Initiatives such as the Industry 4.0 (I40) (see https://www.plattform-i40.de) develop overall solutions and standards to integrate such systems vertically and horizontally over all levels, considering life cycles, value streams, hierarchy levels and layers as derived in the Reference Architectural Model Industrie 4.0 (RAMI 4.0) (DIN SPEC 91345:2016-04, 2016). How these views can be integrated and which new standards will be used as basis, or need to be developed, is currently under discussion and part of many research activities, and exceeds the borders of the manufacturing domain (see e.g. FIRWARE (Ramparany, Marquez, Soriano, & Elsaleh, 2014), the Industrial Data Space (Otto, ten Hompel, & Wrobel, 2018) or GAIA-X (Bundesministerium für Wirtschaft und Energie, 2019)).

Smart Manufacturing infrastructures have to enable the exploitation of these emerged opportunities, considering specific requirements in this domain, such as real-time, security or safety. Figure 2-1 shows a division of a smart manufacturing infrastructure into five different layers described bottom-up in the following:

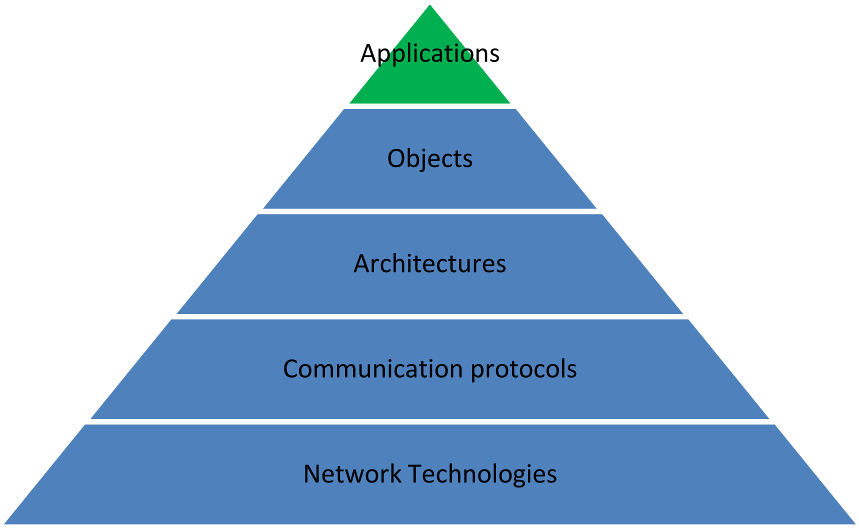


Figure 2-1: Layers of a smart manufacturing infrastructure

2.1.2 Network Technologies

Smart Manufacturing infrastructures have to be based on network technologies which enable a secure and trusted cross-domain and cross-layer communication between stationary and mobile communicating objects. Network technologies have to comply to specific requirements related to e.g. real-time, safety, security, data amounts, wired or wireless, passive or active, etc. (Rathwell & Ing, 2004). Lower field-levels require time frame abilities of seconds or milliseconds for response, reliability, resolution, repair (e.g. control, or real-time statistics of the process) whereas higher levels only require time frames of weeks or months (e.g. for production planning or accounting) (Rathwell & Ing, 2004).

2.1.3 Communication approaches

Communication approaches in Smart Manufacturing systems should enable a robust, loosely coupled, time-synchronized, secured, and semantically supported communication.

In smart manufacturing infrastructures, where thousands of communicating objects are accessible from everywhere, it is hardly possible to exchange information in a conventional loop-based way (e.g. on field level/ISA-95 levels 1-2) via a bus in a defined frequency to enable a deterministic information exchange). This situation generates a need for new approaches to reduce the traffic in networks whilst holding on to all requirements. Such a new approach is e.g. the paradigm "Service-oriented Architecture" (SOA). SOA describes an architectural pattern in which functionalities and features of systems, components, applications, etc. are provided as a service in a network where all services can be found and accessed using features as complex event processing as alternative for control loops and to reduce traffic in the network. Services are *"logical representations of a repeatable business activity that has a specified outcome (e.g. check customer credit, provide weather data, consolidate drilling reports), is self-contained, may be composed of other services and is a "black box" to consumers of the service* (The Open Group, 2016).*"*

One important aspect of the communication in smart manufacturing environments is also the ongoing unification of used semantics. ICPS could understand each other (Fay et al., 2015) if they would use the same lexicon and grammar to describe themselves (properties, functionalities, etc.) (Grangel-González et al., 2016). A more detailed overview about semantics in smart manufacturing is given in section 2.5.4.

Many communication approaches and (complementary) technologies exist (Garcia, Abilio, & Malheiros, 2015) and are candidates for smart manufacturing systems. Some of them are listed below.

Service-oriented Architecture

- **Arrowhead** - is a SoA based framework which provides automation capabilities, such as scalability, security, real-time control, and engineering simplicity while also enabling IIoT and device interoperability at a service level (Delsing, 2017; Paniagua & Delsing, 2020).
- **Simple Object Access Protocol (SOAP)** - is a Web Services Description Language (WSDL) and Extensible Markup Language (XML) based protocol (Zur Muehlen, Nickerson, & Swenson, 2005).
- **Representational State Transfer (REST)** - provides services as Uniform Resource Identifier (URI) through POST, GET and DELTE methods based on Hypertext Transfer Protocol (HTTP) 1.0 (Zur Muehlen et al., 2005) using Javascript Object Notation (JSON). REST is a stateless client-server request-response protocol, mainly used to realise webservices. It enables hypermedia (see e.g. HATEOAS (Varanasi & Belida, 2015)) suitable to navigate through information networks. It has also in Industrie 4.0-compliant ecosystems a high relevance as it is widely used and interoperable with any kind of World Wide Web (WWW) compliant application which makes it (together with the hypermedia support) suitable for AAS as described in DIN SPEC 91345.
- **Devices Profile for Web Services (DPWS)** - is an Organization for the Advancement of Structured Information Standards (OASIS) standard implemented through the Java Multi Edition DPWS Stack (JMEDS) or the Web Services for Devices (WS4D)-

gSOAP framework (Bangemann, 2013; Armando Walter Colombo & Karnouskos, 2009; Nagorny, Colombo, & Barata, 2014).

- **OPC Unified Architecture (OPC-UA)** - is an implementation of the OPC Foundation and is available as implementation approach based on the SoA paradigm (Bangemann, 2013; Armando Walter Colombo & Karnouskos, 2009; Nagorny et al., 2014). OPC UA is in Europe one of the most popular communication technologies in automation. OPC UA is a one-to-one client-server, Transmission Control Protocol (TCP) or HTTP based communication protocol. It supports partly real-time using User Datagram Protocol (UDP) and Time-Sensitive Networking (TSN) as publish-subscribe features (Hoppe, 2016). For cloud-integration, OPC UA uses the Advanced Message Queuing Protocol (AMQP) (Panhelainen, 2017).
- **Complex Event Processing (CEP)** - is the bridge between the occurrence of events, and the reaction through an adaptation that is required by the situation (Hinze & Buchmann, 2010). Situation means that a range of raw events are triggering a complex event by a specific pattern (Hinze & Buchmann, 2010). Some projects with implementations were "Rapide", which generated the Event Processing Language (EPL) called "Rapide" of the Stanford University; Borealis through a cooperation of the Brandeis University, the Brown University and the Massachusetts Institute of Technology (MIT); and "Odysseus" from the University of Oldenburg (Bolles, Grawunder, Jacobi, Nicklas, & Appelrath, 2009). One key driver of CEP technologies is Opher Etzion (Chief Scientist in International Business Machines Corporation (IBM) Research – Haifa), who was among others involved in the European Union (EU) project FI-WARE (part of the Future Internet Private Public Partnership (PPP) program; s. http://www.fi-ware.org/), where they worked on an open source framework with CEP features like processing raw event streams, event pattern and event forwarding (Nagorny et al., 2014).

Multiagent Systems

- **Agent Communication Language (ACL)** - is a specification for agent communication published by the Foundation for Intelligent Physical Agents (FIPA) (Fipa, 2002; O'Brien & Nicol, 1998). Multiagent systems are heavily influencing modern smart manufacturing environments as shown in (Leitao et al., 2016; Pulikottil et al., 2021; Ribeiro & Gomes, 2021).

Internet of Things

- **Message Queuing Telemetry Transport (MQTT)** - is a communication protocol to enable the communication between things using a publish-subscribe approach (Wortmann & Flüchter, 2015), where messages pass a message broker. It is an alternative to OPC UA's AMQP. MQTT is fast and very light-weight (Panhelainen, 2017). It is typically TCP/IP based but other transportation protocols are also supported (see e.g. MQTT-SN (Stanford-Clark & Truong, 2013)).
- **Xively** - is an IoT platform with focus on IoT-specific functionality to complement potentially existing non-IoT platforms (Boman, Taylor, & Ngu, 2014).
- **ThingWorx and Bosch IoT Suite** - are comprehensive platforms which aim to provide an all-in-one solution (Solutions).
- **FIWARE** (FI-WARE Consortium, 2018) - is an open-source platform that defines a universal set of standards for context management in order to support the development of smart solutions in multiple scenarios, such as smart cities, industry, or smart energy grids (Paniagua & Delsing, 2020).
- **Other Frameworks** - (Derhamy, Eliasson, Delsing, & Priller, 2015; Paniagua & Delsing, 2020; Wortmann & Flüchter, 2015) mentioned several IoT frameworks to provide high-level abstraction, including management, security, interoperability, flexibility and interconnection of services, systems and devices. Some of them are: IPSO Alliance framework, IoTivity framework, AllJoyn framework, Thread groups framework, Open Mobile Alliance - Light Weight Machine to Machine (LWM2M) framework and Cumulocity.

2.1.4 Architectures

Architectures in general describe combinations of components/modules and their interaction and should provide a unified structure and wording for used terms. An architecture should include at least a logical-, a development-, a process- and a validation view and should provide scenarios for a validation as proposed by Philippe Kruchten in his 4+1 architectural view model (Kruchten, 1995). A Smart Manufacturing architecture should also provide a unified structure and wording covering major aspects in smart manufacturing as perspectives (Luis M Camarinha-Matos & Afsarmanesh, 2003; Hankel & Rexroth, 2015), hierarchical levels, life-cycle or time & data amount requirements. Some details of these aspects are described below.

Aspect: Perspectives

- The **Business / Coordination** perspective describes the view on business or coordination logic which defines, e.g., KPIs for energy efficiency, wear and tear or produced products per day.
- The **Functional** perspective describes the view on coherent functions to achieve a global system objective. Examples could be functions like system status monitoring, sensor monitoring, meta information request, but also objective related functions usable by other systems such as the control of the position of a robot tool center point.
- The **Information** perspective describes the view on information. The goal is to have a unified information model, a common abstraction, *"naming, semantics and the elimination of redundancies and inconsistencies are some of the aspects*." (Bernus, Nemes, & Schmidt, 2003). Example technologies are eCl@ss or AutomationML (Lüder & Schmidt, 2017) and example information are meta data as Identifiers (IDs), position, etc. but also models for information exchange
- The **Communication** perspective describes the view on communication between systems. This perspective tries to harmonise protocols and interfaces for system integration. Example technologies are Ethernet, WiFi, Global System for Mobile Communication (GSM)/4G, TSN, 5G, IP, IPsec, UDP, TCP, OPC-UA.
- The **Integration** perspective describes the view on integration of a system. Examples are sensors, Quick Response (QR) codes or Human Machine Interfaces (HMIs).
- The **system** perspective describes the system itself. Different kinds of systems need different kinds of integration. Examples are assets in general (software, physical assets, humans, virtual objects).

Aspect: Hierarchy Levels

- The **Consumer** level describes the integration of the consumer through a product interface. For example: a human who is using a product.
- The **Smart Product** level describes the integration of products to provide feedback or to improve configurations, etc. For example: a smartphone.
- The **Inter-Enterprise** level describes the integration of other enterprises through common services, communication standards, etc. to enable inter-enterprise cooperation or collaboration.
- The **Enterprise** level describes the integration and abstraction of enterprise areas for performance measurements and the overall business process management. An example system on this level could be an Enterprise resource planning (ERP) system.

- The **Work Centers** level describes the integration of shop floor systems as warehouses, stations, ERP to administrate and manage the production. An example system on this level could be a Manufacturing Execution System (MES) system.
- The **Station** level describes the integration of work stations and their control devices to enable the supervisory, control and data acquisition. An example system on this level could be a Supervisory Control and Data Acquisition (SCADA) system.
- The **Control Device** level describes the integration of field devices through control devices. An example device on this level could be a Programmable Logic Controller (PLC).
- The **Field Device** level describes the integration of hardware through sensors and actors.
- The **Hardware** level describes the physical hardware itself like a physical robot or a transport system.

Aspect: Life-cycle

- The **Development** life-cycle stage describes the idea creation and development phase of a product. Examples are materials related to development, construction, computer simulation or a prototype.
- The **Maintenance and Usage (Type)** life-cycle stage describes the idea creation and development phase of a product for maintenance and usage. Examples are materials related to software updates, instruction manual or maintenance cycles.
- The **Production** life-cycle stage describes the production phase of a product. Examples are production data as the product itself, serial number, etc.
- The **Maintenance and Usage (Instance)** life-cycle stage describes the usage and maintenance phase of a product. An example is the facility management: usage, service, maintenance.
- The **Recycling** life-cycle stage describes the recycling phase of a product and includes recycling information like for scrapping.

Aspect: Time & Data Amount Requirements

Requirements for time frames (e.g. for response) and the amount of data has to be considered (Macaulay & Singer, 2012) for a system integration. Information exchange for maintenance planning, production planning, etc. can take weeks, and on control device level data exchange has real-time requirements. Radio-frequency identification (RFID) cannot store big amounts of data and cannot meet real-time requirements.

Reference models and architectures of collaborative and networked systems have been analysed by several authors (for instance, (Luis M Camarinha-Matos & Afsarmanesh, 2008; Jagdev & Thoben, 2001; Molina et al., 2007)). Architectures as Purdue Enterprise Reference Architecture (PERA), Computer Integrated Manufacturing Open System Architecture (CIMOSA), GRAI Integrated Methodology (GRAI-GIM) and the Generalised Enterprise Reference Architecture and Methodology (GERAM), or Frameworks such as the Zachman Framework, Architecture of Integrated Information Systems (ARIS), The Open Group Architecture Framework (TOGAF), Federal Enterprise Architecture (FEA), Department of Defense Architecture Framework (DoDAF), Archi-Matem, ATHENA, or standards such as ISO 42010: 2011 (Software Engineering) or ISO 15704 are trying to support the system integration process and trying to cover most of the aspects (D. Chen, Doumeingts, & Vernadat, 2008) identified. Such architectures and frameworks provide suggestions for modelling languages, ontologies, reference models, enterprise modules or engineering methodologies (Romero & Vernadat, 2016). A more comprehensive overview was given in (Romero & Vernadat, 2016) and (Luis M Camarinha-Matos & Afsarmanesh, 2007).

The most modern reference architectures are nowadays the RAMI 4.0, the Smart Grid Architecture Model (SGAM), the China Intelligent Manufacturing System Architecture (IMSA) and the Industrial Internet Reference Architecture (IIRA):

Reference Architecture (RA): The Reference Architecture Model Industrie 4.0 (RAMI 4.0)

Next to worldwide known terms such as smart manufacturing or smart factory many countries have programs in which they develop reference models, standards and technologies to establish smart factories. In Germany a strategic program under the name Industrie 4.0 (Bloem et al., 2014) aims to enable a horizontal integration of value networks, a vertical integration within a factory, integrated engineering lifecycle management and to establish the human role as a conductor of a value added network (VDI, April 2015). In general the Industrie 4.0 vision (Drath & Horch, 2014) aims to establish an industrial ecosystem (industrial internet of things) in which all things are able to exchange information over a network respecting the legacy ISA-95 compliant enterprise architecture and develops among others the RAMI 4.0. This reference model is a further step toward a comprehensive standardisation of smart manufacturing systems.

RAMI 4.0 provides among others a unified basis for discussions around the topic of smart manufacturing systems. This means, RAMI 4.0 has to (1^{st}) cover aspects of smart manufacturing systems as Life Cycle, Value Streams, Information flow layers, and hierarchical layers, (2^{nd}) provide a unified structure and (3^{rd}) provide a unified wording to define a common language for discussions. Therefore, the RAMI 4.0 is a 3-dimensional (3D) model as shown in Figure 2-2 with three axes introduced in the following.

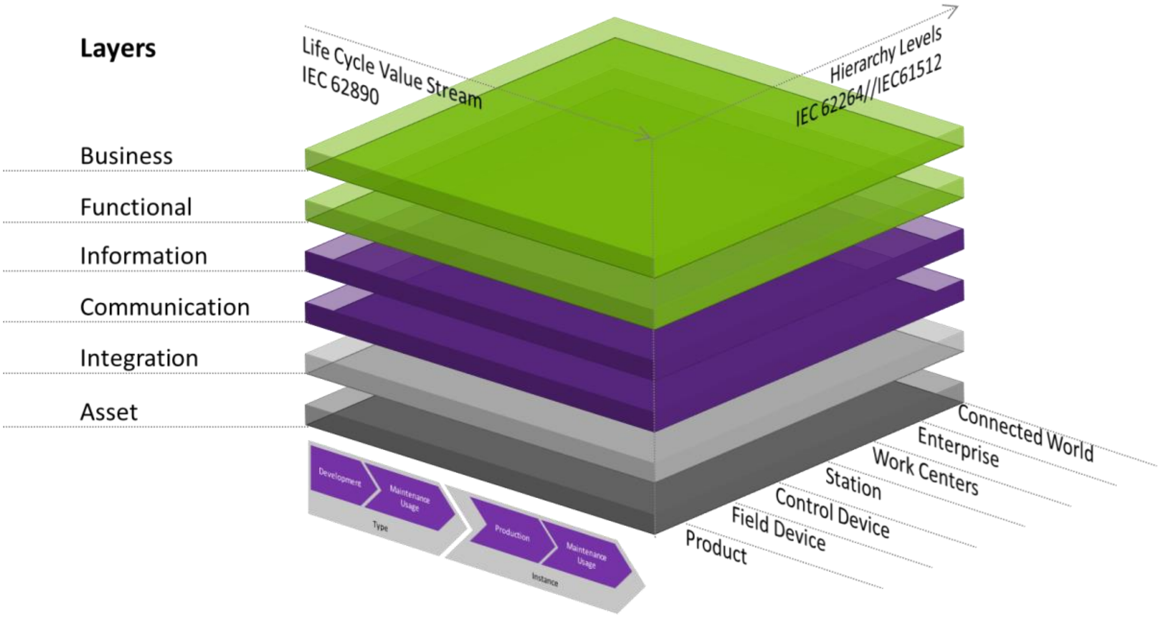


Figure 2-2: Architecture of RAMI 4.0 *(Wolfgang Dorst et al., 2015)*

Axis 1 – Layers: The RAMI 4.0 has six layers in the vertical axis which are representing different aspects of a component as business, functions, information, communication, integration and assets.

Axis 2 - Life Cycle & Value Stream (IEC 62890): The Life Cycle & Value Stream axis covers the full life cycle of components from the development phase, up to the maintenance phase. Thereby, RAMI 4.0 differentiates between components which are still in the development phase (called “types”) and components which are already implemented (called “instances”).

Axis 3 - Hierarchy Levels (IEC 62264 / IEC 61512): The hierarchical levels are based on the classical automation pyramid but are extended through the levels “Field Device”, “Product” and “Connected World”. The hierarchy levels describe in detail, in which level a component is located.

All three axes cover the most important aspects of an Industrie 4.0 component and enable the positioning of such a component in this 3D model as shown in Figure 2-3. The concept of an Industrie 4.0 component and also its digital representation in an AAS is described in section 2.5.1.

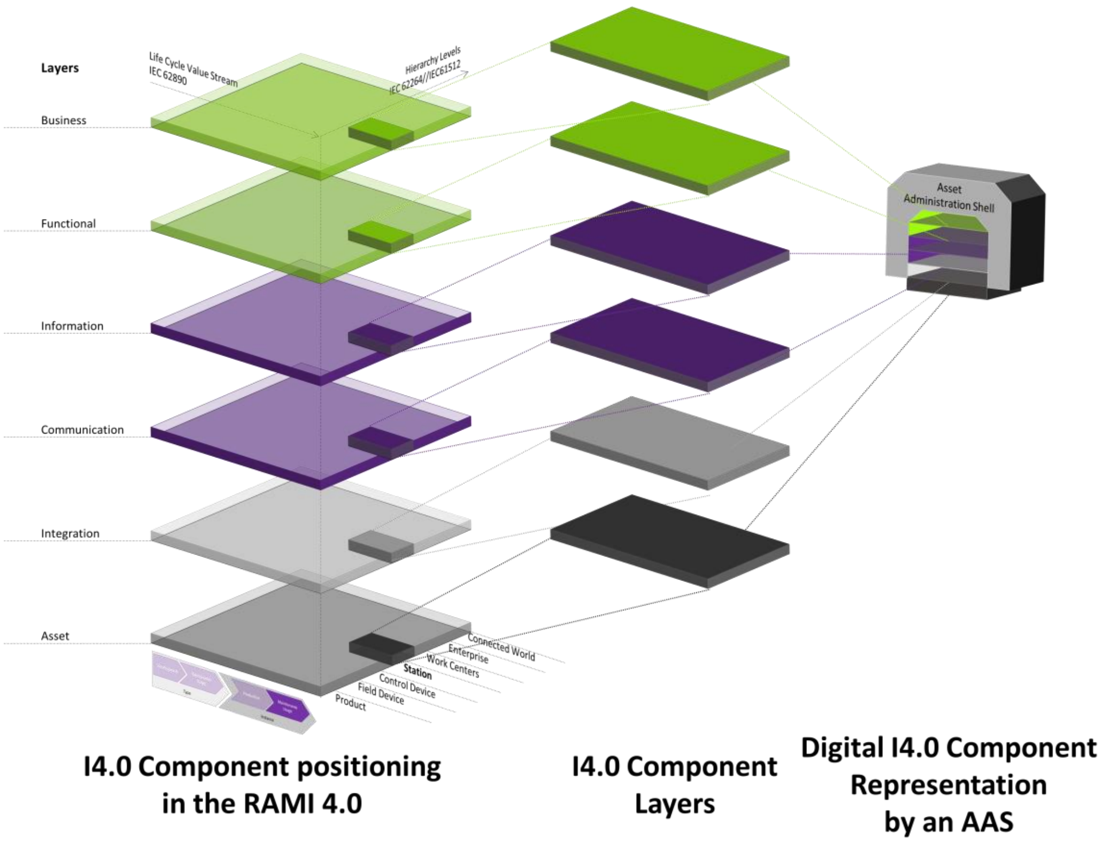


Figure 2-3: Positioning of an Industrie 4.0 Component inside the RAMI 4.0

The RAMI 4.0 was the result of many previous research activities and was based on or influenced by other reference architectures such as e.g. the SGAM:

Reference Architecture: The Smart Grid Architecture Model (SGAM)

The SGAM (Trefke et al., 2013), shown in Figure 2-4, describes an architecture for smart-grids. It is a three-dimensional model which describe interoperability layers, domains and zones.

Comparison with RAMI 4.0: The RAMI 4.0 was strongly influenced by the SGAM architecture. They share, e.g., a range interoperability dimensions/layers and zones/hierarchy levels. A detailed overview of both architectures was given by (Uslar et al., 2019).

Reference Architecture: China Intelligent Manufacturing System Architecture (IMSA)

The IMSA (Wei, Hu, Cheng, Ma, & Yu, 2017) provides a model, terminologies, evaluation indicators, and technology standards for smart manufacturing (B. Wang et al., 2021). It was developed by the Ministry of Industry and Information Technology (MIIT) of the People’s Republic of China and the Standardization Administration of China (SAC).

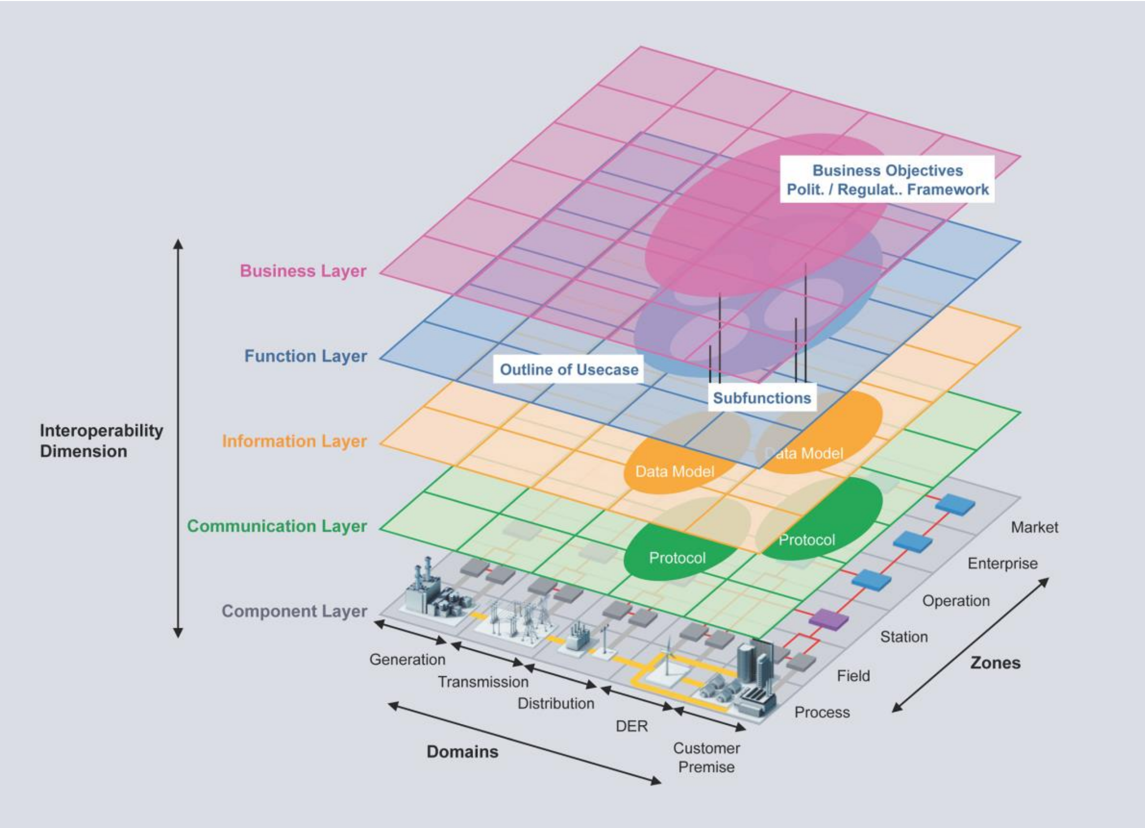


Figure 2-4: Smart Grid Architecture Model (SGAM) (Dietrich Biester, May 2012)

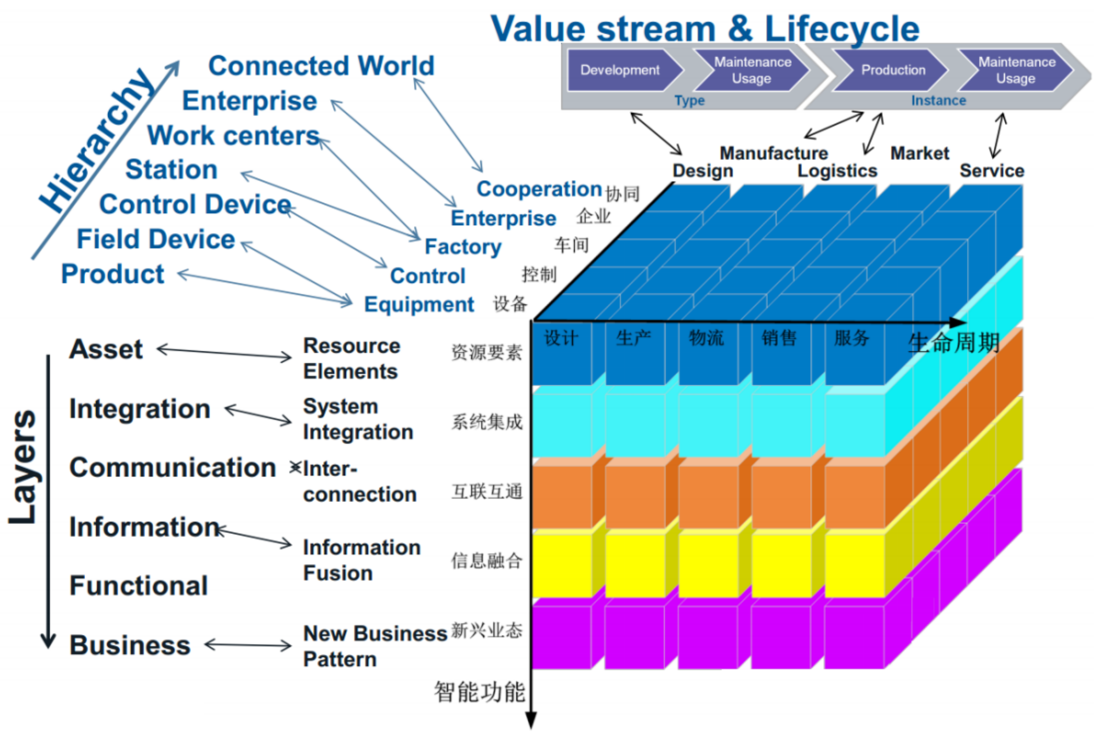


Figure 2-5: Mapping: IMSA <--> RAMI 4.0 (*(SCI 4.0, April 2018)*; M. Hankel, Bosch Rexroth AG)

Comparison with RAMI 4.0: The RAMI 4.0 is also aligned with the IMSA (see Figure 2-5) as presented on the Hannover fair 2018 and documented in (SCI 4.0, April 2018). A collaboration between China and Germany in form of a "Sino-German Industrie 4.0/Intelligent Manufacturing Standardisation Sub-Working Group" is in 2022 still ongoing.

Reference Architecture: Industrial Internet Reference Architecture (IIRA)

A further strong reference architecture (model) which should be mentioned, is the independently developed IIRA of the Industrial Internet Consortium (IIC). The "*IIRA emphasizes a broad applicability and interoperability across industries while the service-oriented RAMI 4.0 reaches deeper in describing models for the digitalization of manufacturing*" (Lin et al., 2015).

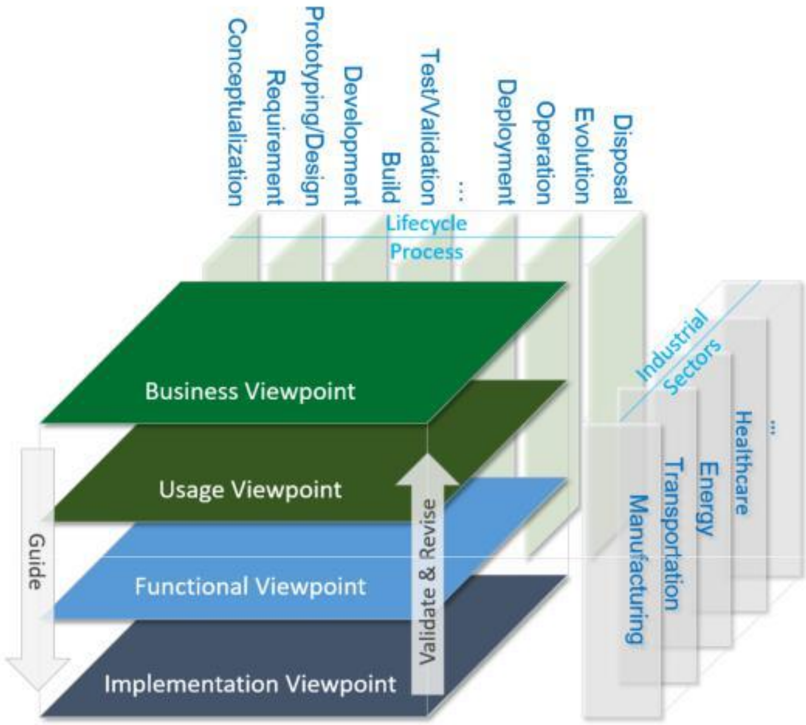


Figure 2-6: Industrial Internet Reference Architecture (IIRA) *(Lin et al., 2017)*

The IIRA is an open standard architecture specifically designed for IIoT systems. The basic structure of this architecture was described in ISO/IEC/IEEE 42010:2011 by the IIC (Lin et al., 2015). Figure 2-6 shows the view model of the IIRA. The model has a three-dimensional structure, and four viewpoints on the first axis: implementation, functionality, usage and business. The subdivision into levels is important for the implementation of different views. The classification into an industrial sector, such as transport, production or energy, is done by the second architectural axis. The third axis describes the life process of an object, from development to devaluation. In addition, the architecture consists of requirement descriptions for data protection, reliability, scalability, user-friendliness, maintainability, transferability and composability (Gunnar Knüpffer, 2015; Matthias Ruhl, 2017).

Comparison with RAMI 4.0: The term IoT emerged in the United States of America (USA) and was defined by the IIC, while the term Industrie 4.0 was defined in Germany by the Plattform Industrie 4.0. IIRA and RAMI 4.0 provide guidelines for development and implementation of systems but are different related to addressed domains (Matthias Ruhl, 2017).

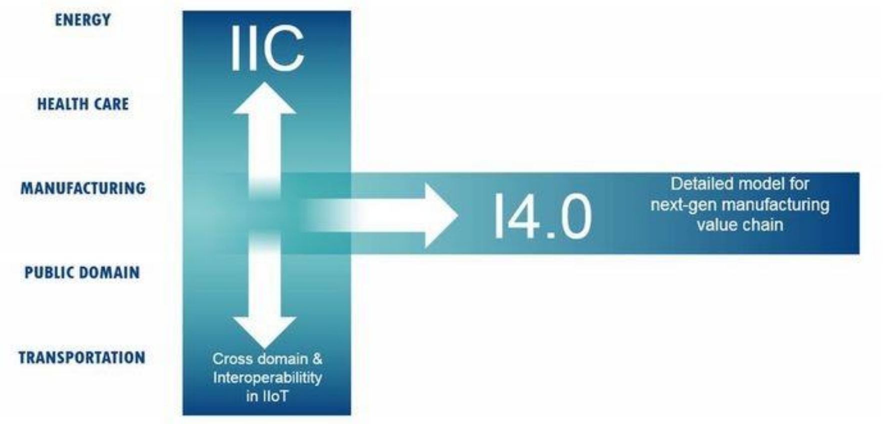


Figure 2-7: Cooperation between Industrie 4.0 and IIC (from Robert Bosch GmbH)

The biggest difference between IoT and RAMI 4.0 is that the IIRA is very open related to domains and the RAMI 4.0 has a strong focus on production processes. The reason for this is that the IIRA applies to the entire industrial sector (energy, health care, production, transport and public sector), while the RAMI 4.0 has a focus on the manufacturing domain ("a focus is on the production of a product over its entire life cycle", (Karoline Kopp, 2016)). Therefore, the RAMI 4.0 can be seen as a specialisation of the IIRA in the field of manufacturing (see Figure 2-7). In conclusion, the IIRA and the RAMI 4.0 are not separable and are linked by their subsets (Matthias Ruhl, 2017). A comprehensive compression of the RAMI 4.0 and the IIRA reference architectures is given in the joint whitepaper (Lin et al., 2017).

2.1.5 Connectable Objects

(Physical) reachable objects inside of a Smart Manufacturing network (e.g. digitalised and virtualised field level devices, systems, material, integrated humans, virtual concepts, etc.), have to fulfil a range of requirements. Objects should communicate using a unified communication protocol, at least at the application level, and should be based on a unified semantic to enable a mutual identifiability and understanding. The object itself should provide its own features as a service (e.g. state information or functionalities) and should be able to provide its own descriptions next to extended information, such as manuals, specifications, or wear information. All this has to be kept next to further requirements related to security, safety or quality of service (DIN SPEC 91345:2016-04, 2016; VDI, April 2015).

2.1.6 Connectable Applications

Various applications can be implemented that use services of deployed objects to realise, for instance, control systems, systems of systems through service orchestration, or Big Data analysis applications.

2.2 Data in Smart Manufacturing Systems

Data in Smart Manufacturing systems are often continuously generated data produced by machines, ambient sensors (temperature, vibration, humidity, etc.), controllers, (manufacturing) systems, etc. (ElMaraghy, Lee, Kao, & Yang, 2014), available in a high variety as e.g. in form of signal streams, log files, master data, events, emails, media data or manually entered operator data (Kurtz & Shockley, 2013). Further data in such environments include data sources from the enterprise level, supply chains, marketing and sales, from Product Lifecycle Management (PLM) systems, social media, website browsing patterns, or from business forecasts (Luckenbach, Stackowiak, Licht, & Mantha, 2015).

2.2.1 Big Data

The diverse landscape of such data is often summarised as Big Data. The definition of Big Data is still under discussion and many suggestions are available. Gartner made a proposal in 2011 where it was suggested to categorise Big Data through 3Vs (Volume of Data, Variety of Data and Velocity of Data) (Wu, Zhu, Wu, & Ding, 2014). This categorisation (/definition) is widely accepted. Newer definitions have also added a fourth V standing for "veracity", which describes the quality of captured data that can vary and affect the accurate analysis (Villanova University Alliance). Big Data and its exploitation is becoming an important topic (Bloem et al., 2014; Labrou & Finin, 1999) and is a broad interdisciplinary research area which covers the idea to extract implicit, previously unknown, and potentially useful information from data (Ghemawat, Gobioff, & Leung, 2003). The volume, velocity and variety of data in smart manufacturing is increasing (Mourtzis, Vlachou, & Milas, 2016) and in parallel also the exploitation potential (Nagorny et al., 2017). Today, smart manufacturing systems must consider Big Data (Kusiak, 2017; Lee, Lapira, Bagheri, & Kao, 2013).

2.2.2 Big Data Value (BDV) Reference Model

The Big Data Value Association (BDVA) as international industry-driven, non–for-profit, organisation and private counterpart to the EU Commission to implement the Big Data Value PPP program should be mentioned. The BDVA aims to create an innovation ecosystem that enables the data and artificial intelligence-driven digital transformation in Europe.

One outcome of the BDVA is the BDV Reference Architecture shown in Figure 2-8 which aims to be a reference framework to locate needed technologies considering the whole Big Data Value chain.

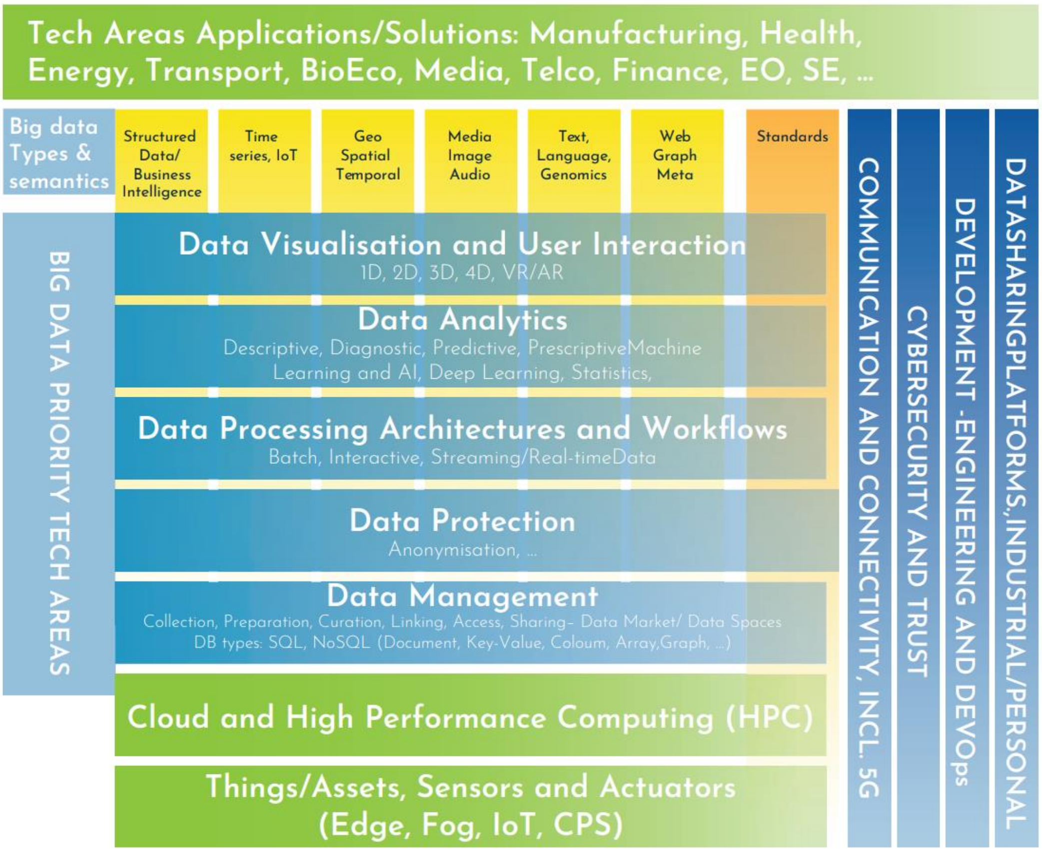


Figure 2-8: Big Data Value (BDV) Reference Model *(Big Data Value Association, 2018)*

The reference model shows the different Big Data priority technology areas that have to be taken into account while developing Big Data solutions:

- **Data Management**: Includes methods for managing data and is addressing the increasing amount of heterogenous Big Data (sources) and their complexity.

- **Data Protection**: Addresses the need of data governance issues, such as data anonymisation and privacy issues.
- **Data Processing Architectures and Workflows**: Addresses the need of scalable data processing and transportation architectures and workflows for Big Data including batch and streaming data, and requirements related to real-time.
- **Data Analytics**: Addresses topics around data analysis to improve data understanding, data mining, deep learning, and meaningfulness of data and to transform data into value.
- **Data Visualisation and User Interaction**: Addresses the need of new approaches for visualisations in the Big Data domain, to handle the continuously increasing complexity and size of data for human users.

Next to the Big Data priority technology areas, the BDV Reference Model addresses also cross-level priorities shown on the right side.

- **Communication and Connectivity, Incl. 5G**: Addresses the need of seamless communication between data producers, also for the mobile area to support Big Data.
- **Cybersecurity and Trust**: Addresses the need of security, since several potential attacking scenarios are possible in complex networked systems, as described in (Group, 2016). It addresses also the need of trust in data to keep control over exchanged data (keywords are certificate, block chain or smart contracts).
- **Development-Engineering and DevOps**: Addresses the need of (next generation) engineering systems and development operations (tool chains and Big Data platforms) to build Big Data Value systems.
- **Data sharing platforms, Industrial/Personal**: Addresses the need of platforms to enable a secure and trustful data exchange. This can be platforms for Business to Business (B2B), Business to Consumer (B2C) or B2G (Business to Government) data exchange, but also platforms without commercial interests as part of smart cities.
- **Standards**: Addresses the need of standards to align approaches developed for different layers.

Next to the BDV reference model also other reference models exist as for example the National Institute of Standards and Technology (NIST) Big Data Reference Architecture (NBDRA), which is described in detail in (Chang & Boyd, 2018).

2.3 Data understanding and data acquisition in Smart Manufacturing

Data understanding and data acquisition are terms that are often mentioned in data analysis processes. Before a data analysis can be performed, it has to be considered that different experts are needed for such a process. At least two experts are needed to make a data analysis itself, namely, a Data Scientist who understands the analysis methodologies and is able to use the tools, and an IT expert who is able to deploy, configure and administrate databases and computer clusters needed for the analysis of data (Weps, July 2016). In addition, two domain experts are commonly needed: one expert who has clear expectations of data analysis outcomes, and another expert who has knowledge about the data needed for the data analysis (Weps, July 2016). In manufacturing this could mean that a production site expert who has clear expectations about the data analysis outcomes (e.g. optimisation of a KPI) is needed, and a manufacturing expert who knows the production processes and the available data (Weps, July 2016). It is very important, and a big challenge, that all involved experts get a clear and common understanding of aimed Big Data analysis expectations/goals (Weps, July 2016).

The workflow of a data analysis process is often following the CRISP-DM (Manyika et al., 2011; Shearer, 2000) which is divided into six phases (Morik, 2008) as shown in Figure 2-9.

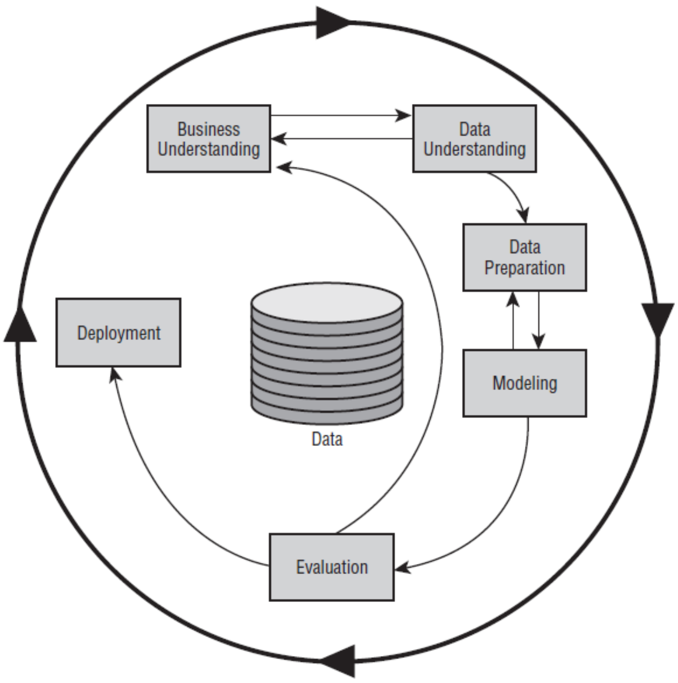


Figure 2-9: The Cross-Industry Standard Process for Data Mining (CRISP-DM) (Jones, 2011)

Each phase includes different tasks:

1st Phase - Understanding the Business: includes the determination of business objectives, situation assessment and a determination of Big Data analysis expectations/goals.

2nd Phase – Data Understanding: includes the collection of initial data, exploration and description of data and the verification of data quality.

3rd Phase – Data Preparation: includes the selection, extraction, cleaning, construction, integration and formatting of data.

4th Phase – Modelling: includes the selection of a modelling technique; the generation of a test design; and the building and assess of a model.

5th Phase – Evaluation: includes the evaluation of Big Data analysis results, knowledge discovery and planning of next steps.

6th Phase – Deployment: includes the exploitation of lessons learned: use of new knowledge, e.g., for predictions, further observations, decision support, automation, etc. (Weps, July 2016).

This doctoral work aims to support the 2nd "Data Understanding" and partially the 3rd "Data Preparation" phases. Both phases according to CRISP-DM should therefore get a stronger focus with a summary from (Shearer, 2000):

Data Understanding

The purpose of the data understanding phase is to get familiar with the data landscape. Therefore, an analyst starts with an initial **data collection** and also notes identified problems that appeared while, for instance, accessing data sources and extracting the data. Then the analyst **describes the data** characteristics of those collected data, such as format, quantity, number of records, table fields, etc. After this step, a first understanding of the data, and fitting of data requirements based on the use case should be given. The next step goes deeper to the **exploration of the data** itself. The analyst performs first analyses to identify which suitable data-sets are included in the data and can be used later for the intended use case. In a final step, the **quality of data** where the analyst checks completeness (missing values, attributes, fields), plausibility, spelling and meaning of data, gets verified.

After this data understanding phase, the analyst should have a good overview of the collected data and should be able to start with its preparation.

Data Preparation

The purpose of the data preparation phase is that the analyst models the data so that they fit into the models of the next phase. First, the analyst starts with the **selection of needed data** (like tables, records or attributes). In the second step, the selected data undergoes a **data cleaning** procedure (if needed). In this procedure the analyst may (e.g.) create estimates for missing data values or delete outliners in a data set and should have at the end only the data which is usable for further processing. In a third step, the analyst may **construct data** out of the cleaned data (like construct/calculate the volume from the side lengths of a rectangle). Cleaned and constructed data will be **integrated** by the analyst in the fourth step: Some data may belong together but - for example - are still in individual, unlinked tables that have to be merged/integrated in this step. Further actions may be aggregations like the calculation of sums, averages, counts, etc. Finally, it may be necessary to **format the data**. Data formats may not fit and need to be converted (as e.g. a String to Date), but it could also be that the data just needs some refinements like trimming of whitespaces or handling of special characters.

2.4 Data pipelining

Data pipelining (Alley, 2019; Densmore, 2021) evolved from ETL (Extract, Transform, Load) approaches. ETL is a term, emerged in the data warehousing domain (Denney, Long, Armistead, Anderson, & Conway, 2016) and is used when data has to be moved or copied from a source system into a target system through an Extraction (source), Transformation (data transformation compliant to the target system(s)) and Loading (move/copy transformed data to the target system(s)). Another form of ETL is Extract, Load, Transform (ELT). Such approaches are needed when it comes to transfer data (buckets or streams) from one point to another while the data transformation happens in the target system. ETL and ELT related technologies provide different I/O interfaces and processors for data cleaning or transformation.

Data pipelining is a broader term that encompasses ETL and ELT as a subset. It refers to a system for moving data from one system to another while it may or may not be transformed. When the data is streamed, it is processed in a continuous flow which is useful for data that needs constant updating, such as data from a sensor monitoring traffic (Alley, 2019).

Many ETL/ELT/Data Pipelining related approaches and technologies are already available in the literature – for instance:

- **Apache NiFi (The Apache Software Foundation, 2019)** - A distributed dataflow engine by the Apache Software Foundation written in Java. It is a flow-based programming approach, is very flexible and includes several processors to handle different kinds of data source interfaces, or to clean and transform data. It also allows to write custom data flow processors (Samal & Panda, 2017).
- **StreamSets Data Collector (StreamSets, 2019)** - A distributed dataflow engine by the StreamSets Inc. written in Java. Provides an attractive user interface and provides typical features for data pipelining.
- **Hevo Data (Hevo Data Inc., 2019)** - A data pipeline tool by Hevo Data Inc. to clean, enrich, and transform data on the fly. It offers several standard data source processors for several kinds of data source interfaces and offers also an automatic data type and schema mapping for selected source and target system technologies.
- **Apache Airflow (Apache Software Foundation, 2019)** - A data pipeline tool by the Apache Software Foundation written in Python. Airflow is a platform to programmatically author, schedule, manage, and monitor workflows which represent these as - so called - directed acyclic graphs (DAGs) of tasks.
- **Talend Open Studio (Bowen, 2012)** - Is an ETL software by Talend written in Java. Talend Open Studio is the open-source version which contains selected core modules of their Data Management Platform and is therefore limited in its functionality.

2.5 Data pipelining compliance with DIN-SPEC-91345 and other industrial norms

This doctoral work provides an approach for DIN-SPEC-91345 compliant Big Data pipelining. While the most data pipelining approaches provide features

- to access several kinds of data sources and extract data from there,
- to transform extracted data e.g. to map data types between selected data sources,
- to load data into a target system,
- to access the data pipeline approach by an API,

they are not going further and combine semantical models to enable a semi-automatic partly deployment of a data pipeline. This doctoral work fills this gap and provides a combination of outcomes of the Industrie 4.0 (e.g. the RAMI 4.0), and other manufacturing related norms, for a semi-automatic data extraction. Therefore, a pipelining approach that uses meta-data descriptions to initialise data source processors and which is compliant to the DIN-SPEC-91345 needs to take in account a range standards and norms. This section introduces the most important standards for this doctoral work in the following sub-sections.

2.5.1 DIN-SPEC-91345 – The Industrie 4.0 Component

While the RAMI 4.0 reference architecture has already been introduced in section 2.1, the associated DIN-SPEC-91345 standard also provides details about an I40 component. This is important for this doctoral work since it provides requirements for the data model to be developed in this work to describe a smart manufacturing environment.

The VDI/VDE Society Measurement and Automatic Control (GMA) defines the Industrie 4.0 component as follows (translated):

The term "component" is general. It describes an object of the physical world or the world of information that can be perceived and handled as a unit and that plays a certain role in its system environment or is intended for such a role. A component can be e.g. a pipe, a function module, a lamp, a valve, an intelligent drive unit etc. Important is to view the component as a unit related to its role (function) that it should or it already perceived in a system. The I40 component is a special type of component. I40 components are characterised by the fact that they fulfil certain requirements with regard to the classification features described. Even in an I40 system there are many components that do not meet these requirements and are therefore not I40 components (Adolphs Peter et al., 2014) .

The definition shows that an object that can be called “Industrie 4.0 component” must be capable of communication and it must be known in the network as a type or instance. Furthermore, the object must also provide information that virtually represents the object. This information must be compliant with the chosen semantic used for the communication in the network, to guarantee an interoperability with Industrie 4.0 compliant network participants (Matthias Ruhl, 2017).

An Industrie 4.0 component is divided into five levels as shown in Figure 2-10.

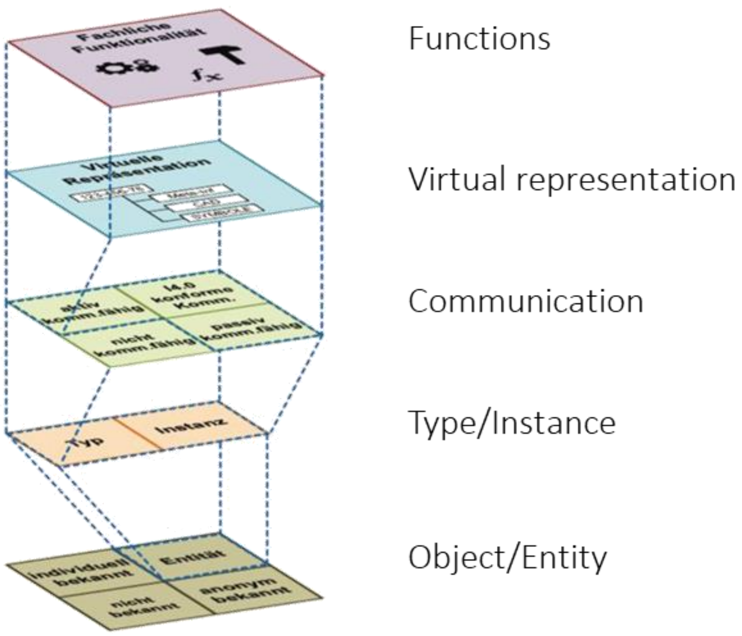


Figure 2-10: Industrie 4.0 Component Levels (from *(DIN SPEC 91345:2016-04, 2016)*)

- **Functions Level**: Processes data and returns calculated conclusions or results or information. The function layer is not only an external interface but addresses also internal functions to enable self-capabilities as self-calibrations or self-maintenance.
- **Virtual representation Level**: describes the component in the information world and up to a digital twin.
- **Communication Level**: shows that the I40 component must be capable of communication. It can also be passively capable of communication (e.g. RFID chip) (Matthias Ruhl, 2017).
- **Type/Instance Level**: describes the I40 component in the life cycle. It can exist either as an instance or as a type or both.
- **Object/Entity Level**: The lowest level shows the level of awareness of an I40 component in the network. Only if an object is an entity or object, it can be transformed into an Industrie 4.0 component and appear in the Industrie 4.0 compliant network (Döbrich, Hankel, Heidel, & Hoffmeister, 2017; Matthias Ruhl, 2017).

Figure 2-11 shows the concept of an AAS. The AAS concept describes a shell which is part of an Industrie 4.0 component. The shell is able to provide a semantical description of an Industrie 4.0 component which includes models for functional and non-functional properties and available kinds of raw data and information over its life-cycle. Not all data needs to be saved in an AAS but references which point to contents saved in other places. Different kinds of meta-data are provided in different life-cycle phases (e.g. design data from the I40 component development phase, or maintenance reports in the maintenance/usage phase).

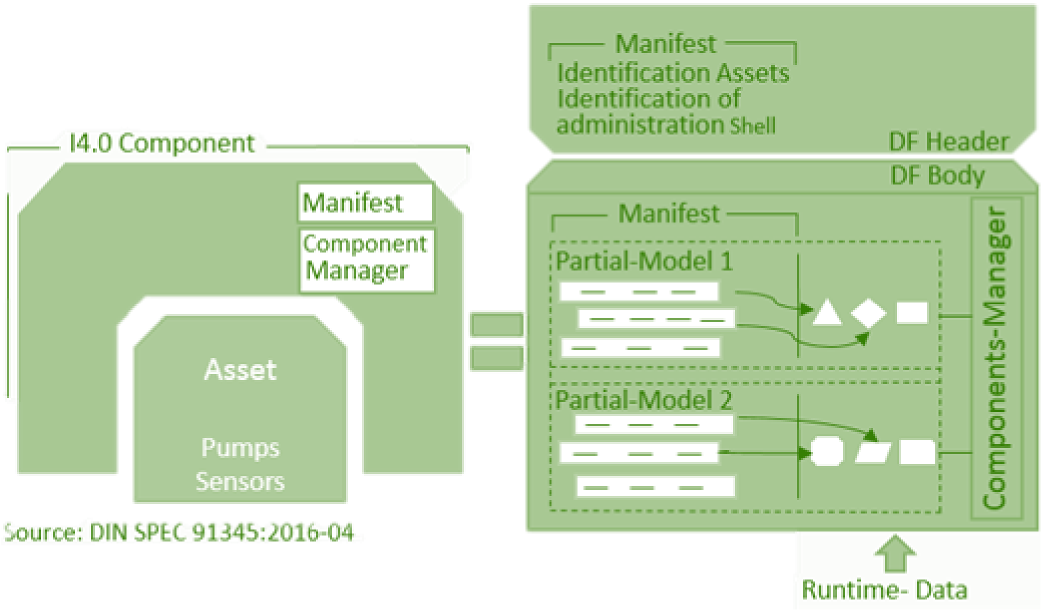


Figure 2-11: Asset Administration Shell (from DIN-SPEC-91345)

Some examples of meta-data are shown in Table 2-1.

Table 2-1: Examples of meta-data

Life-cycle phase	**Category**	**Examples**
Type - Development	Design	Physical dimensions, properties
Type - Maintenance and Usage	Revisions	Type/revision ID, Bill of Materials (BOMs)
Instance - Production	Production data	Quality control data, batch ID, serial number
Instance - Maintenance and Usage	Usage	Maintenance services, recycling

The AAS includes a manifest with the characteristic properties of the Industrie 4.0 component and a Component Manager which references to partial-models which define properties of included services and refers to their individual functions and data (DIN SPEC 91345:2016-04, 2016). More details about the AAS are available in (ZVEI, 2016).

A partial-model has to be linked to at least one of the basic views shown in Figure 2-12 to allow a categorisation and to enable an easier finding.

A semantic model for DIN-SPEC-91345 compliant data pipelining has to be based on the basic structure of an Industry 4.0 component.

There are also other norms in the smart manufacturing domains. One important norm is the eCl@ss standard which is used to classify product data and services.

Defined basic views for the Administration Shell

Basic view	Best practice/ examples
Business	Data and functions are deposited which allow judging on the business suitability and performance of a component in the life cycle phases Procurement, Design, Operation and Realisation. Examples: prices, terms of delivery, order codes
Constructive	Contains properties relevant for the constructive deployment of the component, thus for selection and building structure. Contains a structure classification system pursuant to EN 81346. Contains numerous properties in respect of physical dimensions and regarding start, processing and output values of the component. Contains a modular view of subcomponents or a device structure. Allows an automation view with inputs and outputs of different signal types.
Performance	Describes performance and behavioural characteristics in order to allow a summary assessment and Virtual Commissioning (V-IBN) of an overall system.
Functional	Makes statements on the function pursuant to EN 81346 and on the function of the subcomponents. Here location of the individual functions of the Technical Functionality also takes place, thus for example so-called “skills”, interpretation, commissioning, calculation or diagnosis functions of the component.
Local	Makes statements on positions and local relationships between the component or its parts or inputs and outputs[22].
Security	Can identify a property as security-relevant. This property should be taken into account for an assessment of security.
Network view	Makes statements in respect of electrical, fluidic, materials flow-related and logical cross-linking of the component.
Life cycle	Contains data on the current situation and historical utilisation in the life cycle of the component. Examples: allocation to production, maintenance protocols and past applications.
People	In all views properties, data and functions should appear such that humans can understand individual elements, inter-relationships and causal chains.

Figure 2-12: Basic views for the AAS (ZVEI, 2016)

2.5.2 eCl@ss

A popular candidate for the languages of Industrie 4.0-compliant eco-systems is the eCl@ss standard which is used to describe characteristics of products and services based on international standards. Many participants actively further develop the eCl@ss standard for Industrie 4.0 environments (eCl@ss e. V., 2019a). eCl@ss is a hierarchical system for grouping materials, products and services according to a logical tree structure, which is divided into five levels:

- **Level 1 - Segments:** are the most general categorisation and are used to categorise branches or markets.
- **Level 2 - Main Groups:** are main groups of a segment.
- **Level 3 – Groups:** are finer groupings of the main groups into sub-groups.
- **Level 4 – Commodity or Product Classes:** define a product or a service.

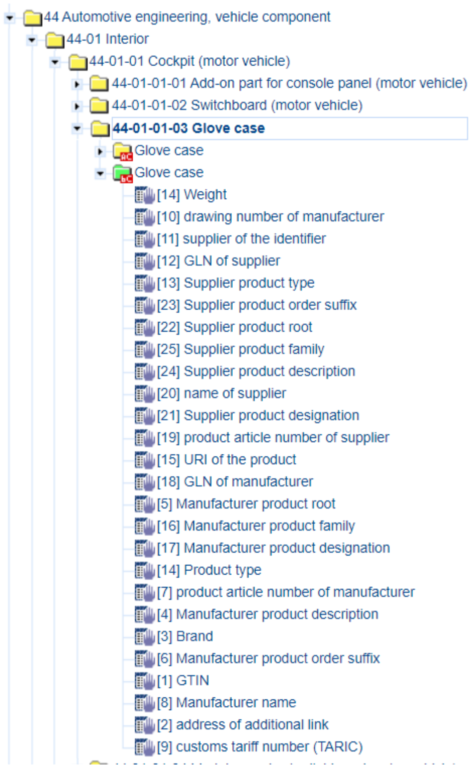


Figure 2-13: eCl@ss example – car glove case

- **Level 5 – Properties:** a product or service is further described by properties. A Property can have a specific property value. The property values can also be defined (e.g. Integer or Enumeration (ENUM))

Example: **44**-00-00-00 (**Segment**: "Automotive engineering, vehicle component"), 44-**01**-00-00 (**Main Group**: "Interior"), 44-01-**01**-00 (**Group**: "Cockpit (motor vehicle)"), 44-01-01-**03** (**Commodity class**: "Glove case"). The commodity class "Glove case" is then described by associated properties as shown in Figure 2-13.

To address segments, main groups, groups, commodity and product classes, and properties, an "International Registration Data Identifier" (IRDI) is used in eCl@ss for their unique identification. The structure of an IRDI is defined as explained in the following:

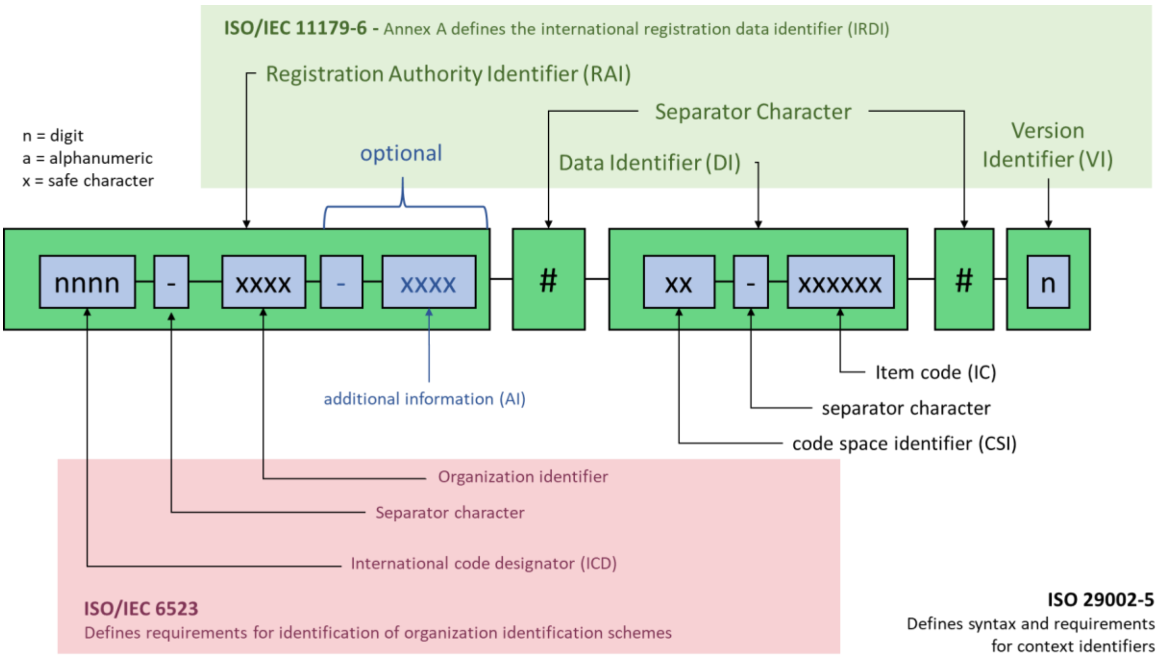


Figure 2-14: Structure of the IRDI and related standards IRDI (compare *(eCl@ss e. V., 2019b)*)

Figure 2-14 shows the globally unique identifiers called IRDI which is based on the international standards ISO/IEC 11179-6, ISO/IEC 6523 and ISO 29002-5 (see (eCl@ss e. V., 2019b)). The IRDI model is next to URI well known as suitable unique identifier in Industrie 4.0 challenges.

The IRDI is separated into six fields and five separators as described in Table 2-2.

Table 2-2: Fields of the IRDI

Part of the IRDI	Based on Standard	Description	Length	Importance	Used in eCl@ss
nnnn-xxxx-xxxx#xx-xxxxxx#n	ISO/IEC 6523 ISO/IEC 11179-6	Defines the international code designation (ICD) (ISO, 2019b). Identifies the authority which issued the code to the organization.	<= 4 digits	Mandatory	Yes
nnnn-xxxx-xxxx#xx-xxxxxx#n	ISO/IEC 6523 ISO/IEC 11179-6	Identifier of an organisation.	<= 35 characters	Mandatory	No
nnnn-xxxx-xxxx#xx-xxxxxx#n	ISO/IEC 6523 ISO/IEC 11179-6	An organisation part identifier (OPI) which can be any kind of entity within an organization. Maybe followed by an OPI source indicator (one additional digit), specifying who attributed the OPI.	<= 35 characters	Optional	Yes
nnnn-xxxx-xxxx#xx-xxxxxx#n	ISO/IEC 11179-6 ISO 29002-5	According to ISO 29002-5 there is a range of Code Space Identifier (CSI) listed in Table 2-3. These CSIs describe the category of administered items.	<= 2 characters	Mandatory	Yes
nnnn-xxxx-xxxx#xx-xxxxxx#n	ISO/IEC 11179-6	The item code (IC). Defines any item inside a company.	<= 6 characters	Mandatory	Yes
nnnn-xxxx-xxxx#xx-xxxxxx#n	ISO/IEC 11179-6	The Version Identifier (VI) defines the version of the item.	1 digit	Mandatory	Yes

Table 2-3: ISO 29002-5 Code Space Identifiers used in eCl@ss (compare *(eCl@ss e. V., 2019b)*)

Code Space Identifiers	Category of administered item
1	class
2	property
5	unit of measure
7	property value
9	data type
11	ontology
Z2	aspect of conversion
Z3	template
Z4	quantity

For example, the Commodity class "Glove case" from the example shown in Figure 2-13, can be addressed. The full eCl@ss (version 10.01) definition is described in Code 1.

```
<ontoml:class
        xmlns:ns11="urn:x-catalogue-extensions:schema:core"
        xmlns:ns10="urn:x-value-extensions:schema:value"
        xsi:type="ontoml:CATEGORIZATION_CLASS_Type"
        guid="bbc7ee96f32c4ec28e25dfbd2b3e8d04"
        id="0173-1#01-AFY260#004"
>
  <date_of_original_definition>2014-11-30Z</date_of_original_definition>
  <date_of_current_version>2018-02-25Z</date_of_current_version>
  <date_of_current_revision>2018-02-25Z</date_of_current_revision>
  <revision>1</revision>
  <status>66</status>
  <source_language country_code="US" language_code="en"/>
  <preferred_name>
        <label country_code="US" language_code="en">Glove case</label>
  </preferred_name>
  <definition>
        <text country_code="US" language_code="en">
                Assembly group or element of a motor vehicle
        </text>
  </definition>
  <its_superclass class_ref="0173-1#01-AFX866#002"/>
  <hierarchical_position>44010103</hierarchical_position>
</ontoml:class>
```

Code 1: eCl@ss definition of the Commodity class "Glove case" with ID "AFY260"

From the highlighted IRDI (0173-1#01-AFY260#004) the following information can be extracted:

- **0173**: is the official ICD for eCl@ss. => It is something from eCl@ss e.V.
- **1**: organization part identifier (OPI). "1" defines the eCl@ss standard of the eCl@ss e.V. => It is something addressed by the eCl@ss standard.
- **01:** It is a classification class
- **AFY260**: That is the ID of the "Glove case" class.
- **004**: That is the version of the "Glove case" class.

More examples can be extracted from the eCl@ss standard or in the work of Döbrich (Döbrich et al., 2017).

```
[001]     <ontoml:property
[002]             xmlns:ns11="urn:x-catalogue-extensions:schema:core"
[003]             xmlns:ns10="urn:x-value-extensions:schema:value"
[004]             xsi:type="ontoml:NON_DEPENDENT_P_DET_Type"
[005]             guid="8bcd1cb7c24e4d609b5016b1be5f098e"
[006]             id="0173-1#02-AAB713#005"
[007]       >
[008]       <name_scope class_ref="0173-1#01-RAA001#001"/>
[009]       <date_of_original_definition>2008-03-29Z</date_of_original_definition>
[010]       <date_of_current_version>2013-11-28Z</date_of_current_version>
[011]       <date_of_current_revision>2013-11-28Z</date_of_current_revision>
[012]       <revision>1</revision>
[013]       <status>66</status>
[014]       <source_language country_code="US" language_code="en"/>
[015]       <preferred_name>
[016]             <label country_code="US" language_code="en">Weight</label>
[017]       </preferred_name>
[018]       <definition>
[019]             <text country_code="US" language_code="en">
[020]                     Mass of weight without packaging and transport unit
[021]             </text>
[022]       </definition>
[023]       <det_classification>K01</det_classification>
[024]       <domain xsi:type="ontoml:REAL_MEASURE_TYPE_Type">
[025]             <unit unit_ref="0173-1#05-AAA731#002"/>
[026]       </domain>
[027]     </ontoml:property>
```

Code 2: eCl@ss 10.01 definition of the property class "Weight" with ID "RAA001"

To continue the Glove case example related to data properties: The Application Class of the Glove case contains the property "weight" (compare Figure 2-13). This property is shown in Code 2. The property contains a list of attribute names as "preferred_name", "definition", "revision", etc. It indicates also, that it is a REAL data type (see line 24) and it references to a unit description. The referenced unit with IRDI "0173-1#05-AAA731#002" is shown in Code 3.

```
<unitsml:Unit
        dimensionURL="0173-1#Z2-AAA012#001"
        xml:id="id0173-1x05-AAA731x002"
        >
        <unitsml:UnitName xml:lang="de-DE">kg</unitsml:UnitName>
        <unitsml:UnitName xml:lang="en-US">kg</unitsml:UnitName>
        <unitsml:UnitSymbol type="ASCII">kg</unitsml:UnitSymbol>
        <unitsml:CodeListValue
                codeListName="IRDI"
                unitCodeValue="0173-1#05-AAA731#002"
        />
        <unitsml:CodeListValue codeListName="SI code" unitCodeValue="kg"/>
        <unitsml:CodeListValue codeListName="DIN code" unitCodeValue="kg"/>
        <unitsml:CodeListValue codeListName="ECE code" unitCodeValue="KGM"/>
        <unitsml:Conversions>
                <unitsml:Float64ConversionFrom
                        initialUnit="kg"
                        divisor="1.0"
                        multiplicand="1.0"
                        xml:id="AAA731-to-AAA731"
                />
        </unitsml:Conversions>
        <unitsml:QuantityReference
                xml:lang="en-US"
                name="mass"
                url="0173-1#Z4-BAJ213#001"
        />
        <unitsml:UnitDefinition xml:lang="de-DE">
Einheit der Masse, die gleich der Masse des internationalen Kilogrammprototyps ist
        </unitsml:UnitDefinition>
        <unitsml:UnitDefinition xml:lang="en-US">
unit of mass, which is equal to the mass of international prototype of the kilogram
        </unitsml:UnitDefinition>
</unitsml:Unit>
```

Code 3: eCl@ss 10.01 definition of the unit "kg" with ID "AAA731"

The unit in Code 3 provides further attribute names, as the "initialUnit", "UnitName" or the "UnitDefinition".

To summarise the "Weight" property example: The weight of a Glove case is described in eCl@ss v 10.01 with the following main attributes listed in Table 2-4.

Table 2-4: Extraction of characteristic description of Code 2 and Code 3.

Attribute Name	Value
Identifier	AAB713
Version	005
Revision	1
Preferred Name	Weight
Synonym	-
Definition	Mass of weight without packaging and transport unit
UnitName	mass
InitialUnit	kg
DataType	REAL
UnitDefinition	unit of mass, which is equal to the mass of international prototype of the kilogram

Although the full eCl@ss standard is a bit more complex,

(for example: next to classification classes, properties and units also other structural elements are existing: Value, Keyword, Synonym, Value List, Application Class, Aspect, Block and Reference Property, etc.)

, the given description provides a good overview to understand the basic aspects of eCl@ss.

As eCl@ss is a candidate for the languages of Industrie 4.0 compliant eco-systems, this standard will be taken into account in this doctoral work, for the data pipelining approach.

As the reader may have already recognised, Code 1 shows a list of meta-data to describe the commodity class (e.g. preferred_name, definition, hierarchical_position, etc.). eCl@ss uses a list[2] of attribute names to describe included elements as classes or properties. The most attribute names are based on the IEC 61360 standard.

[2] for classification classes see e.g. the eC@ss e.V. Wiki: http://wiki.eclass.de/wiki/Classification_Class

2.5.3 IEC 61360

Standardised properties are often used to describe objects in the engineering domain as analysed in (Hadlich, 2015). The IEC 61360 (Reihe, 2009) is a standard to describe electric and

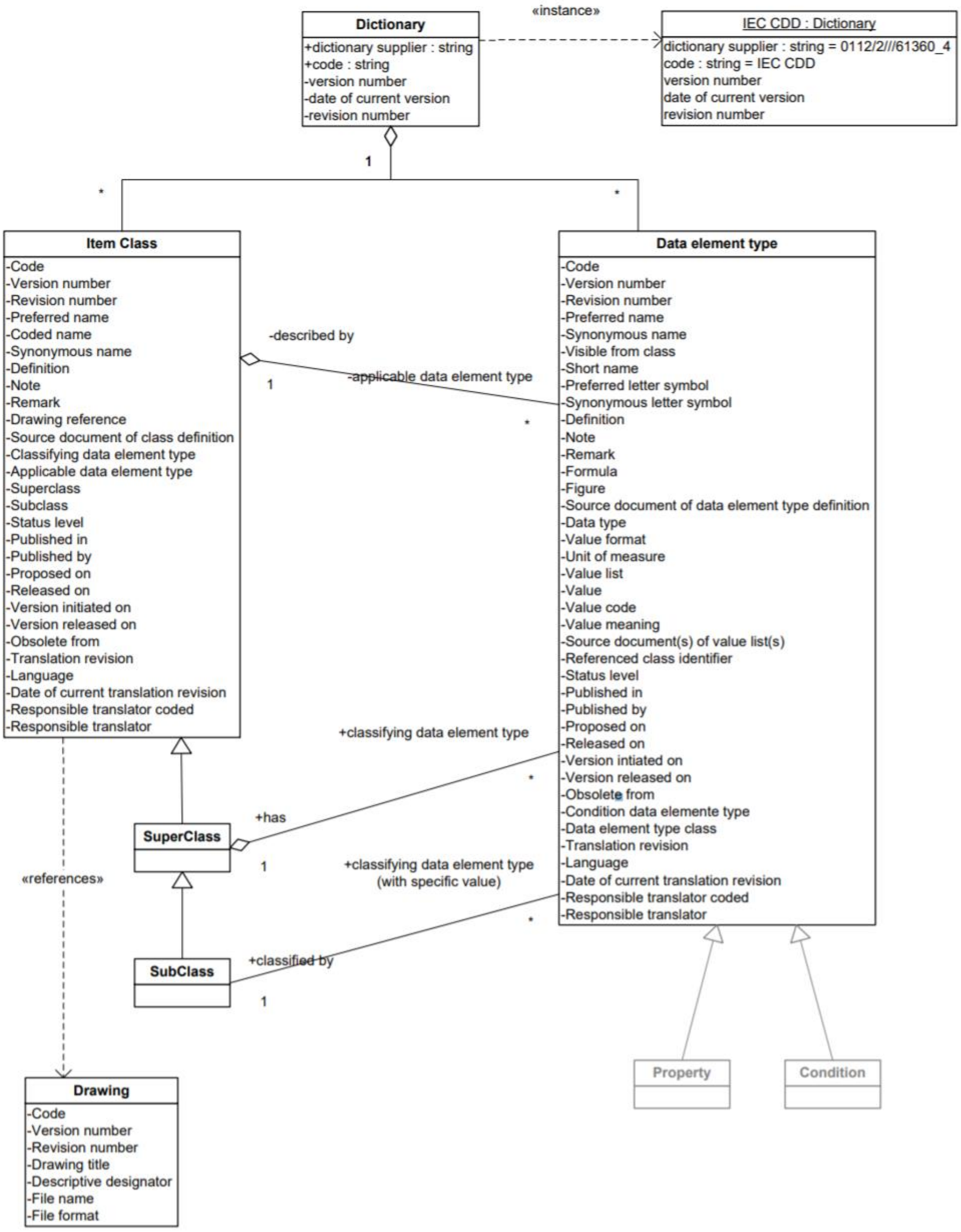


Figure 2-15: IEC 61360 data types *(Hadlich, 2015)*

electronic components. The idea behind this standard was to describe those components in a unified way and enable computers to understand each other across companies. Therefore, it was necessary to define e.g.

- allowed values
- used units
- properties for the identification and characteristic
- data schemas

The IEC 61360 approach had the premise that components, materials and geometry can be fully described by their characteristics in form of technical data element types (VDE Verlag, 2019). Therefore, IEC 61360 introduces several data properties as shown in Figure 2-15.

These data properties will also be used as a basis for the semantical data model developed in this doctoral work.

2.5.4 Semantic models

Many research and innovation activities show that semantics in smart manufacturing systems will be based on semantic models. Those results are for instance reported in VoCol (Grangel-González et al., 2016; Halilaj, Grangel-González, Coskun, Lohmann, & Auer, 2016; Halilaj, Petersen, et al., 2016; IAIS Fraunhofer, 2018), a collaborative space to achieve unified semantics where also an initial semantic model related to the DIN-SPEC-91345 (RAMI 4.0) is available (Irlan Grangel-González, 2017).

Semantic models play an essential role in building a unified description for smart manufacturing systems. The harmonization of semantics is an increasingly important topic in this domain and is one of the key prerequisite to guarantee system interoperability (Alexander Fay et al., 2017; Jardim-Goncalves, Sarraipa, & Steiger-Garcao, 2010). Many networks and organisations like the Semanz40 (Fay et al., 2017), eCl@ss e.V. (see www.eclass.eu), ProSTEP e.V. (Sendler, 2009), AutomationML e.V. (Lüder & Schmidt, 2017), OPC Foundation (OPC Foundation, 2012), PLC Open e.V. (PLCopen, 2019), International Electrotechnical Commission (IEC) (International Electrotechnical Commission, 2019), ISO (ISO, 2019a) or the Institute of Electrical and Electronics Engineers (IEEE) (IEEE, 2019) are currently working on this issue. Several already existing standards, aiming to standardise and unify the used semantics in industry, are basis for their work.

Ontologies

An Ontology is a representation of knowledge using concepts and relations between them. *"An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms."* - (Uschold, King, Moralee, & Zorgios, 1998)

In general, ontologies enable to describe the world in form of entities and their relations to each other. This is done by conceptualisation and specification in technical terms. While the conceptualisation describes an intended-model/implicit-knowledge about a domain, the specification of this intended-model/implicit-knowledge represents the ontology which is describing in a formal way the domain in entities and their relations.

Gruber wrote that the specification of conceptualisations are used to help programs and humans to share knowledge (Gruber, 1995). In this context many movements (Dais, 2017; Kagermann, 2017) show that semantic in Industrie 4.0 compliant systems will be based on ontologies with a unified vocabulary represented in `RDF->RDFS->OWL` formats. Ontologies are basically modelled in "Subject –predicate–> Object" triples (e.g. Alice --knows--> Bob) and they are used to define categories, definitions or properties for the definition of concepts, data, entities and relations (Staab & Studer, 2010).

The theory around ontologies is quite comprehensive. The works of (Bunge, 1977), (Weber, 1997) and (Stevens, Goble, & Bechhofer, 2000) give important insights to the subject of ontologies. Some important selected terms are introduced in Table 2-5.

Table 2-5 Important terms used in the domain of ontologies

Term	Definition
Concept	Describes a class or a set of entities within a domain.
Vocabulary	Can be a context-less list of terms, with no defined interrelationships but also a list of context-less concepts and their relations (terms) that are unique in all contexts. One example for a term could be the eCl@ss "Unit" example which was shown in Code 3 if it would be represented as specified semantic model: The defined unit "kg" has no further context but can be used in many domains as it has a consistent meaning in all contexts.

Taxonomy	Defines the organisation of a vocabulary or concepts in a hierarchical parent-child structure. One example could be: `Car -> Cockpit -> Glove Case.`
Ontology	Defines the formal representation/specification of concepts and their relations using a defined set of vocabularies.
Inference	Defines the extraction of logical conclusions from an ontology based on already known logical rules.

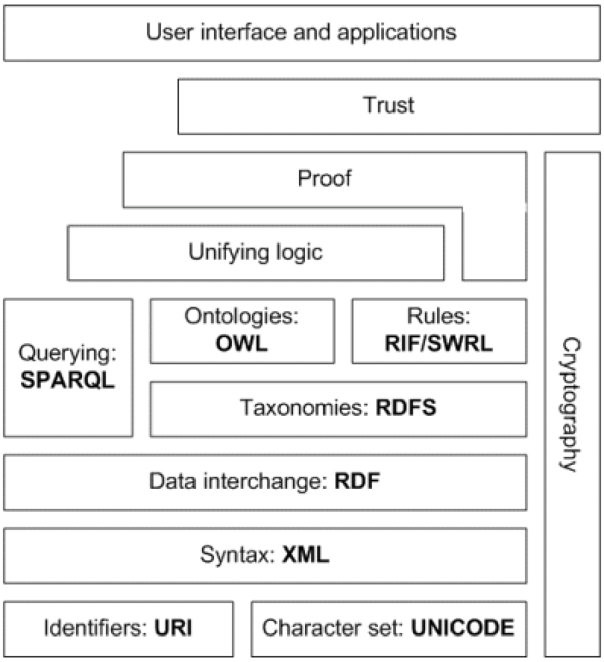


Figure 2-16: Semantic web architecture in layers (M Obitko, 2007)

Typical standards and technologies to implement ontologies are represented in the semantic web architecture shown in Figure 2-16. The architecture builds upon URIs as identifiers using a unicode character set as defined in ISO 10646. The structure is typically given by the World Wide Web Consortium (W3C) standard XML as a serialization format which enables the encoding of data in a machine and human readable way. In the next layers the semantical part begins.

The typical W3C standard to model graphs is the XML based Resource Description Framework (RDF), which is also the standard model to exchange information in the Web. RDF structures data and linked resources ("any identifiable abstract, digital or physical things (Fielding & Taylor, 2000)) in subject–predicate–object triples (e.g. Driver -> uses -> Glove Case) using URIs as identifiers. The result is a directed and labelled graph, where edges are named links between resources, represented by the graph nodes. RDF enables also the use of formal vocabularies to describe information and has some other advantages, useful to describe data in a semantic web, enabling of data merging and the evolution of data schemas (Systap, 2015).

In the next layer the W3C standard Resource Description Framework Schema (RDFS) is defined, which extends RDF with basic/light-weight vocabularies and provides possibilities to describe groups of resources and their relations. RDFS can be already used to build lightweight ontologies through the creation of taxonomies of classes and properties (Marek Obitko, 2007) but is limited in provided features.

Full featured ontologies can be built using the W3C specification Web Ontology Language (OWL). OWL is syntactically embedded in RDF(S), but extends the RDFS vocabulary and enables to describe anything that can be described as data.

RDF/S and OWL formats can be queried by the Simple Protocol and RDF Query Language (SPARQL) which follows a similar structure as SQL (Segaran, Evans, & Taylor, 2009).

2.6 Recap, gaps and contributions of this doctoral work

Smart Manufacturing

The SotA analysis on Smart Manufacturing (see section 2.1) describes how modern manufacturing infrastructures are built and why manufacturing systems are becoming smart.

In addition to conventional systems, several new approaches and paradigms are emerging and evolving to (1.) exploit new technological opportunities and (2.) meet increasing customer demands. This includes new paradigms like SOA, MAS, Systems of Systems (SoS) or (I)IoT or new communication technologies and protocols as OPC-UA, REST, MQTT, ACL, DPWS, 5G or GraphQL. All this while conventional protocols like Controller Area Network (CAN) bus, Profibus, Modbus, Profinet will also still play an important role in the future.

This leads to the assumption that communication technologies will not lose their variety in the next decades. A circumstance that requires new approaches in data acquisition that are compatible with this high variety.

The analysis of reference architectures gave an overview of some of the most popular Information and Communications Technology (ICT) architectures that refer also to smart manufacturing. In many areas, different reference architectures came up (like IIRA in America, IMSA in China or RAMI 4.0 in Europe). Although the focus of the individual architectures is a bit different, the conclusion was that the most reference architectures are mappable and that there is a common ground in the development of modern ICT architectures, which was also confirmed by Martin Hankel, Bosch Rexroth AG, by Prof. Dr.-Ing. Alexander Fay from the Helmut-Schmidt-Universität Hamburg and by experts in the BOOST 4.0 project (Hankel, 2018). Since this doctoral work was done in Europe, it follows the developments in Europe and therefore focus on RAMI 4.0, standardised in DIN-SPEC-91345. The RAMI 4.0 standard already defines a list of further standards which have to be considered as IEC 62890 for the life-cycle and value stream or IEC 62264 and IEC 61512 for the hierarchy levels. These standards represent a widely accepted common ground/basement for further Industrie 4.0-related research and innovation in smart manufacturing in Europe.

This leads to the decision to follow and contribute to the Industrie 4.0 initiative with the RAMI 4.0 as basement, specified in DIN Spec 91345.

Data in Smart Manufacturing

This doctoral work has a focus on data in smart manufacturing. This indicates that the issue needed to be considered more closely, which was done in section 2.2.

Data in smart manufacturing is becoming more and more important. Some reasons are the increasing amounts of data, the hidden value in it, the increasing exploitation field and the globalisation which leads to closely interconnected systems.

The resulting very heterogenous data environments, the data management and the data exploitation creates a need for completely new architectures as exemplary shown with the BDV reference architecture, as well as new approaches for data integration, storing, analysis and evaluation.

This doctoral work uses data located in data sources or storages and forwards them for subsequent processing. For this reason, an overview of storage and processing approaches was provided including some exemplary technologies.

On the one hand, smart manufacturing data environments are becoming more and more complex, on the other hand, data is becoming more and more important. New approaches must follow and consider the entire digital thread of assets and processes along their lifecycle in such very heterogenous data environments.

Data understanding and data acquisition in Smart Manufacturing

The outcomes of this doctoral work aim to support data understanding and data acquisition in Smart Manufacturing. These terms are often used in data analysis processes, as described in the example of CRISP-DM. The related SotA analysis in section 2.3 gave an overview of such data analysis processes and highlighted in detail the needed steps for conventional data understanding and data acquisition.

The analysis showed why the data understanding and data preparation phases require such an effort and indicates why this doctoral work is a valuable contribution to this area. Simplifying "data understanding" and "data acquisition" in heterogenous data environments would strongly support data scientists as well as other experts in their daily work.

Data Pipelining

This doctoral work aims to elaborate a novel approach for data pipelining. Data pipelining itself is not new so that a SotA in this area was given in section 2.4.

As a result of the analysis, it was found that existing data pipelining approaches require high efforts and domain knowledge for their configuration although many features, such as automatically data type mapping, data source technology support and user-friendly interfaces, are provided. The semantical unification can support those kinds of approaches in the future, by using self-descriptive information coming from cyber-physical systems in a smart manufacturing environment suitable to easily initiate data acquisition of selected data in a data pipeline.

This strengthened the idea, that a semantic model-based data pipelining approach could extend existing data pipelining solutions.

Data Pipelining compliance with DIN-SPEC-91345 and other industrial norms

In section 2.5 the data pipelining compliance with DIN-SPEC-91345 and other industrial norms was analysed.

A range of major standards that have to be considered in modern smart manufacturing systems, have been analysed. Standards such as eCl@ss or IEC 61360 as candidates for the language in Industrie 4.0 compliant smart manufacturing systems used in a fundamental framework and embedded in a RAMI 4.0 based structure as a basement can enable Big Data pipelining compliance with DIN-SPEC-91345 and other industrial norms.

The state of the art in the unification of data descriptions was analysed. Introduced were the main organisations which are working semantics in Smart Manufacturing related to Industrie 4.0; collaborative working spaces; standardisation activities and finally outcomes in this domain. The outcome is, that future data models will be based on semantic models. The main reason for this is that traditional RDBMS are too statically/too inflexible to describe and represent all the dynamic relationships between physical and virtual objects and their data produced along their life-cycles; or in short, to represent virtually the real world. Semantic models enable interoperability between other semantic models; they are an open-world approach; they enable to represent the meaning of concepts; they enable reasoning; etc. Semantic models have a lot more benefits than traditional RDBMS systems so that it can be expected that graph-based data representation will evolve more and more in industrial smart manufacturing systems. The state-of-the-arts analysis and its results, leads to the decision for a semantic model-based approach for this doctoral work.

A further outcome is that latest semantical activities in this domain show that standards like eCl@ss are strong candidates for the standardised classification and description of products and services in Industrie 4.0. Their outcomes should be considered in this doctoral work. The collaborative working spaces, like VoCol, showed also that initial ideas for a RAMI 4.0 based semantic models have been elaborated e.g. by Irlan Grangel-González (Grangel-González et al., 2016). This doctoral work contributed beyond these outcomes, but consider them as a basement/as the state-of-the-art in this area.

The following Table 2-6 summarises a list of identified gaps in the SotA where this work will contribute and related research objectives.

Table 2-6: Identified Gaps and Contributions

Addressed SotA sections	Gap	Contribution	Related RO
Smart Manufacturing (2.1) **Data in Smart Manufacturing Systems (2.2)** **Data understanding and data acquisition in Smart Manufacturing (2.3)**	The high heterogeneity of smart manufacturing data environments and its lack of description make, especially for non-experts, data understanding and data acquisition to a quite challenging task (Cios et al., 2007). A suitable solution is not available yet.	This doctoral work addresses this issue and contributes an overall DIN Spec 91345 RAMI 4.0 compliant data pipelining approach to support data understanding and data acquisition in heterogenous (smart) manufacturing data environments.	RO2 RO5 RO6
Data pipelining (2.4)	SotA data pipelining approaches are very flexible and usable for the data acquisition of digitally accessible data. But the configuration of such	This doctoral work contributes an extending approach for SotA data pipelining approaches which enable a pre-configuration and de-	RO2 RO4 RO5 RO6

	data pipelining approaches for data acquisition still requires a lot expert knowledge about connectors, data and data sources, and time.	ployment of data source connectors, ready for the data extraction and sub-sequent processing or forwarding by using a semantic model-based knowledge graph.	
Data pipelining compliance with DIN-SPEC-91345 and other industrial norms (2.5)	"Data pipelining compliance with DIN-SPEC-91345 and other industrial norms" presented a screening and analysis of standards used for the description of smart manufacturing systems that should be considered while building an approach for DIN-SPEC-91345 compliant data pipelining. To follow the reasonable trend of using semantic models, the approach aims to use such a semantic-model as a basis for a knowledge graph to enable (1st) a description of available data in the smart manufacturing environment, and (2nd) to describe the access to the data. Although many semantic model approaches exist, only first rudimentary ideas for building a RAMI 4.0 semantic-model are existing. Gaps are to build a comprehensive version, to show an open approach for the integration of	This doctoral work contributes a DIN-SPEC-91345 compliant semantic model which is open for an integration of further Industrie 4.0 relevant norms. The semantic model, basis for a knowledge graph, shall enable to 1. describe a manufacturing environment in a standardised way based on DIN-SPEC-91345, extendable by other norms, analysed in section 2.5, 2. describe and classify heterogenous data along asset life-cycles, 3. describe the access to data independent from used data related technologies to enable a semantic model-based pre-configuration of available data source processors/connectors in the data pipelining approach. This semantic-model is a basis for a knowledge graph, which is ex-	RO1 RO3 RO5 RO6

	further norms and the linkage between data description and access.	ploited to support data understanding and data acquisition in smart manufacturing systems.	

3

CONCEPT AND APPROACH FOR DIN SPEC 91345 RAMI 4.0 COMPLIANT DATA PIPELINING

Digitalised assets are starting to provide their own meta-data in a standardised way by using an Asset Administration Shell (ZVEI, 2016) approach. These digitalised assets are called Industry 4.0 components in the context of Industry 4.0 (I40) and RAMI 4.0. It is assumed that standardised meta-data are getting standardised by using semantic models, so that the relationships between Industry 4.0 components could be represented in an overall knowledge graph which would describe an overall semantical description of a smart manufacturing environment.

This overall semantical description could be exploited to explore, search and find different kinds of needed data from I40 components along their whole life-cycles as e.g.

- Monitoring Data
- Meta Data
- Context Information
- Historical Data

If the semantical description would also describe how data, e.g. located in a relational database management system (RDBMS), could be accessed, this information could be used to preconfigure data source connectors in a data pipelining approach for a data extraction. The extracted data could then be further pre-processed in the pipeline or directly be forwarded for further subsequent exploitation like in data analysis processes.

3.1 Data pipelining approach

Figure 3-1 shows a conceptual high-level view on a self-descriptive semantical smart manufacturing infrastructure. The first part (see number (1)), visualises I40 components (see robot symbols). These components provide their descriptive standardized meta data and raw data, including the reference to the data coming from/located in a data source, along their life cycle through a self-descriptive semantic model. Following the RAMI 4.0 specification, an I40 component provides its meta-data via an Asset Administration Shell (AAS).

Figure 3-1: Self-descriptive semantical smart manufacturing infrastructure

The part of the figure labelled with the number (2) shows the linkage of compatible I40 component semantical models (see smaller semantic model symbols) with a smart manufacturing environment semantic model (see middle semantic model symbol).

Both linked semantic models build a semantical network of a smart manufacturing environment which is manged by a Semantic Manager module. This environment describes hierarchy levels and main Operational Technology (OT)/IT (Givehchi, Landsdorf, Simoens, & Colombo, 2017) components (for example central databases or message brokers).

In the infrastructure follows a Data Selector module labelled with the number (3). It is responsible for exploring, searching, filtering and selecting data based on the linked semantic models.

Based on selected data (binary object, buckets, streams, etc.), the Data Selector module generates a pre-configuration for data source connectors. These connectors are then deployed in the Data Provider module labelled with the number (4).

The Data Provider deploys the list of connectors/data source processors using parameters described in the pre-configuration file issued by the Data Selector in number (3). One processor could be for instance a MySQL database connector. The last gets pre-configured with access information (Internet Protocol (IP), Port), authorisation information (user, password), as well as a pre-configured query to extract the selected data from a specific table column.

As soon as the pre-configured data source connectors have been deployed, they can be executed (see number (5)). The execution of the connectors allows to extract the selected data from semantic model referenced data located in the data sources addressed in number (2) (for example monitoring data, meta data, context information or historical data along the life-cycle of I40 components).

The data provided in number (5) will have to be prepared for later usage by the data analytics module represented by number (7) as an example use case. This functionality is performed by software components of the infrastructure and identified with the number (6).

The data pipeline approach is basically composed by the sequence 2-5. Since this doctoral work does not deal with comprehensive data transformation tasks nor with data analysis and related use cases, step 6 and 7 in Figure 3-1 are partly shadowed.

3.2 Conceptual Architecture behind the data pipelining approach

To achieve a solution for the approach presented in a high-level view, the conceptual architecture shown in Figure 3-2 is proposed. It shows an architecture with the three core modules "Semantic Manager", "Data Selector" and "Data Provider". Each module is described below.

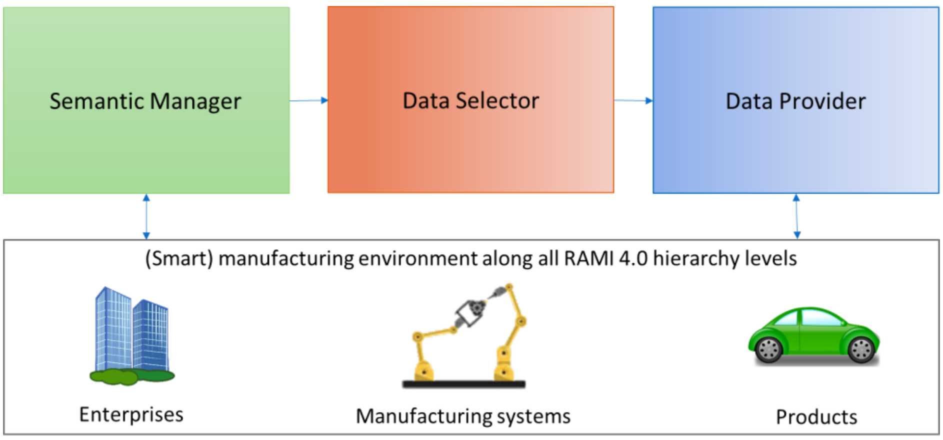


Figure 3-2: Conceptual Architecture

Semantic Manager

The Semantic Manager includes the semantics description of the smart manufacturing environment as a knowledge graph based on a unified semantical data model. I40 components that use this unified semantical data model as self-description can be registered and linked with the manufacturing environment according to the hierarchy levels of the RAMI 4.0. It is also possible to update the semantical description of the linked I40 components.

In the case that an asset in the (smart) manufacturing environment is a legacy system which does not provide its semantical self-description using the unified semantical data model schema, then a manual integration in the Semantic Manager is possible. The legacy systems will be described in the unified semantical data model as I40 component.

The unified semantical data model is based on the DIN-SPEC-91345 and integrates a range of standards related to the Industry 4.0 to support the standardisation activities in this area. The data model also allows to link data related to an I40 component along their life cycle. For example, in the case that specific data, written in a MySQL database, is linked to an I40 component, then the unified semantical data model allows to reference this data, describing the access to the data source (the MySQL database), and the reference to the data (e.g. a specific column in a table).

In general, the Semantic Manager is a central point that integrates the semantical descriptions of I40 components and their linkage to the manufacturing environment. The Semantic Manager is therefore a registry that can be asked to get information about the smart manufacturing environment.

The Semantic Manager provides the unified semantical data model to the Data Selector.

Data Selector

The Data Selector is a search engine that provides a user-friendly interface to explore, understand, search, filter and select data based on the unified semantical data model provided by the Semantic Manager suitable to support the process from data understanding until the data extraction.

The offered search functionalities exploit the unified semantical data model as much as possible including a text-based searching based on meta-data, a search based on filters or further optional extensions and integrations like Google Maps or virtual and augmented reality.

Needed data can be selected and added to a data cart where the Data Selector user is able to set further filter parameters on data for refinements (e.g. to filter only a specific time-frame in a time-series).

To extract the selected data, the Data Selector provides features to deploy pre-configured data source connectors in the Data Provider.

Data Provider

The Data Provider is a data pipeline solution which enables the extraction, transformation and loading of batch and streaming data. It provides a user-friendly interface to build a classical data pipeline using a range of native and custom flow processors to process data.

This module offers an Application Programming Interface (API) that is used to build a data pipeline. The API is used by the Data Selector to semi-automate the deployment of a data pipeline. In this doctoral work the focus for a semi-automation of the data pipeline deployment will be focused on the data extraction. The Data Selector uses the API to deploy pre-configured connectors in the Data Provider.

Extracted data is the basis for a further manual extension of the data pipeline with optional further data transformation/processing steps for a sub-sequent forwarding to use case dependent data exploitation tasks as data analysis processes.

These three modules shall enable to explore, understand, search, filter, find, select and extract data along the life-cycle of assets in a (smart) manufacturing environment to support data understanding and data acquisition in smart manufacturing.

Figure 3-3 shows the conceptual architecture in more detail to describe the internal workflow behind. According to the numbers shown in Figure 3-3, the process is following a sequence described below.

In a first step, available assets, located in the (smart) manufacturing environment, have to be registered in the Semantic Manger (no. 2). They provide their self-describing meta-data and data description along their life-cycle (no. 1). Also, additional context information can be optionally added/updated (e.g. the location of a mobile asset to enable a data source selection of Industrial Cyber-Physical Systems (ICPS) in a defined area (e.g. in sensor network use cases)).

An I40 component registration, linkage and semantic model update engine in the Semantic Manger gets the provided meta and context data and integrates/updates to the semantic model (no. 3). The semantic model is provided to the Data Selector in (no. 4).

The Data Selector queries the semantic model in the Semantic Manager and provides a user-friendly interface for data Searching, Filtering and Selection (SFS) (no. 5).

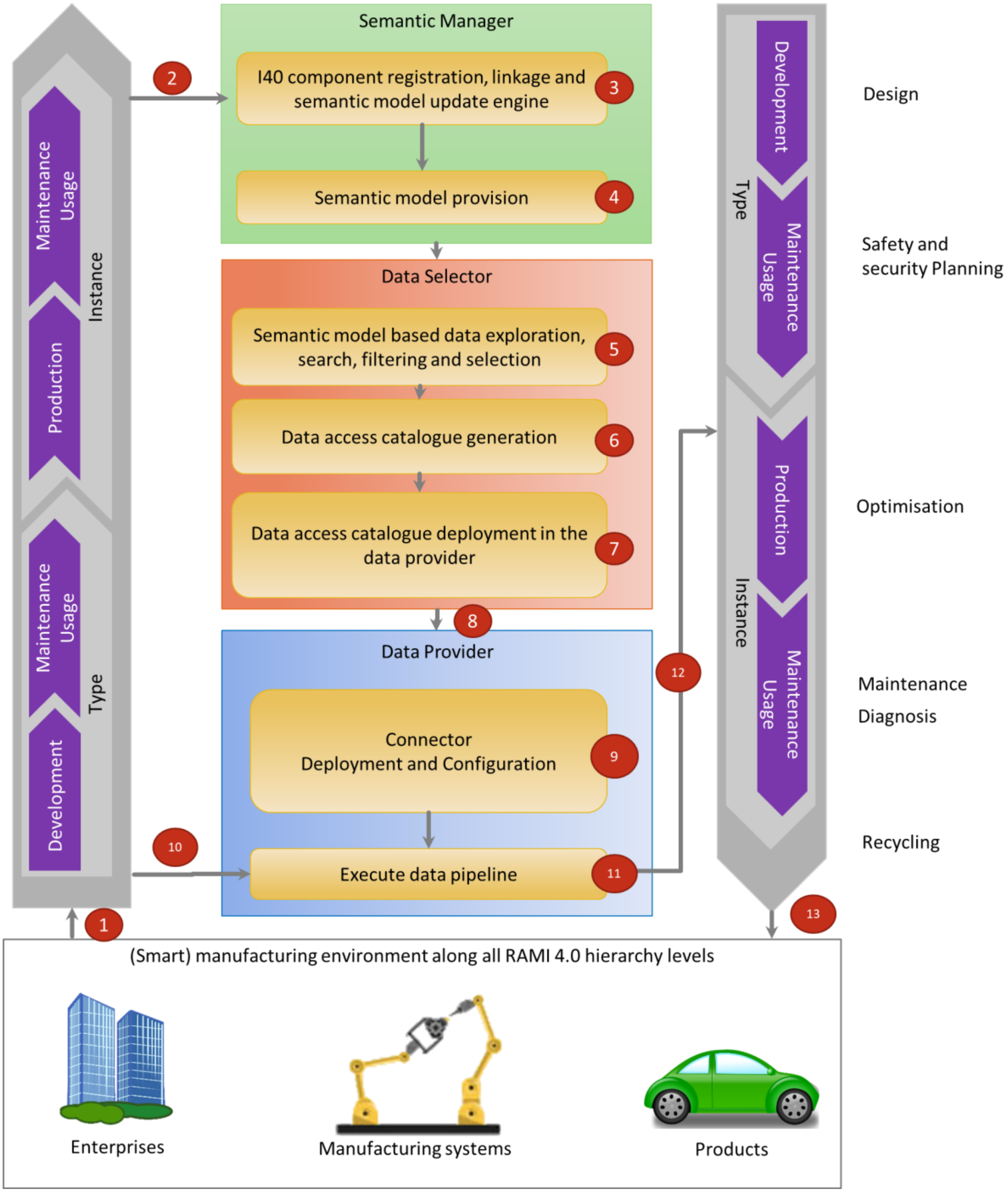


Figure 3-3: Workflow

After the selection of needed data, the "Data access catalogue generation" component generates a catalogue of selected data, enriched with access information to the data sources (no. 6) suitable for the configuration of connectors to be deployed and configured in the Data Provider (no. 7). The Data Selector uses therefore an API to the Data Provider (no. 8).

The API is used to deploy configured connectors based on the generated data access catalogue (no. 9). In a first step all connectors that are needed to extract the data are deployed and in a second step the deployed connectors get configured based on information in the data access catalogue. After this step the data extraction process in the data pipeline can be started (no. 10 and 11) which extracts the needed data from the (smart) manufacturing environment.

The pipeline can be extended by optional data transformation processes before the forwarding/loading of data to sub-sequent use case dependent exploitation tasks (no. 12).

Results of these sub-sequent use case dependent exploitation tasks can be optionally fed back to the smart manufacturing environment (no. 13). Use cases can be located in all life-cycle phases of an Industry 4.0 component. Some high-level use cases could be e.g. feedback to design, safety and security planning, optimisation, maintenance and diagnosis or to recycling.

This approach is proposed to support data understanding and data acquisition in smart manufacturing environments as follows:

Data understanding - The Data Selector shows available data along the life-cycle of available assets and provides SFS features compliant with the DIN-SPEC-91345 and further Industry 4.0 related standardisation activities which simplifies the data understanding, data finding and data selection.

Data acquisition – In a next step the Data Selector and the Data Provider support the data acquisition through a provision of features to:

- Pre-filter data (e.g. a time frame in a time-series)
- Deploy pre-configured data source connectors in a data pipelining approach
- Extract data
- Optionally transform data in the data pipeline
- Forward data to a sub-sequent Big Data analysis as possible use case example

3.3 A Semantic Model for data pipelining

The overall data pipelining approach uses a knowledge graph based on a semantic model as basis to support data understanding and data acquisition in smart manufacturing systems. The model needs to be able to describe whole (smart) manufacturing systems, available data along asset life-cycles and information to access this data. Thereby it should consider or even integrate state-of-the-art reference architectures and standards in smart manufacturing to structure and classify data.

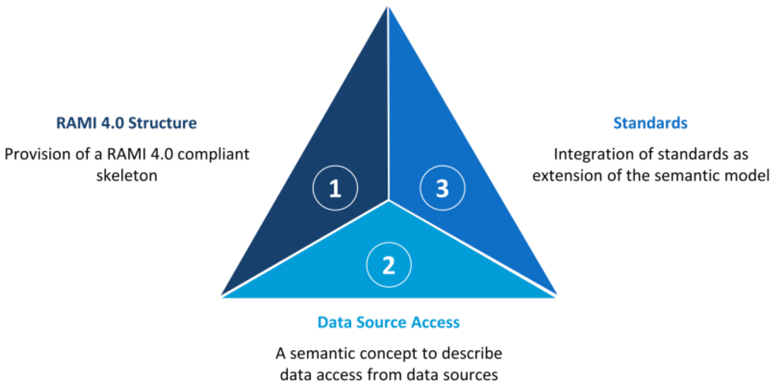


Figure 3-4: Parts of the Semantic Model

The semantic model can be partitioned into three different main parts as indicated in Figure 3-4.

1st - RAMI 4.0 Structure

Smart manufacturing environments are not a series of chaotically coupled systems, instead they follow architectures that have evolved over time. One of the latest reference architectures for smart manufacturing systems is the Reference Architectural Model Industrie 4.0 which is specified in the DIN Spec 91345. This reference architecture describes as a three-dimensional map the most important aspects of assets in smart manufacturing systems and ensures that all participants involved share a common perspective and develop a common understanding (Plattform Industrie 4.0, 2018).

The RAMI 4.0 described in DIN Spec 91345 is one of the most modern reference architectures with a specific focus on smart manufacturing and is therefore selected to be basis for the basic structure of the semantic model. The RAMI 4.0 has three axes that can be used to map a so-called Industry 4.0 component based on its life cycle and value stream, located hierarchy level and layers. Next to theses axes, the DIN Spec 91345 describes meta data that can be used to describe an Industry 4.0 component in a manifest and sub-models, and describes also basic data access views and user roles for accessing the component.

The semantic model uses the DIN Spec 91345 to build a top-level ontology which is further extendable by the data source access part which is used for the data pipelining approach and standards to further structure, classify and describe the smart manufacturing environment and available data in a unified way.

2nd - Data Source Access

The second part of the model is a schema to describe the access to heterogenous data. Some data are single files while others are bundled into data buckets, and others are only available as a data stream or in a request-response mechanism using different kinds of communication protocols. In some environments cyber-physical assets provide data by themselves in a decentral way and sometimes data is centrally collected in data storages/lakes. These heterogenous data characteristics in (smart) manufacturing environments have to be considered in the semantic model. It should also be taken into account that not all data is always needed, which makes filter mechanisms necessary.

Data pipelining approaches provide configurable processors to extract data. The configuration of the processors could be supported by suitable semantical knowledge graph. Therefore, the Data Source Access part of the sematic models proposes a Data Source Bundle concept which allows to describe: (1st) The data source where the data resides, including access information such as IP and port. (2nd) Data source processors and their static configuration options needed to configure the processors of a data pipelining approach. And (3rd), Property Builders to be able to generate processor properties dynamically, which can be useful, for example, if a filter needs to be implemented to select only certain data in a data bucket, such as a certain time frame in an Structured Query Language (SQL) query.

3rd - Standards

The third part extends the semantic model by standards that are used to classify and uniquely describe assets, services and data. Many standards have been proposed to further push the unification process in smart manufacturing. The semantic model shall show that such standards can be integrated in the semantic model and that these integrated standards can support data understanding and data acquisition in smart manufacturing environments.

4

SEMANTIC MODEL AND SOFTWARE ARCHITECTURE FOR DIN SPEC 91345 RAMI 4.0 COMPLIANT DATA PIPELINING

The SotA analysis in section 2 showed that data exploitation in smart manufacturing environments is a huge field and includes several research disciplines. The volume, variety and velocity of data in industry is exponentially rising. Next to legacy data sources as proprietary RDBMS databases or data streams from fieldbus-based systems (PROFIBUS (Felser, 2010), EtherCAT (Jansen & Buttner, 2004), MODBUS (Rinaldi & Lydon, 2015), etc.), new technologies are arising as for data storing (e.g. graph databases or NoSQL databases), for publish-subscribe pattern streaming (MQTT based technologies; IBM WebSphere MQ (Lampkin et al., 2012), RabbitMQ (Dossot, 2014), Apache ActiveMQ (Snyder, Bosanac, & Davies, 2017) and RocketMQ (RocketMQ, 2022)) and also service based approaches (e.g. OPC-UA (Hoppe, 2016) or REST (Richardson & Ruby, 2008)). Smart manufacturing requires the digital availability of data along the whole life cycle of products and assets to enable the exploitation of new business fields and to unlock the full potential hidden in data. In conclusion, data scientists are in the current situation confronted with very heterogenous data (data sources; data models), often without any semantical basis but organically grown data models. This state and the path dependency in this area has to be taken into account while elaborating new approaches for this domain.

This situation is also one of the main reasons why the data understanding and data preparation phases in a data analytics process as CRISP-DM can take more than 60 % of the overall needed time. Analogous to this, the demand for solutions in this area is increasing and calls for research and innovation activities to answer questions like:

- how to understand available data easily?
- how to find/identify needed data in an efficient way?
- how to access (and collect) the data?

, while keeping in mind data environments that are characterized through a high volume, variety and velocity.

It has been shown that one strong movement is currently the standardisation related to data. While associations like eCl@ss e.V. position themselves directly for industrial Industrie 4.0 transformation projects, many research activities revolve around the development of semantic models for a uniform description of smart manufacturing environments.

The provision of a semantic model based on DIN SPEC 91345, generic classifications based on eCl@ss using unified properties as defined in IEC 61360 and a freely configurable individual I40 component taxonomy, could generate a highly flexible data knowledge graph, potentially extendable with other semantic models and open for additional relations enabled through the nature of semantic models not to require a high integrity as in conventional RDBM systems.

This section presents a solution beyond the state of the art for semantic model-based DIN-SPEC-91345 compliant data pipelining to support data understanding and data acquisition in smart manufacturing environments.

4.1 Derived and used core-concepts from SotA standards and technologies

As already mentioned, the current work uses the latest outcomes of the state-of-the-art as fundament, to propose a data pipeline solution, compliant with the DIN-SPEC-91345

The following results have inspired and been used, integrated or combined in this doctoral work:

- The Industrie 4.0 component of the RAMI 4.0: An Industrie 4.0 component provides information about an asset and provides access to its functionalities. It includes an AAS (ZVEI, 2016) which is divided into a header with a manifest and a body which includes several sub-models according to IEC/TS 62832 (International Electrotechnical Commission (IEC), 2016b), to access and describe the component itself.

- RAMI 4.0 dimensions: Hierarchical Levels based on IEC 61512-1(DIN, 1999) and IEC 62264-1 (The International Society of Automation (ISA), 2010); Life-cycle and value stream dimension according to the IEC 62890 (International Electrotechnical Commission (IEC), 2016a); and digitalisation dimension containing six Layers.
- Data Access Views from the RAMI 4.0: Concept of data access views where specific data properties/sub-models are linked to specific user (groups).
- The eCl@ss classification of products and services: the separation between classification classes and product description classes. This concept is mapped to the sub-model concept described in DIN-SPEC-91345.
- Identifying globally and unique I40 components and sub-models through an IRDI identifier.
- Property models from eCl@ss: Derived will be the property models released in eCl@ss 10.0.1, that are derived from several other standards (eCl@ss e.V., 2018).
- FlowFile Processors from Apache NiFi: parametrisation through different attributes to act as pre-configured data source processor/connector (e.g. a MySQL processor with pre-configured access details and a query to read selected data).

4.2 Development Requirements

This section defines a range of development requirements. The development requirements consider the research objectives and research requirements of this doctoral work defined in sections 1.4.1 and 1.4.2, the SotA analysis in section 2, and the concept and approach defined in section 3.

The development requirements are classified as "shall", "should" and "may" (as described in section 1.4.2) and in functional (F) and non-functional (NF) requirements (Clarkson & Eckert, 2010).

4.2.1 DIN-SPEC-91345 compliant Data Model Requirements

As already elaborated, the semantic data model (specified in section 4.3.1) has to fulfil a range of non-functional requirements related to standards or reference models. These requirements are described in Table 4-1.

Table 4-1: List of non-functional data model requirements

ID	DIN-SPEC-91345 compliant Data Model requirement description	F/NF
Data Model Requirements		
DM_ID1_1	The data model shall be semantic.	NF
DM_ID1_2	The semantic data model shall be compliant to DIN-SPEC-91345.	NF
DM_ID1_3	The semantic data model shall allow the integration of eCl@ss classification classes.	NF
DM_ID1_4	The semantic data model shall use IEC 61360-based properties as basis for the used vocabulary.	NF
DM_ID1_5	The semantic data model shall allow the classification and description of data along the life-cycle of I40 components.	NF
DM_ID1_6	The semantic data model shall allow the embedding of I40 components into a smart manufacturing environment based on hierarchy levels according to RAMI 4.0.	NF
DM_ID1_7	The semantic data model shall allow the description of all parameters needed in a data pipelining approach to access and extract the data.	NF
DM_ID1_8	The semantic data model shall be open for the integration of other semantic model-based standards.	NF

4.2.2 Software Architectural Requirements

The software architecture (specified in section 4.3.2) needs to be compliant with the RAMI 4.0, and has to fulfil a range of data related requirements. These requirements are described in Table 4-2.

Table 4-2: List of software architecture requirements

ID	Software Architectural requirement description	F/NF
Data Requirements		
AR_ID1_1	The architecture shall allow the linkage of data produced along the whole life cycles of products and assets to generate a digital thread.	F

AR_ID1_2	The architecture shall allow to handle Big Data according to the three-V model.	F
Requirements coming from influencing (reference) architectures		
AR_ID2_1	The architecture shall be compliant to the RAMI 4.0 described in DIN-SPEC-91345.	NF
AR_ID2_2	The architecture shall allow the integration standards like eCl@ss for data classification.	F

4.2.3 Prototype Requirements

The implemented prototype shall demonstrate the semantic model-based, DIN-SPEC-91345 compliant, data pipelining solution to support data understanding and data acquisition in smart manufacturing environments. This prototype implementation shall fulfil a range of requirements described in Table 4-3.

Table 4-3: List of prototype requirements

ID	Prototype Requirement description	F/NF
Related to data		
FR_ID1_1	The prototype shall provide functionalities to classify heterogenous data in a semantical way, considering latest standards coming from the Industrie 4.0.	F
Related to data provision		
FR_ID2_1	The prototype shall provide functionalities to find and identify data in heterogenous data environments.	F
FR_ID2_2	The prototype shall provide functionalities to select found/identified data as preparation for further sub-sequent processing.	F
FR_ID2_3	The prototype shall provide functionalities to filter selected data as preparation for further sub-sequent processing.	F
FR_ID2_4	The prototype shall provide functionalities to access and extract selected data as preparation for further sub-sequent processing.	F

4.3 Specification

This section specifies the semantic data model and the software architecture, and provides also an alignment to the reference architectures from the BDV and the RAMI 4.0.

4.3.1 DIN-SPEC-91345 compliant Semantic Data Model

The self-descriptive semantical smart manufacturing infrastructure as presented in section 3.1, uses a semantic data model (conceptually introduced in section 3.3) derived from standards that were introduced in section 4.1. This model is described in this section.

4.3.1.1 *Structure of the DIN-SPEC-91345 compliant semantic model*

The DIN-SPEC-91345 compliant semantic model acts as schema for knowledge graph that will enable the description of whole smart manufacturing data environments.

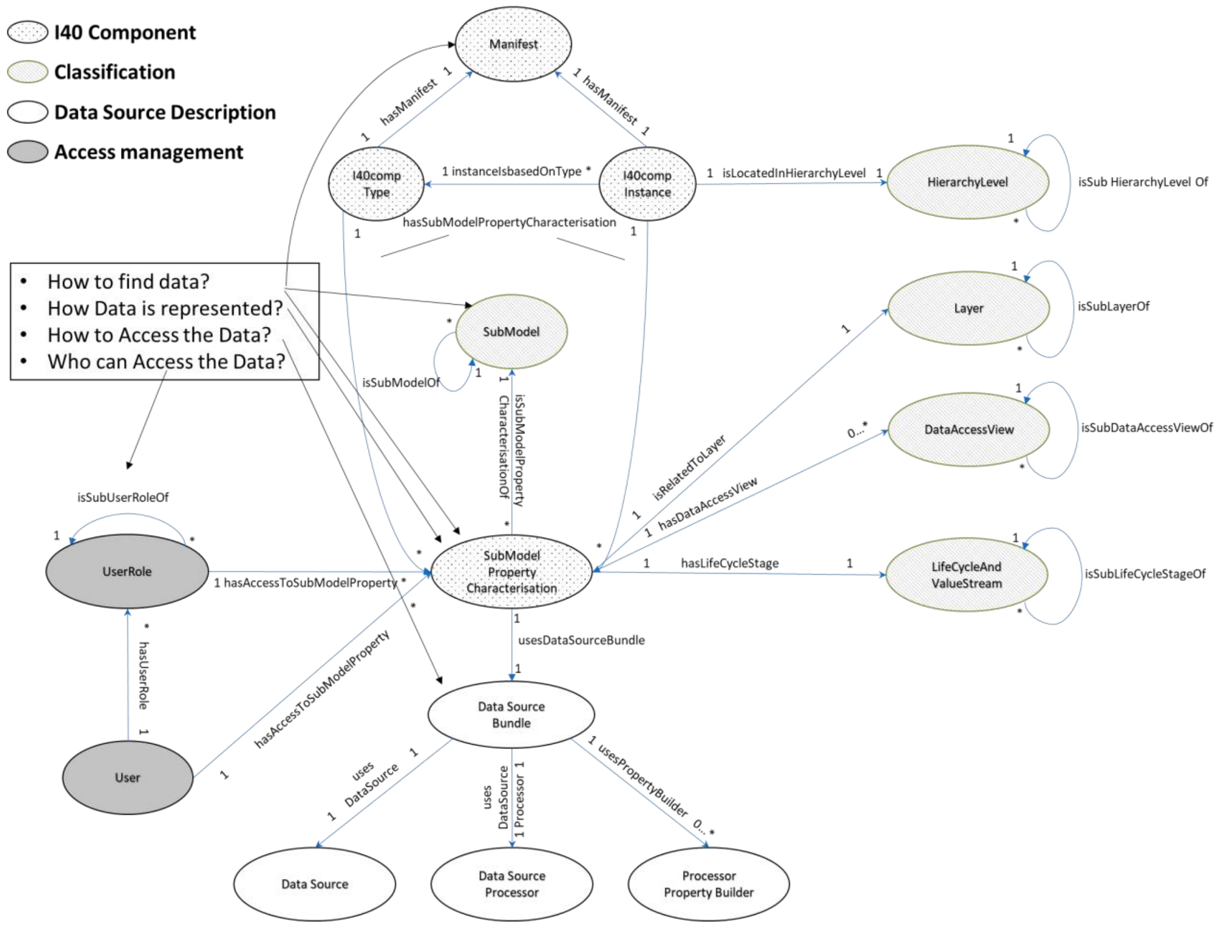


Figure 4-1: Structure of the DIN-SPEC-91345 compliant semantic model

Figure 4-1 shows the main classes of the elaborated semantic model for an I40 component embedded in a RAMI 4.0 oriented smart manufacturing environment.

I40 component related classes follow the I40 component (I40comp) concept described in DIN Spec 91345. An I40 component, according to DIN Spec 91345, is divided into an "I40comp Type", which represents the design/specification phase of an asset and an "I40comp Instance", which represents an instantiation of a specific I40comp type. Types and Instances have a "Manifest", which includes meta data of an "I40comp type" or an "I40comp instance".

An "I40comp instance" can be associated to a specific location in a smart manufacturing environment through the "HierarchyLevel" class. The "HierarchyLevel" class represents the structure of a smart manufacturing environment (for instance a hierarchical structure of an enterprise in a tree or graph representation). Instantiations of hierarchy levels can be used to nest/link I40 component instances and their relations into bundles. Any data linked to an I40comp type or I40comp instance is described in a "SubModel Property Characterisation" (SMPC). An SMPC is according to DIN Spec 91345 linked to

- a "Layer" (SMPCs are linked to the "information" layer), to describe to which RAMI 4.0 layer the data belongs,
- a "DataAccessView", to group/categorise different SMPCs (for instance Energy Efficiency Data),
- a "LifeCycleAndValueStream", to show the life-cycle stage to which the linked data belongs (for instance a Computer-Aided Design (CAD) file in the design life-cycle phase of an I40 component) (Note: According to the IEC 65/617/CDV:2016, DIN EN 62890:2016 the linked data can belong to one or more life-cycles to manage systems and products used in industrial-process measurements, control and automation as e.g. product life-cycle, order life-cycle, factory life-cycle and technology life-cycle.),
- "UserRole" and "User" to configure users and user groups authentication.
- An SMPC describes accessible data of an I40 component (for instance an attribute, a CAD file, a data stream, a specific information in a database, etc.). To enable an easy finding of this data, a standardised classification is needed. Such data classifiers are called "Sub Models". Sub Models are - similar to the eCl@ss approach - standardised data classifiers and identifiable by an IRDI (Döbrich et al., 2017; eCl@ss e. V., 2019b), and can be based on eCl@ss (but also other classification standards are integrable).

In case the data cannot be stored as a simple attribute/data property in a SMPC (as basically done in eCl@ss), because the data is a binary file/object, a document, a data stream, etc., then a "Data Source Bundle" has to be defined. The Data Source Bundle includes detailed information that point/refer to the data in a data source (for instance: a specific MQTT stream that can be subscribed to by an MQTT broker or a query that gathers specific data from a specific database). The "Data Source Bundle" is divided into three classes:

- a "Data Source" class which includes access information to a data source (for instance, a NoSQL database (user, password, IP, etc.));
- a "Data Source Processor" class which represents a parametrizable data source connector. For example, in case Apache NiFi is the chosen data pipelining technology used as basis to extract data, then the list of all Apache NiFi Flow processors that enables access to a data source would be imported to the semantic model, including all required parameters of each processor. Apache NiFi was chosen because it is a scalable and customisable solution which allows data routing, transformation and system mediation also for big amounts of data (streams). The Apache NiFi Flow processors in this doctoral work are called "Data Source Processors". An individual of a data source processor, linked to a data source bundle, initialises the needed parameters for the chosen data flow processor.
- a "Property Builder" class which defines a list of properties needed to generate dynamically a parameter for the data source processor, if needed. Such a property builder could be for instance a MySQL SQL query builder that requires information typically for an SQL query as a table name and a column. The Property Builder individual would contain the values for the table and the column of the data addressed by the data source bundle. An algorithm (in this example an SQL query builder; to be developed) uses these values to build the query which is then used as parameter to initialise the data source processor. A property builder can also be used for dynamic data filtering or aggregations. This enables e.g. the collection of data with spatial and temporal characteristics (different in space and time, like GPS or weather data). An example could be traffic data (live and historical) for a city. Available meta-data, like timestamps and the GPS, can be used to filter the traffic data for a specific time frame in a specific area.

The data source bundle including the three classes introduced above, used to parametrise a data source processor, is described in detail below in section 4.3.1.2.

4.3.1.2 The Data Source Bundle

In case that data is stored in a data source and cannot be represented as property, then a data source bundle has to be instantiated. Some use cases are shown in Table 4-4.

Table 4-4: Use case examples of a data source bundle

Kind of data	Information in Property Characterization
NoSQL Document	Access details to data store and query
Binary Object	Access details to data store and URI
Time-Series	Access details to data store and query
MQTT Stream	Access to MQTT Broker and MQTT Topic

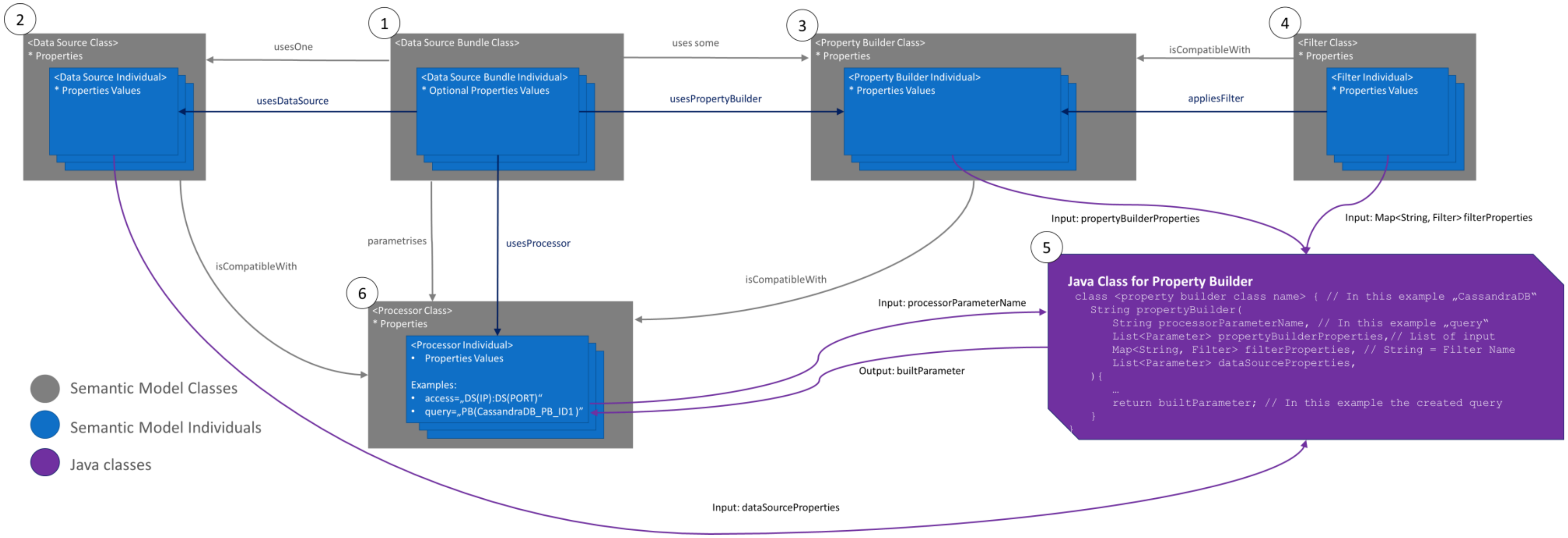


Figure 4-2: Data Source Bundle

Figure 4-2 shows how the data source bundle is built in detail.

Number (1) in Figure 4-2 shows the Data Source Bundle. An individual of a data source bundle is linked to an SMPC (not visible in the figure; therefore, compare Figure 4-1). It is possible to add data properties to describe the data source bundle if needed.

One data source bundle individual uses one data source processor individual (see number (6) in Figure 4-2).

Following the infrastructure described in Figure 3-1 , an essential step is the selection of a technology for implementing the pipelining approach. As a consequence, the data source processor defines a processors/connector to be used based on the selected data pipelining technology (see some examples in section 2.4). For example, if Apache NiFi is used as data pipelining technology, a processor class could be "org.apache.nifi.processors.mqtt.ConsumeMQTT" with all data properties needed for the configuration.

Some data properties do not get initialised in the data source processor individual (number (6)) directly because the values are already initialised somewhere else. This happens if data source access details are needed (see number (2) in Figure 4-2). A data source class defines access parameter to a data source. A data source could be a message broker, a database or another central node for data access. A data source access individual for a specific database will be just instantiated once for a smart manufacturing environment. Data source bundles, that references data in this data source, will be linked by an "usesDataSource" relation (/an usesDataSource object property). The mapping in the data source processor individual follows the syntax:

<data property>=DS(<data property>)

An example (compare Figure 4-2 **number (6)):**

access=„DS(IP):DS(PORT)"

This example shows the simplest/static property mapping.

Properties defined in a data source (DS) are referenced using the function `DS()`. A string builder which uses the semantic model as basis, has to be implemented to build the string by using the linked properties. The generated result in this example would be for instance "`127.0.0.1:8080`", which would be mapped to the parameter/data property "`access`" of the data access processor individual.

Some other data properties do not get initialised in the data source processor individual because the value, which has to be generated, is not static and needs some logic for the generation. This happens for instance if an SQL database query has to be generated based on a table name, optional filters or aggregation rules. For those properties are used property builders (PB) (see number (3) in Figure 4-2). Property builder individuals contain input parameters for different builder classes that have to be implemented (see number (5) in Figure 4-2). The Property Builder concept is extended by optional filters (see number (4) in Figure 4-2). In case that a property builder individual applies a compatible filter, or filters (plural), then these filter properties are also considered by the linked property builder algorithm shown in number (5) of Figure 4-2. Other features, like aggregation rules, can be added to extend this concept.

The use of a property builder in a data source processor individual follows following syntax:

<data property>=PB(<property builder individual name>)

An Example (compare Figure 4-2 number (6)):

query=„PB(CassandraDB_PB_ID1)"

This example shows the dynamic property building. The function name `PB()` stands for Property Builder. `CassandraDB_PB_ID1` in the example is the individual name of a specific PB class. The class brings a list of required properties to be initialised in the individual.

For example: following Cassandra Query Language (CQL) (The Apache Software Foundation, 2016) query would address the needed data.

SELECT columnName FROM keyspaceName.tableName

This would mean that the related CassandraDB PB class would require the properties columnName, keyspaceName and tableName to build a query. Those data properties are initialised in the related individual `CassandraDB_PB_ID1`.

The implemented PB (see number (5)) uses these parameters to build the CQL query.

In addition, Figure 4-2 shows also how a PB can be extended by filters. As example will be extended the CQL query by a WHERE clause:

SELECT columnName FROM keyspaceName.tableName ***WHERE date > '2020-01-01'***

A filter class with the name `beforeDate` is compatible with the PB class `CassandraDB` and would require the property `date`. In case that the PB `CassandraDB` should apply the filter `beforeDate`, then an individual of this filter has to be instantiated which includes the initialised property `date`. The PB would then build the CQL query including the WHERE clause.

How a PB algorithm can be implemented, is introduced in number (5) of Figure 4-2. PB algorithms are part of the Data Selector module introduced in number (3) of Figure 3-1 in section 3.1, and are described in detail in the specification of the software architecture in section 4.3.2. The Data Selector module uses a SPARQL engine to navigate through the semantic model. In case a data source processor has to be configured, the Data Selector module searches in the data source processor individual properties for syntaxes that requires a static mapping (for instance `DS(IP)`), or a dynamic data property creation (for instance `PB(CassandraDB_PB_ID1)`).

Steps in the case of a static Data Source mapping:

- Step 1: Data Selector identifies the function name (for instance `DS()`)
- Step 2: Data Selector collects needed data properties from the linked data source individual (for instance `IP` and `Port`).
- Step 3: Data Selector replaces the static mapping syntax with the property values and uses the result for the data source processor pre-configuration.

Steps in case of a dynamic property building:

- Step 1: Data Selector identifies the function name `PB()`
- Step 2: Data Selector reads the addressed individual (for instance `CassandraDB_PB_ID1`) and identifies the related class name based on its type (for instance `CassandraDB`).
- Step 3: Data Selector searches for the related Java class which has the same name as the PB class in the semantic model (compare number (5)). If successful:
- Step 4: Data Selector collects data property values from the linked individuals as shown as inputs in Figure 4-2.
- Step 5: Data Selector builds based on all inputs the property value (for instance a `query`).

This concept can be next to filters also extended by aggregations.

4.3.1.3 Vocabular Properties

Instantiated classes/individual are described by a range of data properties. In this doctoral work a property definition partly based on the IEC 61360 (Reihe, 2009) to follow the standardisation process in Industrie 4.0 will be used.

According to DIN-SPEC-91345, mandatory (standardised), optional (standardised) and free (not standardized) properties can be defined to describe an I40 component. This concept will be derived for specific classes of the semantic model. Figure 4-3 shows a few derived mandatory vocabularies used for the instantiation of the described semantic model classes. Some classes as “User” get other properties.

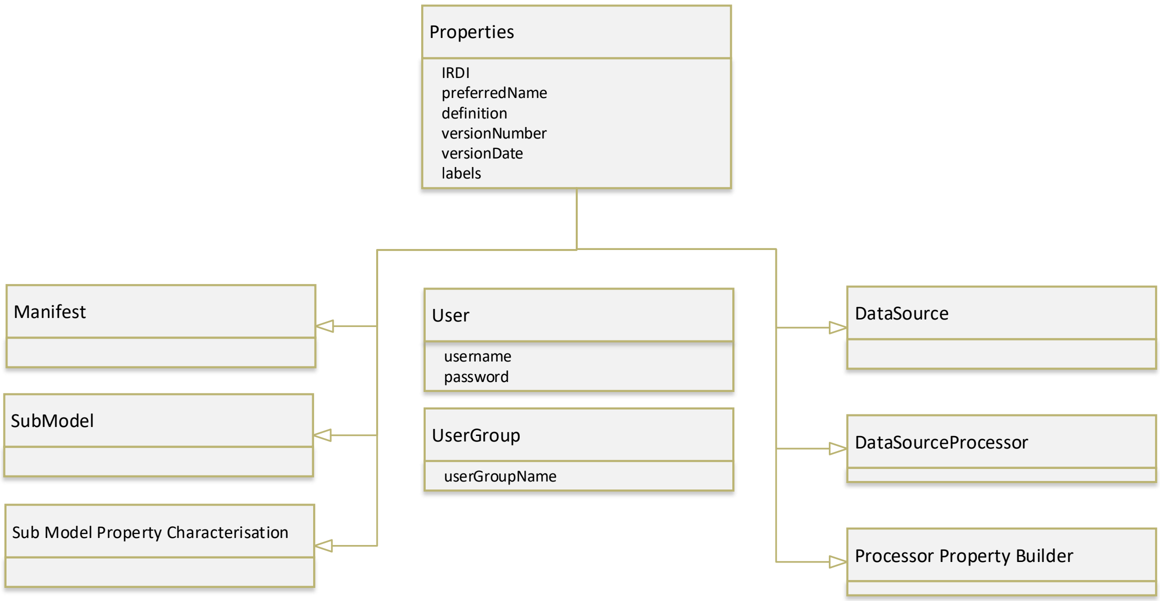


Figure 4-3: Mandatory properties partly based on IEC 61360

The properties shown in Figure 4-3 are described in Table 4-5.

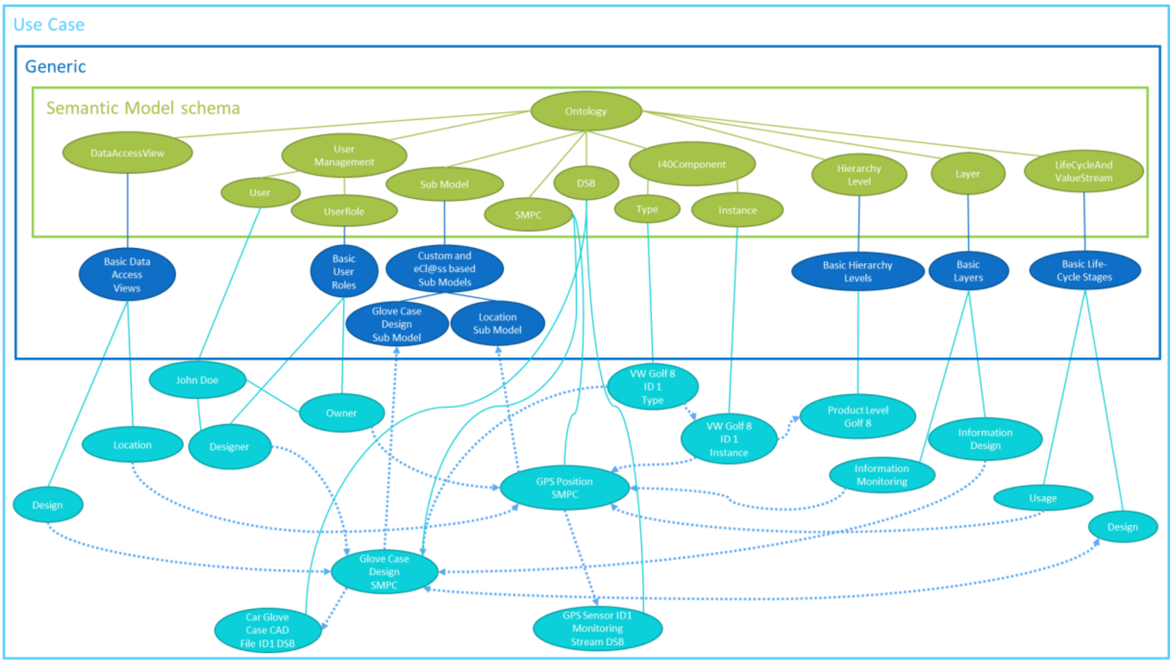
Table 4-5: Description of the properties

Property	Description
IRDI	International Registration Data Identifier (IRDI).
preferredName	Name of a property
definition	Definition of a property
versionNumber	Number of the version
versionDate	Date of the version publication
labels	Labels are keywords related to a property. Format is “[label1, label2, ...]”
username	The username of a user
password	The (hashed) password of a user.
userGroupName	The name of a user group.

For more information concerning ongoing property standardisation activities in the area of smart manufacturing, please see for instance (Hadlich, 2015).

4.3.1.4 *Instantiation example*

As an instatiation example of an I40 component, a car of the model Volkswagen (VW) Golf 8 was chosen.



SMPC = SubModel Property Characterisation; DSB = Data Source Bundle

Figure 4-4: Simplified Instantiation of a VW Golf 8 car

Figure 4-4 shows the simplified instantiation of a VW Golf 8 car specified as an I40 component with the instantiated semantic data model. The *semantic model schema* area describes the basic taxonomy of the semantic model (compare Figure 4-1 to get more details). The *generic* area consists a few generic basic classes and sub-models (e.g. sub models based on eCl@ss (see section 4.3.1.1)). The *individual instantiation* block presents the product, i.e. the car, as an I40 component. The reader can derive the meaning of the relations between individuals and classes by comparing with Figure 4-1.

In order to show a few main parts of the semantic model, Figure 4-5 presents the “Glove Case Design Sub Model”, “Glove Case Design SMPC” and “Car Glove Case CAD File ID1 DSB” (Data Source Bundle)” of the Glove Case Design example of Figure 4-1 with more details. More specifically, Figure 4-5 shows in detail an instantiation of Sub Models - exemplary based on eCl@ss classification classes -, a Sub model property characterization and a Data Source Bundle without the need of a property builder.

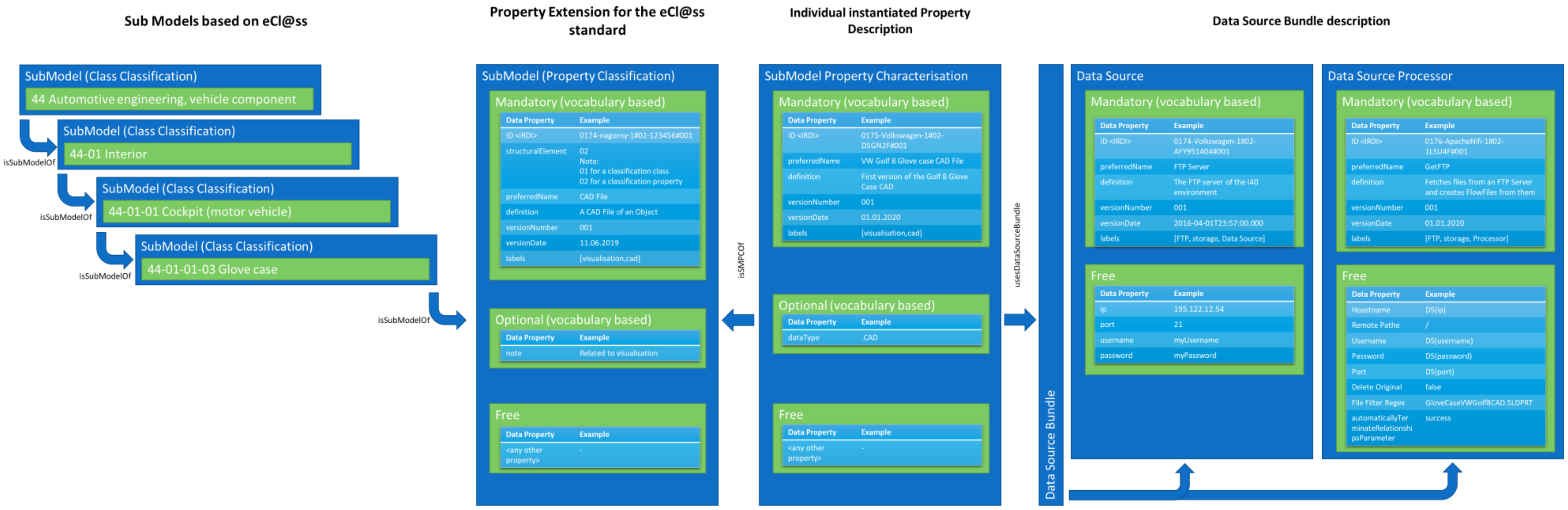


Figure 4-5: Example for a Sub Model, SMPC and Data Source Bundle

The sub-model taxonomy classifies properties of a glove case in a vehicle. Next to the data properties standardised by eCl@ss, this example adds one property classification in form of a sub model. The added property classification gets the IRDI "`0174-nagorny-1#02-123456#001`" and classifies a CAD file for an object using the data source bundle approach. In this example, the Volkswagen AG adds to the I40 component "VW Golf 8" a glove case CAD file with the IRDI "`0175-Volkswagen-1#02-D5GN2F#001`". The CAD file is located in the VW File Transfer Protocol (FTP) server storage which has the IRDI "`0175-Volkswagen-1#02-AFY951404#001`". Therefore, the data source bundle is linked to this data source access individual.

Used as data pipelining technology is Apache NiFi. Apache NiFi provides the data source processor "GetFTP" to extract data from an FTP server. Therefore, a data source processor individual is created with details to reach the CAD file. The data source processor has the IRDI "`0176-ApacheNifi-1#02-1LSU4F#001`" and saves the reference to the CAD file (path and file name). Based on the given information, it is possible to generate a configuration for the "GetFTP" Apache NiFi Flow processor, as described in section 4.3.1.2. A Data Selector Module which uses this information is able to deploy the pre-configured Apache NiFi Flow processor in an Apache NiFi instance using the Apache NiFi API. After this step, the NiFi Flow processor is directly ready to extract the needed data.

The different properties used in Figure 4-5 are described in the following tables.

The Sub Model (Property Classification) properties are described in Table 4-6.

Table 4-6: SubModel (Property Classification) Properties Description

Data Property	Example	Description
Mandatory		
ID <IRDI>	0174-nagorny-1#02-123456#001	A sub model has an IRDI as defined in Table 2-2. The IRDI shows that the property does not belong to eCl@ss but to a fictitious "nagorny" company and shows, how other standards can be integrated in this approach. A CAD File in a fictitious "nagorny" standard has the ID "123456"
structuralElement	2	This property defines that it is property classification (02). 01 would have been a classification of a class which would have been valid for super/parent SubModels.
preferredName	CAD File	Is the preferred name for the sub model property classification of a CAD File.

definition	A CAD File of an Object	A text which defines a CAD File of an object.
versionNumber	1	Is the version number of this SubModel. In case a CAD File will be specified in a different way (e.g. with a changed definition), then another version (2) would exist next to version 1.
versionDate	11.06.2019	Creation date of this version.
labels	[visualisa-tion,cad]	The data property "labels" is an array of strings in the format "[label 1, label 2, label n]" to provide further options for advanced data search features.
Optional		
note	Related to visu-alisation	To provide further information about the SubModel.

The SMPC properties are described in Table 4-7.

Table 4-7: SubModel Property Characterisation Properties Description

Data Property	Example	Description
Mandatory		
ID <IRDI>	0175-Volkswagen-1#02-D5GN2F#001	The IRDI shows the example company who aims to specify their glove case of a Golf 8. The glove case CAD File has the identifier D5GN2F.
preferred-Name	VW Golf 8 Glove case CAD File	The name of this SMPC.
definition	First version of Golf 8 Glove Case CAD	Provides a definition about this data.
version-Number	001	It is the first version of the CAD file.
versionDate	2020-02-01T23:57:00.000	It was created on 01.02.2020
valueType	2	Defines a value type needed to indicate if the data it-self is written in a value as part of the semantic model, or if a reference to a data source is given. 1 stands for "Value Property" and 2 stands for "Data Source". In the example a data source is linked.
labels	[visualisation,cad]	Some labels/keywords, used to enable an easy finding of this SMPC.
Optional		
dataType	.CAD	The object behind the data source is a CAD file.

The Data Source properties are described in Table 4-8.

Table 4-8: Data Source Properties Description

Data Property	Example	Description
Mandatory		
ID <IRDI>	0174-Volkswagen-1#02-AFY951404#001	The data source of the example company Volkswagen has the identifier AFY951404.
preferredName	FTP Server	The preferred name of the data source
definition	The FTP server of the I40 environment	Defines the purpose of this class instantiation.
versionNumber	001	It is the first version this instantiation.
versionDate	2016-04-01T23:57:00.000	It was created on 01.04.2016 at 23:57:00
labels	[FTP, storage, Data Source]	Some labels/keywords, used to enable an easy finding of this Data Source Access class instance.
Free		
ip	xxx.xxx.12.54	Describes the IP of the FTP server (anonymised)
port	21	Describes the port of the FTP server
username	myUsername	Login Username (anonymised)
password	thisIsNotThePwd	Login Password (anonymised)

The Data Source Processor properties are described in Table 4-9.

Table 4-9: Data Source Processor Properties Description

Data Property	Example	Description
Mandatory		
ID <IRDI>	0176-ApacheNifi-1#02-1LSU4F#001	The Apache Software Foundation got an own (invented) identifier. The GetFTP processor got the identifier 1LSU4F.
preferredName	GetFTP	The name of the processor.
definition	Fetches files from an FTP Server and creates FlowFiles from them	The definition of the processor.
versionNumber	001	The version of the processor.
versionDate	01.01.2020	It was created on 01.01.2020
labels	[FTP, storage, Processor]	Some labels/keywords, used to enable an easy finding of this processor.
Free (defined by the processor)		
Hostname	DS(ip)	The fully qualified hostname or IP address of the remote system Linked with the Data Source IP.

Remote Path	/	The path on the remote system from which to pull or push files.
Username	DS(username)	Login Username. Linked with the Data Source username.
Password	DS(password)	Login Password. Linked with the Data Source password.
Port	DS(port)	The port that the remote system is listening on for file transfers. Linked with the Data Source port.
Delete Original	false	Determines whether or not the file is deleted from the remote system after it has been successfully transferred.
File Filter Regex	GloveCaseVWGolf8CAD.SLDPRT	Provides a Java Regular Expression for filtering Filenames; if a filter is supplied, only files whose names match that regular expression will be fetched.
automaticallyTerminateRelationshipsParameter	success	All FlowFiles that are received are routed to success.[3]

4.3.2 Software Architecture

This section describes the specification of the software architecture based on the concept described in section 3. The software architecture describes the modules that are used to build semantic model based, DIN-SPEC-91345 compliant, data pipelining to support data understanding and data acquisition in smart manufacturing environments.

[3] See https://nifi.apache.org/docs/nifi-docs/html/user-guide.html

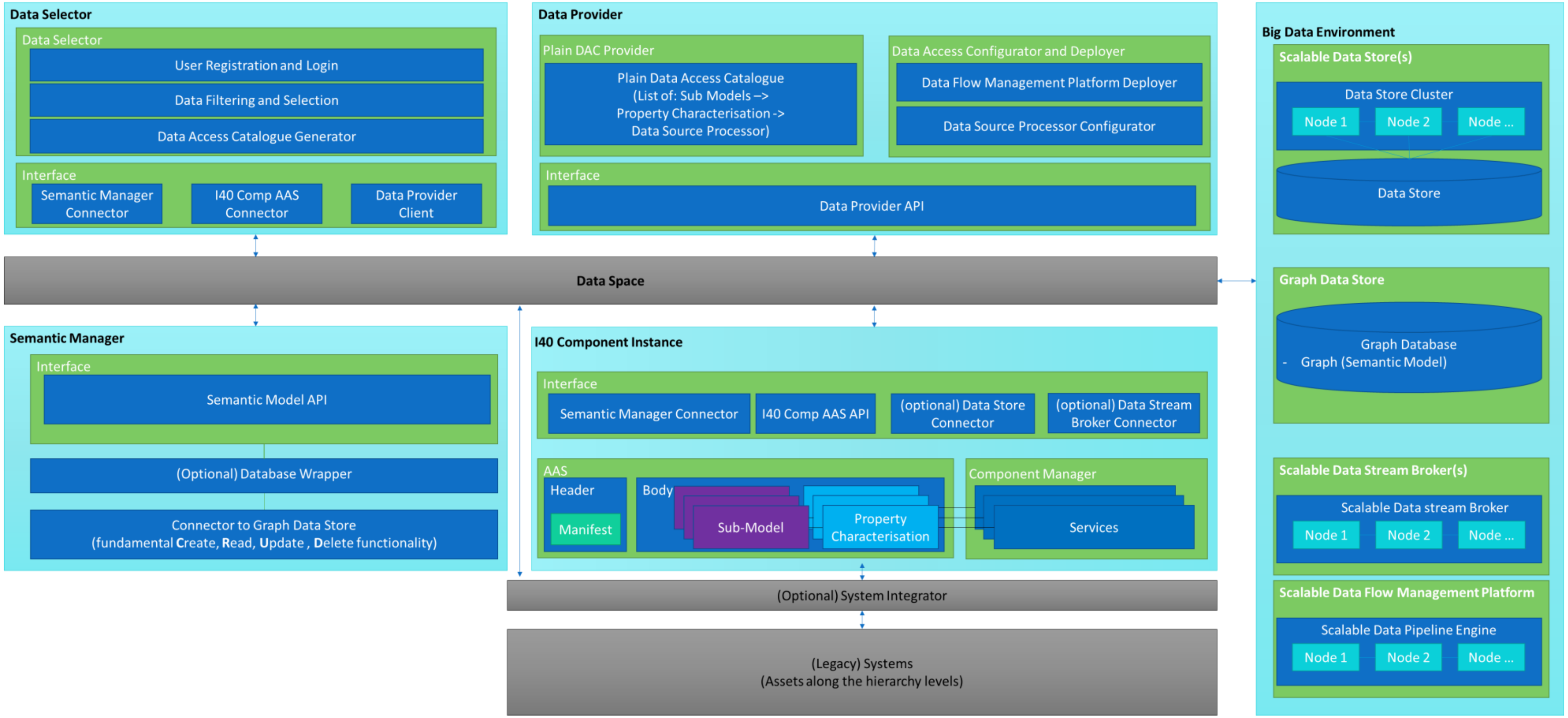


Figure 4-6: A DIN SPEC 91345 compliant software architecture

Figure 4-6 shows the DIN SPEC 91345 compliant software architecture for semantic model-based, DIN-SPEC-91345 compliant, data pipelining. Each module will be described in the following.

4.3.2.1 Data Space

The data space represents the smart manufacturing IT network. Different connected networks (local wired networks, locale wireless networks but also connected world/global networks) can be part of this network.

4.3.2.2 Semantic Manager

The "Semantic Manager" is the access point to manage the semantic model. This module enables to manage (add/remove/update operations) I40 components and the smart manufacturing environment description. The module is connected to a "Graph Data Store" inside the "Big Data Environment", which acts as repository for the semantic model and allows basic create, read, update, and delete (CRUD) operations. The "Semantic Manager" includes the following sub-modules:

- A "Connector to Graph Data Store" to enable the mentioned fundamental CRUD operations on the semantic model.
- A "Database Wrapper" to enable the use of different graph databases as some do not provide native support for datatypes as for instance OWL files. Therefore, this optional sub-module is sometimes needed to transform the semantic model into a supported format (for instance RDF).
- A "semantic model API" provides the interface to create, update and delete I40 components according to the semantic model-based data model as shown in Figure 4-1, and to embed them into a smart manufacturing environment. This API is also used to search, filter and select data within the "Data Selector" module by using a query engine.

4.3.2.3 Big Data Environment

This module consists of IT components to save, process and query data. The term "scalable" indicates that the chosen technologies shall be Big Data compliant (e.g. by clustering and load balancing features). The "Big Data Environment" module includes following sub-modules:

- “Scalable Data Stores“ to save Big Data linked to I40 components along their life-cycle. Examples: NoSQL DBs like Cassandra to store documents (for instance log files). Time series databases (TSDB)s like InfluxDB to store time series. Object DBs like Amazon S3 to store binary data (for instance media-files as pdfs or photos).
- “Graph Data Store” to store the semantic model and to provide fundamental CRUD operations.
- “Scalable Data Stream Broker” represents any message broker needed for instance for publish-subscribe protocols like MQTT (for instance HiveMQ; see www.hivemq.com).
- “Scalable Data Flow Management Platform” represents the chosen data pipeline technology, used as basis to enable a deployment of pre-configured data source processors that extract referenced data.

The Big Data environment is not limited to these modules and can be extended by further data source technologies based on individual application cases.

4.3.2.4 Data Selector

The “Data Selector” enables to explore, search, filter and select the data to support the data understanding and data preparation phase in a (big) data analytics process. The Data Selector is accessible through a user-friendly interface and provides features to navigate through data along the life-cycle of an I40 component embedded in its smart manufacturing environment. This is done by exploiting all information that is stored in the semantic model, using the semantic model API of the Semantic Manager. (References to) data can be selected and bundled into a data access catalogue (DAC) which is a "shopping cart" for selected data. The DAC represents the configuration of the “Data Provider” module. One DAC entry includes a sub model, a SMPC and a data source processor configuration generated by the Data Selector, using the data source bundle as explained in section 4.3.1.2.

The “Data Selector” includes the following sub-modules:

- The “Interface” contains a “*Semantic Manager Connector”* for querying the semantic model taking into account filters configured by the users of the Data Selector. The Data Provider Client is used to send the DAC, including pre-configured data source processors, to the Data Provider.
- The “Data Selector” provides comprehensive data SFS features based on semantic model queries to SFS for I40 components and their data along their life-cycle. Some examples (also combinable) are:
    - Searching based on data property values (IRDI, labels, etc.).
    - Searching based on classification classes (for instance eCl@ss based sub models).

   - Searching based on their (current) context (for instance I40 components in a specific location).

In the "User Registration and Login" sub-module, users can register or login. Based on the logged in user account only authorised data can be found (compare "User" and "UserRole" in Figure 4-1).

The "Data Filtering and Selection" sub-module provides all SFS features. Basic features are – as already introduced - (combinable) search and filter functionalities based on classification and categorisation information presented in Figure 4-1 (life-cycle, hierarchy level, sub models, etc.). Found data can be selected and bundled into a DAC.

The "Data Access Catalogue Generator" generates the DAC which is a configuration for the "Data Provider" module. The DAC includes information to the chosen/selected data, and data-source processor-configurations, in a human readable format (for instance XML, JSON, YAML Ain't Markup Language (YAML), etc.).

4.3.2.5 Data Provider

The "Data Provider" is used to extract and pipeline selected data bundled in a DAC. The module is able to provide the plain DAC to the user and to deploy pre-configured Data Source Processors in the "Data Flow Management Platform" located in the "Big Data Environment". Deployed connectors can then extract the data, ready for further cleaning/transformation or forwarding. The Data Provider is divided in following sub-modules:

- The "Interface" includes the "Data Provider API" used to read and deploy the data source processors defined in the DAC.
- The "Plain DAC Provider" enables the Data Provider user to read the DAC entries provided by the Data Selector.
- The "Data Access Configurator and Deployer" enables the manual configuration (if needed) and deployment of Data Source Processors. This sub-module consists of the following parts:
   - The "Data Source Processor Configurator" pre-configures the processors
   - The "Data Flow Management Platform Deployer" deploys them on the "Data Flow Management Platform".

4.3.2.6 Component Instance (+ Systems and Integrator)

The I40 component instance is optional and fulfils the requirements of an I40 component defined in DIN-SPEC-91345.

Main requirements for an I40 component instance related to this doctoral work are:

- is accessible via an AAS
- produces and provides data along its whole life-cycle
- provides access to asset functionalities
- represents the business, functional, information and communication layer according to RAMI 4.0.

The I40 component instance includes the following sub-modules:

- The "Interface" includes the *Semantic Manager Connector* to access the "Semantic Manager". This interface is used to register the I40 component in the semantic model and to update its description (for instance changed position saved in the manifest). The "I40 Comp AAS API" is used to access the I40 component instance by a web interface. The AAS provides general information about the I40 component (manifest), information about the services (sub-models) and their current status/values/measurements/etc. (SMPCs (and Data Source Bundle, if needed)). It can also be used to access its functionalities. The "Data Store Connector" is used if a direct interface to data storage in the "Big Data Environment" is needed to CRUD data. The "Data Stream Broker Connector" is used in case data streams are consumed or produced over a message broker.
- The AAS includes – according to the introduced semantic model, the "manifest" in the header part, a list of classifying "sub-models" and their SMPCs in the "body". The SMPCs in an I40 component instance are party addressing data produced by I40 component internal services managed in the component manager.
- The "Component Manager" manages I40 component instance internal services that are providing its functionalities.

Below the "I40 component Instance" module, a (legacy) systems module and an optional integrator module that represent the integration and asset layer of an I40 component according to the layers of DIN-SPEC-91345, is shown.

The solution is also suitable for direct integration of legacy systems so that the I40 component instance module is optional.

5

PROTOTYPE IMPLEMENTATION AND TESTING

This section describes an implemented generic[4] prototype based on the specification described in section 4.3 and its test in an industrial relevant environment.

5.1 Technology Screening

Several technologies can be taken into account for the implementation of the software architecture and the semantic data model. This section provides a brief technology screening identifying the most appropriate technologies for implementing the DIN Spec 91345 RAMI 4.0 compliant data pipelining approach.

The following sections are addressing the different modules of the software architecture specified in section 4.3.2.

5.1.1 Technologies for industrial data environments

The software architecture shows modules which characterise a Big Data environment. The environment covers different data stores and data stream brokers that represent different kinds of data sources suitable for the validation of the data pipelining approach.

[4] Generic for use in the manufacturing industry, resulting from compliance with RAMI 4.0.

Next to data sources and message brokers, graph data stores needed to store, query and update the knowledge graph based on the semantic model are also analysed.

Further technologies for a data flow management are analysed to use established technologies as basis for the Data Provider. Suitable technologies are providing functionalities for ETL (Extract, Transform, Load), ELT, data flow or data pipelining.

Scalable Data Storage Technologies

With the growing amount of available data the former RDBMS, centralised and concurrent, became inappropriate to deal with the huge chunks of data (Michalski, Carbonell, & Mitchell, 2013). Hence, to cope with this change in the amount of data, a new type of database emerged. This new kind of databases (Freitag, 2000) are suited for distributed systems and parallel access. NoSQL databases are very scalable and provide good performance rates on unstructured Big Data (Klein et al., 2015). Instead of tables as in SQL databases, NoSQL databases are structured in collections where data is saved as key-value pairs that are represented in the data itself. Compared to SQL databases they have a lower normalization degree caused by disjointed data collections (Li & Manoharan, 2013). This makes them better scalable and improves the data exchange performance as there is no need to block whole tables for writing actions or to deal with exponentially growing consistency processing's. Some example for such databases are e.g. MongoDB (Andrieu, Freitas, Doucet, & Jordan, 2003), Cassandra (Sebastiani, 2002) or Hadoop Distributed File System (HDFS) (Jung, Youn, Bae, & Choi, 2015; Vora, 2011).

Modern approaches to save and process big amounts of data are also scalable object and TSDB databases. An extensive overview of scalable Big Data storage systems was given in (Strohbach, Daubert, Ravkin, & Lischka, 2016).

Table 5-1 provides a technology screening of such approaches.

Table 5-1: List of selected Data Storage Technologies

Technology	Characteristics	Description
Selected Distributed File Systems		
HDFS	• Open Source (Apache License 2.0) • Scalable • Part of Apache Hadoop • Locally managed • Well established	Is a popular distributed, scalable and Java based file system for the Hadoop framework based on the Map Reduce algorithm. Is suitable for bigger files and many compliant extensions coming from the Hadoop framework are available (HBase, Hive, Pig, etc.) (Borthakur, 2008).
Amazon S3	• Proprietary by Amazon.com • Scalable • Remote access • Well established	Is an Amazon Webservices (AWS) scalable object store with the aim to provide a high scalability and availability. Objects are addressable by bucket-key IDs. E.g.: `http://s3.amazonaws.com/`**`bucket`**`/`**`key`** See (Palankar, Iamnitchi, Ripeanu, & Garfinkel, 2008).
Microsoft Azure	• Proprietary by Microsoft • Scalable (some DBs) • Remote access • Cloud computing platform	A cloud computing platform which includes several kinds of local (Azure Virtual Machine (VM)) and remote (REST API), partly scalable, data storage systems (queue, table, blob, file, disc) (Copeland, Soh, Puca, Manning, & Gollob, 2015).
Selected NoSQL Databases (Key-Value, Column, Document Databases)		
CouchDB	• Open Source (Apache License 2.0) • Scalable • Document-oriented/JSON format • Open Source (Apache)	A NoSQL DB by the Apache Software Foundation mainly written in Erlang, which stores data/documents in JSON format. It is/was used in several applications as in Ubuntu or Facebook. It is highly scalable as it

		uses map-reduce to query the independently stored documents (Lennon, 2010).
MongoDB	• Open Source (SSPL) • Document-oriented/JSON format • Well established	A NoSQL DB by MongoDB, Inc. written in C++. Several documents with different schemas can be grouped into collections inside a MongoDB (MongoDB, 2014).
Apache Cassandra	• Open Source (Apache License 2.0) • Scalable • Column-oriented • High I/O performance (Abramova & Bernardino, 2013) • Well established	A NoSQL DB by the Apache Software Foundation and written in Java. Cassandra is well established and used by Apple, Twitter, Digg and Reddit (Cassandra, 2014).
Graph Databases and Triple Stores		
Neo4J	• Community-Edition: Open Source (GPLv3) • Popular Graph Database • Scalable (Enterprise Edition)	A graph database by Neo4j written in Java. The enterprise edition allows clustering, hot backups, and monitoring. As in many graph databases, Neo4J stores data as edges and nodes which can be labelled with attributes (Van Bruggen, 2014).
Allegro-Graph	• Proprietary by Franz's • Document and Graph DB	A Document and Graph DB by Franz's which supports JSON, JSON-LD, SPARQL 1.1, RDFS++, Shapes Constraint Language (SHACL), and Prolog (Aasman, 2006). AllegroGraph is W3C/ISO standards compliant (Fernandes & Bernardino, 2018).
Blazegraph	• Open Source (Aduna BSD license) • Scalable • Native graph DB • High performance	A graph database by Systap written in Java. It is used in the Wikidata SPARQL Endpoint. It supports SPARQL and RDF (Blazegraph™ DB, 2020).

Apache Jena TDB2	• Open Source (Apache License 2.0) • RDF storage and query • High performance RDF store on single machine	A graph database by the Apache Software Foundation which support the full range of Jena APIs and is written in Java. This includes also the support of SPARQL and RDF (Apache Software Foundation, 2020).
Time Series Databases		
InfluxDB	• Open Source (MIT) • Scalable (commercial) • Well established	A TSDB for storing time series, events and metrics by InfluxData built to store Big Data volumes of time-stamped data (Ahmad & Ansari, 2017).
Prometheus	• Open Source (Apache License 2.0) • Scalable with Cortex on top of Prometheus • Popular	A TSDB by the Cloud Native Computing Foundation for numeric data and written in Go. It enables real-time metrics storage and supports alerts based on them (Brazil, 2018).
OpenTSDB	• Open Source (LGPL) • Scalable • Works with HBase from Hadoop	A scalable TSDB written in Java which runs on HBase from the Hadoop framework using map-reduce. It was built for big amounts of time-series data and supports numeric data for metrics and strings for tags (Spaggiari & O'Dell, 2016).

For the implementation of the software architecture, the presented technologies are not necessarily needed but they represent a selection of different kinds of potential data sources. A selection of them is used for the validation of the prototype.

Message Broker

For a message-based data transfer (scalable) message brokers are needed. Table 5-2 provides a list of selected Message Broker technologies.

Table 5-2: List of selected Message Broker Technologies

Technology	Characteristics	Description
MQTT Broker		
HiveMQ[5]	• Proprietary by dc-square GmbH (a community edition is open source (Apache License version 2.0)) • Highly Scalable through load balancing in a cluster • Many extensions available (Apache Kafka, SQL DB, InfluxDB, etc.)	A commercial MQTT Broker written in Java which provides the possibility to process incoming data directly in the broker via a plugin interface that can be used e.g. to save data in a database. It is highly scalable and provides several extensions.
Eclipse Mosquitto	• Open Source (Eclipse Public License 1.0, Eclipse Distribution License 1.0 (BSD))	An MQTT Broker by Eclipse which is well known and open source (Light, 2017).
Apache ActiveMQ	• Open Source (Apache License 2.0) • Scalable • Supports Java Message Service (JMS), MQTT, AMQP 1.0, Simple (or Streaming) Text Oriented Message Protocol (STOMP), OpenWire, REST and WebSockets	A multi message broker by the Apache Software Foundation which supports several transport protocols (Snyder et al., 2017).
Other Message Brokers		
Apache Kafka	• Open Source (Apache License 1.0) • Uses a binary TCP protocol	A message broker implemented in Java and Scala by the Apache Software Foundation

[5] See https://www.hivemq.com/hivemq/mqtt-broker/

	• Scalable • Provides a feature for a persistent storage of messages • Well known	which combines storage, transmission and further processing of data (Garg, 2013).
RabbitMQ	• Open Source (Mozilla Public License) • Uses AMQP	An AMQP Broker by Rabbit Technologies Ltd which is implemented in Erlang (Johansson & Dossot, 2020).

For the implementation of the prototype, the presented message broker technologies represent different kinds of potential data sources. Some of them will be used for the validation of the prototype.

Data Flow Management (used for Data Provider)

See section 5.1.4 about technologies for the Data Provider.

5.1.2 Technologies for the Semantic Manager

The Semantic Manager manages the semantic data model and the resulting knowledge graph. Some selected technologies suitable as a basis for development are listed in Table 5-3.

Table 5-3: List of selected Semantic Web Technologies

Technology	Characteristics	Description
Ontology/Semantic Model Editors		
Fluent Editor	• Proprietary from Cognitum • Free for individual developers, open-source projects, academic research, education, and small professional teams • Interoperable with Protégé • W3C compliant	An ontology editor from Cognitum that focuses on the meaning of taxonomy, vocabulary, ruleset, etc. of an ontology (Seganti, Kapłański, & Zarzycki, 2015).

Protégé	• Open Source (Mozilla Public License) • Well known	An ontology editor from the Stanford University + Community written in Java. The editor is well established and enables a detailed development of ontologies. Next to a desktop version also a web version is available (Gennari et al., 2003).
Semantic Web Stack related Technologies		
Apache Jena[6]	• Open Source (Apache License 2.0) • Well known	A Java implementation of semantic web technologies including OWL from the Apache Software Foundation. It provides APIs to query and manipulate RDF graphs. It includes the TBD2 to store and query RDF graphs. It includes "Fuseki" which is a HTTP server to present RDF data and answer SPARQL queries over HTTP.
RDF4J[7]	• Open Source (Eclipse Distribution License (EDL), v1.0)	A framework for storing, querying, and analysing RDF data on a server. It is an Eclipse project and written in Java. It supports SPARQL and Sesame RDF Query Language (SeRQL) (Broekstra & Kampman, 2003).

For the implementation of the Semantic Manager, a semantic model editor and a framework to query and manipulate the semantic data model are needed. Further, a graph data store is needed to store the semantic model and the resulting knowledge graph.

As basis for the implementation of the semantic model the editor Protégé is used because it provides a user-friendly Graphical User Interface (GUI), is open source and provides features to develop OWL files in detail. Further, the editor is well known and has a large supporting community.

[6] See https://jena.apache.org/getting_started/

[7] See https://rdf4j.org/documentation/

As framework for the implementation of the Semantic Manager, Apache Jena will be used as it fulfils all needed requirements. Apache Jena Fuseki provides a HTTP server for querying and updating semantic models. It is W3C compliant and the integrated TBD2 database, described in section 5.1.1, provides a suitable graph data store.

5.1.3 Technologies for the Data Selector

For the implementation of the Data Selector Java as programming language will be used. For Java there are several suitable frameworks available for the implementation of the user-friendly Web Interface such as Primefaces (Varaksin, 2013) or Java Server Faces (E. Burns, Schalk, & Griffin, 2010). Java also has a large community for the Spring framework that provides features for an efficient development of web application including security issues using Spring Security[8].

5.1.4 Technologies for the Data Provider

The Data Provider is needed to pipeline/transfer big amounts of data from one point to another, including different I/O interfaces and optional data pre-processing or transformations. Table 5-4 provides a list of selected data pipelining compliant technologies.

Table 5-4: List of selected data pipelining compliant technologies

Technology	Characteristics	Description
Data pipelining compliant technologies		
Apache NiFi[9]	• Open Source (Apache License 2.0) • Scalable • Allows custom Connectors • Well known • Provides REST API	A distributed dataflow engine by the Apache Software Foundation written in Java. It is a flow-based programming approach, is very flexible and include several processors to handle different kinds of interfaces or to process data. It also allows to write custom data processors (Samal & Panda, 2017).

[8] See https://spring.io/projects/spring-security

[9] See https://nifi.apache.org/

	• Comprehensive number of connectors	
StreamSets Data Collector[10]	• Open Source (Apache License 2.0) • Scalable • Provides REST API	A distributed dataflow engine by the Stream-Sets Inc. written in Java. Provides an attractive user interface.
Hevo Data[11]	• Proprietary by Hevo Data Inc. • Company offers custom data source integration in case no standard connector is available	A data pipeline tool by Hevo Data Inc. to clean, enrich, and transform data on the fly. It offers several standard connectors for several kinds of data sources and offers also an automatic data type and schema mapping.
Apache Airflow[12]	• Open Source (Apache License 2.0)	A data pipeline tool by the Apache Software Foundation written in Python. Airflow is a platform to programmatically author, schedule, manage, and monitor workflows which represents workflows as - so called - directed acyclic graphs (DAGs) of tasks.
Talend Open Studio	• Apache License • Eclipse based framework + Java • Allows custom Connectors • Well known	Is an ETL software by Talend written in Java. Talend Open Studio is the limited open-source version which contains selected core modules of their Data Management Platform and is limited in its functionality (Bowen, 2012).

[10] See https://streamsets.com/products/dataops-platform/data-collector/

[11] See https://hevodata.com/platform/

[12] See https://airflow.apache.org/

For the implementation of the Data Flow Management and the Data Provider Apache NiFi will be used as basis. It fulfils the needed requirements of a wide range of available connectors/processors, it is a scalable software solution and it provides a comprehensive API to manage data flows. Next to these advantages it is also open source, has a large community and is customizable.

5.1.5 Technologies for the I40 component instance

Similar to the Data Selector, for the I40 component instance there are no specific technologies available that can be used as basis. For the implementation of the I40 component Java and the Spring Boot library will be used.

5.1.6 Technologies for the Containerisation and Clustering

To enable an operating system (OS) independent use of applications, to make them portable and to securely connect specific applications in an isolated environment, an OS-level virtualization software should be used. A typical software therefore is the open-source (Apache 2.0 license) software Docker (Anderson, 2015) from the Docker Inc. Docker isolates applications in – so called – containers from the rest of the system. Docker enables to run multiple containers on a single machine and enables also a connection between these containers.

If scalable solutions are needed to be executed on multiple machines, a solution for an automated application deployment, scaling, and management is needed. Such a solutions is the open-source (Apache License 2.0) software Kubernetes (B. Burns, Beda, & Hightower, 2018) from the Cloud Native Computing Foundation. Kubernetes allows an automated provision, scaling, load-balancing and networking of containers (called “pods” in Kubernetes) and can be managed using a single dashboard or a command line.

While Docker is specialized on the OS-level virtualization, Kubernetes can run on top of Docker to manage scalable clusters.

Although other technologies are existing next to Docker and Kubernetes, both technologies are currently by far the most popular ones with a large community (see 2018 Docker usage report by Eric Carter (Carter, 2018)) and are therefore used in this work.

5.1.7 Physical architecture for data pipelining

Figure 5-1: Mapping of selected technologies on the software architecture

Figure 5-1 and Table 5-5 conclude the technological screening with a mapping of selected technologies used as basis for the implementation of the architecture described in section 5.3.

Table 5-5: Summary of selected technologies

Software Architecture Module	Technologies
Big Data Environment	• An Apache Cassandra Cluster running on Kubernetes. • An HiveMQ Cluster running on Kubernetes. • A Jena Fuseki server running in a Docker container to store, query and update the knowledge graph based on the semantic model. • Apache NiFi running in a container as basis to deploy data source connectors.
Semantic Manager	• Protégé as semantic model editor • Apache Jena for the semantic model management • Docker to containerize the Semantic Manager
Data Selector	• Java as programming language • Spring Boot as comprehensive framework for the development of the web application

	• Spring Security as library for the implementation of security features. • MySQL as database for internal configurations and user management. • PrimeFaces for the implementation of a graphical web interface
Data Provider	• Apache NiFi as basis to configure and deploy data source connectors. • Docker to containerize the Data Provider
I40 component	• Java as programming language • SpringBoot as comprehensive framework for the development web related features as REST for the AAS.

For the implementation of the semantic model in section 5.2 is used the editor Protégé.

5.2 DIN-SPEC-91345 compliant semantic model implementation

The semantic model implementation based on the specification described in section 4.3.1 is divided in different namespaces (/Internationalized Resource Identifiers (IRI)), as shown in Figure 5-2. One major requirement for this implementation is to enable an easier integration of new vocabularies, I40 components, smart manufacturing environments and sub models, but also to achieve a better overview, simplifying the engineering and the maintaining of the semantic model.

Figure 5-2 shows the different namespaces represented as bubbles.

The schema bubbles include classes, object properties, data properties, some generic individuals and their relations to describe main elements of the semantic model.

The instantiations bubbles show the instantiated individuals that uses the related schemas. One instantiation could be for instance an I40 component that uses the I40 component schema.

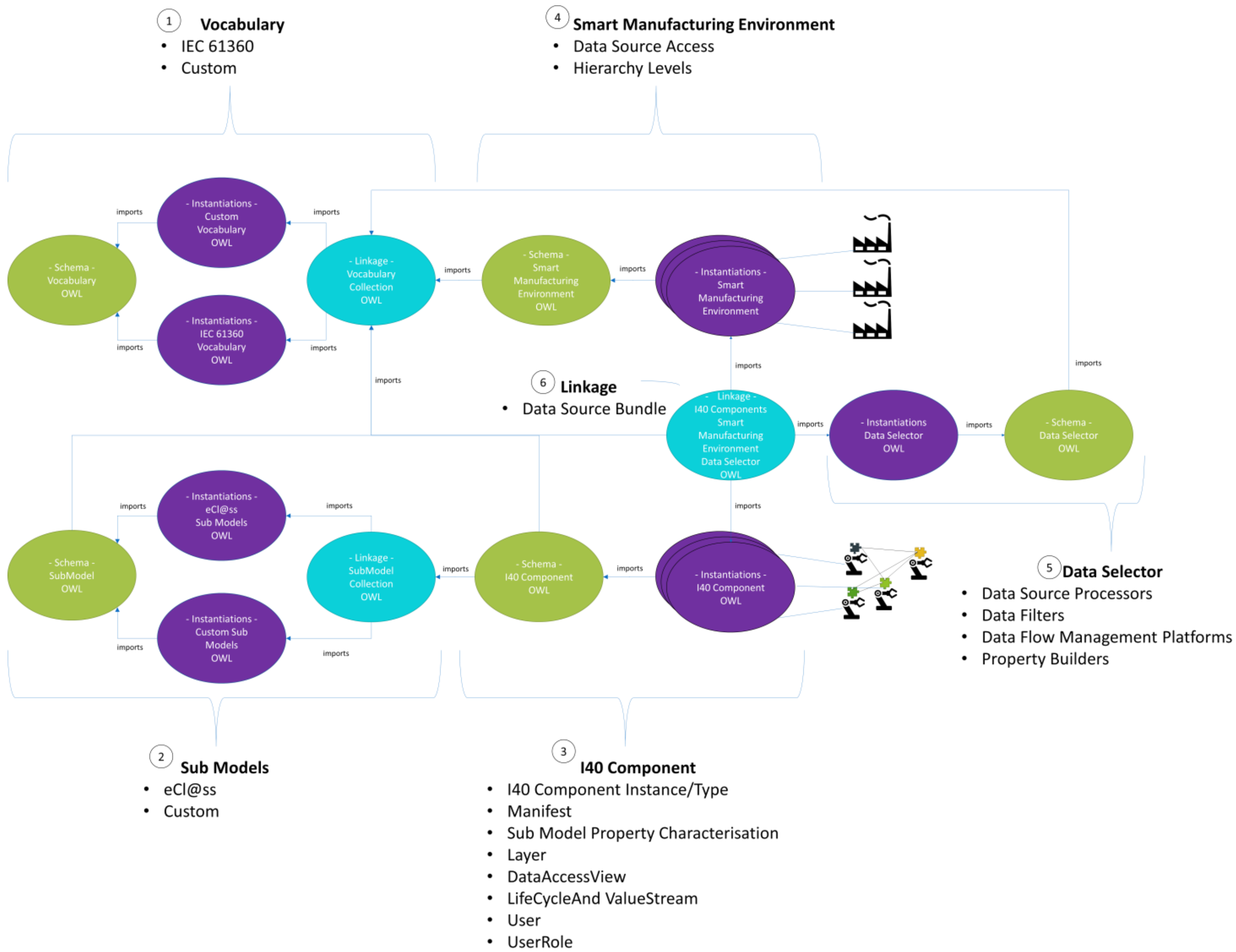


Figure 5-2: Semantic model division into different namespaces/IRIs

The linkage bubbles show linkages and collection of/between main elements. They act as a collection of individuals to generate one central point of access and to simplify the engineering process. For instance, the sub model collection includes all eCl@ss based sub models and all custom sub models.

The main elements of the semantic model are:

- **Number 1**: Vocabulary – which describes the mandatory and optional vocabularies in form of data properties partly based on standards like the IEC 61360. This bundle is divided in a schema namespace, instantiation namespaces (IEC 61360; Custom) and a collection namespace that bundles all instantiations.
- **Number 2**: Sub models – which describes the structure of classification classes partly based on standards like eCl@ss. This bundle is divided in a schema namespace, instantiation namespaces (eCl@ss; Custom) and a collection namespace that bundles all instantiations.
- **Number 3**: I40 component – which describes classes and object properties that are directly linked to an I40 component. An I40 component includes classes to describe the I40 component category (is it an instance or a type), the manifest, layers, sub model property characterisations, data access views, life-cycles and value streams and user access. The bundle is divided in a schema namespace and in instantiation namespaces that initialise I40 components.
- **Number 4**: Smart manufacturing environment – which defines classes and object properties to describe a smart manufacturing environment. A smart manufacturing environment includes hierarchy levels and data sources. The bundle is divided in a schema namespace and in instantiation namespaces that initialise smart manufacturing environments.
- **Number 5:** Data selector – the Data Selector part includes for instance data source processors, data filters, data flow management platforms and property builders.
- **Number 6:** Linkage – the central point where all individuals of smart manufacturing environments, I40 components and the Data Selector get joined. In this namespace are also defined the data source bundles.

Each main element is described in the following sub-sections. The shown screenshots were made with Protégé (Gennari et al., 2003) which was used for the engineering of the semantic model.

5.2.1 Vocabulary

This section describes the implementation of the vocabulary which is a list of data properties in a prescribed taxonomy.

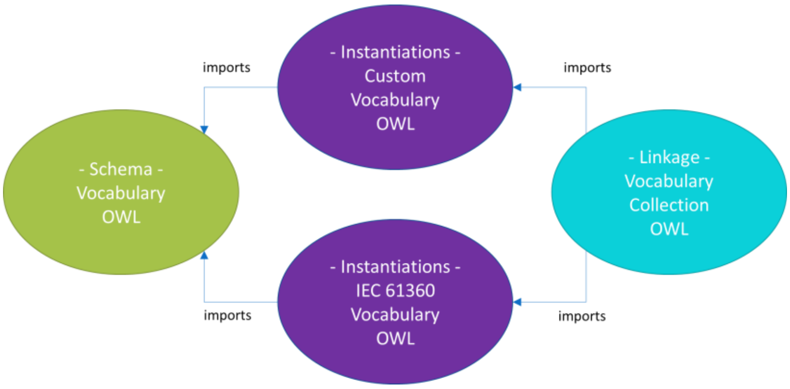


Figure 5-3: Vocabulary Structure

Figure 5-3 shows an overview of the implementation. The vocabulary schema is used by individuals for their description. Implemented is a vocabulary based on IEC 61360 and a custom vocabulary. Both vocabularies are bundled/linked in the vocabulary collection.

Each namespace is described below.

Schema - Vocabulary

The vocabulary schema has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/vocabulary-schema` and describes one `Vocabulary` data property as top-level data property for all integrated vocabularies as shown in Figure 5-4.

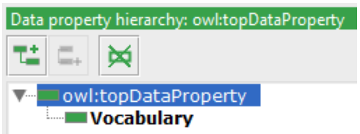


Figure 5-4: Top-level Data Property "Vocabulary"

The vocabulary schema is imported by the IEC 61360 based vocabulary and the custom vocabulary described in the following.

Instantiations – IEC 61360 Vocabulary

The IEC 61360 OWL has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/iec61360` and implements a vocabulary based on IEC 61360. As described in section 4.3.1.3, the IEC 61360 describes properties for data elements and for item classes as shown in Figure 5-5.

Figure 5-5: IEC 61360 based data properties

Instantiations – Custom Vocabulary

The custom vocabulary OWL has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-vocabulary` and implements a list of custom data properties.

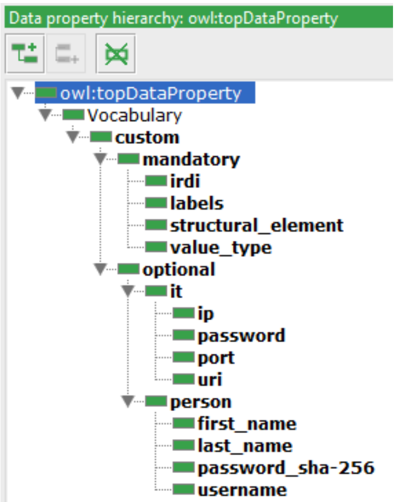


Figure 5-6: Custom data properties

Figure 5-6 shows a snippet of the implemented vocabulary. It shows mandatory data properties as described in section 4.3.1.3 and some additional optional data properties.

Linkage – Vocabulary Collection

The vocabulary collection has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary` and imports the IEC 61360 vocabulary and the custom vocabulary (see Figure 5-7). The collection represents a central point to access the whole vocabulary.

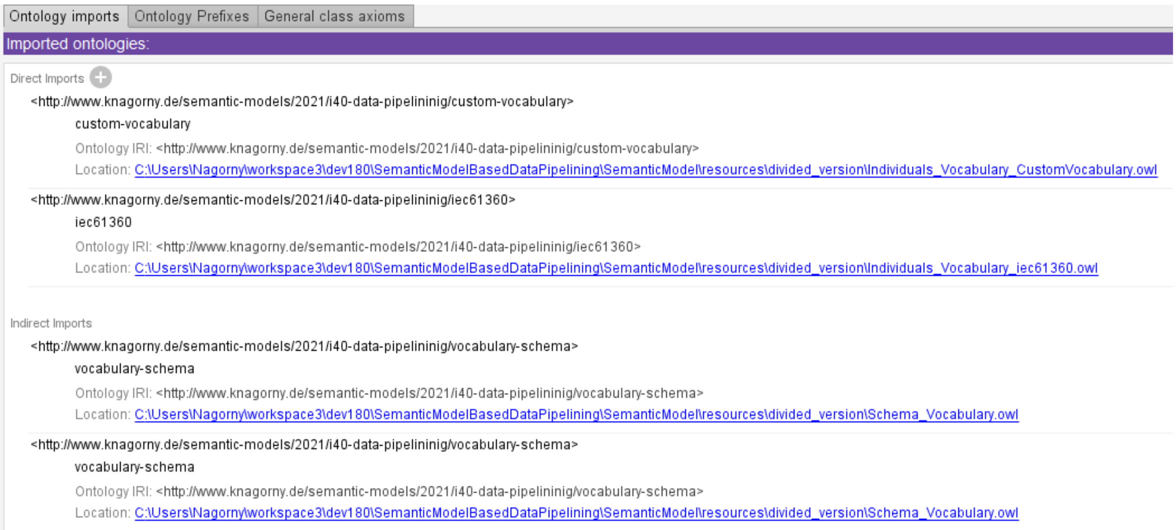


Figure 5-7: Vocabulary Collection - Imports

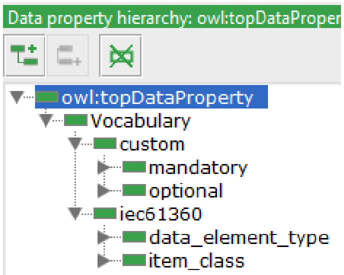


Figure 5-8: Vocabulary Collection – integrated data properties

Figure 5-8 shows the integrated vocabulary collection which includes the data properties coming from the IEC 61360 vocabulary and the custom vocabulary as sub-level data properties of the Vocabulary data property, described in the schema.

The vocabulary collection is used by all other main elements to describe their individuals.

5.2.2 Sub Models

This section describes the implementation of the sub models that are used to classify the data described in sub model property characterisations in a standardised way.

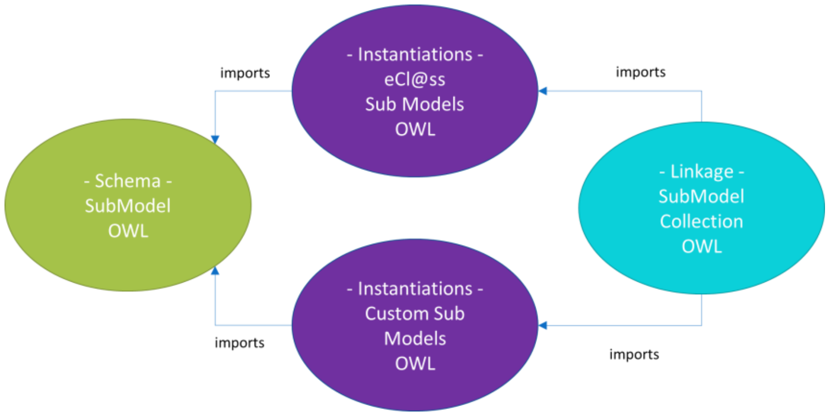


Figure 5-9: Sub Model Structure

Figure 5-9 shows an overview of the implementation. The sub model schema is used by individuals for their description. Implemented are sub models based on eCl@ss 10.1 and custom sub models. Both instantiated sub models are bundled/linked in the sub model collection.

Each namespace is described in the following.

Schema – Sub Model

The sub model schema has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/sub-model-schema`, imports the vocabulary collection and describes one `SubModel` class used as top-level class for all integrated sub models, and an `isSubModelOf` object property to describe sub model hierarchies (see Figure 5-10).

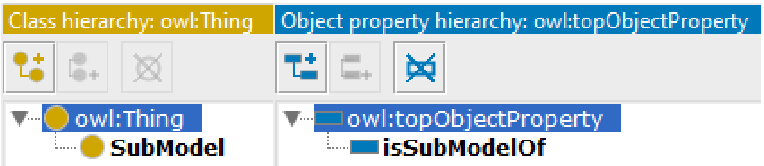


Figure 5-10: Class and object property of the sub model schema

The sub model schema is imported by the eCl@ss and the custom sub model individuals described in the following.

Instantiations - eCl@ss Sub Models

The eCl@ss OWL has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/eclass10_01` and implements sub models based on eCl@ss version 10.1. Used for this doctoral work was the English Comma-Separated Values (CSV) eCl@ss basic version[13] with all included classification classes (needed to classify data) located in file *eClass10_1_CC_en.csv* available in the eCl@ss download portal (eCl@ss e. V., 2020).

The *eClass10_1_CC_en.csv* file needs to be transformed to a semantic model structure that fits to the sub model schema. Therefore, is used the Protégé Desktop plugin Cellfie[14] which is used to import spreadsheet data into OWL semantic models.

Figure 5-11 shows the import and transformation procedure for all 42.220 eCl@ss 10.1 classification classes (see start row: 2; end row: 42221). The transformation rule (see column "rule" in the figure) shows the mapping between columns in the CSV file and the semantic model:

- *Individual: @T**: Describes that an *individual* has to be created with the name written in cell *@T**. Column T in the CSV file includes the IRIDIs of the classification class which are used as individual names.
- *Types: <class name>*: Describes which type the individual has. For instance: classificationClass.
- *Facts: <data property name> @A**: Describes the data property and its value for the individual. For instance: the data property *supplier* with the value written in the cell of column *A*. In eCl@ss the *supplier* is *0173-1* which represents the eCl@ss e.V. Some other data properties are also added. Some data properties are based on the IEC 61360 vocabulary.

[13] See http://wiki.eclass.eu/wiki/read_me

[14] See https://github.com/protegeproject/cellfie-plugin

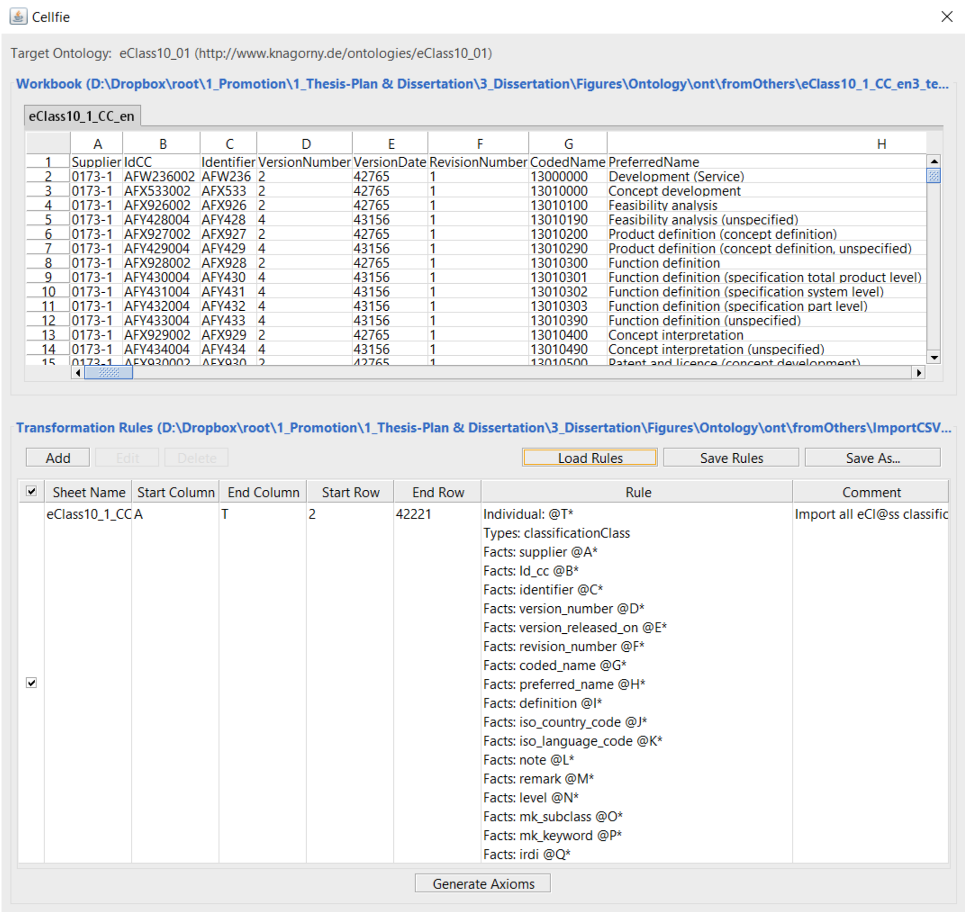


Figure 5-11: eCl@ss CSV import and OWL transformation using Cellfie

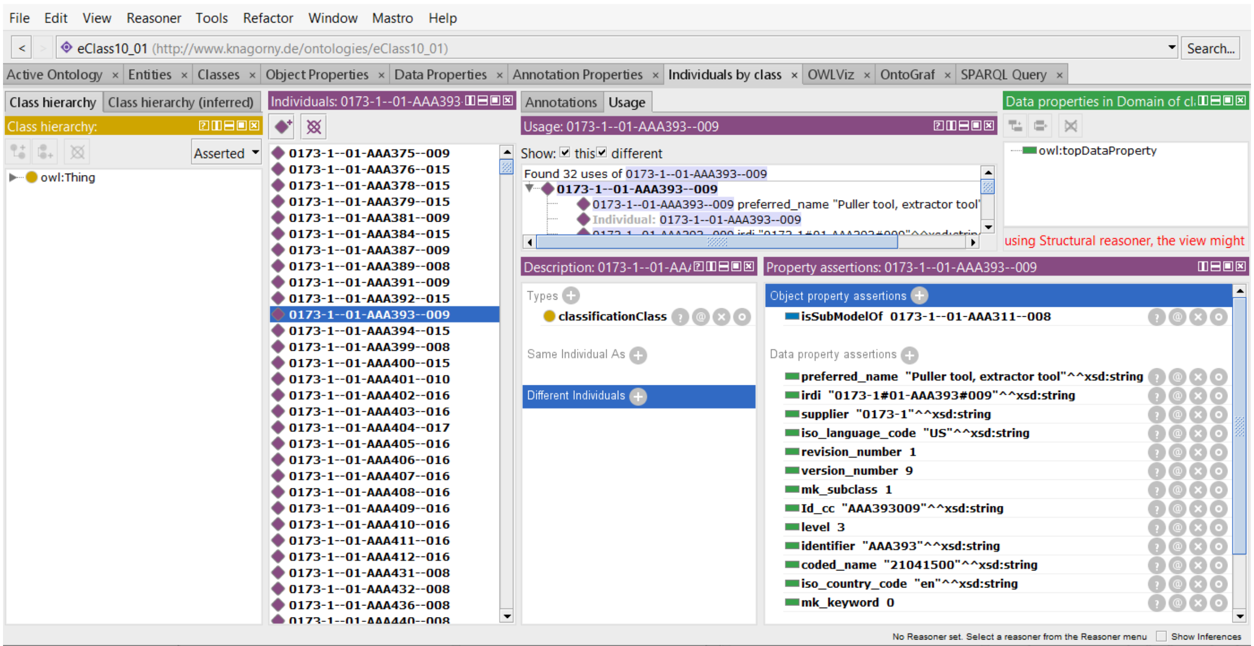


Figure 5-12: Individuals - eCl@ss Sub Models after import

Figure 5-12 shows one snippet of the transformed result. The class taxonomy in this ontology is "`SubModel --> eClass10_01 --> classificationClass`".

Individuals - Custom Sub Models

The custom sub model OWL has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-submodels` and implements a list of sub models that are not addressed by eCl@ss.

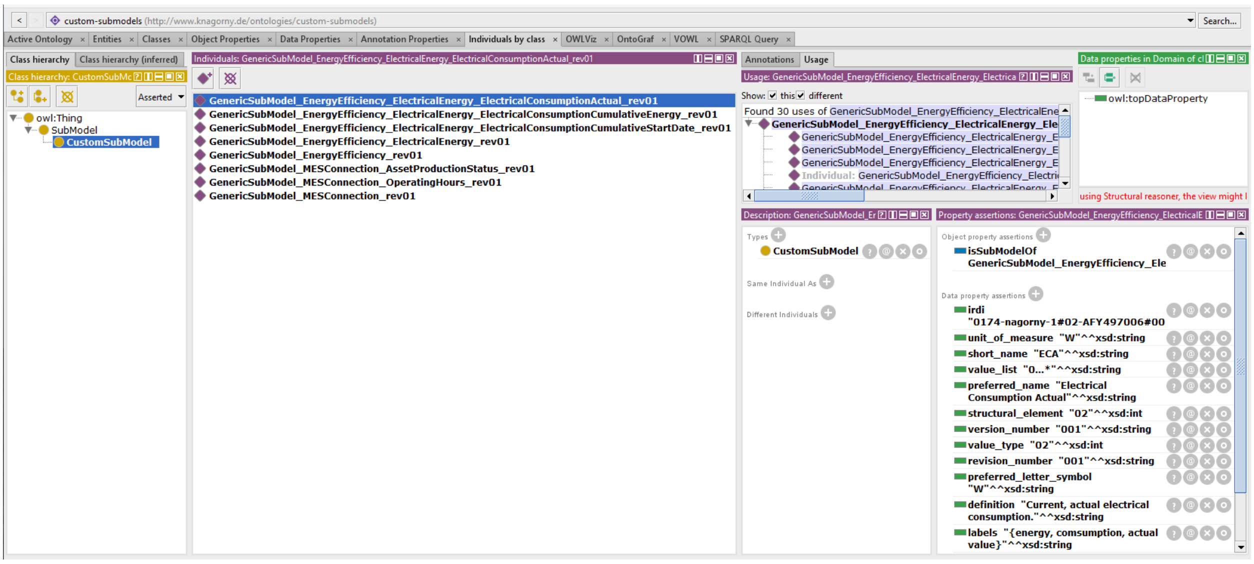


Figure 5-13: Individuals – Custom Sub Models

Figure 5-13 shows a snippet of the custom sub models that have the type `CustomSubModel`, are arranged in a tree structure by using the `isSubModelOf` object property and are defined by a list of instantiated data properties.

Linkage – Sub Model Collection

The sub model collection has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-submodels` and imports the eCl@ss10_1 based sub models and the custom sub models (see Figure 5-14). The collection represents a central point to access all sub models.

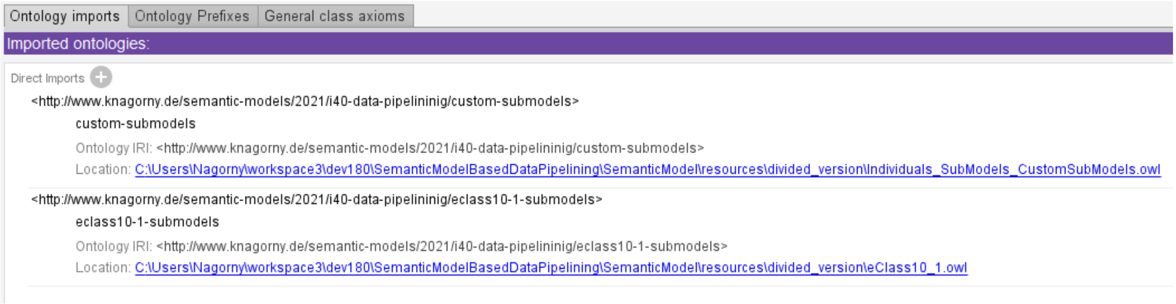


Figure 5-14: Sub Model Collection - Imports

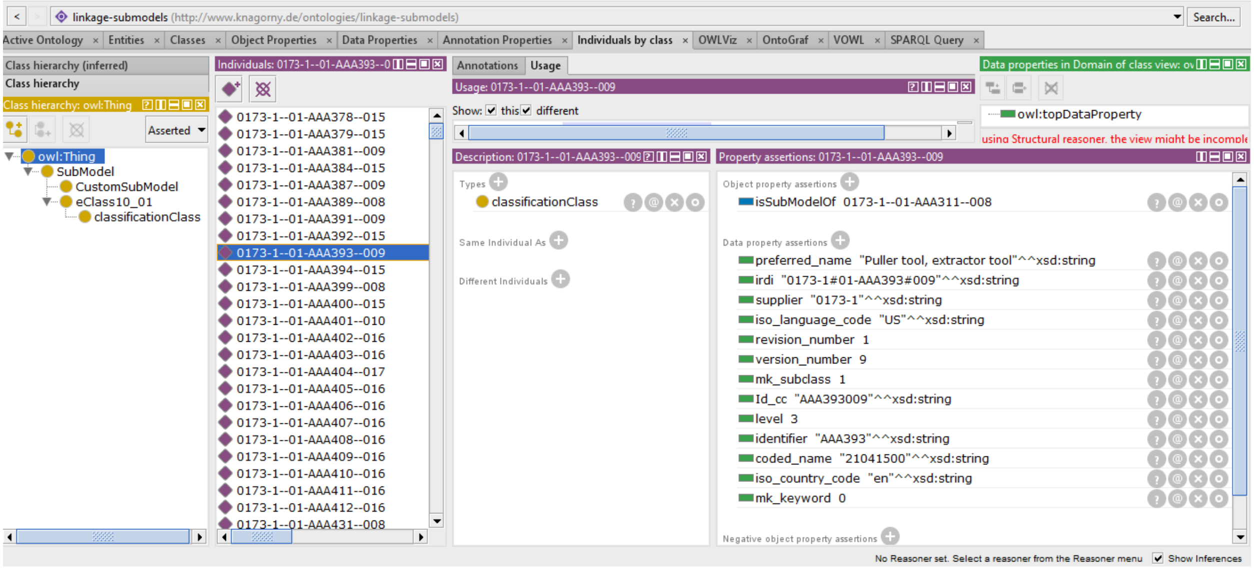


Figure 5-15: Sub Model Collection – Snippet of integrated sub models

Figure 5-15 shows a snippet of the integrated sub model collection. The sub model collection is used by I40 components described in the next section.

5.2.3 I40 component

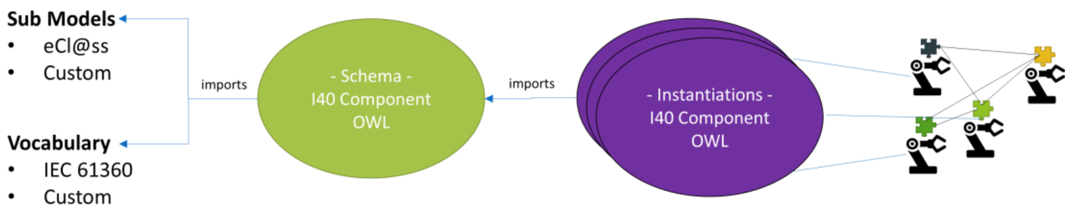


Figure 5-16: I40 component Structure

Figure 5-16 shows an overview of the implementation of the I40 component namespaces. The I40 component schema is used by individuals for their description. The schema and the instantiations are described below.

Schema – I40 component

The I40 component schema has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema` and imports the vocabulary collection and the sub model collection.

The schema describes classes, object properties, data properties, their relations and some generic individuals as basis for the instantiation of I40 components.

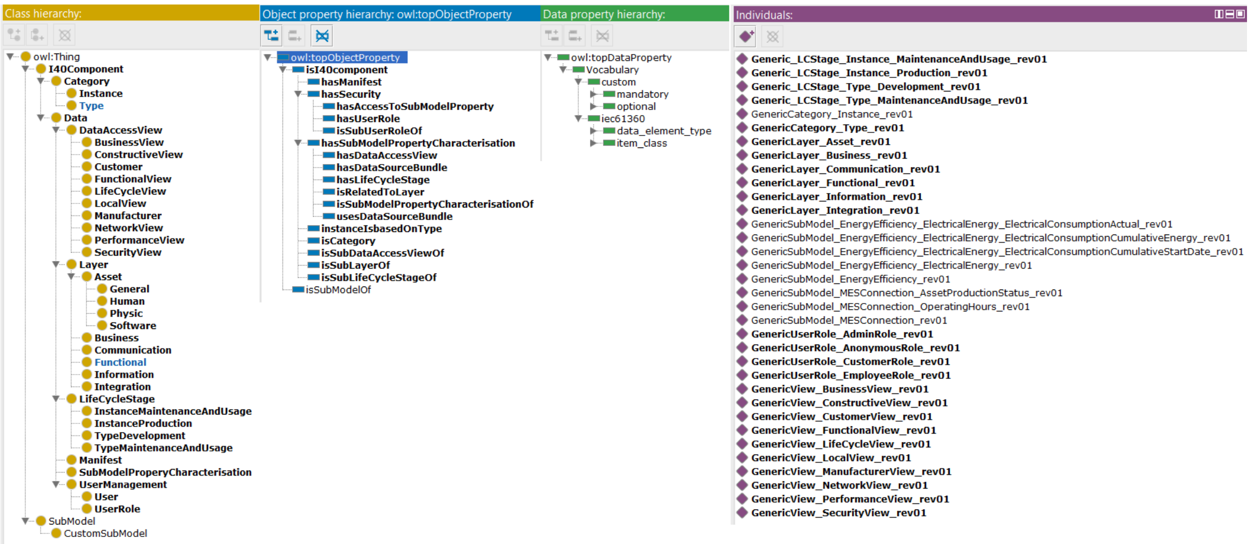


Figure 5-17: I40 component schema

Figure 5-17 shows the defined I40 component semantic model elements (Classes, Object Properties, Data Properties and Individuals). Elements shown in **bold** are elements directly described in the I40 component schema. Elements that are shown in a normal text format are imported from the vocabulary collection and from the sub model collection.

The **class taxonomy** describes the concepts of an I40 component based on DIN-SPEC-91345 as introduced in section 4.3.1.1. The *Category* class has the sub classes *type* and *instance*. An I40 component type describes the type of an I40 component while an instance describes instantiations of a type. The sub classes of the *Data* class show all concepts used to describe an I40 component. An individual with the type *Manifest* describes meta-data of an I40 component type or instance by using data properties of the imported vocabulary but also further freely defined data properties defined in the semantic model instance namespace. An individual with the type *SubModelPropertyCharacterisation* (SMPC) describes data related to an I40 component. A SMPC is linked to a Layer, a Data Access View, a Life-cycle stage, Users and User Roles, and is classified by a Sub Model as described in section 4.3.1.1. For the Data Access View, Layers and Life-cycle stages are defined basic sub-classes according to DIN-SPEC-91345.

The **object properties** are used to describe relations between Individuals related to an I40 component. A description of these relations can be derived from Figure 4-1.

The **data properties** are imports of the vocabulary collection used to describe individuals related to an I40 component.

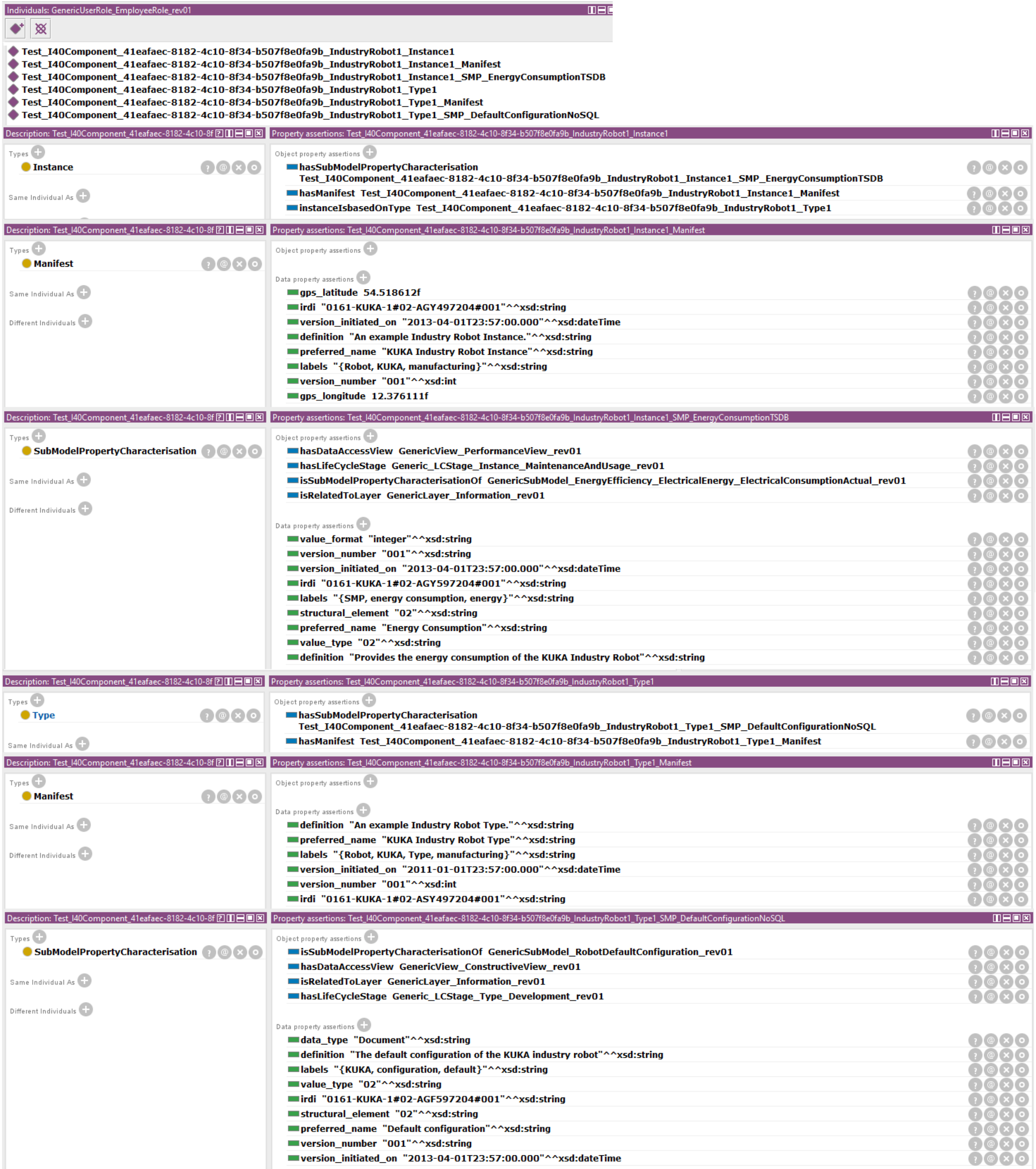


Figure 5-18: Exemplary I40 component instantiation

The **individuals** show next to imported sub models some instantiations of generic layers (see bold text) according to DIN-SPEC-91345. They are used as basis to describe an SMPC. From basis individuals can be derived further individuals for a further specification. An example for a Data Access View could be:

- *ThroughputView* → *isSubDataAccessViewOf* → *GenericView_PerformanceView_rev01*

After the description of the I40 component schema, the following part describes an instantiation example.

Instantiations – I40 component

Instantiations of I40 components are made based on the I40 component schema. Figure 5-18 shows a small exemplary instantiation.

5.2.4 Smart Manufacturing Environment

Figure 5-19 shows an overview of the Smart Manufacturing Environment implementation. The Smart Manufacturing Environment schema is used by instantiations for their description.

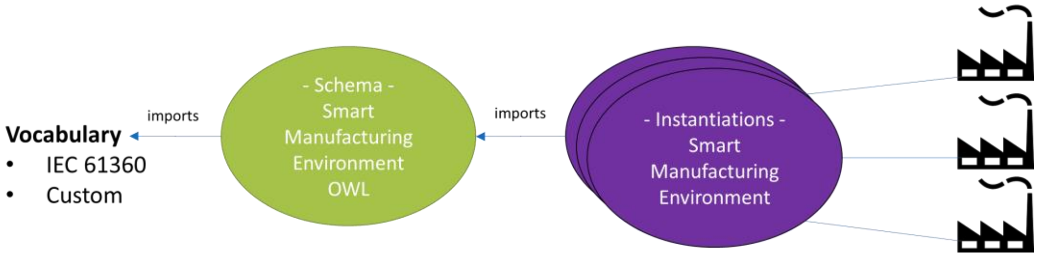


Figure 5-19: Smart Manufacturing Environment Structure

Schema – Smart Manufacturing Environment

The Smart Manufacturing Environment schema has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-environment-schema` and imports the vocabulary collection. The schema describes classes, object properties and data properties to be used as basis for the instantiation of Smart Manufacturing Environments as shown in Figure 5-20. The **class taxonomy** shows the class *DataSource* which describe any data source located in a smart manufacturing environment. The sub-classes describe concepts for basic data sources: services, data stores and message brokers. To describe the structure of smart manufacturing environment, the *HierarchyLevel* class is used which has a range of sub-classes according to RAMI 4.0. The **object property** *isSubHierarchyLevelOf* is used to build a structure of hierarchy level individuals. Available **data properties** are coming from the imported vocabulary collection.

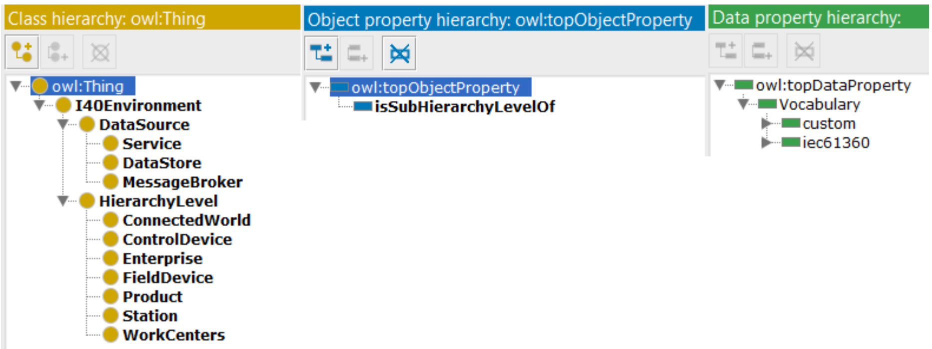


Figure 5-20: Smart Manufacturing Environment schema

Individual – Smart Manufacturing Environment

Instantiations of Smart Manufacturing Environments are implemented based on the Smart Manufacturing Environment schema. Figure 5-21 shows a small exemplary instantiation.

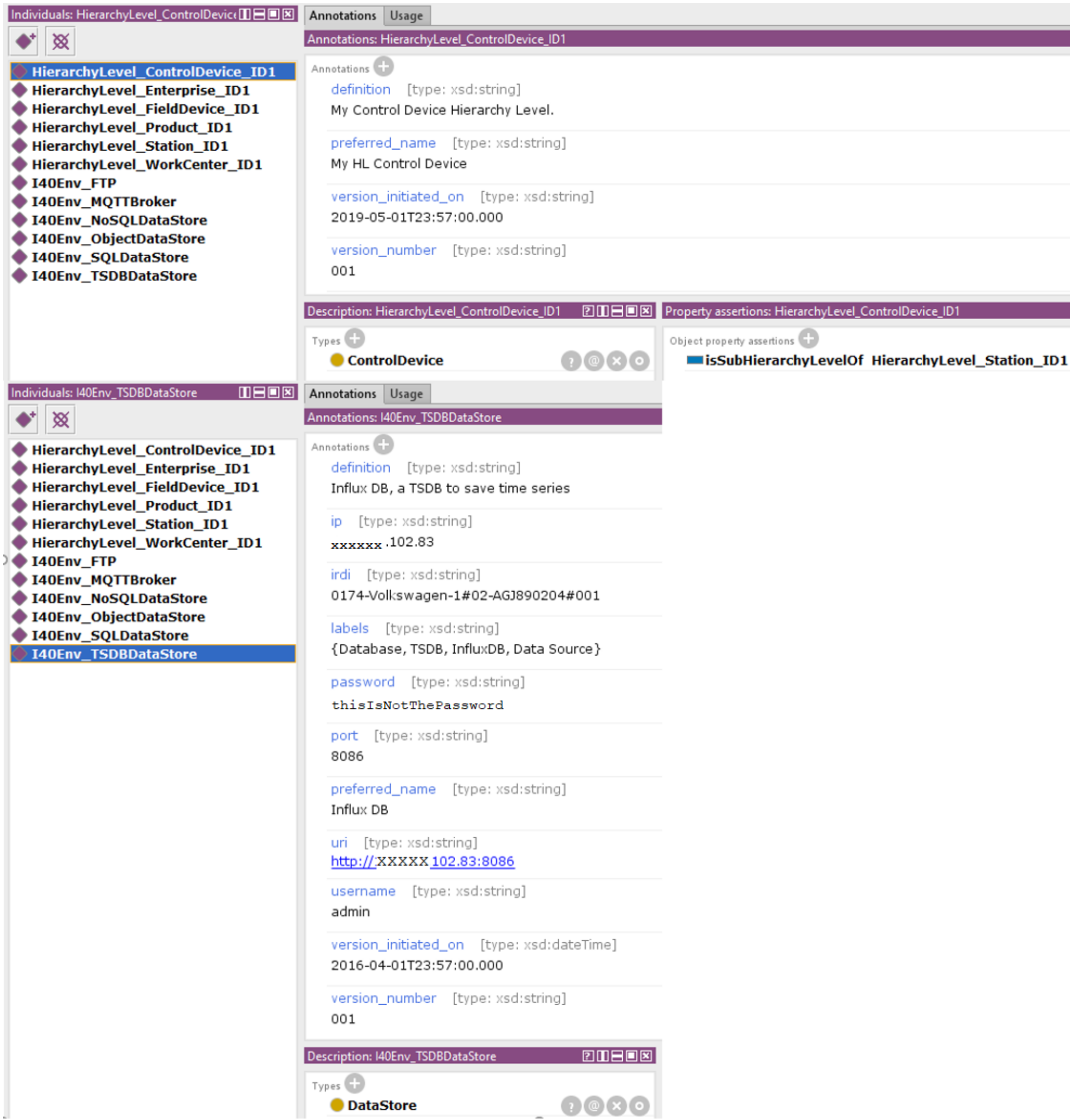


Figure 5-21: Exemplary Smart Manufacturing Environment instantiation

5.2.5 Data Selector

Figure 5-22 shows an overview of the Data Selector implementation. The Data Selector schema is used by individuals for its description.

Figure 5-22: Data Selector Structure

Schema – Data Selector

The Data Selector schema has the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/data-selector-schema` and imports the vocabulary collection. The schema describes classes and object properties to be used as basis for the instantiation of a Data Selector as shown in Figure 5-23.

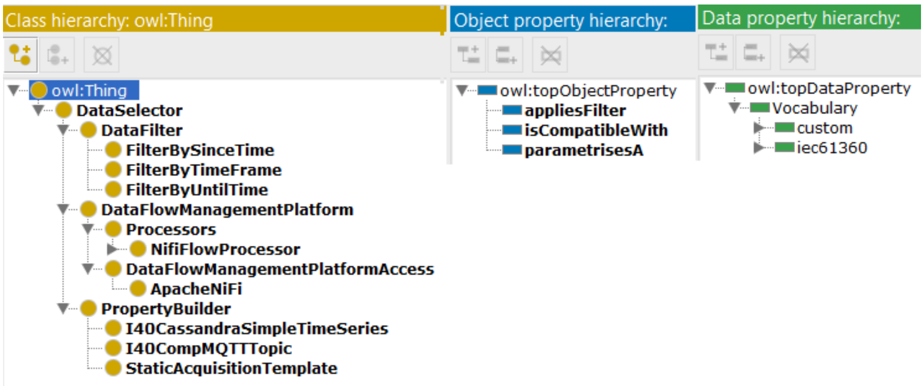


Figure 5-23: Data Selector schema - Classes, Object Properties and Data Properties

The **class taxonomy** shows the class *DataFlowManagementPlatform* which describe the platform used for the implemented approach. The sub-class *DataFlowManagementPlatformAccess* is the concept for individuals that describe the access details as the IP, Port and authorisation to a data flow management platform. The platform used in this doctoral work as basis is Apache NiFi. The sub-class *Processors* describe available data source processors of the data flow management platform. A list of available Apache NiFi Flow Processors is imported in this work. As described in section 4.3.1.2, the Data Source Bundle includes Filters and Property Builders. Both concepts are represented in the sub-classes *DataFilter* and *PropertyBuilder*. Relations between these classes are engineered according the description in section 4.3.1.2 using the **object properties** *appliesFilter*, *isCompatibleWith* and *parametrisesA*. Shown **data properties** are coming from the imported vocabulary collection.

Instantiation – Data Selector

Individuals of the Data Selector instantiate and describe data filters, processors and property builders that are linked to a data source bundle (as described in section 4.3.1.2). The data source bundle is described in the "linkage" namespace that links I40 components, Smart Manufacturing Environments and the Data Selector. This "linkage" namespace is described below.

5.2.6 Linkage - I40 components, Smart Manufacturing Environments and Data Selector

This "linkage" namespace imports individuals of I40 components, Smart Manufacturing Environments and the Data Selector (see Figure 5-24).

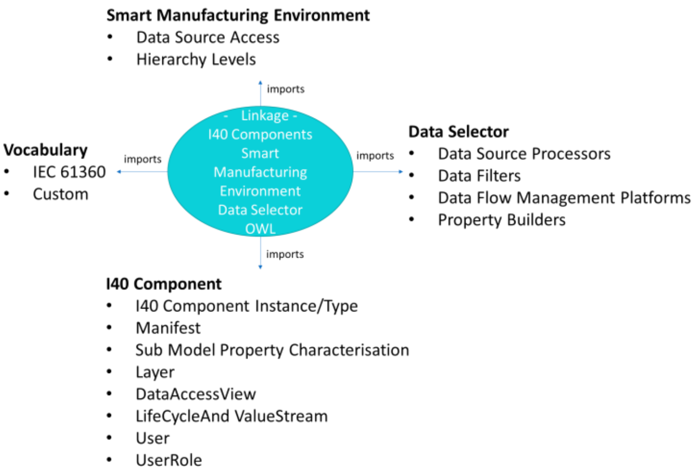


Figure 5-24: Linkage of the semantic model

The link between these individuals is necessary to instantiate a data source bundle. The data source bundle uses the data sources accesses from Smart Manufacturing environments and the processors and property builders of the Data Selector. Furthermore, the linkage is necessary to link I40 component instances to hierarchy levels in a smart manufacturing environment. Therefore, the IRI `http://www.knagorny.de/semantic-models/2021/i40-data-pipelini-nig/linkage-environment-i40comp-dataselector` includes besides the imported individuals also the class *DataSourceBundle* and the object properties

- *isLocatedinHierarchyLevel*
- *usesDataSourceBundle*
- *usesDataSource*
- *usesDataSourceProcessor*
- *usesPropertyBuilder*

An integrated version of the overall semantic model including some example individuals is shown in Figure 5-25. It shows all classes, object properties, data properties, individuals, and their relations caused by the imported individuals and all related indirect imports (compare Figure 5-26 and Figure 5-27).

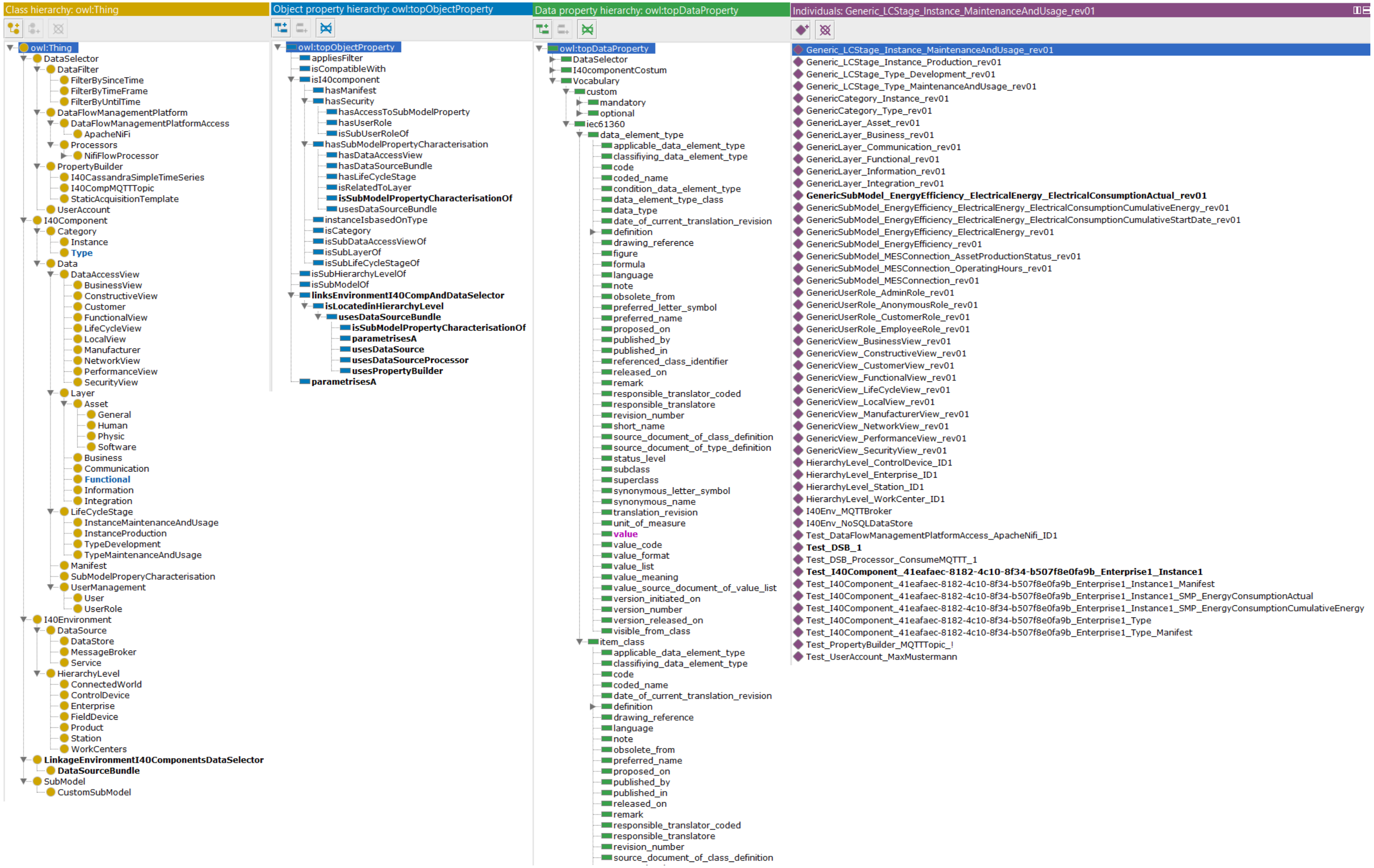


Figure 5-25: Integrated semantic model including some example individuals

Imported ontologies:

Direct Imports

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-01>
example-i40-components-01
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-01>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_I40Component_Enterprise_Examples01.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01>
example-i40-components-industryrobot-01
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_I40Component_IndustryRobot_Examples01.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01>
example-i40-components-vwgolf8-01
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_I40Component_Golf8_Examples01.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-workstation-01>
example-i40-components-workstation-01
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-workstation-01>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_I40Component_WorkCell_Examples01.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01>
individual-data-selector-example01
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_DataSelector_Example01.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01>
individuals-i40-environment-example01
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_I40Environment_Examples01.owl

Indirect Imports

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/sub-model-schema>
sub-model-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/sub-model-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_SubModel.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/vocabulary-schema>
vocabulary-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/vocabulary-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_Vocabulary.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary>
linkage-vocabulary
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Linkage-Vocabulary.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/sub-model-schema>
sub-model-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/sub-model-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_SubModel.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema>
i40-component-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_I40Component.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema>
i40-component-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_I40Component.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema>
i40-component-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-component-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_I40Component.owl

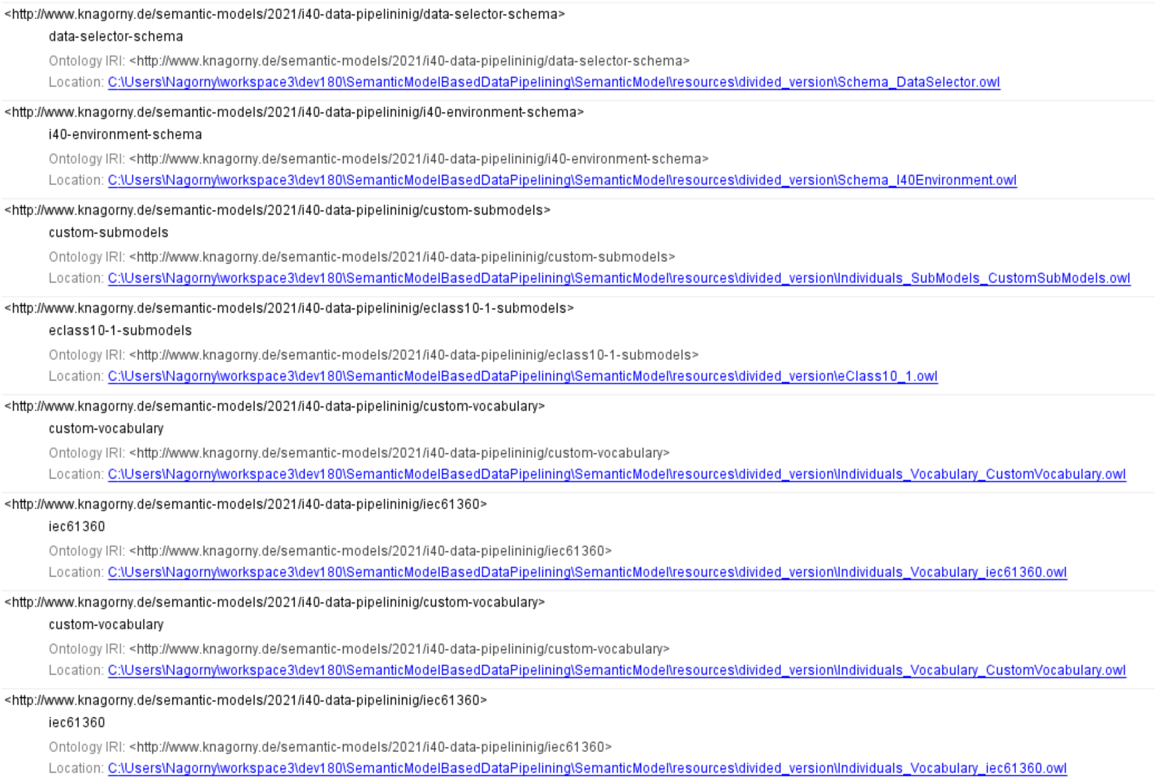
<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-submodels>
linkage-submodels
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-submodels>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Linkage-SubModels.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary>
linkage-vocabulary
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Linkage-Vocabulary.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary>
linkage-vocabulary
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-vocabulary>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Linkage-Vocabulary.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/vocabulary-schema>
vocabulary-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/vocabulary-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_Vocabulary.owl

Figure 5-26: Direct and indirect imports (1/2)



<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/data-selector-schema>
data-selector-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/data-selector-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_DataSelector.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-environment-schema>
i40-environment-schema
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/i40-environment-schema>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Schema_I40Environment.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-submodels>
custom-submodels
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-submodels>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_SubModels_CustomSubModels.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/eclass10-1-submodels>
eclass10-1-submodels
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/eclass10-1-submodels>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\eClass10_1.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-vocabulary>
custom-vocabulary
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-vocabulary>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_Vocabulary_CustomVocabulary.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/iec61360>
iec61360
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/iec61360>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_Vocabulary_iec61360.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-vocabulary>
custom-vocabulary
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/custom-vocabulary>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_Vocabulary_CustomVocabulary.owl

<http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/iec61360>
iec61360
Ontology IRI: <http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/iec61360>
Location: C:\Users\Nagorny\workspace3\dev180\SemanticModelBasedDataPipelining\SemanticModel\resources\divided_version\Individuals_Vocabulary_iec61360.owl

Figure 5-27: Direct and indirect imports (2/2)

A full documentation of the implemented semantic model schema, generated with the tool "WIzard for DOCumenting Ontologies (WIDOCO)" (Garijo, 2017), is referenced in the appendix section 9.3.

5.3 Architecture implementation

The implementation of the architecture based on the specification described in section 4.3.2 is described in this section. Each module is described in detail in the following subsections.

5.3.1 Semantic Manager module

The Semantic Manager module as part of the overall architecture is used to perform typical CRUD operations on the knowledge graph based on the semantic model. As a basis, it uses the Apache Jena Framework that provides a range of sub-systems, suitable to manage smart manufacturing environments, I40 components and the Data Selector inside the data model.

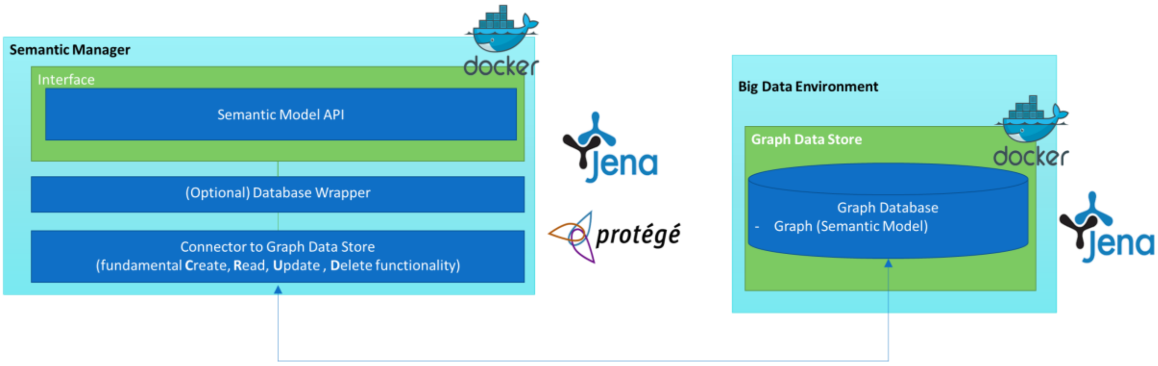


Figure 5-28: Semantic Manger module

Figure 5-28 shows a part of the overall architecture which addresses the Semantic Manager. The graph data store in the Big Data environment stores the knowledge graph based on the semantic model. This knowledge graph is accessed by a graph data store connector in the Semantic Manger. On the highest level, the Semantic Manager module provides a semantic model API interface. It can be used by other modules of the architecture, mainly through the Data Selector module.

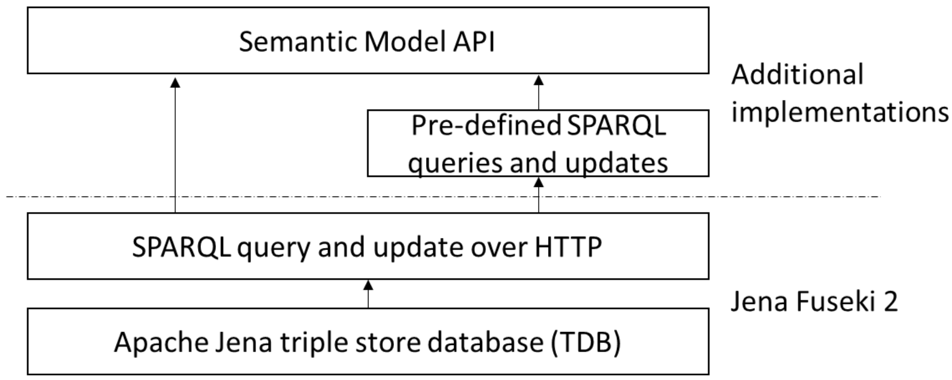


Figure 5-29: Implemented Semantic Manager structure

Figure 5-29 shows the implemented structure of the Semantic Manager. Jena Fuseki 2 is used for the graph data store and a basic semantic model API. It is an element of the Jena Framework and represents a SPARQL 1.1 server with a web User Interface (UI), backed by Apache Jena's Triple Data Base (TDB). This server provides SPARQL update and SPARQL query features using

Jenas SPARQL protocol over HTTP. Jena Fuseki 2 is already available in a containerized version on Docker Hub[15]. The latest version with the digest/identifier *2a65a093b164*[16] was used.

The **Apache Jena triple store database (TDB)**, inside the Jena Fuseki 2 server, is used as a graph database, to store the semantic model in a RDF format. The **SPARQL query and update over HTTP** interface is used for fundamental SPARQL operations.

A range of **pre-defined SPARQL queries and updates** are implemented on top of Jena Fuseki 2. Thereby they create further SPARQL abstractions, provided additionally by the **Semantic Model API**.

```
[001]     public class ManifestInstanceByI40comp extends AbstractSPARQLQueryEngine {
[002]       //...
[003]       private SelectBuilder getQuery(String i40ComponentIndividualName) {
[004]
[005]               SelectBuilder sb2 = getPreparedQuery().clone();
[006]               sb2.addVar("?Manifest");
[007]               sb2.addVar("?dataProperty");
[008]               sb2.addVar("?datatypeValue");
[009]
[010]               sb2.addWhere("?I40Component", "rdf:type", "i40comp:Instance");
[011]               sb2.addWhere("?I40Component", "i40comp:hasManifest", "?Manifest");
[012]               sb2.addWhere("?Manifest", "rdf:type", "i40comp:Manifest");
[013]               sb2.addWhere("?Manifest", "?dataProperty", "?datatypeValue");
[014]
[015]               sb2.addFilter("?I40Component IN (i40comp:" + i40ComponentIndividualName + ")");
[016]               sb2.addFilter("isLiteral(?datatypeValue)");
[017]               sb2.addFilter("!isBlank(?Manifest)");
[018]
[019]               return sb2;
[020]       }
[021]     }
```

Code 4: SPARQL Query – Get Manifest of I40 comp. instance by Individual Name

Code 4 shows an example of a pre-defined SPARQL query using the Jena Query Builder[17]. It gets the manifest of an I40 component instance. In the shown example, each pre-defined query has an own Java class as *ManifestInstanceByI40comp* (see line 001). The pre-defined query is defined in the method *getQuery*. It gets as an input parameter the individual name of the I40 component instance (see line 003). The *AbstractSPARQLQueryEngine* class (see line 001) includes all methods needed to execute the SPARQL query and to return the results.

The SPARQL query and update features provided by Jena Fuseki 2 and the pre-defined queries and updates build the semantic model API of the Semantic Manager.

---

[15] See https://hub.docker.com/r/stain/jena-fuseki

[16] See https://hub.docker.com/layers/stain/jena-fuseki/latest/images/sha256-2a65a093b164be99da3b8f566fdfb5aec1d71ad29496a18d4bda12fffa5a847e

[17] See https://jena.apache.org/documentation/extras/querybuilder/index.html

5.3.2 Data Selector module

As introduced in section 4.3.2, the Data Selector is a search, filter and selection engine to support data understanding and data acquisition. It is accessible through a web-based graphical user interface and provides features to navigate through data along the life-cycle of I40 components located in a smart manufacturing environment. Therefore, it uses the knowledge graph based on the semantic model provided by the Semantic Manager. Searched data can be selected and bundled into a DAC, which is used as configuration for the Data Provider module.

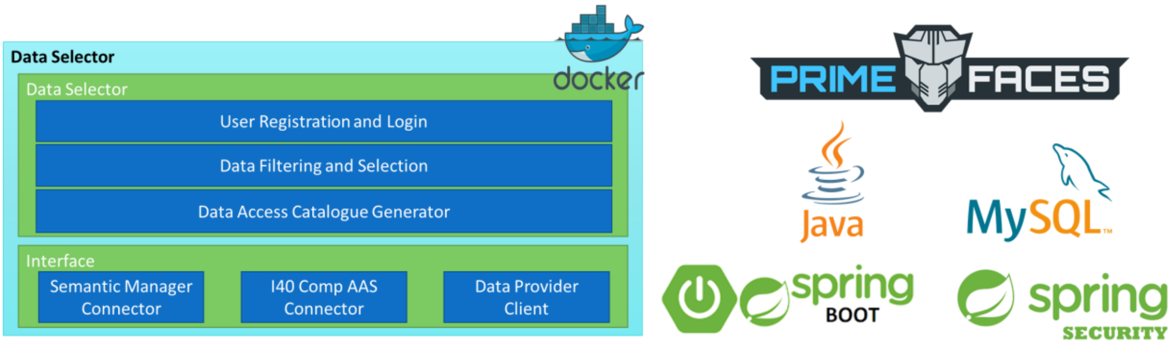


Figure 5-30: Data Selector module and used main technologies

The implementation of the Data Selector is a Java web-application based on JavaServer Faces (JSF), using PrimeFaces. It is supported by the Spring Framework. For data that is not stored in the semantic model, is used a MySQL database. JSF is a component-based Java framework for developing web-based applications. It supports the developer to easily integrate software components, to define their user web-interface and to manage the navigation through the web application. JSF is extended by PrimeFaces. PrimeFaces was the first choice because it provides a library with several pre-built customizable UI components, additional features for the development of web applications[18] and several prepared layouts that enable a fast development. The basis for this doctoral work was the PrimeFaces ULTIMA layout[19]. The Spring framework is used, because it simplifies the development of complex Java applications[20]. In particular Spring Boot was used which simplifies the configuration of the Spring framework and comes with an embedded Apache

[18] See the PrimeFaces show case: https://www.primefaces.org/showcase/

[19] See https://www.primefaces.org/layouts/ultima/

[20] Features overview: https://www.dev-insider.de/was-ist-das-spring-framework-a-829846/

Tomcat webserver, that handles the Hypertext Markup Language (HTML) pages of the web application. And secondly by Spring Security that offers a range of security features to secure a web application (for instance functionalities to configure authentication and authorization and provides an integrated attack protection)[21].

5.3.2.1 *Application layers*

Figure 5-31 shows the layers in which the Data Selector web-application is implemented. Each layer is described bottom-up below.

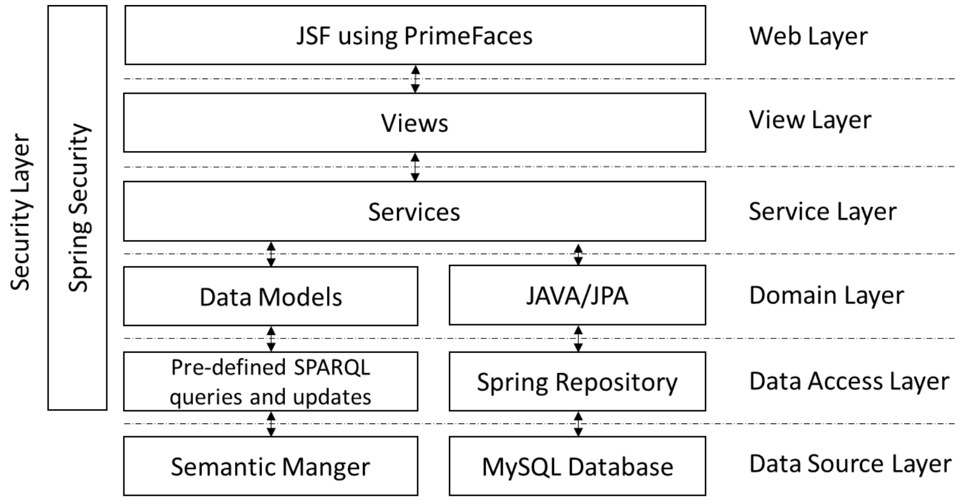


Figure 5-31: Layers of the Data Selector

- **Data Source Layer** – Describes data sources used by the Data Selector. The Data Selector has two data sources:
  - The 1st data source is the *Semantic Manager* which provides the semantic model. It provides all information available in the semantic model and enables also manipulations.
  - The 2nd data source is a MySQL database which stores all information that are not handled by the semantic model (for instance configurations, user credentials and data access catalogues).

[21] See https://spring.io/projects/spring-security

- **Data Access Layer** – Describes the functionalities to access data sources in the data source layer.
    - To access the Semantic Manager, the Apache Jena ARQ is used. It is a SPARQL query engine for Jena[22]. In this layer, pre-defined SPARQL queries are also defined. They are used for searching and filtering data in the semantic model. The approach is open to add further queries for extending the filtering and searching functionalities of the Data Selector.
    - To access the MySQL database, the repository functionality of the Spring framework to implement data accesses for various persistence data stores[23] is used.
- **Domain Layer** – Describes domain entities that are building blocks of data, used by the Data Selector. For example: One entity could be a SMPC of an I40 component as shown in Code 5. These entities are implemented for data located in the semantic model and data located in the MySQL database.
- **Service Layer** – Represents the business logic that is implemented inside the Data Selector in form of internal services. Services represent functionalities that are called indirectly in the backend by the user in the upper layers. Such functionalities/services are for instance: user registration, data filtering, data selection, data access catalogue generation, deployment of data source processors in the Data Provider, etc.
- **View Layer** – Describes views that are the logical counterpart of web pages in the web layer. They bundle needed services of the service layer and provide functions that are linked with pages on the web layer to provide data and to handle user requests.
- **Web Layer** – Describes the HTML web pages that the semantic Data Selector users see in a rendered form in the web browser. The Data Selector implements three main pages: Login, Data Searching and selection, data filtering and DAC deployment.
- **Security Layer** – Describes basic technology features manged by Spring security. Spring security is used to implement the user authentication.

---

[22] See https://mvnrepository.com/artifact/org.apache.jena/jena-arq

[23] See https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html

```
package de.atb.dataselector.model;
//imports...

public class SubModel {
    private String irdi;
    private String preferredName;
    private String definition;
    private String LifeCycleStage;
    private String DataAccessView;
    private int  versionNumber;
    private Date versionDate;
    private String Layer;
    private String categorySearchSubModelcolumnTwoDescription;
    private String individualName;
    private String structuralElement;
    private String valueType;
    private List<String> labels;
    private Map<String, String> additionalDataProperties;

    //getter & setter...
}
```

Code 5: Entity Example in the Domain Layer

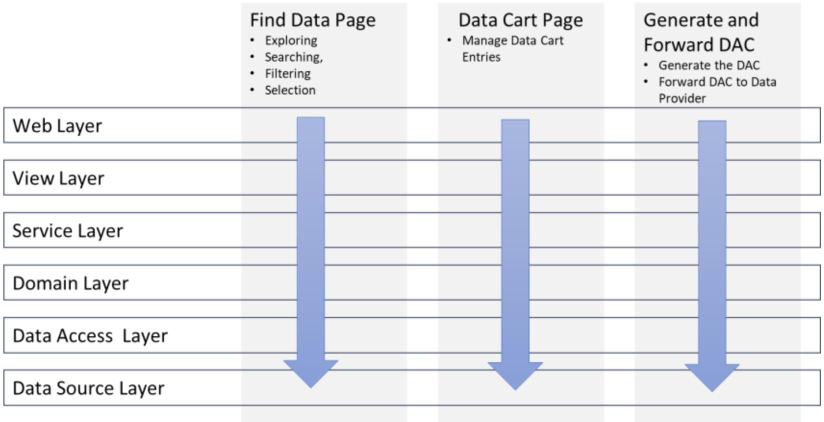


Figure 5-32: Description top-down

The next sections describe the implemented key functionalities across the introduced layers. As visualised in Figure 5-32, these sections will be described top-down.

5.3.2.2 *Find Data: Exploring, Searching, Filtering and Selection*

This section describes the implemented functionalities for the searching, filtering and selection of data. The description is structured top-down according to the introduced layer model.

On the Web & View Layer

On the web layers the visual web-pages used by the user are implemented. The view layer represents the logical counterpart of the web-page.

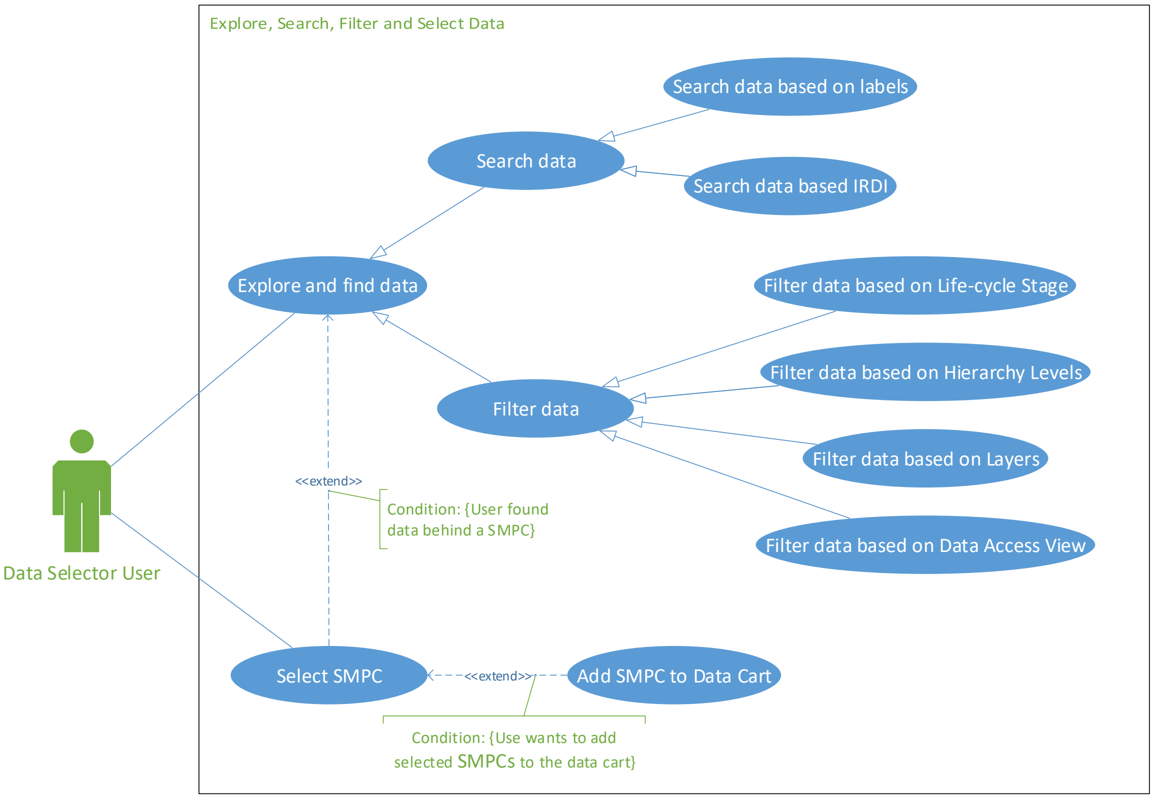


Figure 5-33: Use cases for Data Searching, Filtering and Selection

Figure 5-33 shows the implemented use cases for the web-page for data exploration, searching, filtering and selection. The web-page offers basic search and filter functionalities to query data inside the semantic model. The figure shows some combinable basic search and filter options. Search and filter results are presented in a hierarchical structured tree-table[24] which is divided into three levels: I40 component --> Sub Model --> Sub Model Property Characterisation. If the Data Selector user finds needed SMPCs, the web-page offers their selection. The selection of SMPCs can be added to the data cart if the user wishes to do so.

Service Layer

On the service layer the *MarketplaceDocumentService* is located. It takes all the search and filtering options and returns the results. Those results are the data for the tree-table on the web layer described before.

[24] See https://www.primefaces.org/showcase/ui/data/treetable/selection.xhtml

```
[001]     @Named("marketplaceDocumentService")
[002]     @ViewScoped
[003]     public class MarketplaceDocumentService implements Serializable {
[004]       //...
[005]       public TreeNode createCheckboxDocumentsForCategorySearch() {
[006]         List<String> lifeCycleAll = new ArrayList<>();
[007]
[008]         lifeCycleAll.addAll(
[009]            filterCategorySearchSelectManyView.getSelectedOptionsLifeCycleType()
[010]         );
[011]         lifeCycleAll.addAll(
[012]            filterCategorySearchSelectManyView.getSelectedOptionsLifeCycleInstance()
[013]         );
[014]
[015]         ResultSet result = sqCategorySearch.executeQueryCategorySearch(
[016]                 lifeCycleAll,
[017]                 filterCategorySearchSelectManyView.getSelectedOptionsHierarchyLevels(),
[018]                 filterCategorySearchSelectManyView.getSelectedOptionsLayers(),
[019]                 filterCategorySearchSelectManyView.getSelectedDataAccessViews()
[020]         );
[021]         return fillTreeBasedOnResultSet(result);
[022]       }
[023]       //...
[024]     }
```

Code 6: Snippet: Get filter results for the tree-table in the web layer

Code 6 shows a snippet in the *MarketplaceDocumentService*. Line 9, 12, and 17-19 show the filter parameters located in the view layer. These data is used to parametrises a SPARQL query which is located in the lower layers (see line 15). The returned result is used to fill out the tree-table in the view layer. The tree-table is visualised on the web layer. Found SMPCs can be selected and added to the data cart.

Domain and Data Access Layer

In the data access layer, the SPARQL queries using the Jena API to execute the query are located. As the snippet Code 6 in line 15 shows, the *executeQueryCategorySearch* method in the class *SparqlQueryCategorySearch* gets as parameters all defined filtering parameters. Inside this method is SPARQL query builder that built the query according to the filter parameters and collects all needed variables (see Code 7). The query result will be returned as *org.apache.jena.query.ResultSet*[25] data model in the domain layer.

[25] See https://jena.apache.org/documentation/javadoc/arq/org/apache/jena/query/ResultSet.html

```
// Define variables to be queried
sb2.addVar("?class");
sb2.addVar("?I40Component");
sb2.addVar("?irdi");
sb2.addVar("?name");
sb2.addVar("?description");
sb2.addVar("?instanceIsbasedOnType");
sb2.addVar("?SubModel");
sb2.addVar("?irdi");
sb2.addVar("?preferred_name");
sb2.addVar("?isSubModelOf");
sb2.addVar("?SubModelPropertyCharacterisation");
sb2.addVar("?LifeCycleStage");
sb2.addVar("?DataAccessView");
sb2.addVar("?Layer");

//Variables that are constructed for the tree-table
sb2.addVar("?columnOneI40Component");
sb2.addVar("?columnTwoDescription");
sb2.addVar("?columnThreeType");
sb2.addVar("?SubModelcolumnOneI40Component");
sb2.addVar("?SubModelcolumnTwoDescription");
sb2.addVar("?SubModelcolumnThreeType");
```

Code 7:Part of the SPARQL query: Variables to be queried

Data Source Layer

The data source layer completes the top-down layers description. The SPARQL queries get forwarded to the Semantic Manager by using the Apache Jena HTTP based API. The Semantic Manager executes the query and sends the result back to the Data Selector.

5.3.2.3 *Data Cart: Manage data cart entries and configured data filters*

This section describes the implemented functionalities of the data cart page where the DAC entries are managed. The description is again structured top-down according to the introduced layer model.

On the Web & View Layer

Figure 5-34 shows the implemented use cases for the data cart web-page. The web-page offers management functionalities on selected SMPCs that are added to the data cart. The page offers the possibility to delete and preview SMPCs, as well as the configuration of data filters, if available.

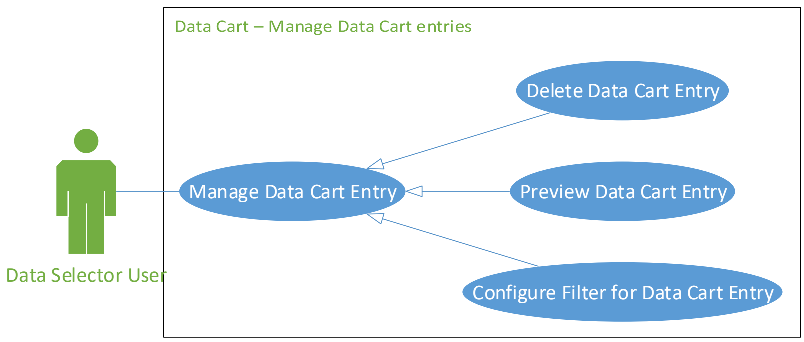


Figure 5-34: Use cases on the Data Cart page visualised on the web layer

The data filters are associated to the filters linked to property builders which were described in the data source bundle section 4.3.1.2. The Data Selector user can refine the needed data here. For instance: Configuration of a time frame in a time series.

The deletion of a data cart entry is a simple database operation on the lower layers. The preview of a data cart entry shows a modal window with all already queried meta data, located in the I40 component manifest, the Sub Model and the SMPC. Both use cases are not strongly related to the key contributions of this work and therefore, they are not described in detail in this section. To keep the focus on the key contributions of this work, this section describes the implementation of the filter configuration.

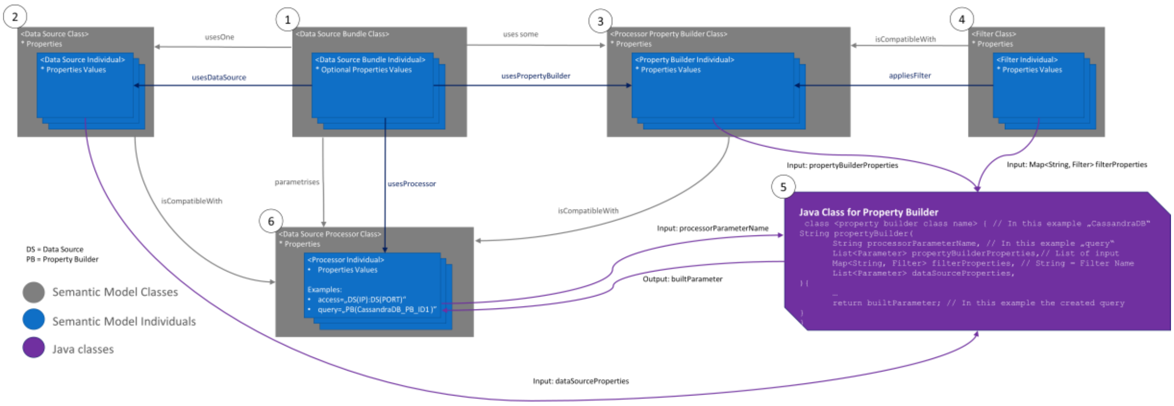


Figure 5-35: Data Source Bundle structure

The concept of the data source bundle was introduced in section 4.3.1.2. As a reminder, the Figure 5-35 shows the structure of the data source bundle again. For the data cart page, the logic for the initialisation of the filters (see number 4 in the figure) is implemented. If a data cart entry/a SMPC is linked to a data source bundle that uses Property Builders compatible with a list of filters, these filters can be enabled and initialised.

```
<p:sidebar id="sidebarFiltersID"
              widgetVar="sidebarFilters"
              position="right"
              dynamic="true"
           style="width: 800px; overflow: scroll; background-color: rgb(247, 247, 247)">

   <h1 style="font-weight:normal">Apply Data Filters</h1>
   <h:panelGroup id="includeContainer">
      <c:forEach items="#{dataCartFilterButtonView.listOfClassNamesOfCompatibleFilters}"
                 var="filterClassName">
         <ui:include src="/dataselector/datasourcefilters/#{filterClassName}.xhtml"/>
      </c:forEach>
   </h:panelGroup>

</p:sidebar>
```

Code 8: Dynamic inclusion of compatible filters

Code 8 shows a snippet for dynamic inclusion of compatible filters on the web layer. On the main page of the data cart, implemented in the datacart.xhtml, the Data Selector user may aim to configure filters for an SMPC. Each filter is rendered in a sidebar[26] (line 1) and there in a panel group (line 8). Filters are loaded based on a *listOfClassNamesOfCompatibleFilters* (line 9 and 11). The class names are equal with class names in the semantic model. This means, each filter class in the semantic model has also a web-layer page in the Data Selector (for instance *filterByTimeFrame.xhtml*) and a view layer class (for instance *FilterByTimeFrameView.java*). The collection of compatible filters for each SMPC is returned by the service layer.

Service Layer

```
[001]     public List<String> getListOfClassNamesOfCompatibleFilters() {
[002]        //...
[003]        // Get List of Property Builders
[004]        List<String> pbIndividualNames = new ArrayList<>();
[005]        ProcessorModel processorModel =
[006]           dataSourceBundleService.getProcessor(DSBundleIndividualName);
[007]        for (Map.Entry<String, LiteralModel> lm : processorModel.getAllLiterals().entrySet()) {
[008]           String builtParameter = lm.getValue().getValueAsString();
[009]           String[] pbTemplateList = StringUtils.substringsBetween(builtParameter, "PB(", ")");
[010]           if (pbTemplateList != null) {
[011]              for (String dapbIndividualName : dapbTemplateList) {
[012]                 pbIndividualNames.add(pbIndividualName);
[013]              }
[014]           }
[015]        }
[016]        //Get class names of property builder individuals
[017]        List<String> pbClassNames =
[018]           dataSourceBundleService.getPBClassNamesByIndividualNames(pbIndividualNames);
[019]
[020]        //Identification of compatible filter class names
[021]        List<String> compatibleFilterClassNames = new ArrayList<>();
[022]        for (String pbClassName : pbClassNames) {
[023]                //Validate if Java classes are available in Data Selector
[024]                List<PB> dataselectorBPs = dataselectorPBs.getDataSelectorPBs();
[025]                PB foundPB = dataselectorBPs.stream()
[026]                      .filter(entry ->
[027]                            pbClassName.equals(entry.getPBName())
[028]                      )
[029]                      .findAny()
[030]                      .orElse(null);
[031]              // Collect compatible filters from property builders
[032]              compatibleFilterClassNames.addAll(
[033]                 foundPB.getCompatibleFiltersListAsStringFirstCharLowerCase()
[034]              );
[035]       }
[036]       return compatibleFilterClassNames;
[037]     }
```

Code 9: Service layer snippet – Get Compatible Filters

Code 9 shows the acquisition of compatible filters. Each data property of the processor (line 7) in a related data source bundle (line 6) gets analysed if the pattern "PB(<Property Builder Individual Name>)" is included (line 9). Is that the case, the included Property Builder Individual Names are saved in a list (line 12). Based on this list, the related class names of the Property Builder Individual names (line 17 and 18) are identified. Based on the class names, the classes available in the Data Selector are checked through a validation. Compatible filters of used data property builders are collected in line 32-34. Based in this collection, the filters are loaded as explained in Code 8.

[26] See https://www.primefaces.org/showcase/ui/overlay/sidebar.xhtml

Based on needed data properties, the rendered filters are used by the Data Selector user to instantiate them. For example, the *filterByTimeFrameView* filter needs a start timestamp and an end timestamp. The instantiated filter will be linked to the data cart entry and to the related property builder and will be saved in the MySQL database in the domain and data access layer.

Domain and Data Access Layer

In the domain and data access layer are the pre-defined SPARQL queries (see Code 4 for an example) located. They are used to query the data source bundle of data cart entries and data models that store the query results.

This domain also contains the data models for the entries of the data cart and the applied filters of the logged-in user. These are stored in the MySQL database on the data source layer by using simple database CRUD operations.

Data Source Layer

The SPARQL queries gets forwarded to the Semantic Manager using the Semantic Manager API. The Semantic Manager executes the queries and sends the result back to the Data Selector.

The information to be stored in the MySQL database is forwarded by using Spring data repositories.

5.3.2.4 *Generate and forward the DAC to the Data Provider*

This section describes the generation and forwarding of the DAC to the Data Provider used for a deployment of the pre-configured processors. The description is structured top-down according to the introduced layer model again.

On the Web & View Layer

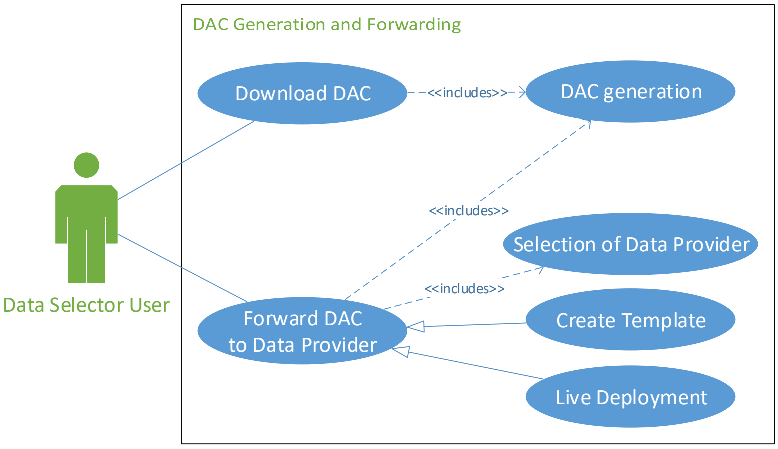


Figure 5-36: Use cases for DAC generation and forwarding to Data Provider

Figure 5-36 shows the implemented use cases for the DAC generation and for the forwarding of the DAC to the Data Provider. The web-page offers functionalities to:

- download the plain DAC as a JSON file
- create a template on the Data Provider side which contains all pre-configured data processors for a later use
- deploy the processors directly in a selected process group[27] which runs on the Data Provider.

To download the DAC, the Data Selector user presses a download button on the web page. On the View Layer a service for the DAC generation is called which creates the DAC, based on all data cart entries and configured filters. The plain DAC is downloadable in a JSON structure and includes for each data cart entry the related I40 component, the sub model, the SMPC and the processor pre-configuration. The pre-configuration is generated based on the data source bundle, as shown in Code 10.

```
{
  "DataAccessCatalogue": {
    "DataEntries": [
      {
        "I40Component": {
                      "I40compIndividualName": "<individual name>",
          "Manifest": {
            "ManifestProperties": {
                      //all manifest data properties and values
            }
          },
          "SubModel": {
            "SubModelProperties": {
                      //all submodel data properties and values
            },
            "SubModelIndividualName": "<individual name>"
          },
          "SubModelPropertyCharacterisation": {
            "SubModelPropertyCharacterisationProperties": {
                      // all SMPC data properties and values
            },
            "SubModelPropertyCharacterisationIndividualName": "<individual name>"
          },
          "DataSourceBundle": {
            "DataSourceBundleIndividualName": "<individual name>",
            "Processor": {
              "ProcessorIndividualName": "<individual name>",
              "ProcessorProperties": {
                      // all pre-configured processor data properties and values
              }
            }
          }
        }
      }
    ]
  }
}
```

Code 10: Structure of the DAC for one data cart entry

To forward the DAC to a Data Provider instance, the Data Selector user has to select the wanted Data Provider instance. The Data Selector will check if the Data Provider is online. If this is

[27] A Process Group is a group of processors (see https://nifi.apache.org/docs/nifi-docs/html/user-guide.html)

the case, the Data Selector user can choose whether to create a DAC template on the Data Provider or to deploy the processors in a process group directly. For a direct deployment, the Data Selector checks process groups on the Data Provider side that are available for a deployment. The basis technology for the Data Provider in this thesis is Apache NiFi. To communicate with the Apache NiFi platform, the Apache NiFi API is used.

The main aspects that are explained in the lower layers are the generation of the DAC and how the DAC gets forwarded to the Data Provider.

Service Layer

The DAC generator is implemented in the java class *GenerateDataConfig.java*. There the manifest, Sub Model and SMPC data properties for each data cart entry are collected by using pre-defined SPARQL queries in the domain and data access layer.

To generate the processor configuration, it has to be analysed weather a data source property mapping needs to be executed and if property builders have to be applied.

The implemented concept for this pattern matching was already introduced in the example of Code 9. The mapping of data source properties to a processor property is a simple replacement. The algorithm in the *GenerateDataConfig.java* replaces a processor property value reference with the syntax *DS(<data source property name>)* with the value behind the reference.

```
/**
 * Note: Property builder class names have to be equal to the Semantic model property
 * builder class names semantic model in order to achieve a correct mapping!
 */
public interface DynPBInterface {
    /**
     * @param processorParameterName: Name of the processor data property
     * @param pbInputParameterList: PB input parameters from PB individual
     * @param dataSourceDescription: input parameters from the data source individual
     * @param filterList: List of applied filters including parameters
     * @return PBParameter: The results as List (a query, a topic, etc.)
     */
    public List<Parameter> builder(
            String processorParameterName,
            List<Parameter> pbInputParameterList,
            DataSourceDescription dataSourceDescription,
            Map<String, DataSourceFilter> filterList
    );


    // Three methods for validation purposes (equalize Semantic model and Java implementation):

    // Returns a list of required input Parameter which are required for the builder
    public List<Parameter> getRequiredPBInputProperties();

    // Returns a list of required input Parameter from the data source
    public List<Parameter> getRequiredDataSourceInputProperties();

    // Returns a list of compatible filter classes
    public List<Class> getCompatibleFilters();
}
```

Code 11: Property Builder Interface

For the creation of a property builder value (syntax: *PB(<property builder individual name>)*), the related property builder java class has to be executed. Implemented property builders in the Data Selector must implement a defined interface shown in Code 11. The *builder(...)*

method of a Property Builder, in which the property building algorithm is implemented, gets the following parameters:

- *processorParameterName* – The data property of the processor, where the syntax PB(<property builder individual name>) has to be replaced by the data property result.
- *pbInputParameterList* – The list of key-value pairs of data properties that are defined in the Property Builder semantic model individual. These key-value pairs are collected by a SPARQL query in the data access layer.
- *dataSourceDescription* – Includes key-value pairs of data properties from the data source bundle linked data source (again collected by a SPARQL query).
- *filterList* – The list of filters that are instantiated in the data cart. Initialised filters are read from the MySQL database.

Next to the builder method three validation methods have to be implemented:

- getRequiredPBInputProperties() - Returns a list of required properties that have to be included in the *pbInputParameterList*. The *pbInputParameterList* has to be compared with the return value of the *getRequiredPBInputProperties()* method. All required parameters have to be included in the *pbInputParameterList*.
- *getRequiredDataSourceInputProperties()* – Returns a list of required properties that have to be included in the data source. The *dataSourceDescription* has to be compared with the return value of the *getRequiredDataSourceInputProperties()* method. All required parameters have to be included in the *dataSourceDescription*.
- *getCompatibleFilters* – Returns a list of compatible filters. All filters in the *filterList* have to be compatible with the property builder.

To build the parameter by the Property Builder, the *builder(…)* method is executed and the result replaces the *PB(<property builder individual name>)* syntax in the related processor data property.

Note: The logic inside the builder method is free for the developer and depends on the use case (for instance: build a query, a MQTT topic or something else).

After the replacement of *DS(<data source property name>)* and *PB(<property builder individual name>)* placeholders with values, the processor pre-configuration is ready.

After these steps are finished for all data cart entries, the DAC generation is done. The DAC is now ready to be downloaded or to be forwarded to the Data Provider. The explanation about how the DAC is forwarded to the Data Provider is following below.

The Data Provider used as template is Apache NiFi. Apache NiFi provides a comprehensive API[28] for a remote access to run NiFi instances. To communicate with the Data Provider, the Data Selector implements the following methods in the class *NifiRestApiController.java*:

- *getClientID()* - To check if the instance is online an if yes, to establish a connection
- *createTemplate()* - To create a template of the DAC in the Data Provider for a later use
- *createProcessor()* - To create NiFi Flow Processors in a Process Group
- *getListOfAllProcessGroups()* - To enable the Data Selector user to deploy the DAC in specific available NiFi process group.
- *getRootProcessGroupID()* - The root process group is the default process group for a live deployment.

This implementation is specific for Apache NiFi, as a basis for the Data Provider. In case that another technology is chosen, an additional Data Provider interface has to be implemented according to the related API.

Domain and Data Access Layer

In the domain and data access layer, pre-defined SPARQL queries to query the data source bundle of related data cart entries and data models that store the query results are located.

Also in this domain the data models for data cart entries and applied filters that relate to the logged in user are located. These models get filled with data read from the MySQL database located in the Data Source layer.

Data Source Layer

The SPARQL queries gets forwarded to the Semantic Manager using Semantic Manager API to execute the query and to send back the result to the Data Selector.

The information to be stored in the MySQL database are read by using Spring data repositories.

[28] See https://nifi.apache.org/docs/nifi-docs/rest-api/index.html

5.3.3 Data Provider Module

The Data Provider module, as part of the overall architecture, is used to deploy pre-configured data source processors for a data extraction. In a sub-sequent step, the data pipeline can be extended by further data cleaning, data transformation and data forwarding operations.

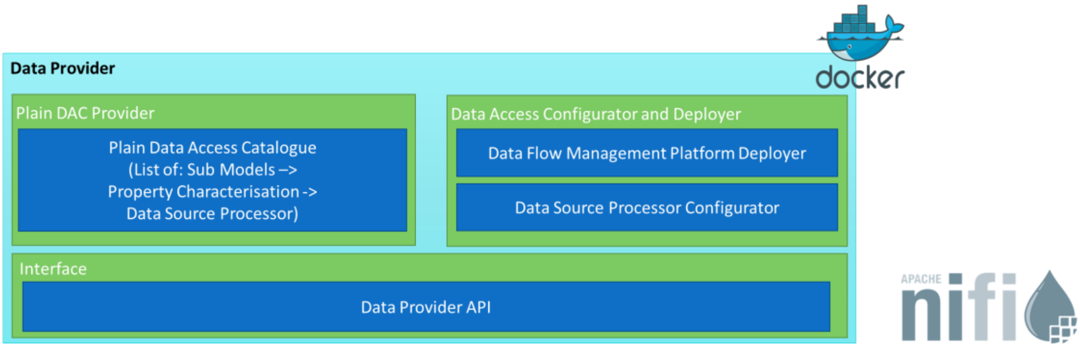


Figure 5-37: Data Selector Module and used main technologies

As basis for the date provider, Apache NiFi is used. It is, as introduced in section 5.1.4, a distributed dataflow engine by the Apache Software Foundation and written in Java.

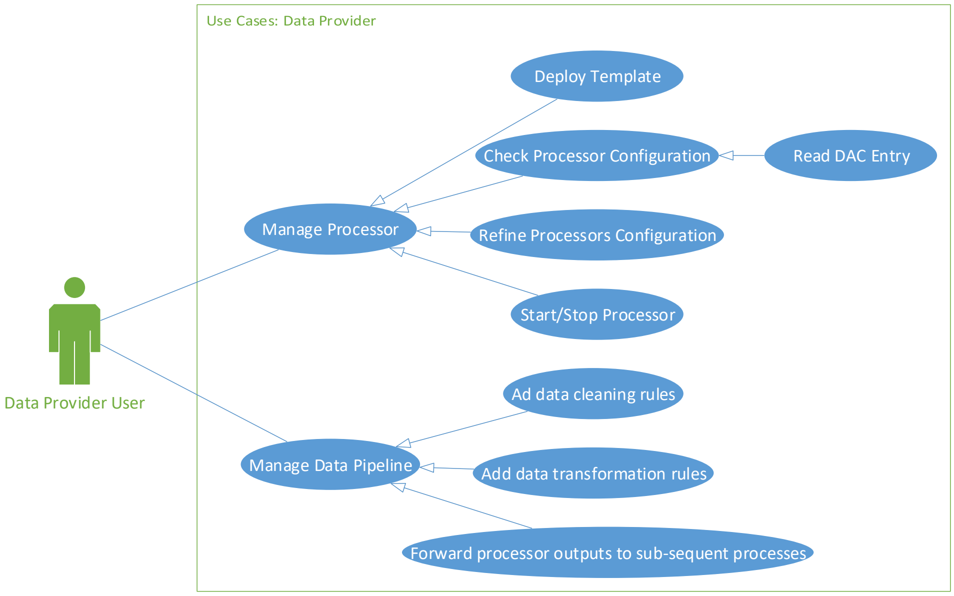


Figure 5-38: Data Provider use cases

Figure 5-38 shows the use cases of the Data Provider, which are covered by the functionality of Apache NiFi.

- Manage Processors
    - Deploy Template: The Data Selector offers the storage of a DAC in the Data Provider as a template. A template combines flow processors into a large building

block. This template can be deployed by the Data Provider user or can be exported as an XML file for sharing with others[29].

  - Check and refine processor configuration: Apache NiFi provides the functionality to check and refine the configuration of a data source processor. The comment section includes the related DAC entry.
  - Start/Stop processor: The deployed processors in a process group can be started, to begin the fetching of data, and can be stopped.

- Manage Data Pipeline: Apache NiFi offers functionalities to refine, extend or reduce the initiated data pipeline by e.g. sub-sequent data cleaning, data transformation and data forwarding operations.

Apache NiFi is available in a containerized version on Docker Hub[30]. The used version has the digest/identifier *c7e2bbb01fc2*[31].

5.3.4 I40 component instance module

The prototype developed in this doctoral work includes a minimal I40 component framework for a proof of concept.

In relation to this doctoral work, an I40 component instance is able to produce data (streams) and to change their state (for instance a changed GPS based location). Therefore, the implemented I40 component instance framework includes functionalities to create services, which are used to produce data. It also provides functionalities to update its part in the semantic model, by using the Semantical Manager.

The framework has to be configured with the I40 component IRDI. The framework fetches the information saved in the semantic model about the I40 component instance and provides this information in an AAS REST API interfaces with hypermedia (Jakubetz, 2011) support.

---

[29] See https://cwiki.apache.org/confluence/display/NIFI/Example+Dataflow+Templates

[30] See https://hub.docker.com/r/apache/nifi/

[31] See https://hub.docker.com/layers/apache/nifi/latest/images/sha256-c7e2bbb01fc2af650ae1df53a21eebe121c6676c4fedff933c0986694825ebfc

5.4 Generic Testing Scenario

The implemented prototype is tested by a generic testing scenario. Figure 5-39 shows the four main steps of this scenario.

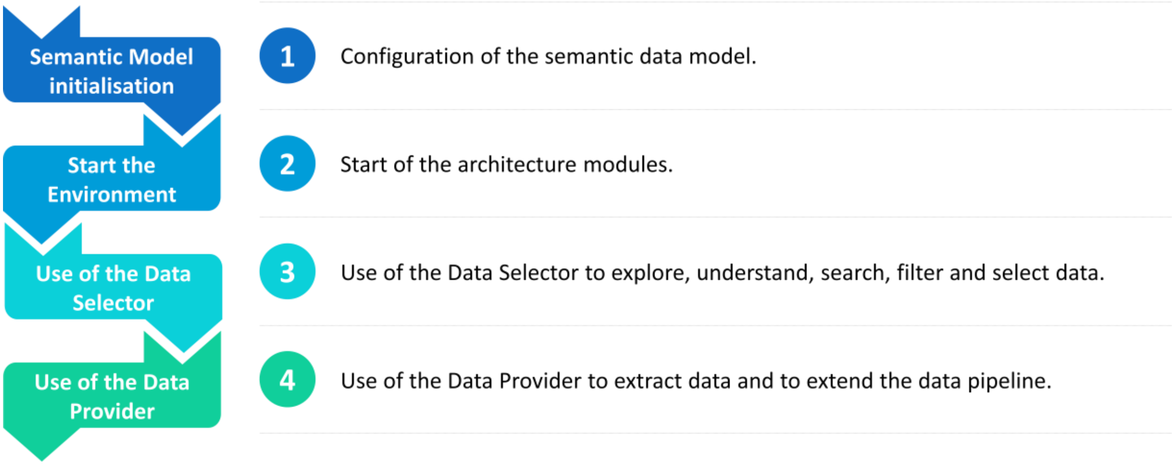


Figure 5-39: Generic testing scenario

Each step is described in detail in the next sub-sections.

5.4.1 Semantic model initialisation

In a first step, the smart manufacturing environment (hierarchy levels and data sources) has to be initialised in the semantic model by creating related instantiations/individuals. The hierarchy level individuals and their relations represent the structure of a smart manufacturing environment, for instance, "Station_ID1 isSubHierarchyLevelOf WorkCenter_ID1". Further, individuals for data sources that are used as central nodes for data storages (e.g. databases) or data distribution (e.g. message brokers) in the smart manufacturing environment, are created.

In a next step, I40 components need to be initialised in the semantic model. This includes the creation of individuals for the I40comp Type/Instance, the related manifest, sub model property characterisations including their relations, and data source bundles, if needed. Created I40 component instances have to be linked to the smart manufacturing environment.

The initialised smart manufacturing environment with all I40 components builds the knowledge graph.

5.4.2 Start the Environment

After initialising the semantic model, the Semantic Manger, Data Selector and Data Provider modules can be started. It is assumed, that data stores, data stream brokers and I40comp instances (if existing) are already online.

First, the Semantic Manager is started and the initial knowledge graph is uploaded. In case the semantic model in the Semantic Manager for I40comp instances is outdated, I40comp instances will update their individuals in the knowledge graph, by using the Semantic Manager API.

Finally, the Data Selector module and the Data Provider can be started.

5.4.3 Use of the Data Selector

After having all modules online, the Data Selector can be used. The user accesses the Data Selector in a web browser and logs in. After this step, the web-interface shows functionalities for data search, filtering and selection:

Search, Filter and Select Data

The user uses the Data Selector to search and select data based on classification classes, context and labels along I40 components life-cycles.

The Data Selector provides multiple combinable search and filtering features. Logically the user sets the filters and the Data Selector uses the Semantic Manager API to execute queries on the knowledge graph. Search results are presented to the user. The user can select needed data (streams) and add them to the data cart. The Data Selector offers also a preview button for an SMPC that can be used to show an overview of defined properties of the related I40 component manifest, Sub Model and SMPC.

Data Cart: Set filters if available & needed

After the user finished the selection of data, it is possible to delete entries or to set additional data filters in the data cart page. It is possible to link multiple (compatible) filters to a Property Builder. For the user it is also possible to enable filters and to set required filter properties. Based on configured data filters, the data selection can be refined (e.g. values of a specific time-frame in a time-series).

Download or preview the generated DAC

After the user has finished setting the filters, it is possible to download or preview the DAC in a JSON format.

Data Provider Deployment

After the selection of data and the configuration of all filters, the final DAC can be generated (including data source processor configurations) and deployed in the Data Provider. The initial prototype uses as a basis Apache NiFi as data pipelining technology for the Data Provider. In a first step a running Apache NiFi instance is selected. The user can create a new NiFi template, to be saved in the selected NiFi instance for a later use; or deploy the pre-configured Data Source Processors live in a selected NiFi Process Group.

5.4.4 Use of the Data Provider

The Data Provider is used to extract the data. In case of a chosen direct data source processor deployment, the user sees all the deployed pre-configured data source processors in the related process group and is able

- to read the DAC entries in the related data source processor comment section
- to extend the data pipeline by adding further data cleaning, transformation or forwarding processors
- to start single processors or the whole workflow on the data flow management platform, for extracting and optionally further clean/transform/forward the data.

Apache NiFi as selected basis technology offers several further data processing, transformation and data forwarding processors to extend the data pipeline according to use case dependent requirements.

5.5 Development Requirements Validation

Section 4.2 defines a range of development requirements for the prototype design. These requirements are validated in this section.

5.5.1 Validation of Data Model Requirements

Table 5-6 validates the non-functional requirements for the semantic data model specified in section 4.2.1.

Table 5-6: Validation on non-functional data model requirements

ID	Data Model requirement description	Validation
Data Model Requirements		
DM_ID1_1	The data model shall be semantic.	The data model is a semantic model created in OWL which describes a smart manufacturing environment and related data in a semantical way.
DM_ID1_2	The semantic data model shall be compliant to DIN-SPEC-91345.	The semantical data model is strongly aligned to DIN-SPEC-91345 and enables the description of I40 components according to this standard.
DM_ID1_3	The semantic data model shall allow the integration of eCl@ss classification classes.	The implemented semantical data model shows successfully the integration of eCl@ss classification classes.
DM_ID1_4	The semantic data model shall use IEC 61360-based properties as basis for the used vocabulary.	The schema of the semantical data model includes a vocabulary according to IEC 61360.
DM_ID1_5	The semantic data model shall allow the classification and description of data along the life-cycle of I40 components.	The semantical data model allows the description of data related to I40 components, along their life-cycle. The model allows also the classification of this data.
DM_ID1_6	The semantic data model shall allow the embedding of I40 components into a smart manufacturing environment based on hierarchy levels according to RAMI 4.0.	The semantical data model allows the description of a smart manufacturing environment and the embedding of I40 components.

DM_ID1_7	The semantic data model shall allow the description of all parameters needed in a data pipelining approach to access and extract the data.	The semantical data model allows to describe parameters needed for the pre-configured connector deployment in a data pipelining approach according to the data source bundle concept.
DM_ID1_8	The semantic data model shall be open for the integration of other semantic model-based standards.	The semantical data model is open for the integration of other semantic model-based standards.

5.5.2 Validation of Software Architectural Requirements

Table 5-7 validates the non-functional requirements for the software architecture defined in section 4.2.2.

Table 5-7: Validation on non-functional architectural requirements

ID	Software Architecture requirement description	Validation
Data Requirements		
AR_ID1_1	The architecture shall allow the linkage of data, produced along the whole life cycles of products and assets to generate a digital thread.	The designed solution allows to categorize and nest data, based on defined life-cycle stages and value streams, which link data, produced by Industrie 4.0 components and generates a digital thread.
AR_ID1_2	The architecture shall allow to handle Big Data according to the three V model.	The designed solution allows to categorize and describe Big Data, by using the semantic model, used by the Data Selector. Furthermore, to acquire Big Data according to the three-V model using the Data Provider.

Requirements coming from influencing (reference) architectures		
AR_ID2_1	The architecture shall be compliant to the RAMI 4.0 described in DIN-SPEC-91345.	The semantic model of the designed solution is strongly related on DIN-SPEC-91345 and covers all mayor aspects described in this standard (see section 6.1.1).
AR_ID2_2	The architecture shall allow the integration standards like eCl@ss for data classification.	The designed solution shows that eCl@ss can be integrated basically and used in the proposed semantic model as sub models (see section 5.2.2).

5.5.3 Validation of Functional Requirements

Table 5-8 validates the functional requirements defined in section 4.2.3.

Table 5-8: Validation of functional requirements

ID	Functional requirement description	Validation
Related to data		
FR_ID1_1	The prototype shall provide functionalities to classify heterogenous data in a semantical way, considering latest standards coming from the Industrie 4.0.	The prototype allows to classify heterogenous data in a semantical way using the proposed semantic model. The developed semantic model considers latest standards coming from the Industrie 4.0. Concerning the semantic model aspects related to the classification and description of data, among others, the standard eCl@ss (Artur Bondza, Christian Eck, Roland Heidel, Markus Reigl, & Dr. Sven Wenzel, 2018) in combination with IRDI (ISO/IEC, 2015) as identifiers was considered. Further, IEC 62890 (International Electrotechnical Commission

		(IEC), 2016a) was considered, because it describes requirements for life-cycle management of systems and products used in industrial-process measurement, control and automation.
Related to data provision		
FR_ID2_1	The prototype shall provide functionalities to find and identify data in heterogenous data environments.	The Data Selector provides functionalities for users to find and identify data in heterogenous Big Data environments.
FR_ID2_2	The prototype shall provide functionalities to select found/identified data as preparation for further sub-sequent processing.	The Data Selector provides functionalities for users to select found/identified data.
FR_ID2_3	The prototype shall provide functionalities to filter selected data as preparation for further sub-sequent processing.	The Data Selector provides functionalities for users to further filter data, if compatible filters were defined (e.g. a time-frame in a time series) using the property builder approach.
FR_ID2_4	The prototype shall provide functionalities to access and extract selected data as preparation for further sub-sequent processing.	The deployment of pre-configured data extraction processors in the Data Provider enables to access and acquire selected data as preparation for further sub-sequent processing.

5.6 Testing in an industrially relevant smart manufacturing environment

The prototype was tested according to the generic testing scenario in section 5.4 in an industrially relevant smart manufacturing environment. The test provided a collection of data and results suitable for the subsequent assessment and interpretation in section 6.

Note: User names, passwords and IPs were anonymised.

5.6.1 The smart manufacturing test environment

Table 5-9 presents the deployed prototype modules running as Docker containers.

Table 5-9: Deployed Architecture Core Modules

Architecture module	Technology Container	Description
Data Selector	Data Selector	The Data Selector to explore, find and select data.
Data Provider	Apache NiFi	An Apache NiFi instance acting as Data Provider.
Semantic Manager	Jena Fuseki	A Jena Fuseki instance acting as Semantic Manager.

In the test environment different types of data sources were deployed that are described in Table 5-10.

Table 5-10: Used Data Sources and Message Brokers

Data Sources and Message Brokers	Description
Apache Cassandra	A NoSQL database as database for documents.
HiveMQ	A MQTT Broker for a publish-subscribe based message exchange.
MySQL	A RDBMS as conventional database system.
InfluxDB	A TSDB to store time series.
REST Webservices	Some REST interfaces to simulate typical webservices as data source.
FTP	An FTP server to store binary files.

Although many other technologies for storing and distributing data are available, the selected technologies consider some of the mayor technologies used in modern and older manufacturing environments.

5.6.2 Testing and Result Collection

The testing procedure followed the generic testing scenario described in section 5.4. It includes the semantic model initialisation, the start of the environment, the use of the Data Selector and the use of the Data Provider.

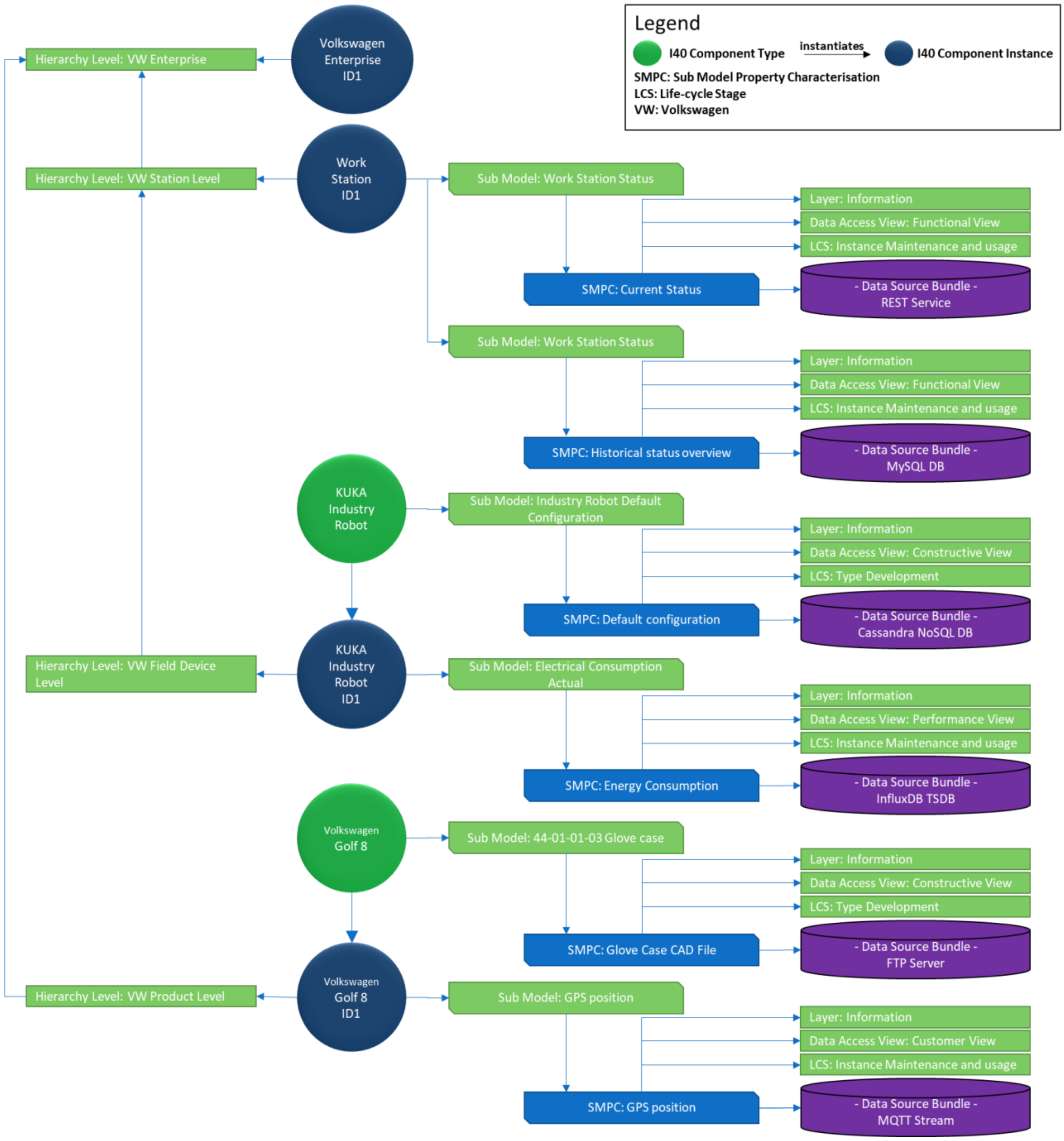


Figure 5-40: Use Cases: Multiple instantiated I40 component

5.6.2.1 Semantic model initialisation

For the testing a test configuration was set up, to build a smart manufacturing environment. This smart manufacturing environment contains a range of I40 components. Each I40 component represents one use case.

Figure 5-40 shows the different I40 components instantiated for the testing and collection of results. The I40 components are related to the Volkswagen AG and the KUKA AG and are typical I40 components in a smart manufacturing environment.

The data produced along the I40 components life-cycles is stored in the data sources located in the smart manufacturing environment, introduced in section 5.6.1. The data source bundle approach is used to link the I40 components with related data inside the data sources.

All I40 components are introduced below. Their full configuration gives the appendix in section 9.5.1.

I40 component: Volkswagen Enterprise ID1

The Volkswagen Enterprise represents a production plant of the car manufacturer Volkswagen. The I40 component consists a manifest which provides meta-data about the Volkswagen Enterprise. The manifest includes the IRDI, the location, the name, the definition and some labels as shown in Table 5-11.

Table 5-11: Volkswagen Enterprise ID1 – Manifest Properties

Data Properties	*Irdi: "0174-Volkswagen-1#02-AFY497004#001"^^xsd:string* *version_number: "001"^^xsd:int* *gps_latitude: 54.518612f* *gps_longitude: 12.376111f* *definition: "An example Volkswagen manufacturing plant."^^xsd:string* *preferred_name: "Volkswagen Manufacturing Plant"^^xsd:string* *version_initiated_on: "2012-12-31T23:57:00.000"^^xsd:dateTime* *labels: "{Enterprise, Volkswagen, manufacturing, plant}"^^xsd:string*

The Volkswagen Enterprise ID1 is linked/nested into the VW enterprise hierarchy level instance. In this test configuration, the VW enterprise hierarchy level instance is the top-level hierarchy level while other level instantiations belong directly or indirectly to this instance.

I40 component: Work Station ID1

The Work Station ID1 is an I40 component on the station hierarchy level. The work station is described in a manifest and refers to two SMPCs, which are linked to classifying sub models.

The first SMPC provides data about the current status. The SMPC is linked to a custom Sub Model which standardises data about a work station status. For the classification and categorisation, the SMPC is linked to following individuals:

- Layer - Information: The SMPC provides information about the Work Station ID1 I40 component.
- Data Access View - Functional: The current status of the work station is categorised as functional information.
- Life-cycle Stage - Maintenance and Usage: The SMPC provides data in the maintenance and usage life-cycle stage of the Work Station ID1 I40 component.

Finally, the SMPC is linked to Data Source Bundle, which addresses a REST Service that provides information about the current status of the work station. All the relevant information to pre-configure a REST access NiFi Flow Processor, is described in the Data Source Bundle.

The second SMPC provides data about the historical status of the work station. The SMPC is linked again to the custom Sub Model, which standardises data about a Work Station status. For the classification and categorisation, the SMPC is linked to following individuals:

- Layer - Information: The SMPC provides information about the Work Station ID1 I40 component.
- Data Access View - Functional: The historical status of the work station is categorised as functional information.
- Life-cycle Stage - Maintenance and Usage: The SMPC provides data in the maintenance and usage life-cycle stage of the Work Station ID1 I40 component.

Finally, the SMPC is linked to Data Source Bundle which references a table in a MySQL database. The database provides information about the historical status of the workstation. All needed information to pre-configure a database query NiFi Flow Processor is described in the Data Source Bundle.

I40 component: KUKA Industry Robot

Type

The KUKA Industry Robot (Type) is an I40 component which describes the design and specification of a KUKA industry robot. The robot type is described in a manifest and refers in this test setup to one SMPC. The SMPC is linked to the classifying sub model.

The SMPC provides data about the default configuration of the robot. It is linked to a custom Sub Model which standardises data of the default configuration from the industry robot. For the classification and categorisation, the SMPC is linked to following individuals:

- Layer - Information: The SMPC provides information about the KUKA Industry Robot (Type) I40 component.
- Data Access View - Constructive: The default configuration of the industry robot is categorised as information about its construction.
- Life-cycle Stage - Type Development: The SMPC provides data in the Type Development life-cycle stage of the KUKA Industry Robot (Type) I40 component.

Finally, the SMPC is linked to a Data Source Bundle which addresses a key space in a Cassandra NoSQL DB. The Cassandra NoSQL DB provides the default configuration of the KUKA Industry Robot (Type). All the necessary information to pre-configure a Cassandra Query Flow Processor is described in the Data Source Bundle.

Instance

The KUKA Industry Robot ID1 is an instantiated I40 component based on the KUKA Industry Robot (Type). The robot is located in the VW field device hierarchy level which is nested in the VW station hierarchy level. The robot instance is described in a manifest and refers in this test setup to one SMPC which is linked to the classifying sub model.

The SMPC provides data about the actual energy consumption of the robot. It is linked to a custom Sub Model which standardises data about the electrical energy consumption of an industry robot. For the classification and categorisation, the SMPC is linked to following individuals:

- Layer - Information: The SMPC provides information about the KUKA Industry Robot ID1 (Instance) I40 component.
- Data Access View - Performance: The actual energy consumption of the industry robot is categorised as information about its performance.
- Life-cycle Stage – Instance Maintenance and Usage: The SMPC provides data in the maintenance and usage life-cycle stage of the KUKA Industry Robot ID1 I40 component.

Finally, the SMPC is linked to Data Source Bundle which references a measurement in an InfluxDB TSDB. The InfluxDB TSDB saves all received measurements about the actual energy consumption with a timestamp. All the necessary information to pre-configure an InfluxDB Query Flow Processor is described in the Data Source Bundle.

I40 component: Volkswagen Golf 8

Type

The Volkswagen Golf 8 (Type) is an I40 component which describes the design and specification of a VW Golf 8 vehicle. The vehicle type is described in a manifest and refers in this test setup to one SMPC which is linked to the classifying sub model.

The SMPC provides data about the glove case design for a VW Golf 8. The SMPC is linked to an eCl@ss Sub Model which standardises data of a vehicles glove case. For the classification and categorisation, the SMPC is linked to following individuals:

- Layer - Information: The SMPC provides information about the Volkswagen Golf 8 (Type) I40 component.
- Data Access View - Constructive: The design of the glove case is categorised as information about its construction.
- Life-cycle Stage - Type Development: The SMPC provides data in the Type Development life-cycle stage of the Volkswagen Golf 8 (Type) I40 component.

Finally, the SMPC is linked to Data Source Bundle which references a CAD file located in an FTP Server. All the necessary information to pre-configure a GetFTP Flow Processor is described in the Data Source Bundle.

Instance

The Volkswagen Golf 8 ID1 (Instance) is an instantiated I40 component based on the Volkswagen Golf 8 Type. The instance represents a physical VW Golf 8. The vehicle instance is described in a manifest and refers in this setup to one SMPC.

The SMPC provides data about the GPS position of the VW Golf 8. The SMPC is linked to a custom Sub Model which standardises the GPS position data of a vehicle. For the classification and categorisation, the SMPC is linked to following individuals:

- Layer - Information: The SMPC provides information about the Volkswagen Golf 8 ID1 (Instance) I40 component.

- Data Access View - Customer: The GPS position is categorised as information for the customer.
- Life-cycle Stage - Instance maintenance and usage: The SMPC provides data in the instance maintenance and usage life-cycle stage of the Volkswagen Golf 8 ID1 I40 component.

Finally, the SMPC is linked to Data Source Bundle which references a MQTT stream subscribable in the MQTT broker. All the needed information to pre-configure an MQTT consumer Flow Processor is described in the Data Source Bundle.

Each I40 component is configured according to the semantic model, described in section 4.3.1. The configured knowledge graph, based on this semantic model, is used for the initialisation of the Semantic Manager.

5.6.2.2 *Start of the test environment*

The Docker containers for the Data Selector, Data Provider and Semantic Manager are deployed and started.

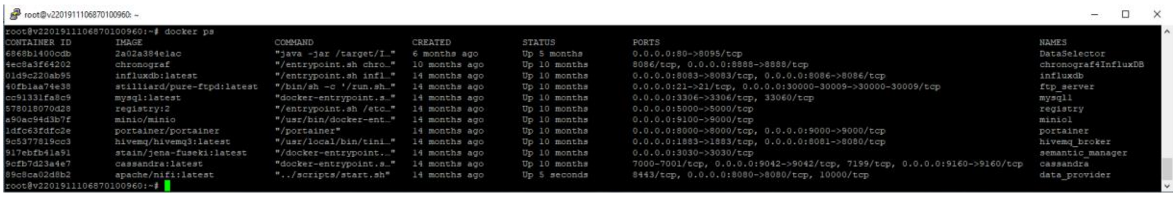


Figure 5-41: Running containers on Docker

Figure 5-41 shows the running containers after the start of the docker environment. The data sources and Industrie 4.0 components are already online.

In a next step the running Semantic Manager gets pre-configured with the initialised knowledge graph based on the semantic model.

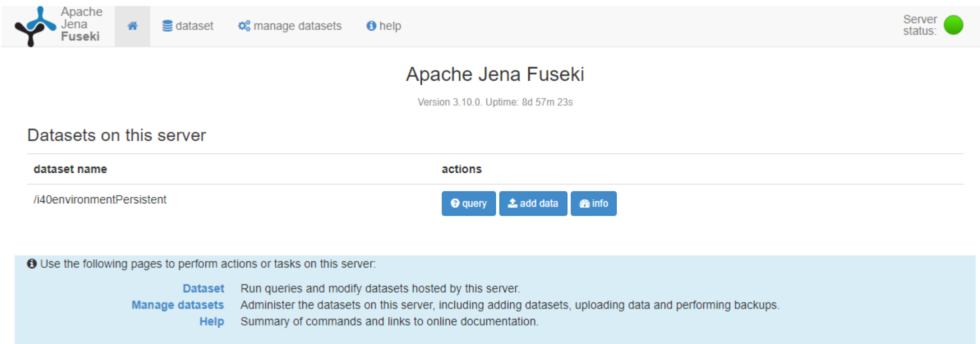


Figure 5-42: Initial Dataset Upload

Figure 5-42 shows the Apache Jena Fuseki instance used as Semantic Manager. The knowledge graph is uploaded as data set with the name *I40environmentPersistant*. This name is also used as endpoint for CRUD operations on the semantic model (http://<serverIP>:<semanticManagerPort>/i40environmentPersistent/).

After this step the initialised prototype is ready to be used.

5.6.2.3 *Test of the Data Selector*

The Data Selector is used to (1st) explore, understand, find and select data, (2nd) to pre-filter data and (3rd) to deploy the data processors in the Data Provider.

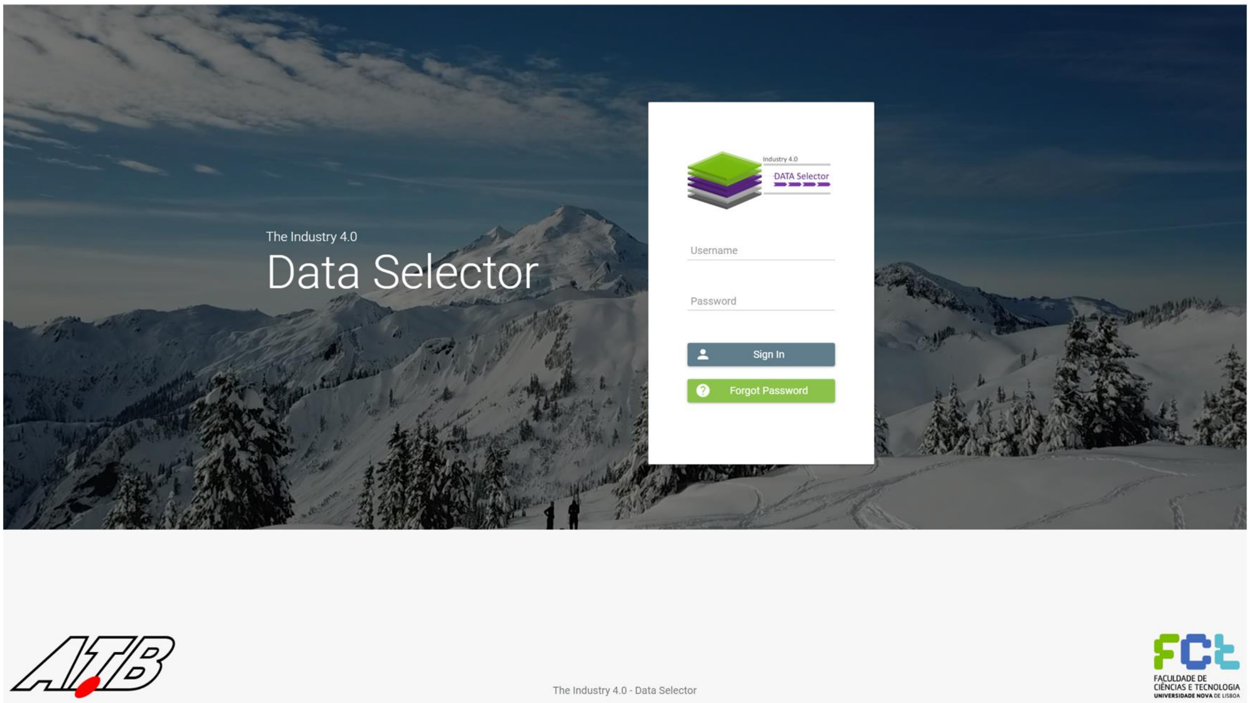


Figure 5-43: Data Selector – Login

Figure 5-43 shows the login procedure. The Data Selector user enters the username and the password and presses the *Sign In* button.

Explore, understand, search, filter and select data

After the login procedure the "Category Search" tab provides various filter options. The filter options are the input for the SPARQL query builder, running as background functionality to search for data using the Semantic Manger API.

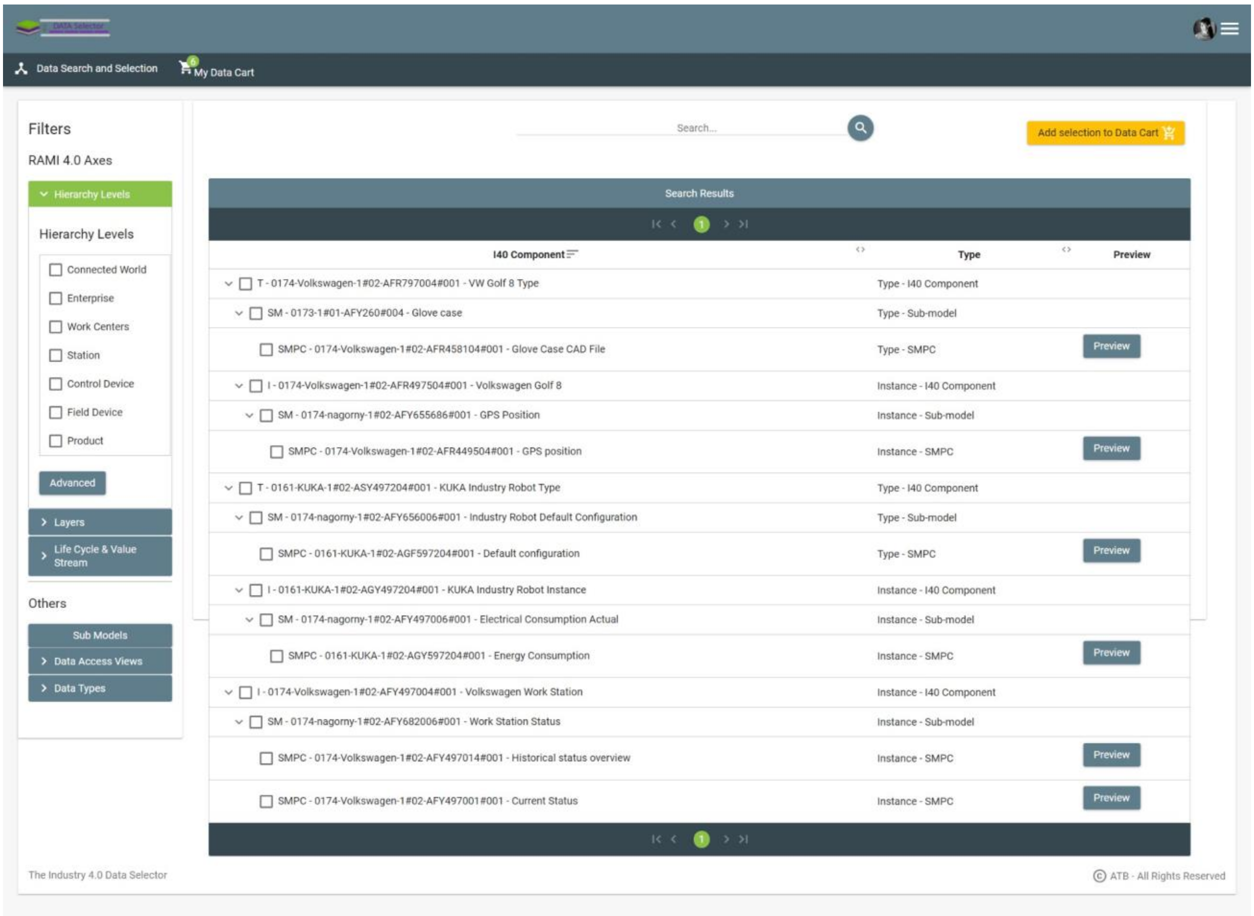


Figure 5-44: Data Selector – Explore, understand, search, filter and select data

Figure 5-44 shows the GUI of the Data Selector. It offers multiple functionalities to simply explore the knowledge graph. Next to the visually shown filters, the test procedure included a range of further comprehensive functionalities like:

- Filtering based on nested data access views
- Filtering based on nested hierarchy levels
- Filtering based on nested life-cycle stages
- Filtering based on sub-model hierarchies
- Filtering based on labels
- Filtering based on used data sources
- Filtering based IRDIs
- Filtering based on creation date

The results show that the semantic model is suitable to find needed data based on multiple (combinable) filter options.

Once the needed data (data source bundles linked to an SMPC) are identified, they can be selected and added to the data cart for further raw data-based filtering, DAC generation and processor deployment in the Data Provider. For the testing all SMPC instantiations are added to the data cart.

Data filtering, DAC generation and processor deployment

After the selection of needed data, the data can be refined further (e.g. if used property builders have compatible filters), the DAC can be generated to prepare the processor deployment in the Data Provider and the processors can be deployed in the Data Provider. In the Data Selector these steps happen in the "My Data Cart" tab.

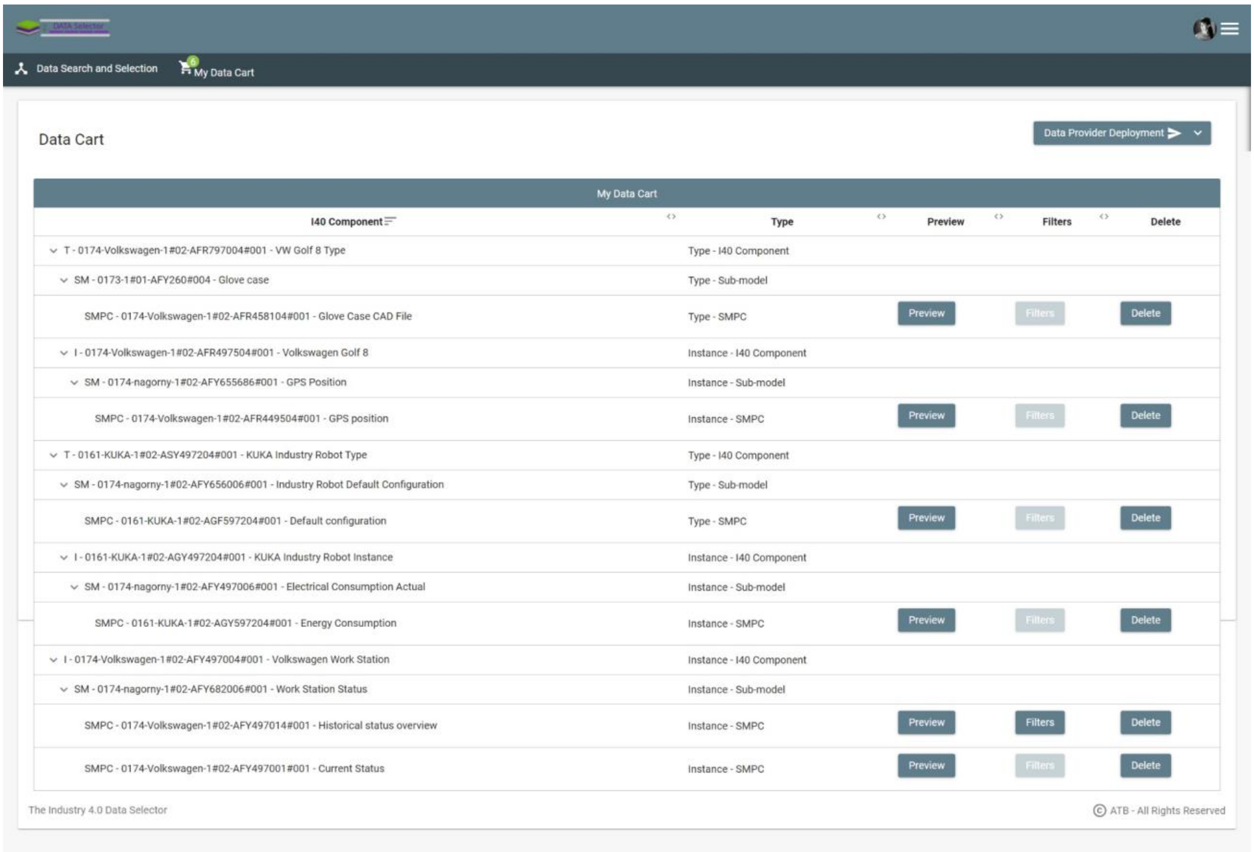


Figure 5-45: Data Cart – All selected SMPCs

Figure 5-45 shows the "My Data Cart" tab which includes all selected SMPCs.

For the testing of the filtering functionality a filter for the *SMPC - 0174-Volkswagen-1#02-AFY497014#001 - Historical status overview* was created. It uses a property builder to generate an SQL query. The related Property Builder has one optional time frame filter which can be configured as shown in Figure 5-46.

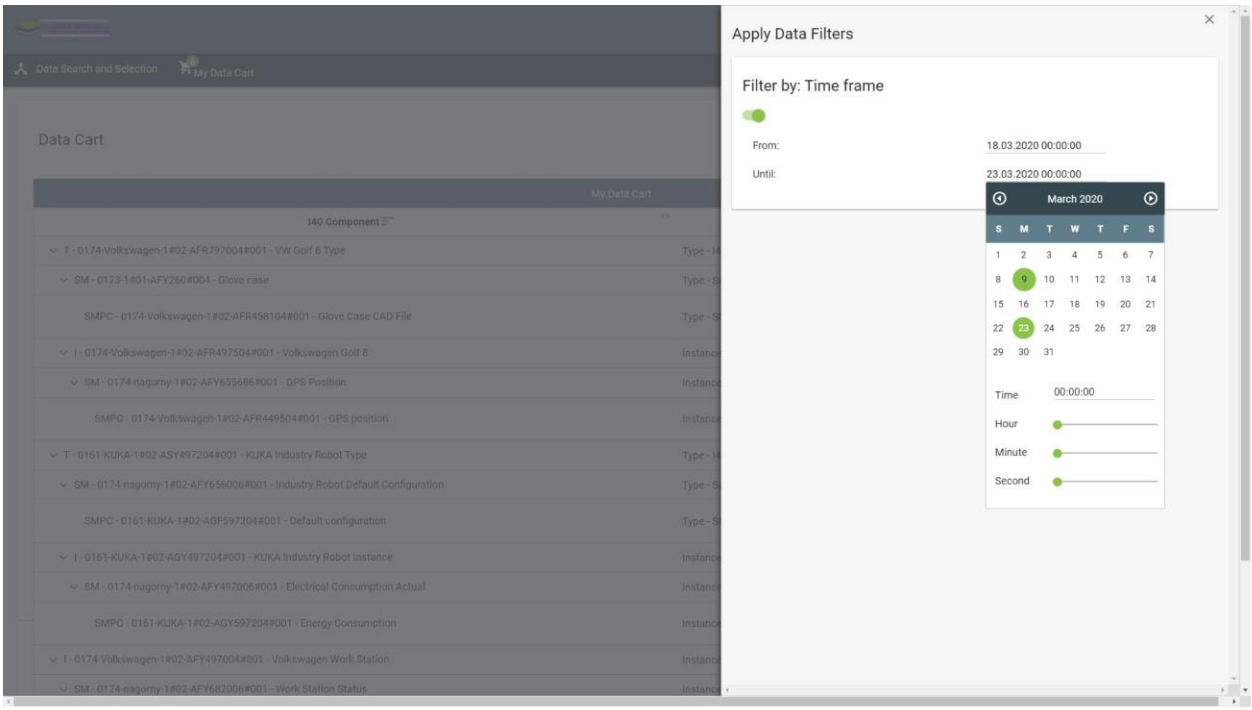


Figure 5-46: Time Frame Filter

A test of the framework which allows to add multiple other filters, listed in the sidebar shown Figure 5-46, was successful.

After the configuration of the filtering options, the DAC generation is tested. The Data Selector offers to download the DAC, to give an in-application preview or to deploy the DAC in the Data Provider as shown in Figure 5-47.

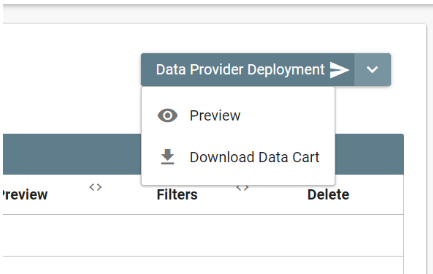


Figure 5-47: DAC - Download, Preview or Deployment

The Data Selector is generating the DAC based on all SMPCs located in the Data Cart properly. Code 12 shows one entry of the generated DAC. The entry includes the meta data of the related I40 component manifest, sub model, SMPC and the processor. The fully generated DAC is provided in the appendix in section 9.5.2.

```
          "I40Component": {
            "SubModelPropertyCharacterisation": {
              "SubModelPropertyCharacterisationProperties": {
                "value_type": "02",
                "structural_element": "02",
                "definition": "CAD File of the Glove Case",
                "version_number": "001",
                "preferred_name": "Glove Case CAD File",
                "irdi": "0174-Volkswagen-1#02-AFR458104#001",
                "labels": "{SMP, CAD, binary, Glove Case}",
                "version_initiated_on": "2019-12-01T23:57:00.000"
              },
              "SubModelPropertyCharacterisationIndividualName":
      "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type1_SMP_GloveCaseCADObjectStore"
            },
            "DataSourceBundle": {
              "Connector": {
                "ConnectorProperties": {
                  "File Filter Regex": "GloveCaseVWGolf8CAD.SLDPRT",
                  "Username": "myuser",
                  "automaticallyTerminateRelationshipsParameter": "success",
                  "Port": "21",
                  "Hostname": "xxx.xxx.102.83",
                  "Remote Path": "/",
                  "Delete Original": "false",
                  "Password": "thisIsNotThePwd"
                },
                "ConnectorIndividualName": "Test_DataSourceBundle_FTP_1"
              },
              "DataSourceBundleIndividualName": "Test_DataSourceBundle_FTP_1",
              "DataSourceBundleProperties": {},
              "DataSource": {
                "DataSourceIndividualName": "Test_DataSourceBundle_FTP_1",
                "DataSourceProperties": {
                  "password": "thisIsNotThePwd",
                  "port": "21",
                  "ip": "xxx.xxx.102.83",
                  "definition": "The FTP server of the I40 environment",
                  "version_number": "001",
                  "preferred_name": "FTP Server",
                  "irdi": "0174-Volkswagen-1#02-AFY951404#001",
                  "labels": "{FTP, storage, Data Source}",
                  "version_initiated_on": "2016-04-01T23:57:00",
                  "username": "myuser"
                }
              }
            },
            "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type",
            "Manifest": {
              "ManifestProperties": {}
            },
            "SubModel": {
              "SubModelProperties": {
                "coded_name": "44010103",
                "revision_number": "1",
                "definition": "Assembly group or element of a motor vehicle",
                "version_number": "4",
                "preferred_name": "Glove case",
                "irdi": "0173-1#01-AFY260#004",
                "labels": "{0173-1,AFY260,en,US,4,0,0}",
                "version_initiated_on": "2018-02-25T00:00:00"
              },
              "SubModelIndividualName": "0173-1--01-AFY260--004"
            }
          }
```


Code 12: One entry of the DAC

In the next testing phase, the DAC deployment in the Data Provider is tested. The Data Selector offers to select a Data Provider instance (if online) based on the instances defined in the knowledge graph as shown in Figure 5-48.

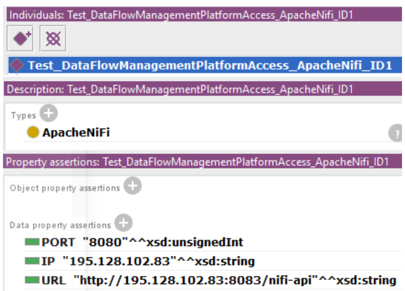


Figure 5-48: Data Provider instantiation in the knowledge graph

After the selection of the Data Provider instance, configured as shown in Figure 5-48, the Data Selector allows to deploy the DAC in a Data Provider template or to deploy the DAC directly in a process group. For the direct process group deployment, the Data Selector scans the Data Provider for all available flows and offers all available process groups in a list, as shown in Figure 5-49 with "NiFi Flow", to the user.

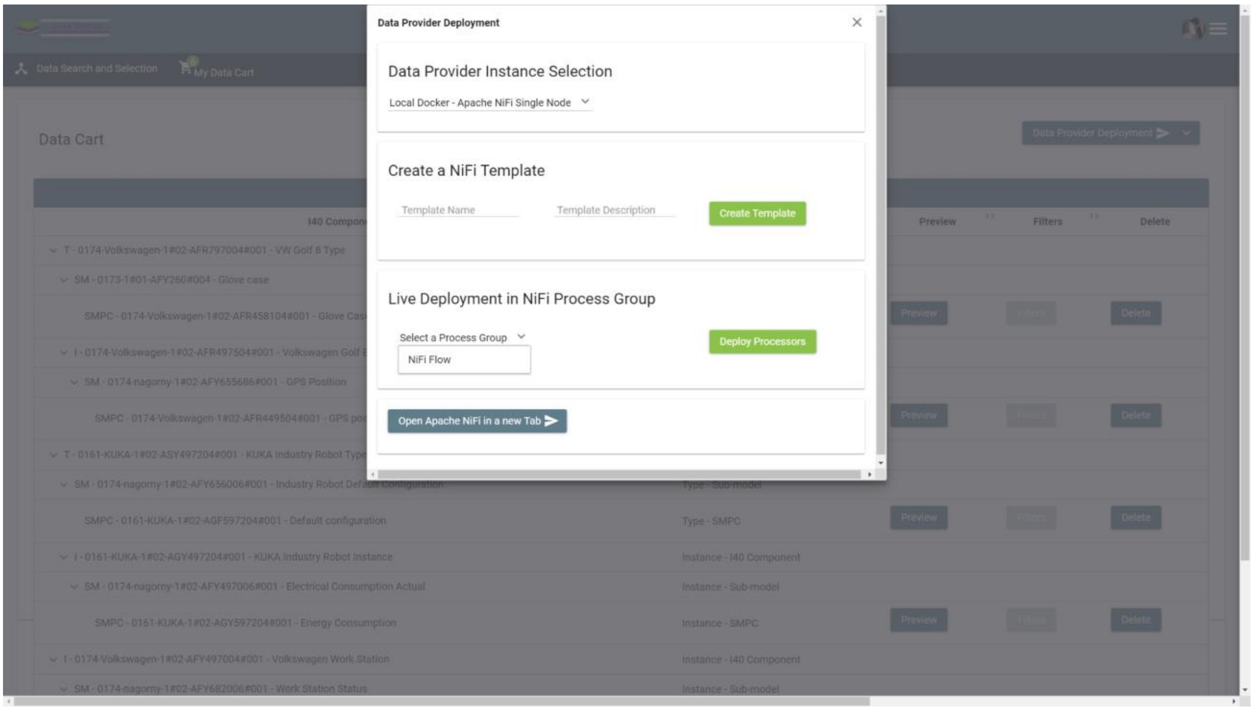


Figure 5-49: DAC Deployment

Both, the template creation and the direct deployment in a process group, works properly. The result (/the deployed DAC) is described in the next section.

5.6.2.4 *Test of the Data Provider*

The Data Provider deploys the processors that build a connection to the data sources for the extraction of the needed data. The test procedure in the Data Selector included the creation of a template as well as the direct deployment in the “NiFi Flow” process group. The results are described below.

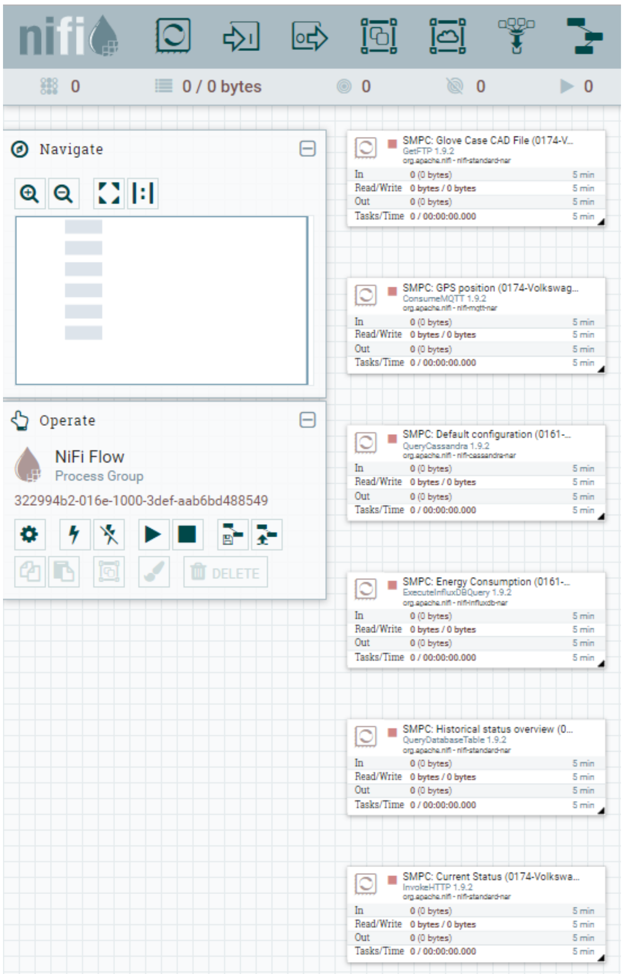


Figure 5-50: Data Provider – Deployed Processors

Figure 5-50 shows the result of the direct deployment. Each deployed processor is deployed properly and is ready for a direct execution to extract the data. The processor titles include the SMPC preferred name and the IRDI for a comparison.

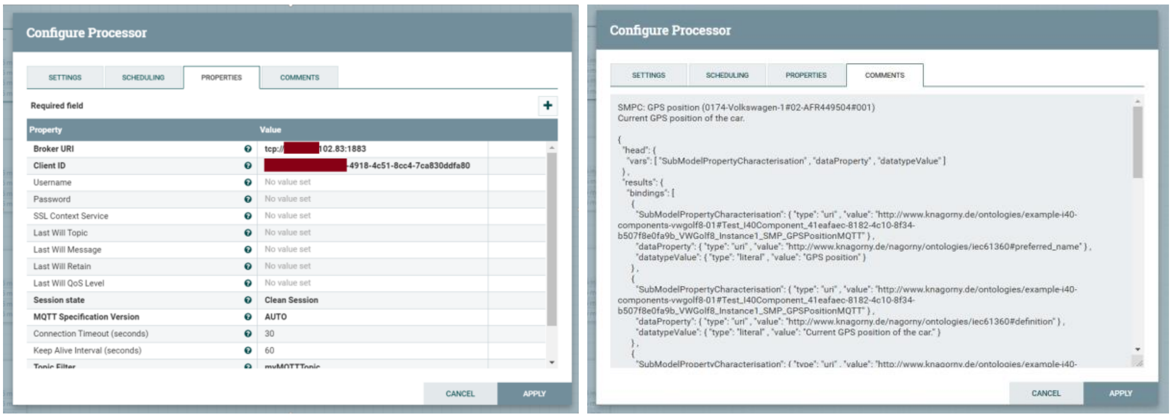


Figure 5-51: Processor pre-configured and ready for execution

The comment section in the processor configuration includes the full related DAC entry for further detailed information and the other tabs show the processor pre-configuration as shown in Figure 5-51.

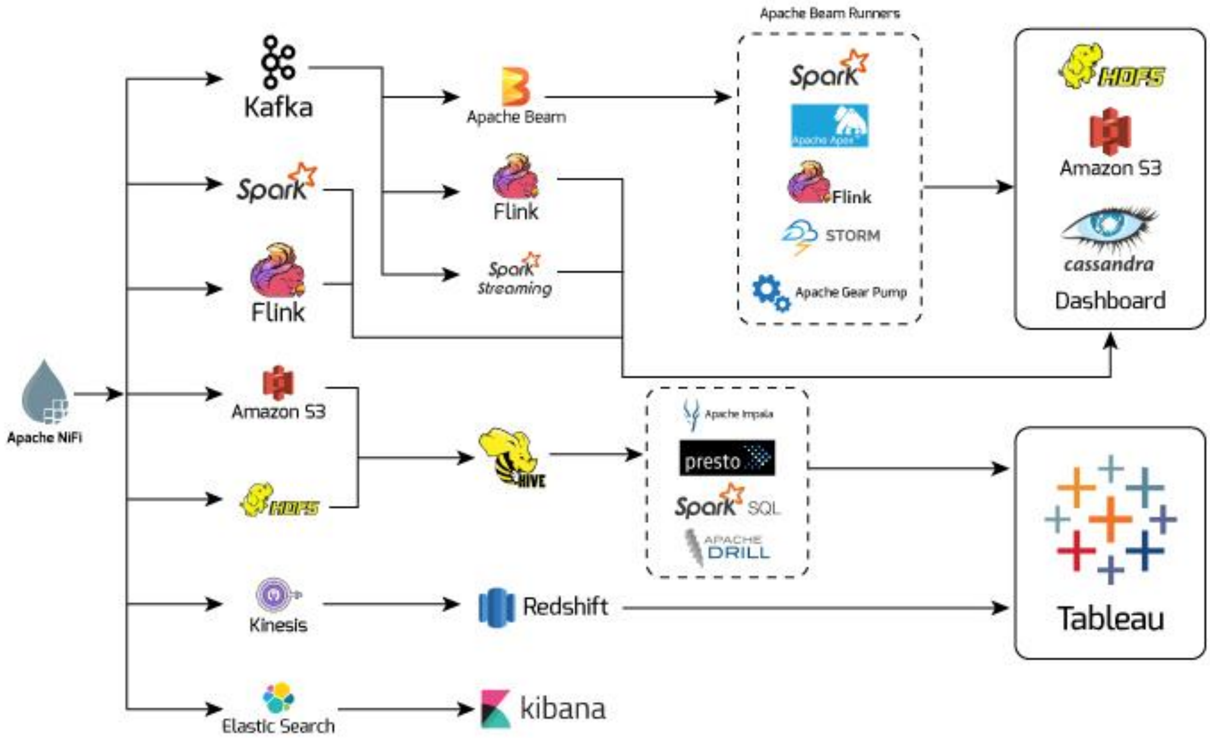


Figure 5-52: Extending the data pipeline (compare *(Navdeep Singh Gill, Jul 07, 2017)*)

The Data Provider is now ready to extend the data pipeline according to use case dependent requirements. Apache NiFi, chosen as the basis for the Data Provider, offers therefore several native processors for data cleaning, transformation and forwarding as indicated in Figure 5-52. But also, a large community provides further custom processors as e.g. for OPC-UA[32].

5.7 Limitations of the prototype solution

The current prototype implementation has some limitations. Table 5-12 provides a list of identified limitations, their causes and possible solutions to be implemented in future versions of the architecture.

[32] See https://github.com/linksmart/nifi-opc-ua-bundles or https://github.com/hashmapinc/nifi-opcua-bundle

Table 5-12: Identified Limitations of the implementation

Limitation	Cause	Possible Solution(s)
Apache NiFi requires for some kinds of Data Sources a pre-defined Controller Service.	For some types of data sources Apache NiFi requires an instantiation of controller services which are shared services, that can be used by reporting tasks, processors, and other services to utilize for configuration or task execution[33]. Native processors to access e.g. MySQL or Cassandra databases are using these controller services.	In the performed tests these controller services have been configured in Apache NiFi manually using the Apache NiFi API. For a deeper integration of Apache NiFi these controller services have to be added automatically, using the DSB approach for data sources.
Apache NiFi requires, for some native data extraction procedures, two processors. This is e.g. the case in S3 based object stores.	Some natively provided data source processors requires an additional processor to perform the data extraction. This is e.g. the case for S3 related data sources. Apache NiFi requires here a "ListS3" processor to enumerate the S3 bucket and to generate flow files referencing each object; and a FetchS3Object processor to access a single object based on the referencing flow files.	In future this issue can be solved with two different approaches. 1) Write custom processors that do not require two processors. 2) Add a second DSB to SMPCs so that both processors are deployed and connected.
Not all aspects of the BDV architecture are covered.	This doctoral work has a focus on data understanding and	The prototype could be extended by additional aspects

[33] See https://nifi.apache.org/docs/nifi-docs/html/user-guide.html#Controller_Services

Figure 5-53 (see below) shows the BDV Reference Architecture introduced in section 2.2.2. Layers with green dots mean that these layers are covered by the solution, and layers with red dots mean that they are not covered. See Table 5-13 and Table 5-14 below for a detailed description.	data acquisition. The implemented approach should be understood as a supplement for a comprehensive Big Data solution. Currently it is applicable company internal secured environments.	which are not covered yet by the prototype. Also, it could be integrated as supplement in a comprehensive Big Data environment.

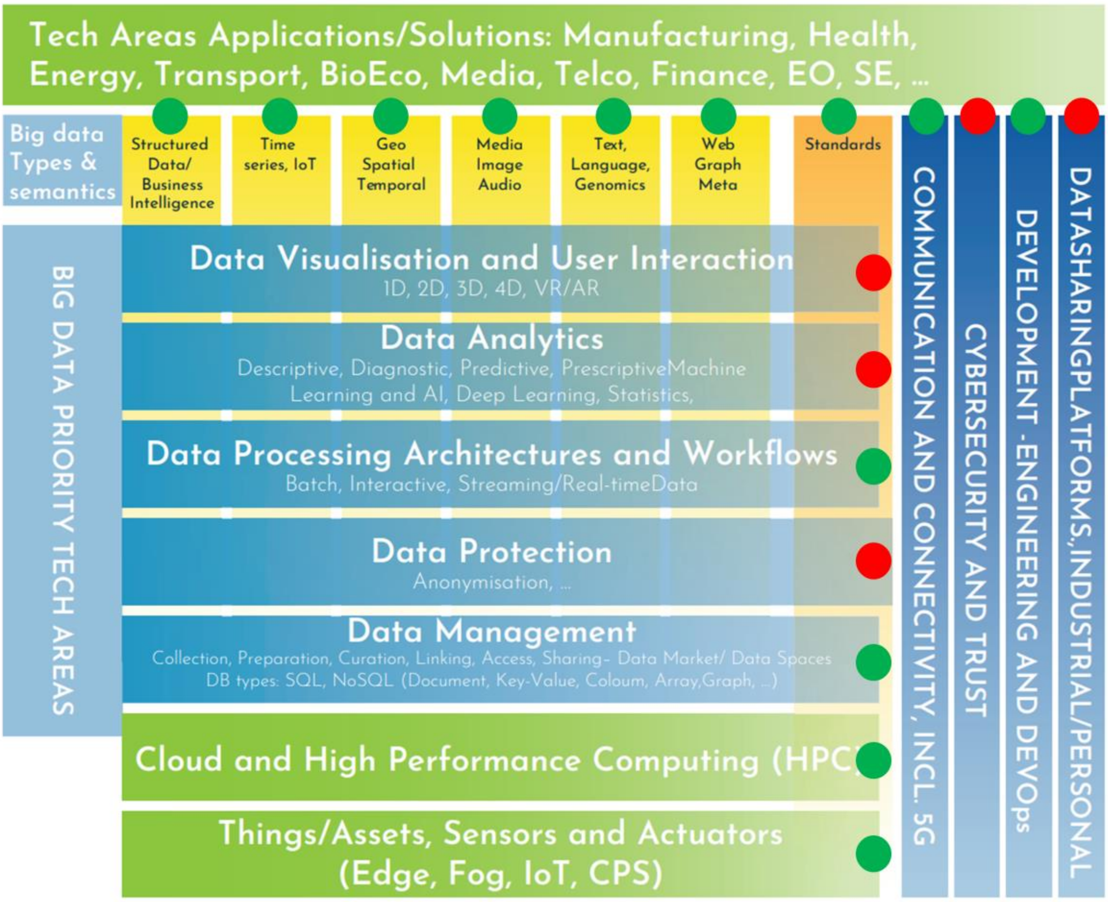


Figure 5-53: BDV Reference Architecture Mapping (figure based on (Association, 2016))

Table 5-13 provides a mapping of covered layers of the BDV Reference Architecture.

Table 5-13: Covered layers of the BDV Reference Architecture

BDV RA (green dots)	Software architecture Mapping
Things/ Assets, Sensors and Actuators	(Legacy) Systems and the System Integrator Covered by the (Legacy) Systems and the System Integrator module.
HPC	Big Data Environment Covered by the Big Data Environment. Depends also on the detailed architecture implementation, used technologies and technical IT infrastructure.
Data Management	Big Data Environment Covered by the Big Data Environment module.
Data Processing Architectures and Workflows	Data Space The data space and modules in the architecture enable data processing for batch, interactive and streaming data. Especially the Data Provider can be used for data pipelining.
Big Data Types and Semantics	Overall data pipelining solution Heterogenous data (see yellow vertical layers) are considered and can be stored, forwarded and processed.
Standards	Semantic model and software architecture The specification has shown that many I40 relevant standards have been considered and used for facilitating data integration, exchange and interoperability.
Communication and Connectivity, incl. 5G	Optional System Integrator The Optional System Integrator module allows the integration of several kinds of assets up to the developer which includes also 5G connected devices.
Development, Engineering and DevOps	Protégé, IntelliJ, Configuration Files This solution can be configured and further engineered by external tools like Protégé to modify the semantic model or IntelliJ to improve the overall solution. But it also can be configured by configuration files.

Table 5-14 shows a list of not covered layers.

Table 5-14: NOT covered layers of the BDV Reference Architecture

BDV RA (red dots)	**Description**
Data Analytics	This work describes an approach to support especially the data understanding and data acquisition in smart manufacturing environments. Data Analytics is seen as an extension on top of the Data Provider.
Data Visualization and User Interaction	Similar to Data Analytics. This would be an extension on top of the Data Provider and Data Analytics.
Data Protection	Security is not a particular focus of this work, as it concentrates on enterprise internal data environments. To enable/provide data protection, security, trust and data sharing platforms, the approach needs to be extended.
Cybersecurity and Trust	
Data sharing platforms Industrial/ Personal	

6

VALIDATION

The four major research and innovation results described in sections 2, 3 and 4 are:

1. A concept and approach for semantic model-based DIN Spec 91345 compliant data pipelining.
2. A DIN Spec 91345 RAMI 4.0 compliant semantic data model, which enables a standardized description of smart manufacturing environments, available data and the access to them.
3. A software architecture which exploits the semantic model for DIN Spec 91345 compliant data pipelining.
4. A generic implemented semantic model-based DIN Spec 91345 compliant data pipelining prototype validated in an industrially relevant environment.

These results have to be validated proofing the fulfilment of the requirements, as well as the achievement of the doctoral work objectives.

Note: Camarinha-Matos described in (Luis M. Camarinha-Matos, 2016) a range of validation and discussion instruments in research, to provide evidence on research results. Suitable for a validation are (i) a comparison between results in similar research fields (trend screening), (ii) frequent publication of research and/or innovation results to confirm their novelty and impact, (iii) a comprehensive and convincing discussion with experts in the scientific and technical areas covered by the research/innovation work, (iv) feedback from relevant stakeholders and experts in the application field, and (v) an implemented and tested prototype to demonstrate the applicability.

The five validation instruments addressed above have been applied in this doctoral work. The SotA in section 2 provides a trend screening in similar research fields and the detailed contribution beyond the SotA, described in section 4 which is implemented and tested in form of a prototype in section 5.

Figure 6-1 depicts how the doctoral work has been performed and organized, as well as how the dissertation is organised particularly in relation to validation and assessment of results. The Sankey diagram (Sankey, 1896) allows to understand the relationships between the research work and the mapping of research results. The green/grey arrow shows the path through chapters of the dissertation with emphasis on the validation and assessment of research and innovation results.

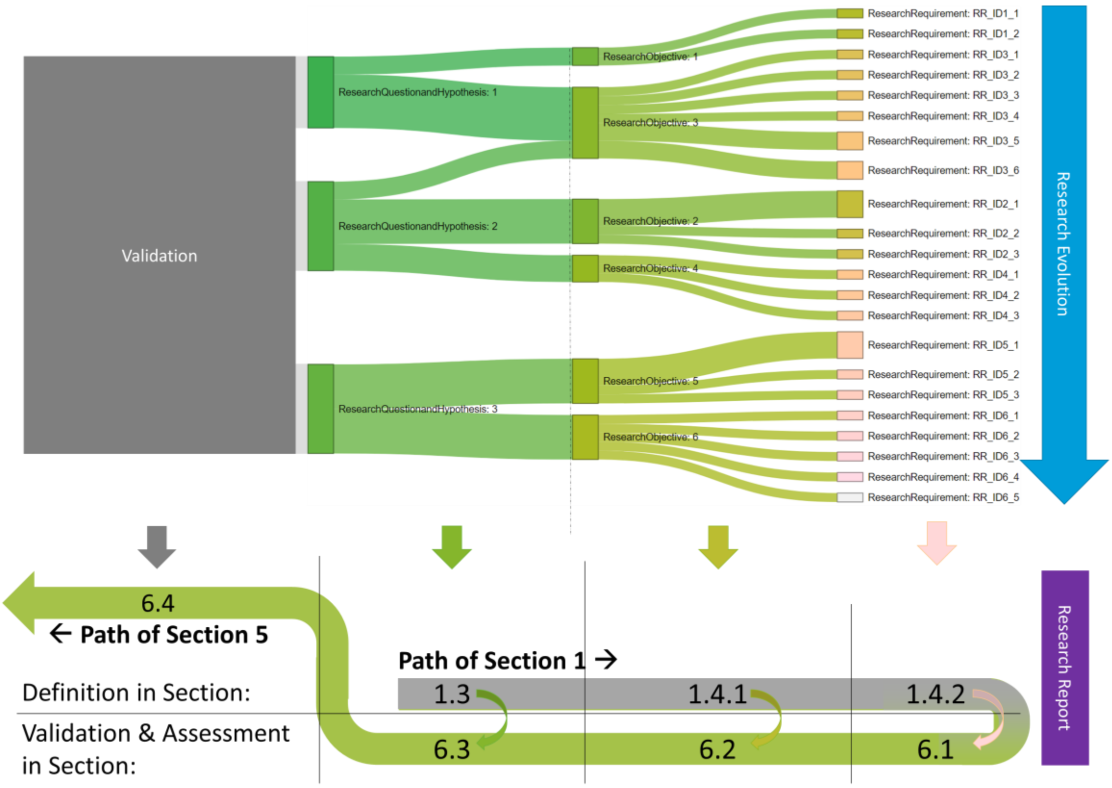


Figure 6-1: Organisation of the Validation & Assessment of results.

In detail:

- The research questions and hypotheses defined in section 1.3 are addressed in section 6.3.
- The research objectives defined in section 1.4.1 are validated in section 6.2.
- The research requirements defined in section 1.4.2 are validated in section 6.1.
- The overall research and innovation contributions defined in section 1.5 are validated and assessed in section 6.4.

This doctoral work cannot be proofed by mathematics as there is no mathematical model behind. It also cannot be tested by benchmarks because no competitive solutions for semantical based data pipeline could be found for a comparison.

6.1 Validation of research requirements

This section validates the research requirements defined in section 1.4.2 (RR_IDx_y), derived from the research objectives, defined in section 1.4.1, and the research questions and hypotheses (RQ&H_x). The validation is described in Table 6-1.

Table 6-1: Validation of research requirements

ID	Research requirement description	Associated RQ & H	Validation
Addressing the first research objective			
RR_ID1_1	The report shall provide an analysis of **reference architectures** relevant for data representation and structuring in smart manufacturing environments, with emphasis in considering proposals of Germanys strategic program under the name Industrie 4.0.	RQ&H_1	The doctoral work provides an analysis of reference architectures in the SotA sections 2.1.4 for smart manufacturing and 2.2.2 for Big Data.
RR_ID1_2	The report shall provide an analysis of technology **standards** relevant for data representation and structuring in smart manufacturing environments, with emphasis in considering proposals of Germanys strategic program under the name Industrie 4.0.	RQ&H_1	The doctoral work provided an analysis of standards in the SotA section 2.5.

Addressing the second research objective			
RR_ID2_1	The concept and approach shall **conceptually** show that a knowledge graph that describes a smart manufacturing data environment based on a unified semantical data model can be used as basis to realise the features for data **exploration, search, filtering, identification, understanding, selection and extraction**.	RQ&H_2	The concept and approach showed conceptually a data pipelining approach (introduced in section 3.1), which is built on the basis of a software architecture (introduced in section 3.2) and exploits a knowledge graph based on a unified semantic model (introduced in section 3.3) to realise the features addresses in the requirement.
RR_ID2_2	The concept and approach should show that it is suitable to elaborate a semantic model.	RQ&H_1, RQ&H_2	The overall concept and approach (described in section 3) showed in section 4.3.1 that it is suitable to elaborate the semantic model.
RR_ID2_3	The concept and approach should show that it is suitable to elaborate a software architecture that uses the semantic model to realise data search, filtering, selection and extraction.	RQ&H_2	The overall concept and approach (described in section 3) showed in section 4.3.2 that it is suitable to elaborate the software architecture.
Addressing the third research objective			
RR_ID3_1	The semantic data model shall be DIN-SPEC-91345 standard (RAMI 4.0) compliant.	RQ&H_1	The semantic data model strongly followed the DIN-SPEC-91345 standard (RAMI 4.0). The model covers the main aspects of this standard, which are described in section 2.1.4 (the architecture) and 2.5.1 (the Industrie 4.0 component).

			A detailed compliance analysis is given in sub-section 6.1.1.
RR_ID3_2	The semantic data model following the industrial digitalisation framework described by the DIN-SPEC-91345 standard (RAMI 4.0), should support the semantical unification in the smart manufacturing domain.	RQ&H_1	The elaborated semantic model (described in section 4.2.1) used the DIN-SPEC-91345 standard (RAMI 4.0) as basis for its structure. The model provides core concepts to describe smart manufacturing systems in a RAMI 4.0 compliant way. The model is open for a linkage with other standards and is therefore a suitable contribution for the semantical unification in the smart manufacturing domain.
RR_ID3_3	The semantic data model shall show that available Industrie 4.0-related classification standards, like e.g. eCl@ss, can be integrated.	RQ&H_1	The SotA in 2.5 presented a range of standards that are used to unify the classification and description of data environments (see 2.5.2 for eCl@ss or 2.5.3 for IEC 61360). The taxonomy of classification standards like eCl@ss can be linked to the SubModel class defined in the elaborated semantic model (see Figure 4-1). Vocabularies that provide standardised data properties as IEC 61360 can be integrated as described in 4.3.1.3.
RR_ID3_4	The semantic data model shall show that it is suitable to classify, structure and to describe assets	RQ&H_1	The instantiation example, given in section 4.3.1.4, showed that the semantic model is suitable to

	and associated digitalised data along the whole asset life-cycle.		structure and to describe assets and associated digitalised data along the whole asset life-cycle.
RR_ID3_5	The semantic data model shall show that it is suitable for heterogeneous brownfield and greenfield smart manufacturing data environments.	RQ&H_1	The model can be applied in heterogeneous brownfields by describing the whole assets and their data sources manually according to the semantic model. The data source bundle approach, described in section 4.3.1.2, enables the linkage of data in digitally accessible data sources. The model is also suitable in greenfields where assets are built as Industrie 4.0 components that bring an own description based on the elaborated semantic data model.
RR_ID3_6	The semantic data model shall show that it can be one major key element to support data understanding and data acquisition in smart manufacturing.	RQ&H_1	The semantic model enables a full description of assets in a manufacturing environment according to DIN-SPEC-91345 and their data along the whole life-cycle. This virtual representation is suitable for an exploitation for data understanding approaches. The data source bundle approach described in section 4.3.1.2 enables the description of the access to data. This description is suitable for an exploitation for acquisition approaches.

Addressing the fourth research objective			
RR_ID4_1	The software architecture should enable the management of a semantic model.	RQ&H_2	The Semantic Manager module, described in section 4.3.2.2, enables the management of the semantic model.
RR_ID4_2	The software architecture shall enable the exploitation of a semantic model to realise data **exploration, search, filtering, identification, understanding and selection**.	RQ&H_2	The Data Selector module, described in section 4.3.2.4, offers multiple functionalities to realise the addressed operations by exploiting the semantic model.
RR_ID4_3	The software architecture shall enable the exploitation of a semantic model to realise data **extraction**.	RQ&H_2	The Data Provider module, described in section 4.3.2.5, enables the data extraction based on the semantic model Data Source Bundle approach, described in section 4.3.1.2.
Addressing the fifth research objective			
RR_ID5_1	The generic prototype should be suitable for heterogenous data environments.	RQ&H_3	The designed solution shows that the data source bundle approach is flexible enough to parametrise several kinds of data source connectors deployable in the Data Provider. The Data Provider prototype with Apache NiFi as basis, enables to implement even proprietary and custom processors, so that any digitally accessible data source is theoretically accessible. The generic prototype is therefore suitable for heterogenous data environments.

RR_ID5_2	The generic prototype shall be suitable for industrial Industrie 4.0 compliant greenfields.	RQ&H_3	The generic prototype is suitable for Industrie 4.0 compliant greenfields from two perspectives: Industrie 4.0 components that uses the semantic model as a basis for the description of their data, are able to synchronise their description with the knowledge graph, by using the Semantic Manager. The generic prototype enables the application for typical communication technologies used in greenfields, as for example OPC-UA[34] or MQTT[35], using the data source connectors of the Data Provider.
RR_ID5_3	The generic prototype shall be suitable for industrial brownfields.	RQ&H_3	The generic prototype is suitable for brownfields from two perspectives: Assets that do not describe themselves by a semantic description, can be manually added to the knowledge graph by initiating assets based on the semantic model, as shown in section 5.6.2.1.

[34] See https://github.com/linksmart/nifi-opc-ua-bundles

[35] See https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-mqtt-nar/1.5.0/org.apache.nifi.processors.mqtt.ConsumeMQTT/

			The designed generic prototype is suitable for brownfield environments with very proprietary data sources and models through the variety of available data source connectors and the customisation possibility offered by Apache NiFi that is used as basis for the Data Provider.
Addressing the sixth research objective			
RR_ID6_1	The generic prototype shall prove that the solution is suitable for **exploration, search, filtering, identification and understanding** of data in heterogenous data environments.	RQ&H_3	The generic prototype was validated and tested in an industrially relevant smart manufacturing environment (see section 5.5). The Data Selector module shows the usage of the prototype for data **exploration, search, filtering, identification and understanding.** The Data Selector user gets an overview of the available data, which is structured and described in a standardised way, according to the semantic model suitable for data **exploration**. The used vocabulary shows that the approach is suitable to **understand** needed data by initialised meta data for the I40 components (manifest), Sub Models and SMPCs. The Data Selector offers also **search and filtering** functionalities which enables the user to search data based on IRIDIs, labels and

			other initiated data properties. Used unique identifiers as the IRDIs in I40 components, Sub Models and SMPCs also enable data **identification**. The semantic model enables the description of heterogeneous data environments, so that the prototype is suitable for such environments.
RR_ID6_2	The generic prototype shall prove that the solution is suitable for **selection** of data in heterogenous data environments.	RQ&H_3	The validation and testing of the prototype (see section 5.5) shows that the Data Selector of the designed solution enables to select needed data, located in heterogenous data environments.
RR_ID6_3	The generic prototype shall prove that the solution is suitable for **extraction** of data in heterogenous data environments (independently from used data storage or streaming technologies, communication technologies, protocols, data formats or data models).	RQ&H_3	The validation and testing of the prototype (see section 5.5) shows that the Data Provider, demonstrated with Apache NiFi as basis, is suitable for data extraction in heterogenous data environments. The solution shows that any data storage or streaming technology, communication technology, protocol, data format or data model is manageable as long as data is available in the network, accessible by the Data Provider.
RR_ID6_4	It may be proved that Graph Triple Stores can be used as basis to manage the contents of the	RQ&H_3	The generic prototype of the Semantic Manager uses as a basis the Jena Fuseki 2, with the Apache

	knowledge graph based on the semantic data model.		Jena TDB database as graph triple store (see section 5.3.1). The validation and testing of the prototype (see section 5.5) shows that such graph triple stores can be used successfully as basis to manage the contents of the knowledge graph, based on the semantic data model.
RR_ID6_5	It shall be proved that existing data pipelining technologies can be extended to enable semantic model-based data pipelining.	RQ&H_3	The generic prototype of the Data Provider used as data pipelining technology Apache NiFi (see section 5.3.3). This technology was extended with the pre-configuration approach enabled by the data source bundle (see section 4.3.1.2) and the DAC (see sections 4.3.2.4 and 5.3.2.4) creation. The validation and testing of the prototype (see section 5.5) shows successfully that such an extension is suitable to enable semantic model-based data pipelining.

6.1.1 RR_ID3_1 - Compliance validation to the RAMI 4.0

Addressing the research requirement "RR_ID3_1", the solution aims to be "DIN Spec 91345 RAMI 4.0 compliant". This section describes the compliance of the solution to the RAMI 4.0, introduced in section 2.1.4.

Figure 6-2 depicts the 3D RAMI 4.0 model, identifying 28 potential data sources. For each potential assets, positioned in one of the seven hierarchy levels (according to the IEC 62264/IEC 61512), can be considered four major faces of the life-cycle of those assets, according to the IEC

62890. These faces are building the digital thread of an asset. The 28 potential data sources are positioned on the information layer, which is the digitalisation dimension of the RAMI 4.0.

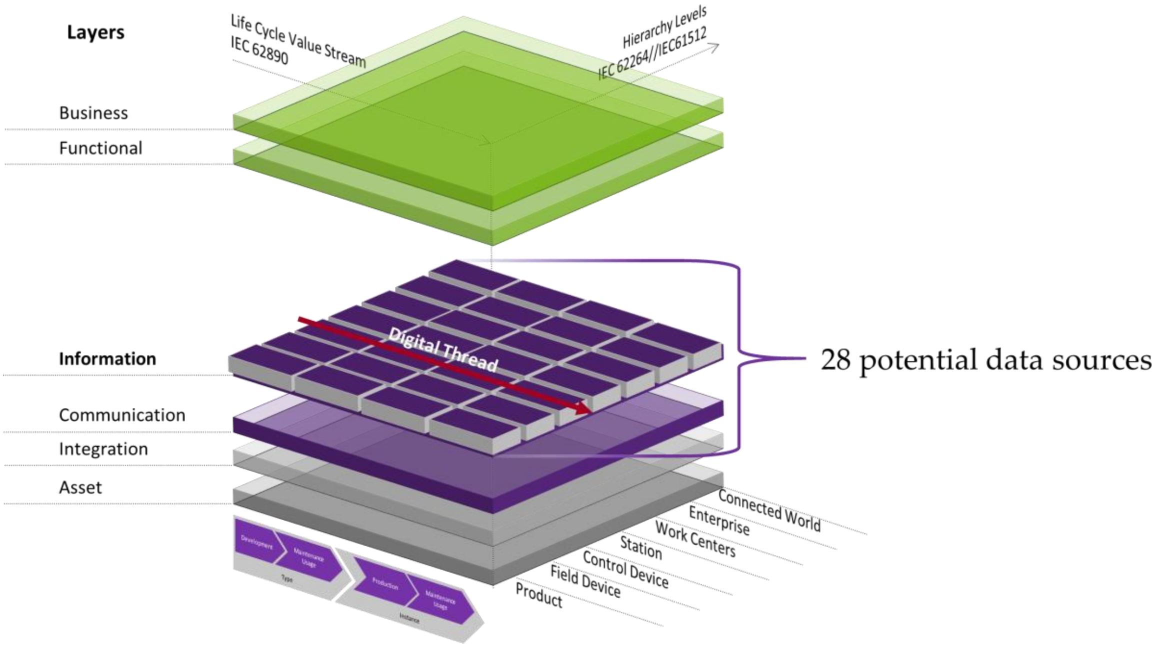


Figure 6-2: (4x7) 28 Potential data sources in the information layer of the RAMI 4.0

According to DIN-SPEC-91345, the **information layer** in general consists data that is used, generated or modified by or for an asset. In the semantic data model this data is described by a standardised vocabulary (see section 4.3.1.3) in the manifest, sub models and SMPCs (see section 4.3.1.1), and are referenced by the Data Source Bundle approach (see section 4.3.1.2). The implemented prototype allows to extract data which is located in I40 components themselves or in any other digitally accessible data source (see section 4.3.2.5).

The **life-cycle and value stream axis** represents the whole life-cycle of an asset. It distincts between "type" and "instance". A "type" becomes an "instance" as soon as an instance based on a type is created (SCI 4.0, April 2018). For example: A VW Golf type is a collection of CAD files and specifications and the produced physical VW Golf is an instance of this type. The semantic data model allows the description of life-cycles and value streams by instantiating individuals based on the "LifeCycleAndValueStream" class (see section 4.3.1.1). The individuals are linked to the SMPCs of an I40 component, so that a digital thread is created (see "Digital Thread" in Figure 6-2).

The **Hierarchy Level axis** describes the location of different kinds of assets according to IEC 62264-1 and IEC 61512-1. The assets can be mapped into the semantic model where the related hierarchy layer is associated to the asset. The semantic data model allows the description of hierarchy levels by instantiating individuals based on the "HierarchyLevel" class (see section 4.3.1.1).

The semantic model is also strongly compliant to the Industrie 4.0 component. As described in section 2.5.1, the DIN SPEC 91345 standard specifies details about the structure of an Industrie 4.0 component. The standard specifies (among others) how meta-data about an Industrie 4.0 component needs to be represented in form of a manifest partial models and basic views. The semantic model considers this structure (compare in section 4.3.1) by integrating the following concepts:

- A manifest by the definition of a "Manifest" class
- Partial models by defining the "SubModel" and "SubModelPropertyCharacterisation" classes.
- Basic views by defining the "DataAccessView" class.

Also, the concepts for users and their roles are integrated by defining the "User" and "UserRole" classes.

The knowledge graph, which is generated out of all these instantiated individuals, based on the semantic data model, is able to represent the manufacturing environment. This knowledge graph is exploited by the implemented early prototype (see section 5.3) which was elaborated in form of a software architecture in section 4.3.2.

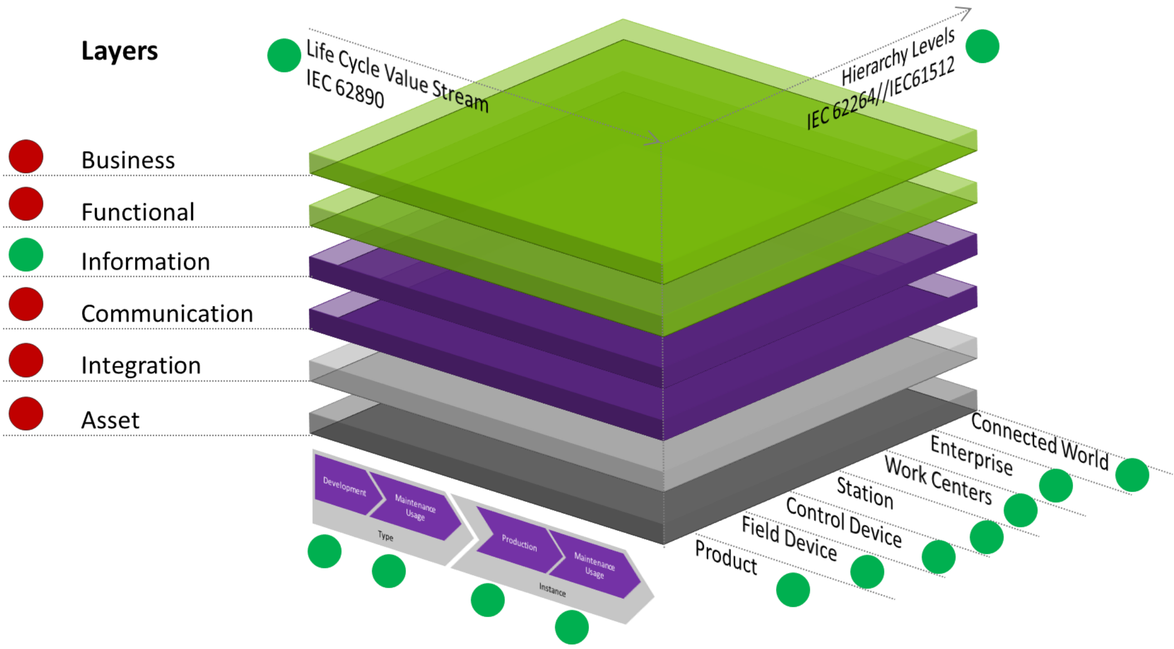


Figure 6-3: RAMI 4.0 architecture mapping (derived from *(Wolfgang Dorst et al., 2015)*)

In conclusion, the solution elaborated in this doctoral work is compliant to the whole life cycle and value stream axis according to IEC 62890, the hierarchy level axis according to IEC

62264/IEC 61512 and to the information layer of the RAMI 4.0 as shown in Figure 6-3. Therefore, all 28 potential data sources on the information layer, shown in Figure 6-2, can be addressed.

6.2 Validation of research objectives

Section 1.4.1 defines the objectives of this doctoral work. Each objective is validated below.

6.2.1 Research Objective 1 – Analysis of standards and reference architectures

The first objective was an analysis of the most relevant standards and reference architectures for data representation and structuring in smart manufacturing environments. It considers proposals from Germanys strategic program directed towards the year 2020, under the name Industrie 4.0.

The state-of-the-art analysis given in section 2 provides a comprehensive description of smart manufacturing environments (see section 2.1), their data (see section 2.2) and standards (see section 2.5). The analysis also included a comprehensive introduction in modern smart manufacturing reference architectures (see section 2.1.4) including:

- Reference Architecture Model Industrie 4.0 (RAMI 4.0)
- Smart Grid Architecture Model (SGAM)
- China Intelligent Manufacturing System Architecture (IMSA)
- Industrial Internet Reference Architecture (IIRA)

Especially the RAMI 4.0 (see section 2.1.4), specified by the Plattform Industrie 4.0 and, standardised in DIN-SPEC-91345 (see section 2.5.1), is one of the main orientations in the context of Industrie 4.0 and was strongly influencing this doctoral work.

The RAMI 4.0 was basis for the analysis of relevant standards for data representation and structuring in smart manufacturing environments. Several standards have been identified, analysed and considered. The most relevant were DIN-SPEC-91345 (RAMI 4.0), IEC 62890, IEC 62264 (see section 2.1.4 and 2.5.1) or IEC 61360 (see section 2.5.3), but also standards apart from the RAMI 4.0 as eCl@ss or IRDI (see section 2.5.2).

The analysis of standards and reference architectures was used to design concept and approach (objective 2).

6.2.2 Research Objective 2 – A concept and approach to search, filter, select and extract data

The second objective was the development of a concept and approach to search, filter, select and extract data to realize comprehensive data search, filter, selection and extraction to support data understanding and data acquisition (see section 2.3) in smart manufacturing systems.

The presented concept and approach (see section 3) described how a semantic model (see section 3.3) can be exploited to search, filter, select and extract data along the life-cycle of Industrie 4.0 components (see section 3.2).

The concept and approach also presented how data can be extracted by using the semantic model to parametrise a data source processor which accesses and extracts data. The approach showed how data, located in digitally accessible data sources, can be described, searched, filtered, selected and extracted to support users in data understanding and data acquisition.

Based on the concept and approach a semantic model (research objective 3) and a software architecture (research objective 4) was specified.

6.2.3 Research Objective 3 – Development of a semantic model

The third objective is the development of a semantic model to standardise, categorise and structure data along asset life-cycles, including their linkages, compliant with DIN-SPEC-91345 which specifies the RAMI 4.0 and allows to add further standards.

Section 4.3.1 described the specification of a semantic model in line with DIN-SPEC-91345 (as validated in section 6.1.1). The semantic model enables to define I40 components, to structure and categorise data produced along their life-cycle and to embed them into a smart manufacturing environment. The semantic model enables also to integrate data classification standards like eCl@ss to standardise data related to I40 components.

The data source bundle approach (see section 4.3.1.2) enables also to describe the parameters needed to access the data in a data source. Therefore, it is divided into a "data source" part that configures a data source (e.g. a database or a message broker) in the smart manufacturing environment, a "processor" part that configures static processor parameters, and a "property builder" part that dynamically creates properties as queries for data filtering or aggregation.

The defined semantic model was used as a basis to build the knowledge graph for the software architecture (research objective 4).

6.2.4 Research Objective 4 – A software architecture

The fourth objective was the specification of a software architecture that uses the semantic model to realise comprehensive data search, filter, selection and extraction.

Section 4.3.2 specifies the software architecture that enables these features for smart manufacturing environments. The architecture introduced the Semantic Manager to manage the semantic model (see section 4.3.2.2), the Data Selector to understand, explore, search, filter and select data (see section 4.3.2.4), and the Data Provider to extract data from data sources in smart manufacturing environments (see section 4.3.2.5).

Based on the specifications of the semantic model and the software architecture, a generic prototype was implemented (research objective 5).

6.2.5 Research Objective 5 – A generic prototype implementation

The fifth objective was the implementation of a generic prototype which supports data understanding and data acquisition in smart manufacturing environments.

This prototype implementation was described in section 5 and includes the implementation of the semantic model (see section 5.2) and the software architecture (section 5.3) which is divided into the Semantic Manager (see section 5.3.1), the Data Selector (see section 5.3.2), the Data Provider (see section 5.3.3) and a basic implementation of an I40 component instance (see section 5.3.4).

The prototype was successfully verified according to the development requirements in section 5.5 and was validated in a relevant industrial environment (research objective 6).

6.2.6 Research Objective 6 – Validation of the prototype

The sixth objective was the integration, validation and demonstration of the prototype in a relevant industrial environment (TRL 6).

Section 5.6 presented how the prototype was configured for the validation according to the generic testing scenario (see section 5.4). The generic prototype was integrated in a TRL 6 compliant environment (see section 5.6.1) and collected results (see section 5.6.2) that are suitable to answer the research questions (see section 6.3).

A demonstration is provided in form of a video which is also summarising the outcomes of this doctoral work. The video is part of a Journal publication and is accessible in the IEEE Xplore library (see section 9.1).

6.3 Validation of hypotheses and answering research questions

6.3.1 Related to the first Research Question and Hypothesis

Question: *Is it possible to classify, to structure and to describe heterogenous brownfield and greenfield smart manufacturing data environments by considering latest standards coming from the Industrie 4.0 platform?*

Hypothesis: *If latest norms issued by the Industrie 4.0 platform, such as the DIN Spec 91345 RAMI 4.0, as well as other data classification standards, are used to elaborate a semantic data model, then it is possible to classify, structure and describe heterogeneous brownfield and greenfield smart manufacturing data environments.*

Answer: ***The validation of the DIN Spec 91345 RAMI 4.0 compliant semantic data model which integrates standards like eCl@ss, IEC 62890, IEC 62264, IEC 61512, IEC 61360 and IRDI showed, that the classification, structuring and description of heterogeneous brownfield and greenfield smart manufacturing data environments is possible.***

Notes: This doctoral work shows how standards provided by the Industrie 4.0 platform can be used to build a unified DIN Spec 91345 RAMI 4.0 compliant semantic data model (conceptually introduced in section 3.3, described in section 4.3.1 and implemented in section 5.2). In summary, this is demonstrated by applying DIN SPEC 91345 (acting as basis for the structure of an Industrie 4.0 component, described in section 2.5.1), IEC 62890 (to model the Life Cycle & Value Stream of an Industrie 4.0 component, described in section 2.1.4), IEC 62264 & IEC 61512 (representing the Hierarchy Levels, described in section 2.1.4), IEC 61360 (for the provision of a vocabulary, described in section 2.5.3), eCl@ss (used for data classification, described in section 2.5.2) and the IRDI (acting as globally unified identifier, described in section 2.5.2).

The elaborated semantic model is suitable to classify, structure and describe heterogeneous brownfield and greenfield smart manufacturing data environments, based on those applied standards (as tested in an industrial relevant environment in section 5.6, based on the generic testing scenario described in section 5.4). Therefore, the hypothesis is correct.

The ongoing standardisation increases the common ground in the area of smart manufacturing. Reference architectures and semantical standardisation support the development of a common vocabulary, a common focus, a common structure and common definitions. The transformation of standards into semantic models and their sub-sequent mapping makes it possible to identify (and merge or erase) overlaps/seminaries, so that an overall unified semantic model, used to classify and structure heterogenous data environments, can be reached closely. However, a static semantic model which fits it all will be probably never reached, because the area is very dynamic and volatile.

6.3.2 Related to the second Research Question and Hypothesis

Question: *Can a concept and an approach be elaborated, and a software architecture be specified, which support the exploitation of a unified semantic data model to enable an easy exploration, search, filtering, identification, understanding, selection and extraction of data, supporting also data understanding and data acquisition, in smart manufacturing environments?*

Hypothesis: *If a knowledge graph that describes a smart manufacturing data environment based on a unified semantical data model is used as a basis, then it is possible to define a concept and approach, and to specify a software architecture, which support the exploitation of a unified semantic data model, to enable an easy exploration, search, filtering, identification, understanding, selection and extraction of data, supporting also data understanding and data acquisition, in smart manufacturing environments.*

Answer: ***The knowledge graph, based on the DIN Spec 91345 RAMI 4.0 compliant semantic data model used to describe a smart manufacturing data environment, was successfully used as a basis to define a concept and approach, and to specify a software architecture, which support the exploitation of the***

semantic data model, to enable an easy exploration, search, filtering, identification, understanding, selection and extraction of data, supporting also data understanding and data acquisition, in smart manufacturing environments.

Notes: The overall concept and approach (described in section 3) was successfully elaborated by using a semantic data model as basis to classify, structure and describe heterogenous brownfield and green-field smart manufacturing data environments in form of a resulting knowledge graph. The concept and approach conceptually describe the basis for a software architecture in section 3.2 which is successfully specified in section 4.3.2. The software architecture exploits the knowledge graph by defining three major modules: While the Semantic Manager enables the management of the knowledge graph, the Data Selector enables an easy exploration, search, filtering, identification, understanding and selection of data, based on visualisation, search and filtering functionalities. The Data Provider enables the data extraction. The exploitation of the knowledge graph by these modules supports the data understanding and data acquisition in smart manufacturing environments. The hypothesis is thus evaluated as correct.

6.3.3 Related to the third Research Question and Hypothesis

Question: *Is it possible to implement and validate in a relevant industrial environment a semantic model-based data pipelining tool prototype, using current data pipelining and software engineering technologies?*

Hypothesis: *If a triple store, using SPARQL to navigate over a knowledge graph, is combined with software engineering frameworks and existing data pipelining technologies, then it is possible to implement and validate an industrial-compliant data pipeline tool prototype.*

Answer ***The Apache Jena TDB2 triple store with SPARQL features to navigate over a knowledge graph, was successfully combined with software engineering frameworks and existing data pipelining technologies like Spring Boot and Security, PrimeFaces and Apache NiFi, which enabled the successful implementation and validation of an industrial-compliant data pipeline tool prototype.***

Notes: The concept and approach (described in section 3) were successfully specified in section 4.3 and implemented in section 5. The prototype is implemented by using as basis a range of SotA technologies, selected based on a technology screening, given in section 5.1. The Semantic Manager is successfully implemented using the Apache Jena TDB2 triple store (see section 5.3.1) as a basis. The Data Selector is successfully implemented by using software engineering frameworks like Spring Boot and Security and PrimeFaces (see section 5.3.2). The Data Provider is successfully implemented by using the existing data pipelining technology Apache NiFi as a basis (see section 5.3.3). The overall implemented data pipelining prototype was successfully tested in an industrial relevant environment (see section 5.6) and is applicable in brownfield and greenfield smart manufacturing systems as described in the validation of "RR_ID3_5" (see section 6.1). In conclusion, the hypothesis is thus evaluated as correct.

6.4 Validation and Assessment of Results

This doctoral work makes four major research and innovation contributions, as schematically described in Figure 1-2 of section 1.5. These results are used to validate and assess the overall achievements.

6.4.1 Result 1: Concept and approach

The concept and approach for semantic model-based DIN Spec 91345 compliant data pipelining (described in section 3) was used in the European project BOOST 4.0 (see https://boost40.eu/). BOOST 4.0 was the biggest European initiative in Big Data for Industrie 4.0 in the year 2020. In this project the concept and approach addressed the needs of the project partner "Volkswagen AG" and the path dependencies of their brownfields towards a digital transformation.

This concept and approach was claimed and shared in (Nagorny, Scholze, Ruhl, & Colombo, 2018) and received positive feedback (see section 9.4.1).

The results are in line with the related research requirements RR_ID2_(1-3) (validated in section 6.1) and research objectives (validated in section 6.2).

The approach has a huge field for exploitation and could also be integrated in diverse ETL, ELT, data pipelining or data analytics tools.

6.4.2 Result 2: DIN-SPEC-91345 compliant semantic model

A DIN Spec 91345 RAMI 4.0 compliant semantic data model (described in section 4.3.1) enables a standardised description of smart manufacturing data environments including the access to data.

All related research requirements RR_ID3_(1-6) (validated in section 6.1) and the related research objective 3 (validated in section 6.2.3) are achieved. The semantic model is also compliant to the DIN Spec 91345 (RAMI 4.0) as validated in detail in section 6.1.1.

The literature review in section 2 (published (Nagorny et al., 2017) and (Nagorny et al., 2020)) shows that the most results in similar areas are very focused on specific areas like vocabularies, classification, custom semantic models for specific environments, self-description of assets, etc. An overall approach which allows to describe a RAMI 4.0 compliant smart manufacturing data environment, using standards for classification, vocabularies and especially the access to data in a DIN Spec 91345 compliant semantic model-based way, was missing.

The semantic model, claimed and shared in (Nagorny et al., 2020), is a novel approach which integrates the RAMI 4.0, standardised vocabularies and classification classes, and data access descriptions to linked data. The enabled nesting of hierarchy levels, layers, life-cycle stages and data access views, allows to build complex and highly customised structures of smart manufacturing environments and I40 component descriptions, based on the RAMI 4.0. The nature of semantic models enables to link additional models which makes the overall approach flexible and evolvable.

The results above were published to the journal paper (Nagorny et al., 2020) which was well accepted in the community (see section 9.4.1).

Although the semantic model combines many aspects of smart manufacturing, some aspects are still missing. Aspects like the description of functionality, business logic, functional relations (orchestrations) between I40 components, processes descriptions or extended trust and security are not considered in this doctoral work, but could be considered as a potential outlook, as described in section 7.3.4 and 7.3.6.

6.4.3 Result 3: Software architecture

The software architecture (described in section 4.3.2) exploits the semantic model for DIN Spec 91345 compliant data pipelining.

The architecture was published in (Nagorny et al., 2020) which was well accepted in the community (see section 9.4.1).

The related requirements RR_ID4_(1-3) (validated in section 6.1) are successfully fulfilled, as well as the related research objective 4 (validated in section 6.2.4).

The architecture has a focus on data and does not cover all requirements for a Big Data solution which has been shown in the mapping with the BDV reference architecture in section 5.7. The architecture does not consider data governance aspects, security, data protection (e.g. anonymisation) or trust which would be needed for a secure data exchange. Such additional features would require further research activities that are addressed in the outlook section 7.3.4. The architecture is therefore designed in its current state for an already secured company internal environment. The application outside of such networks would require further improvements and is therefore not foreseen yet but addressed as an outlook in section 7.3.4.

6.4.4 Result 4: Generic prototype

The generic implemented semantic model-based DIN Spec 91345 compliant data pipelining prototype (described in section 5) was validated in an industrially relevant environment (see section 5.6).

The prototype is successfully demonstrated in form of a video (see section 9.1) which was part of (Nagorny et al., 2020) (see section 9.4.1). The sources are available in GitLab (see section 9.1).

The related requirements RR_ID5_(1-3) and RR_ID6_(1-5) are fully fulfilled (see section 6.1), as well as the related research objectives 5 and 6 (validated in section 6.2.5 and 6.2.6).

The genericity of the prototype is still to be further validated. The prototype is based on the RAMI 4.0 which is a reference architecture with a focus on the manufacturing industry. This leads to the hypothesis that the prototype is generic for the manufacturing domain. The RAMI 4.0 is also mappable with the IIRA as shown in section 2.1.4. This leads to the hypothesis that the prototype is also generic for further industrial sectors like energy, health care, production, transport or the public sector. In this doctoral work the prototype was only validated in the car manufacturing sector. Further validation in other sectors is needed to fully validate the genericity. This is addressed in outlook section 7.3.7

How performant and how scalable the elaborated prototype is, when it is used in very large manufacturing systems where the knowledge graph can be very large, is not yet clear. The analysis of such aspects on the scalability, reliability, performance and other technologies are further discussed as an outlook in section 7.3.5.

The prototype has also a high potential for further extensions. These are addressed in outlook sections 7.3.1, 7.3.3 and 7.3.6.

7

CONCLUSIONS AND OUTLOOKS

7.1 Conclusions

This doctoral work makes four major research and innovation contributions:

1. A concept and approach for semantic model-based DIN Spec 91345 compliant data pipelining.
2. A DIN Spec 91345 RAMI 4.0 compliant semantic data model, which enables a standardized description of smart manufacturing environments, available data and the access to them.
3. A software architecture which exploits the semantic model for DIN Spec 91345 compliant data pipelining.
4. A generic implemented semantic model-based DIN Spec 91345 compliant data pipelining prototype validated in an industrially relevant environment.

The research and innovation contributions have been backed up by a set of publications (conference papers and Journal manuscripts) that reveals the novelty of the generated foreground knowledge.

7.2 Concluding remarks

A starting point of this doctoral work were the three identified challenges described in section 1.2. What is achieved, what is not and what are the lessons learned?

Achievements of challenge 1 - Structuring the data chaos

The first challenge aimed to support structuring the growing volume, variety, velocity and the growing complexity of data in the smart manufacturing domain. This challenge addressed mainly the first research question (see section 1.3.1) with the hypothesis that a semantic data model could support the structuring of such complex heterogenous data environments.

A semantic model was achieved which is suitable to classify, structure and describe heterogeneous brownfield and greenfield manufacturing data environments in a standardised way. This model enables the generation of a catalogue of data mapped according to RAMI 4.0 and standardised by using standards as demonstrated with eCl@ss for data classification or IEC61360 for data description. The model itself is suitable for data in different volumes, varieties and velocities.

A more comprehensive integration and extension of the semantic model by available standardisation actions was not achieved. This issue is addressed in the outlook in section 7.3.2. Also not achieved, but also not directly addressed in this work, was a unification of communication technologies and data models themselves. It can be expected that in future, further communication technologies, protocols and data model schemas will emerge for a variety of reasons to solve specific challenges in data management (e.g. high volumes, real-time, event-based communication, streams, energy efficient communication, etc.). Semantic models are a suitable approach for further unification and standardisation, but require a strong collaboration with various interest groups.

A lesson learned is therefore: In future we will continue to be surrounded by a high variety of communication technologies, protocols and data models. Semantic models that structure, classify and standardise heterogenous environments, as presented in this doctoral work, are a good approach to structure the data chaos and to push the unification process towards a standardised data representation. A strong collaboration of interest groups could even reach extensive standardisation in some areas.

Achievements of challenge 2 – Ease data exploration in heterogeneous data environments.

The second challenge aimed to simplify the exploration, searching, filtering, identification, understanding and selection of data in heterogeneous data environments. This challenge addressed mainly the second and third research question (see sections 1.3.2 and 1.3.3). While for the second research question a suitable concept and an approach is elaborated, for the third research question a suitable prototype for a testing in an industrial relevant environment is implemented.

A suitable implementation was achieved, following the suggested concept and approach in form of a prototype, which was tested in a relevant industrial environment. The Data Selector module shows how a knowledge graph which represents the description of available data in a smart manufacturing data environment based on the elaborated semantic model can be exploited to offer comprehensive functionalities for simplifying the exploration, searching, filtering, identification, understanding and selection of data in heterogeneous data environments. A graphical user interface, which exploits the knowledge graph, simplifies such complex tasks.

A range of suitable additional prototype features for better exploitation of the knowledge graph to further simplify data understanding and data acquisition were not achieved. These features are explained as outlook in detail in section 7.3.3. Also, the management of the semantic model could be further improved by additional features for the Semantic Manager module. Such features are addressed as outlook in detail in section 7.3.1.

A learned lesson is that exploitation of a RAMI 4.0 compliant semantic model in a well-structured web-interface, as shown with the Data Selector, is a suitable approach to simplify data exploration in heterogeneous data environments. This doctoral work has a focus on data, the results lead to the assumption that the approach could also be extended to exploit other aspects like asset functionalities and business logic. These ideas are discussed in more detail in section 7.3.6.

Achievements of challenge 3 – Simplify data extraction for heterogeneous data environments.

The third challenge aimed to simplify data extraction in heterogeneous data environments. This challenge addressed again the second and third research question (see sections 1.3.2 and 1.3.3). While for the second question a suitable concept and approach was elaborated, for the third research question a suitable data extraction prototype for a testing in an industrial relevant environment was implemented.

The prototype considers a variety of communication and data storage technologies, data formats and data schemas, and is therefore usable in greenfield as well as in brownfield smart manufacturing environments. The approach shows how data in can be described by defining data source bundles, usable independently from used data source technologies, as long as the data is digitally accessible by the Data Provider. The approach shows also how the data source bundle enables to parameterise data source processors in a data pipelining solution, as demonstrated with Apache NiFi. In summary, it is successfully demonstrated that the approach is suitable for simple extraction of data in heterogeneous data environments.

A range of opportunities for data preparation that could enable a transformation of extracted raw data into standardised data formats or/and schemas are not achieved. These opportunities are further discussed in the outlook section 7.3.6.

A learned lesson is that the elaborated approach is quite flexible and applicable in most heterogenous data environments, as long as data is digitally accessible. However, it is interesting how data will be managed in future manufacturing systems. Will it be a stronger centralised approach in form of Big Data lakes that act as central point for data access? Or will it be a more decentralised approach where data is accessible from several data sources? Will it be a mix of both? However, the presented approach will be suitable for a simple data extraction in heterogenous centralised and decentralised data environments.

As already said in the motivation, people say "data" is the new gold of our century. The value within the growing amounts of data and its broad field of exploitation leads to the expectation that data will increasingly become the focus of many industrial sectors. On the other side, smart manufacturing systems are becoming more and more complex. In order to use their data efficiently, new approaches are needed to enable efficient work with it. This doctoral work makes a small contribution and provides a DIN Spec 91345 RAMI 4.0 compliant data pipelining approach based on a semantic model to support data understanding and data acquisition in such complex heterogenous smart manufacturing data environments.

Overall Conclusion

*This work **contributed** a DIN Spec 91345 RAMI 4.0 compliant data pipelining approach to support data understanding and data acquisition in smart manufacturing.*

*On a **research** level this work successfully combined the ongoing semantic standardisation and unification processes in smart manufacturing with data pipelining approaches to push data exploitation in industrial brownfields and greenfields.*

*The contributed **innovation** is a generic prototype, configurable for individual use cases in (smart) manufacturing environments.*

*The outcomes are **applicable and replicable in use cases** where data structuring, classification and access is needed in heterogenous (smart) manufacturing data environments and where humans need easily to understand, find and use data for sub-sequent forwarding, transformation and/or processing.*

*The outcomes of this work evolved from **international research projects** and were finally validated in the Volkswagen pilot of the H2020 BOOST 4.0 project which was in 2020 the biggest European initiative in Big Data for Industry 4.0.*

*A **video** summarising the results of this work, including a demonstration of the generic prototype, was part of a Journal publication and is accessible in the IEEE Xplore library:*

https://ieeexplore.ieee.org/abstract/document/9296293

7.3 Outlook

The outcomes of this doctoral work open a wide field for further Research, Development and Innovation (R&D&I).

Many potential outlooks were mentioned already (mainly in the Limitations of the prototype solution section 5.7 and in the Validation section 6). This section describes a structured detailed collection.

7.3.1 On engineering tools

For the presented prototype of this doctoral work the ontology editor Protégé (Gennari et al., 2003) was used as engineering support. The Semantic Manager should be extended in future with engineering features that enable to manage I40 components over their life-cycle in a more user-friendly way so that a plain ontology editor is not necessary anymore.

7.3.2 On semantics

The presented semantic model shows an integration of a range of derived standards like eCl@ss classification classes, IEC 61360 vocabulary or a DIN-SPEC-91345 compliant structure. However, many other standards are available and standardisation activities are ongoing. This semantic model could be integrated and extended further in future. Some examples:

- Integration of further standards like AutomationML, outcomes of the SemAnz (Hildebrandt et al., 2017), the presented work of Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS) (IAIS Fraunhofer, 2018), or integration of ontologies like FOAF (Friend of a Friend) (Graves, Constabaris, & Brickley, 2007), W3C Data Catalog Vocabulary (DCAT) (World Wide Web Consortium, 2014) or OntoCAPE (Morbach, Wiesner, & Marquardt, 2009).
- An extension of the schema semantic models e.g. to improve the description of relationships between I40 components. Currently a relationship is only given based on hierarchical levels. More information could generate also new data search functionalities.

7.3.3 On search and filtering in the Data Selector

The current prototype of the Data Selector includes basic functionalities for searching and filtering of data. These functionalities can be extended in future to unleash the full potential of the developed semantic model. This could be done e.g. through following development actions:

- Integration of Google Maps (see www.google.de/maps) based search to find I40 components visually based e.g. on the GPS position.
- Integration of ontology visualisers as WebVOWL (Lohmann, Link, Marbach, & Negru, 2014) to find I40 components and data in a visualised ontology.
- Integration of graph crawlers like Sparklis (Ferré, 2017) to search data in a natural language way.

Also, further research could be done based on this work. For example:

- Further research on visualisations for the Data Selector to identify how visualisations could support the finding of data in smart manufacturing environments. The work of Cambridge Intelligence shows first steps in this direction (see (Disney, 2016; Lanum, 2020)).
- Use the presented semantic model for an integration into virtual or augmented reality to find and select data for a subsequent extraction in the Data Provider.

7.3.4 On security and trust aspects

One gap of this doctoral work is related to security and trust issues. This gap should be filled in future through the integration of security, trust and data anonymisation features – also to cover more aspects of the BDV reference architecture. Some examples:

- If possible, I40 components should communicate in an encrypted and trusted way. Although the presented approach should be kept open for all smart manufacturing environments and stay open for the integration into brownfields, it should be elaborated how newer greenfield I40 components can be integrated in the Data Selector and Data Provider approach in a secure and trusted way. The question is, how certificates and encryption can increase the cybersecurity in smart manufacturing environments and how this could be integrated with the presented approach.
- Advanced data access control: Although the presented approach includes a mechanism for user and user role management, it is only a basic approach. Further investigations could be made on the aspect how a comprehensive data access control should be designed.
- Another open point is to add privacy and anonymisation criteria. Two questions would be:
    - How to enable the data, related to an I40 component, to be anonymised (e.g. in case the data sets contain sensitive private data such as customer names that are not required for subsequent processing).

- How data privacy can be ensured if the Data Selector and Data Provider are deployed as a cloud solution? It could be investigated if smart contracts using blockchain technology could be used to ensure data privacy and data traceability to enable, next to paper-written policies, also a digital enforcement.

7.3.5 On scalability, reliability, performance and other technologies

The presented prototype is successfully validated. However, it has not been investigated how the current implementation will perform in environments with thousands or millions of I40 components. The currently used Apache Jena Fuseki server provides a TDB2 triple store which may reach technological limits as soon as many parallel users use the Data Selector, or if thousands of I40 components are managed. Therefore, a further activity could be the creation of benchmarks: How does the approach perform with thousands of I40 components? What about scalability and reliability? What happens if the Semantic Manager uses as basis technology other triple stores like Neo4J (Van Bruggen, 2014)? First general database technology benchmarks in context of the industrial manufacturing domain were made in (Ramis Ferrer et al., 2021).

A further outlook could be the integration of the Data Provider approach in different technologies. The basis technology used for the Data Provider prototype was Apache NiFi, but how does the approach perform if other data pipeline technologies or also analytics frameworks are used? Like in KNIME (Berthold et al., 2009), Rapid Miner (Ristoski, Bizer, & Paulheim, 2015) or NodeRED[36] to easily find and extract data for a sub-sequent processing or analysis. Or in other data pipelining approaches from StreamSets, Talend or Hevo. Is it possible to initiate a data pipeline also in such frameworks, using the generated DAC of the Data Selector?

7.3.6 On functionality

One thought during the implementation was to extend the functionality of the approach presented in this doctoral work. As this doctoral work had a focus on data, the Data Selector only allows to search for data, but other aspects could be added. Some research questions could be:

- How could the approach be extended to address also I40 component functionalities and business logic?

[36] See https://nodered.org/

- Are classification classes according to eCl@ss and vocabularies according to IEC 61360 also suitable to describe functions, business logic and functional orchestrations in a semantic model based way?
- Could the Data Selector be extended to execute functions/services provided by I40 component instances?

A second thought during the implementation was, that the data source bundle approach could be extended. In this doctoral work only one data source bundle per data set was allowed to parametrise one connector that extracts the data. Partly this would also be needed, like in the native S3-based storage system connector of Apache NiFi. Could this concept be extended with further data transformation operations using multiple data source bundles per data set? Further, this could enable to generate standardised data transformation templates that transform extracted raw data into a specific format. This would require an extension of the single data source bundle to a multi data source bundle approach which adds additional pre-configured data transformation processors to the data pipeline that further pre-processes extracted data.

7.3.7 On genericity

The prototype is compliant to the RAMI 4.0 which focuses on the manufacturing industry. The aim was to build a generic (/domain independent) prototype for the manufacturing industry, but a comprehensive validation on genericity was not achieved yet.

One outlook is therefore, to perform further tests to validate the genericity in the manufacturing and other domains.

7.3.8 On standardisation

The concept, specification and implementation of the DIN-SPEC-91345 compliant semantic model could be used to support the standardisation of semantical data models in the area of smart manufacturing. The "Plattorm Industrie 4.0" has the working group "Reference Architectures, Standards and Standardisation"[37] which develops the foundations for uniform, open standards and carries its ideas into national and international standardisation processes. Contacting this group could be a suitable starting point for collaboration on standardisation.

[37] See www.plattform-i40.de/IP/Redaktion/DE/Standardartikel/arbeitsgruppe-01.html

8

REFERENCES

Aasman, J. (2006). Allegro graph: RDF triple database. *Cidade: Oakland Franz Incorporated, 17*.

Abramova, V., & Bernardino, J. (2013). *NoSQL databases: MongoDB vs cassandra*. Paper presented at the Proceedings of the International C* Conference on Computer Science and Software Engineering, Porto, Portugal.

Adolphs Peter, Heinz Bedenbender, Dagmar Dirzus, Martin Ehlich, Ulrich Epple, & u.a. (2014). Statusreport: Industrie 4.0 - Gegenstände, Entitäten, Komponenten. In White Paper (Ed.).

Ahmad, K., & Ansari, M. (2017). Hands-on InfluxDB. *NoSQL Database for storage and retrieval of data in cloud*, 341-354.

Alexander Fay, Christian Diedrich, Tizian Schröder, Martin Dubovy, Christian Eck, & Wiegand, R. (2017). Semantik für Industrie 4.0-Systeme - Die Basis für den Informationsaustausch in Industrie 4.0-Anwendungsszenarien: Semantische Allianz für Industrie 4.0.

Alley, G. (2019). What Is a Data Pipeline?   , 2019, from https://dzone.com/articles/what-is-a-data-pipeline

Anderson, C. (2015). Docker [software engineering]. *IEEE software, 32*(3), 102-c103.

Andrieu, C., Freitas, N. d., Doucet, A., & Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning, 50*, 5-43. doi: 10.1023/A:1020281327116

Apache Software Foundation. (2019). Apache Airflow Documentation. 2019, from https://airflow.apache.org/

Apache Software Foundation. (2020). Documentation - TDB2.   Retrieved 07.04.2021, from https://jena.apache.org/documentation/tdb2/

Artur Bondza, Christian Eck, Roland Heidel, Markus Reigl, & Dr. Sven Wenzel. (2018). Mit Daten und Semantik auf dem Weg zur Industrie 4.0: eCl@ss e.V.,.

Association, B. D. V. (2016). Big Data Value Strategic Research and Innovation Agenda: Version.

Bangemann, T. (2013, 25.-27.02.2013). *Where is SoA Technology going?* Paper presented at the IEEE International Conference on Industrial Technology (ICIT 2013), Cape Town, South Africa.

Bernus, P., Nemes, L., & Schmidt, G. J. (2003). *Handbook on Enterprise Architecture*: Springer Berlin Heidelberg.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., . . . Wiswedel, B. (2009). KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explor. Newsl., 11*, 26–31. doi: 10.1145/1656274.1656280

Big Data Value Association. (2018). Big Data Challenges in Smart Manufacturing - A Discussion Paper on Big Data challenges for BDVA and EFFRA Research & Innovation roadmaps alignment.

Blazegraph™ DB. (2020). Blazegraph Wiki. Retrieved 07.04.2021, from https://github.com/blazegraph/database/wiki

Bloem, J., Doorn, M. v., Duivestein, S., Excoffier, D., Maas, R., & Ommeren, E. v. (2014). The Fourth Industrial Revolution - Things to Tighten the Link Between IT and OT *VINT research report 3 of 4*: Sogeti VINT.

Bolles, A., Grawunder, M., Jacobi, J., Nicklas, D., & Appelrath, H.-J. (2009). *Odysseus: Ein Framework für massgeschneiderte Datenstrommanagementsystem*. Paper presented at the GI Jahrestagung.

Boman, J., Taylor, J., & Ngu, A. H. (2014). *Flexible IoT middleware for integration of things and applications.* Paper presented at the Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on.

BOOST 4.0 Consortium. (2019a). Big Data Value Spaces for COmpetitiveness of European COnnected Smart FacTories 4.0. 2019, from https://cordis.europa.eu/project/rcn/213926/factsheet/en

Boost 4.0 Consortium. (2019b). Joining forces towards an European Industrial Data Space. Retrieved 05.12.2019, from https://boost40.eu/

Borangiu, T., Morariu, O., Raileanu, S., Trentesaux, D., AO, P. L., & Barata, J. (2020). Digital transformation of manufacturing. Industry of the future with cyber-physical production systems. *Sci. Technol, 23*, 3-37.

Borthakur, D. (2008). HDFS architecture guide. *Hadoop Apache Project, 53*(1-13), 2.

Bowen, J. (2012). *Getting Started with Talend Open Studio for Data Integration*: Packt Publishing Ltd.

Brazil, B. (2018). *Prometheus: Up & Running: Infrastructure and Application Performance Monitoring*: O'Reilly Media.

Broekstra, J., & Kampman, A. (2003). *Serql: A second generation rdf query language.* Paper presented at the Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval.

Brown, B., Chui, M., & Manyika, J. (2011). Are you ready for the era of 'big data'. *McKinsey Quarterly, 4*(1), 24-35.

Bundesministerium für Wirtschaft und Energie. (2019). Das Projekt GAIA-X - Eine vernetzte Dateninfrastruktur als Wiege eines vitalen, europäischen Ökosystems.

Bunge, M. (1977). *Treatise on basic philosophy: Ontology I: the furniture of the world* (Vol. 3): Springer Science & Business Media.

Burns, B., Beda, J., & Hightower, K. (2018). *Kubernetes*: Dpunkt.

Burns, E., Schalk, C., & Griffin, N. (2010). *JavaServer faces 2.0: the complete reference*: McGraw-Hill Education.

Camarinha-Matos, L., Afsarmanesh, H., & Ollus, M. (2008). *Methods and tools for collaborative networked organizations*: Springer.

Camarinha-Matos, L. M. (2016). Scientific Research Methodologies and Techniques - Unit 5: Thesis organization and validation. Retrieved 26.04.2016, from https://sites.google.com/a/uninova.pt/cam/docs/srmt-docs/SRMTunit5-2016.pdf?attredirects=0&d=1

Camarinha-Matos, L. M., & Afsarmanesh, H. (2003). Designing the information technology subsystem *Handbook on Enterprise Architecture* (pp. 617-680): Springer.

Camarinha-Matos, L. M., & Afsarmanesh, H. (2007). A comprehensive modeling framework for collaborative networked organizations. *Journal of Intelligent Manufacturing, 18*(5), 529-542.

Camarinha-Matos, L. M., & Afsarmanesh, H. (2008). On reference models for collaborative networked organizations. *International Journal of Production Research, 46*(9), 2453-2469.

Carter, E. (2018). Docker usage report. *Sysdig [online]. 29 May 2018 [cit. 2019-02-08]. Dostupné z: https://sysdig. com/blog/2018-docker-usagereport*.

Cassandra, A. (2014). Apache cassandra. *Website. Available online at http://planetcassandra. org/what-is-apache-cassandra*, 13.

Chang, W. L., & Boyd, D. (2018). NIST Big Data Interoperability Framework: Volume 6, Big Data Reference Architecture.

Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Computers in industry, 59*(7), 647-659.

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications, 19*, 171-209. doi: 10.1007/s11036-013-0489-0

Cios, K. J., Pedrycz, W., Swiniarski, R. W., & Kurgan, L. A. (2007). *Data mining: a knowledge discovery approach*: Springer Science & Business Media.

Clarkson, J., & Eckert, C. (2010). Design process improvement: a review of current practice.

Colombo, A. W. (2014). *Industrial Cloud-based Cyber-physical Systems: The IMC-AESOP Approach*: Springer.

Colombo, A. W., Gepp, M., Oliveira, J. B., Leitão, P., Barbosa, J., & Wermann, J. (2019). *Digitalized and Harmonized Industrial Production Systems: The PERFoRM Approach*: CRC Press.

Colombo, A. W., & Karnouskos, S. (2009). Towards the Factory of the Future: A Service-oriented Cross-layer Infrastructure. In E. T. S. I. Etsi (Ed.), *ICT Shaping the World: A Scientific View* (Vol. 65-81): John Wiley and Sons.

Colombo, A. W., Karnouskos, S., Yu, X., Kaynak, O., Luo, R. C., Shi, Y., . . . Haase, J. (2021). The IES 70 Year Evolution Journey through Industrial Revolutions-ICT Perspective. *IEEE Industrial Electronics Magazine*.

Copeland, M., Soh, J., Puca, A., Manning, M., & Gollob, D. (2015). Microsoft Azure. *Apress: New York, NY, USA*.

Dais, S. (2017). Industrie 4.0–Anstoß, Vision, Vorgehen *Handbuch Industrie 4.0 Bd. 4* (pp. 261-277): Springer.

Delsing, J. (2017). *Iot automation: Arrowhead framework*: Crc Press.

Denney, M. J., Long, D. M., Armistead, M. G., Anderson, J. L., & Conway, B. N. (2016). Validating the extract, transform, load process used to populate a large clinical research database. *International journal of medical informatics, 94*, 271-274.

Densmore, J. (2021). *Data Pipelines Pocket Reference*: " O'Reilly Media, Inc.".

Derhamy, H., Eliasson, J., Delsing, J., & Priller, P. (2015). *A survey of commercial frameworks for the internet of things.* Paper presented at the Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on.

Dietrich Biester. (May 2012). Siemens entwickelt europäisches Architekturmodell für Smart Grids - Methodik zum Aufbau von Smart-Grid-Projekten für Energieversorger und Industrie. Retrieved 23.03.2019, from https://www.siemens.com/press/de/pressebilder/?press=/de/pressebilder/2012/infrastructure-cities/smart-grid/icsg201205018-01.htm&content[]=ICSG&content[]=EM&content[]=EMSG

DIN, E. (1999). 61512-1, Chargenorientierte Fahrweise, Teil 1, Modelle und Terminologie. *DIN Deutsches Institut für Normung e, 2000*.

DIN SPEC 91345:2016-04. (2016). Reference Architecture Model Industrie 4.0 (RAMI4.0). from http://www.beuth.de/de/technische-regel/din-spec-91345/250940128

Disney, A. (2016). Visualizing very large networks: an update. Retrieved 10.02.2020, from https://cambridge-intelligence.com/visualizing-very-large-networks-an-update/

Döbrich, U., Hankel, M., Heidel, R., & Hoffmeister, M. (2017). *Basiswissen RAMI 4.0: Referenzarchitekturmodell und Industrie 4.0-Komponente Industrie 4.0*: Beuth Verlag.

Dossot, D. (2014). *RabbitMQ essentials*: Packt Publishing Ltd.

Drath, R., & Horch, A. (2014). Industrie 4.0: Hit or Hype? [Industry Forum]. *Industrial Electronics Magazine, IEEE, 8*(2), 56-58. doi: 10.1109/MIE.2014.2312079

eCl@ss e. V. (2019a). eCl@ss WIKI. 2019, from http://wiki.eclass.de/wiki/Main_Page eCl@ss e. V. (2019b). IRDI (International Registration Data Identifier). Retrieved 09.04.2019, 2019, from http://wiki.eclass.de/wiki/IRDI

eCl@ss e. V. (2020). eCl@ss-DownloadPortal. Retrieved 07.02.2020, from https://www.eclassdownload.com/catalog/index.php

eCl@ss e.V. (2018, 9 October 2018). eCl@ss - Property. Retrieved 02.05.2019, from http://wiki.eclass.eu/w/index.php?title=Property&oldid=9069

ECSEL PMB. (2016). 2016 Multi Annual Strategic Research and Innovation Agenda for ECSEL Joint Undertaking. Retrieved 30.05.2016, from http://www.smart-systems-integration.org/public/documents/publications/ECSEL%20MASRIA%202016.pdf

ElMaraghy, H., Lee, J., Kao, H.-A., & Yang, S. (2014). Product Services Systems and Value Creation. Proceedings of the 6th CIRP Conference on Industrial Product-Service SystemsService Innovation and Smart Analytics for Industry 4.0 and Big Data Environment. *Procedia CIRP, 16*, 3-8. doi: http://dx.doi.org/10.1016/j.procir.2014.02.001

Fay, A., Diedrich, C., Dubovy, M., Eck, C., Hildebrandt, C., Scholz, A., . . . Wiegand, R. (2017). Vorhandene Standards als semantische Basis für die Anwendung von Industrie 4.0 (SemAnz40).

Fay, A., Diedrich, C., Thron, M., Scholz, A., Schmidt, P. P., Ladiges, J., & Holm, T. (2015). Wie bekommt Industrie 4.0 Bedeutung? *atp edition, 57*(07-08), 30-43.

Felser, M. (2010). *PROFIBUS-Handbuch: eine Sammlung von Erläuterungen zu PROFIBUS-Netzwerken*: Epubli.

Fernandes, D., & Bernardino, J. (2018). *Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB.* Paper presented at the DATA.

Ferré, S. (2017). Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web, 8*(3), 405-418.

FI-WARE Consortium. (2018). FIWARE: The Open Source Platform for Our Smart Digital Future: Accessed: Sep.

Fielding, R. T., & Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures* (Vol. 7): University of California, Irvine Doctoral dissertation.

Fipa, A. (2002). Fipa acl message structure specification. *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)*.

Forbes Africa. (2019). Data Is The New Gold. Retrieved 17.03.2021, from https://www.forbesafrica.com/technology/2019/07/18/data-is-the-new-gold/#disqus_thread

Freitag, D. (2000). Machine Learning for Information Extraction in Informal Domains. *Machine Learning, 39*, 169-202. doi: 10.1023/A:1007601113994

Garcia, C., Abilio, R., & Malheiros, N. (2015). Approaches and Technologies for Systems Integration: A Case Study at the Federal University of Lavras. *Revista de Sistemas de Informação da FSMA*(15), 11-22.

Garg, N. (2013). *Apache kafka*: Packt Publishing Ltd.

Garijo, D. (2017). *WIDOCO: a wizard for documenting ontologies.* Paper presented at the International Semantic Web Conference.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., . . . Tu, S. W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International journal of human-computer studies, 58*(1), 89-123.

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003, 2003). *The Google File System*.

Givehchi, O., Landsdorf, K., Simoens, P., & Colombo, A. W. (2017). Interoperability for industrial cyber-physical systems: An approach for legacy systems. *IEEE Transactions on Industrial Informatics, 13*(6), 3370-3378.

Golden, B. (2014). As IDC Sees It, Tech's 'Third Platform' Disrupts Everyone. Retrieved 20.07.2016, from http://www.cio.com/article/2377568/cloud-computing/as-idc-sees-it-tech-s-third-platform-disrupts-everyone.html

Grangel-González, I., Halilaj, L., Coskun, G., Auer, S., Collarana, D., & Hoffmeister, M. (2016). *Towards a semantic administrative shell for industry 4.0 components.* Paper presented at the Semantic Computing (ICSC), 2016 IEEE Tenth International Conference on.

Graves, M., Constabaris, A., & Brickley, D. (2007). Foaf: Connecting people on the semantic web. *Cataloging & classification quarterly, 43*(3-4), 191-202.

Group, T. T. W. (2016). The treacherous 12: cloud computing top threats in 2016. *Cloud Security Alliance*.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies, 43*(5-6), 907-928.

Gunnar Knüpffer. (2015). Die Referenzarchitekturen für das Internet of Things und Industrie 4.0. from https: //www.produktion.de/trends-innovationen/die-referenzarchitekturen-fuer-das-internet-of-things-und-industrie-4-0-235.html

Hadlich, T. (2015). *Verwendung von Merkmalen im Engineering von Systemen.* Universitätsbibliothek.

Halilaj, L., Grangel-González, I., Coskun, G., Lohmann, S., & Auer, S. (2016). Git4Voc: collaborative vocabulary development based on git. *International Journal of Semantic Computing, 10*(02), 167-191.

Halilaj, L., Petersen, N., Grangel-González, I., Lange, C., Auer, S., Coskun, G., & Lohmann, S. (2016). *VoCol: an integrated environment to support version-controlled vocabulary development.* Paper presented at the European Knowledge Acquisition Workshop.

Hankel, M. (2018). *Neues zum Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)*. Paper presented at the Forum Industrie 4.0, Bosch Rexroth AG

Hankel, M., & Rexroth, B. (2015). The Reference Architectural Model Industrie 4.0 (RAMI 4.0). *ZVEI*.

Hermann, M., Pentek, T., & Otto, B. (2015). Design principles for Industrie 4.0 scenarios: a literature review. *Technische Universität Dortmund, Dortmund*.

Hevo Data Inc. (2019). Unified Data Platform for Customer Focused Companies - Break Data Silos and Put your Data into Action. 2019, from https://hevodata.com/

Hildebrandt, C., Scholz, A., Fay, A., Hadlich, T., Diedrich, C., Dubovy, M., . . . Wiegand, R. (2017). Semantische Allianz 4.0: Semantische Inhalte für Industrie 4.0. *Tagungsband Automation. VDI*.

Hinze, A., & Buchmann, A. P. (2010). *Principles and Applications of Distributed Event-based Systems*: Information Science Reference.

Hoppe, S. (2016, 13. Juli 2016). OPC UA: Standard für Industrie 4.0 - OPC UA wird (neben anderen) Industrie 4.0-Standard. Retrieved 02.05.2019, from https://www.produktion.de/trends-innovationen/opc-ua-wird-neben-anderen-industrie-4-0-standard-334.html

IAIS Fraunhofer. (2018). VoCol - An Integrated Environment for Collaborative Vocabulary Development. Retrieved 08.10.2018, from https://vocol.iais.fraunhofer.de/

IEEE. (2019). IEEE at a Glance. Retrieved 20.11.2019, from https://www.ieee.org/about/today/at-a-glance.html

International Electrotechnical Commission. (2019). About the IEC - What we do - Facts & figures. 2019, from https://www.iec.ch/about/activities/facts.htm

International Electrotechnical Commission (IEC). (2016a). IEC 62890 Life-cycle management for systems and products used in industrial-process measurement, control and automation: IEC.

International Electrotechnical Commission (IEC). (2016b). IEC TS 62832-1.

Irlan Grangel-González. (2017). RAMIVocabulary. Retrieved 08.10.2018, from https://vocol.iais.fraunhofer.de/rami/

ISO. (2019a). About us: what we do, structure, members, news and events. Retrieved 20.11.2019, from https://www.iso.org/about-us.html

ISO. (2019b). ISO 6523 ICD list. Retrieved 2019, from http://docs.peppol.eu/poacc/billing/3.0/codelist/ICD/

ISO/IEC. (2015). ISO/IEC 11179-6 - Information technology -- Metadata registries (MDR) -- Part 6: Registration. ISO.

ISO/IEC. (2018). ISO/IEC Directives, Part 2 – Principles and rules for the structure and drafting of ISO and IEC documents. 2022, from https://www.iso.org/sites/directives/current/part2/index.xhtml

Jagdev, H. S., & Thoben, K.-D. (2001). Anatomy of enterprise collaborations. *Production planning & control, 12*(5), 437-451.

Jakubetz, C. (2011). *Crossmedia*: Herbert von Halem Verlag.

Jansen, D., & Buttner, H. (2004). Real-time Ethernet: the EtherCAT solution. *Computing and Control Engineering, 15*(1), 16-21.

Jardim-Goncalves, R., Sarraipa, J., & Steiger-Garcao, A. (2010). Semantic harmonization for seamless networked supply chain planning in the future of internet. *Enterprise Architecture, Integration and Interoperability*, 78-89.

Jasperneite, J. (2012). Was hinter Begriffen wie Industrie 4.0 steckt. Retrieved 11.03.2016, 2016, from http://www.computer-automation.de/steuerungsebene/steuern-regeln/artikel/93559/0/

Johansson, L., & Dossot, D. (2020). *RabbitMQ Essentials: Build distributed and scalable applications with message queuing using RabbitMQ*: Packt Publishing Ltd.

Jones, M. (2011). CRISP-DM model, Cross-Industry Standard Process for Data Mining. from http://geuder.tumblr.com/post/3028113424/crisp-dm-model-cross-industry-standard-process

Jung, M. G., Youn, S. A., Bae, J., & Choi, Y. L. (2015, November 2015). *A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment.* Paper presented at the 2015 8th International Conference on Database Theory and Application (DTA).

Kagermann, H. (2017). Chancen von Industrie 4.0 nutzen *Handbuch Industrie 4.0 Bd. 4* (pp. 237-248): Springer.

Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry; Final Report of the Industrie 4.0 Working Group*: Forschungsunion.

Karoline Kopp. (2016). RAMI und IIRA verschmelzen jetzt zum Welt-Modell 4.0. from https://www.produktion.de/iot-by-sap/iot-by-sap/rami-und-iira-verschmelzen-jetzt-zum-welt-modell-4-0-318.html

Kelleher, J. D., & Tierney, B. (2018). *Data science*: MIT Press.

Klein, J., Gorton, I., Ernst, N., Donohoe, P., Pham, K., & Matser, C. (2015). *Performance evaluation of NoSQL databases: a case study.* Paper presented at the Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems.

Kroes, N. (2011). Opening Remarks, Press Conference on Open Data Strategy - Data is the new gold. Retrieved 17.03.2021, from file:///D:/MyDownloads/Neelie_Kroes__Vice-President_of_the_European_Commission_responsible_for_the_Digital_Agenda__Data_is_the_new_gold__Opening_Remarks__Press_Conference_on_Open_Data_Strategy__Brussels__12th_December_2011.pdf

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software, 12*(6), 42-50.

Kurtz, J., & Shockley, R. (2013). Analytics: The real-world use of big data in manufacturing. *IBM Institute for Business Value*.

Kusiak, A. (2017). Smart manufacturing must embrace big data. *Nature News, 544*(7648), 23.

Labrou, Y., & Finin, T. (1999, 1999). *Yahoo! As an Ontology: Using Yahoo! Categories to Describe Documents*.

Lampkin, V., Leong, W. T., Olivera, L., Rawat, S., Subrahmanyam, N., Xiang, R., . . . Keen, M. (2012). *Building smarter planet solutions with mqtt and ibm websphere mq telemetry*: IBM Redbooks.

Lanum, C. (2020). How to visualize very large networks and still uncover insight. Retrieved 10.02.2020, from https://cambridge-intelligence.com/visualize-large-networks/

Lee, J., Lapira, E., Bagheri, B., & Kao, H.-a. (2013). Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing letters, 1*(1), 38-41.

Leitao, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., & Colombo, A. W. (2016). Smart agents in industrial cyber–physical systems. *Proceedings of the IEEE, 104*(5), 1086-1101.

Leitão, P., Pires, F., Karnouskos, S., & Colombo, A. W. (2020). Quo Vadis Industry 4.0? Position, Trends, and Challenges. *IEEE Open Journal of the Industrial Electronics Society, 1*, 298-310.

Lennon, J. (2010). *Beginning CouchDB*: Apress.

Li, Y., & Manoharan, S. (2013). *A performance comparison of SQL and NoSQL databases.* Paper presented at the 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM).

Light, R. A. (2017). Mosquitto: server and client implementation of the MQTT protocol. *Journal of Open Source Software, 2*(13), 265.

Lin, S.-W., Miller, B., Durand, J., Joshi, R., Didier, P., Chigani, A., . . . Bleakley, G. (2015). Industrial internet reference architecture. *Industrial Internet Consortium (IIC), Tech. Rep*.

Lin, S.-W., Murphy, B., Clauer, E., Loewn, U., Neubert, R., Bachmann, G., . . . Hankel, M. (2017). Architecture Alignment and Interoperability: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper *White Paper, Industrial Internet Consortium*.

Lohmann, S., Link, V., Marbach, E., & Negru, S. (2014). *WebVOWL: Web-based visualization of ontologies.* Paper presented at the International Conference on Knowledge Engineering and Knowledge Management.

Luckenbach, T., Stackowiak, R., Licht, A., & Mantha, V. (2015). ORACLE - Improving Manufacturing Performance with Big Data - Architect's Guide and Reference Architecture Introduction. Retrieved 24.06.2016, from http://www.oracle.com/us/technologies/big-data/big-data-manufacturing-2511058.pdf

Lüder, A., & Schmidt, N. (2017). AutomationML in a Nutshell. In B. Vogel-Heuser, T. Bauernhansl & M. ten Hompel (Eds.), *Handbuch Industrie 4.0 Bd.2: Automatisierung* (pp. 213-258). Berlin, Heidelberg: Springer Berlin Heidelberg.

Lukac, D. (2015, 24-26 Nov. 2015). *The fourth ICT-based industrial revolution "Industry 4.0" HMI and the case of CAE/CAD innovation with EPLAN P8.* Paper presented at the Telecommunications Forum Telfor (TELFOR), 2015 23rd.

Macaulay, T., & Singer, B. L. (2012). *Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS*: CRC Press.

MAESTRI Consortium. (2016). The MAESTRI project vision.   Retrieved 06.08.2016, from http://maestri-spire.eu/project/

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. (2011). Big data: The next frontier for innovation, competition, and productivity.

Matthias Ruhl. (2017). *Entwicklung und Implementierung einer Industrie 4.0 Komponente.* (Bachelor Bachelor Thesis), University of Applied Sciences Emden/Leer; ATB Institute for Applied Systems Technology Bremen GmbH.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (2013). *Machine Learning: An Artificial Intelligence Approach*: Springer Science & Business Media.

Molina, A., Panetto, H., Chen, D., Whitman, L., Chapurlat, V., & Vernadat, F. (2007). Enterprise integration and networking: challenges and trends. *Studies in Informatics and Control, 16*(4), 353-368.

MongoDB, I. (2014). Mongodb. *URL https://www. mongodb. com/. Cited on (2014), 9*.

Morbach, J., Wiesner, A., & Marquardt, W. (2009). OntoCAPE—A (re) usable ontology for computer-aided process engineering. *Computers & Chemical Engineering, 33*(10), 1546-1556.

Morik, K. (2008). Der CRISP-DM Prozess für Data Mining. *Wissensentdeckung in Datenbanken SoSe* Retrieved 12.07.2016, from http://www-ai.cs.uni-dortmund.de/LEHRE/VORLESUNGEN/KDD/SS08/02_CRISP_4p.pdf

Mourtzis, D., Vlachou, E., & Milas, N. (2016). Industrial big data as a result of IoT adoption in manufacturing. *Procedia CIRP, 55*, 290-295.

Nagorny, K., Colombo, A. W., & Barata, J. (2014). *A survey of service-based systems-of-systems manufacturing systems related to product life-cycle support and energy efficiency.* Paper presented at the Industrial Informatics (INDIN), 2014 12th IEEE International Conference on.

Nagorny, K., Lima-Monteiro, P., Barata, J., & Colombo, A. W. (2017). Big Data Analysis in Smart Manufacturing: A Review. *Int. J. Communications, Network and System Sciences, 10*.

Nagorny, K., Scholze, S., Colombo, A. W., & Oliveira, J. B. (2020). A DIN Spec 91345 RAMI 4.0 Compliant Data Pipelining Model: An Approach to Support Data Understanding and Data Acquisition in Smart Manufacturing Environments. *IEEE Access, 8*, 223114-223129.

Nagorny, K., Scholze, S., Ruhl, M., & Colombo, A. W. (2018). *Semantical support for a CPS data marketplace to prepare Big Data analytics in smart manufacturing environments.* Paper presented at the 2018 IEEE Industrial Cyber-Physical Systems (ICPS).

Navdeep Singh Gill. (Jul 07, 2017). Data Lake Services for Real-Time and Streaming Analytics. Retrieved 04.12.2019, from https://www.xenonstack.com/blog/data-lake-services/

O'Brien, P. D., & Nicol, R. C. (1998). FIPA—towards a standard for software agents. *BT Technology Journal, 16*(3), 51-59.

O’Brien, R. T. (2018). Data is the new gold.   Retrieved 17.03.2021, from https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Public-Sector/gx-real-estate-data-new-gold.pdf

Obitko, M. (2007). Semantic web architecture. *Ontologies and Semantic Web, available: http://www. obitko. com/tutorials/ontologies-semantic-web/semantic-webarchitecture. html*.

Obitko, M. (2007). Semantic Web Architecture-Introduction to ontologies and semantic webtutorial. Retrieved 03.02.2022, 2022, from https://www.obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

OPC Foundation. (2012). What is the OPC Foundation?: OPC Foundation: The Interoperability Standard for Industrial Automation & Other Related Domains. 2019, from https://web.archive.org/web/20120220234456/http://www.opcfoundation.org/Default.aspx/01_about/01_history.asp?MID=AboutOPC

Otto, B., ten Hompel, M., & Wrobel, S. (2018). Industrial Data Space *Digitalisierung* (pp. 113-133): Springer.

Palankar, M. R., Iamnitchi, A., Ripeanu, M., & Garfinkel, S. (2008). *Amazon S3 for science grids: a viable solution?* Paper presented at the Proceedings of the 2008 international workshop on Data-aware distributed computing.

Panhelainen, A. (2017). Open Messaging Interface and Open Data Format delivers semantic interoperability to the internet of things exploiting Open API.

Paniagua, C., & Delsing, J. (2020). Industrial frameworks for internet of things: A survey. *IEEE Systems Journal, 15*(1), 1149-1159.

Peres, R. S., Jia, X., Lee, J., Sun, K., Colombo, A. W., & Barata, J. (2020). Industrial Artificial Intelligence in Industry 4.0-Systematic Review, Challenges and Outlook. *IEEE Access, 8*, 220121-220139.

Plattform Industrie 4.0. (2018). RAMI4.0 - a reference frameworkfor digitalisation.   Retrieved 04.03.2021, from https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.pdf?__blob=publicationFile&v=7

PLCopen. (2019). What is PLCopen. 2019, from https://plcopen.org/what-plcopen

ProSEco Consortium. ProSEco Project.   Retrieved 06.08.2016, 2013, from https://www.proseco-project.eu/

Pulikottil, T., Estrada-Jimenez, L. A., Rehman, H. U., Barata, J., Nikghadam-Hojjati, S., & Zarzycki, L. (2021). *Multi-agent based manufacturing: current trends and challenges.* Paper presented at the 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).

Ramis Ferrer, B., Mohammed, W. M., Ahmad, M., Iarovyi, S., Zhang, J., Harrison, R., & Martinez Lastra, J. L. (2021). Comparing ontologies and databases: a critical review of lifecycle engineering models in manufacturing. *Knowledge and Information Systems*. doi: 10.1007/s10115-021-01558-4

Ramparany, F., Marquez, F. G., Soriano, J., & Elsaleh, T. (2014). *Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform.* Paper presented at the Big Data (Big Data), 2014 IEEE International Conference on.

Rathwell, G., & Ing, P. (2004). Design of enterprise architectures. Retrieved 30.08.2016, from http://www.pera.net/Levels

Reihe, I. (2009). 61360 IEC 61360-1: 2009-07. *Standard data elements types with associated classification scheme for electric items–Part, 1*, 61360-61362.

Ribeiro, L., & Gomes, L. (2021). Describing Structure and Complex Interactions in Multi-Agent-Based Industrial Cyber-Physical Systems. *IEEE Access, 9*, 153126-153141.

Richardson, L., & Ruby, S. (2008). *RESTful web services*: " O'Reilly Media, Inc.".

Rinaldi, J. S., & Lydon, W. (2015). *Modbus: The Everyman's Guide to Modbus*: CreateSpace Independent Publishing Platform.

Ristoski, P., Bizer, C., & Paulheim, H. (2015). Mining the Web of Linked Data with RapidMiner. *Web Semantics: Science, Services and Agents on the World Wide Web, 35, Part 3*, 142-151. doi: 10.1016/j.websem.2015.06.004

RocketMQ, A. (2022). An open source distributed messaging and streaming data platform. Retrieved 02.01.2022, 2022, from https://kafka.apache.org/

Romero, D., & Vernadat, F. (2016). Enterprise information systems state of the art: past, present and future trends. *Computers in industry, 79*, 3-13.

SAFIRE Consortium. (2017). Cloud-based Situational Analysis for Factories providing Real-time Reconfiguration Services. Retrieved 13.03.2017, from http://www.safire-factories.org/

Samal, B., & Panda, M. (2017). Real Time Product Feedback Review and Analysis Using Apache Technologies and NOSQL Database. *Int. J. Eng. Comput. Sci., 6*(10), 22551-22558.

Sankey, H. R. (1896). *THE THERMAL EFFICIENCY OF STEAM-ENGINES.(INCLUDING APPENDIXES).* Paper presented at the Minutes of the Proceedings of the Institution of Civil Engineers.

Schwab, K. (2016). *The Fourth Industrial Revolution*: World Economic Forum.

SCI 4.0. (April 2018). Alignment Report for Reference Architectural Model for Industrie 4.0/Intelligent Manufacturing System Architecture - Sino-German Industrie 4.0/Intelligent Manufacturing Standardisation Sub-Working Group. In Standardization Council Industrie 4.0 - DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (Ed.): Federal Ministry of Economic Affairs and Energy Department of Public Relations.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Comput. Surv., 34*, 1–47. doi: 10.1145/505282.505283

Seganti, A., Kapłański, P., & Zarzycki, P. (2015). *Collaborative editing of ontologies using fluent editor and ontorion.* Paper presented at the International Experiences and Directions Workshop on OWL.

Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the Semantic Web: Build Flexible Applications with Graph Data*: " O'Reilly Media, Inc.".

Sendler, U. (2009). ProSTEP iViP Verein *Das PLM-Kompendium* (pp. 377-382): Springer.

Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing, 5*(4), 13-22.

Snyder, B., Bosanac, D., & Davies, R. (2017). Introduction to apache activemq. *Active MQ in action*, 6-16.

Soltanpoor, R., & Sellis, T. (2016). *Prescriptive Analytics for Big Data*, Cham.

Solutions, P. E. I. Platform Technology: ThingWorx. 2016. *URL: https://www. thingworx. com/(cited on page 25)*.

Spaggiari, J. M., & O'Dell, K. (2016). *Architecting HBase Applications: A Guidebook for Successful Development and Design*: O'Reilly Media.

Staab, S., & Studer, R. (2010). *Handbook on ontologies*: Springer Science & Business Media.

Stanford-Clark, A., & Truong, H. L. (2013). Mqtt for sensor networks (mqtt-sn) protocol specification. *International business machines (IBM) Corporation version, 1*.

Stevens, R., Goble, C. A., & Bechhofer, S. (2000). Ontology-based knowledge representation for bioinformatics. *Briefings in bioinformatics, 1*(4), 398-414.

StreamSets. (2019). StreamSets Data Collector - Data Collection Challenges with Building Robust Dataflow Pipelines. Retrieved 2019, from https://streamsets.com/products/sdc

Strohbach, M., Daubert, J., Ravkin, H., & Lischka, M. (2016). Big data storage *New horizons for a data-driven economy* (pp. 119-141): Springer, Cham.

Systap. (2015). Blazegraph - Concepts. Retrieved 19.09.2019, from https://wiki.blazegraph.com/wiki/index.php/Concepts

The Apache Software Foundation. (2016). The Cassandra Query Language (CQL). Retrieved 06.11.2019, from https://cassandra.apache.org/doc/latest/cql/

The Apache Software Foundation. (2019). An easy to use, powerful, and reliable system to process and distribute data. Retrieved 2019, from https://nifi.apache.org/

The International Society of Automation (ISA). (2010). ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) Enterprise-Control System Integration - Part 1: Models and Terminology. Retrieved 08.03.2016, 2016, from https://www.isa.org/store/products/product-detail/?productId=116636

The Open Group. (2016). Service Oriented Architecture : What Is SOA? Retrieved 01.04.2016

Trefke, J., Rohjans, S., Uslar, M., Lehnhoff, S., Nordström, L., & Saleem, A. (2013). *Smart Grid Architecture Model use case management in a large European Smart Grid project.* Paper presented at the IEEE PES ISGT Europe 2013.

Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The enterprise ontology. *The knowledge engineering review, 13*(1), 31-89.

Uslar, M., Rohjans, S., Neureiter, C., Andrén, F., Velasquez, J., Steinbrink, C., . . . Strasser, T. (2019). Applying the Smart Grid Architecture Model for Designing and Validating System-of-Systems in the Power and Energy Domain: A European Perspective. *Energies, 12*, 258. doi: 10.3390/en12020258

Van Bruggen, R. (2014). *Learning Neo4j*: Packt Publishing Ltd.

Varaksin, O. (2013). *PrimeFaces Cookbook*: Packt Publishing Ltd.

Varanasi, B., & Belida, S. (2015). HATEOAS *Spring REST* (pp. 165-174): Springer.

VDE Verlag. (2019). IEC 61360 Datenbank für elektrische/elektronische Bauteile. Retrieved 11.01.2020, from https://www.vde-verlag.de/iec-normen/iec-datenbanken/iec-61360-elekronische-bauteile.html

VDI. (April 2015). Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0) *Statusreport*.

Villanova University Alliance. (2021, 2022). What is Big Data? Retrieved 2022, 2022, from http://www.villanovau.com/resources/bi/what-is-big-data/

Vora, M. N. (2011, 24-26 Dec. 2011). *Hadoop-HBase for large-scale data.* Paper presented at the Computer Science and Network Technology (ICCSNT), 2011 International Conference on.

Wang, B., Tao, F., Fang, X., Liu, C., Liu, Y., & Freiheit, T. (2021). Smart Manufacturing and Intelligent Manufacturing: A Comparative Review. *Engineering, 7*(6), 738-757. doi: https://doi.org/10.1016/j.eng.2020.07.017

Wang, Y. (2015). China Automotive Industry IT Application Market Forecast 2015–2019. Retrieved 14.03.2016, 2016, from https://www.idc.com/getdoc.jsp?containerId=CHE40707015

Weber, R. (1997). *Ontological foundations of information systems*: Coopers & Lybrand and the Accounting Association of Australia and New ….

Wei, S., Hu, J., Cheng, Y., Ma, Y., & Yu, Y. (2017). *The essential elements of intelligent manufacturing system architecture.* Paper presented at the 2017 13th IEEE Conference on Automation Science and Engineering (CASE).

Weps, W. (July 2016). Webinar: Manufacturing Analytics in der Umsetzung. *Bosch Software Innovations GmbH.* Retrieved 07.07.2016, from https://www.bosch-si.com/de/newsroom/events/veranstaltungen-72384.html

Wermann, J., Kliesing, N., Colombo, A. W., & Moraes, E. C. (2015, 9-12 Nov. 2015). *Impact of new ICT trends for the educational curriculum in the area of Industrial Automation and engineering.* Paper presented at the Industrial Electronics Society, IECON 2015 - 41st Annual Conference of the IEEE.

Wolfgang Dorst , Carsten Glohr , Thomas Hahn , Frank Knafla , Dr. Ulrich Loewen , Roland Rosen , . . . Michael Sandner (2015). Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0: BITKOM e.V., VDMA e.V. and ZVEI e.V.

World Wide Web Consortium. (2014). Data catalog vocabulary (DCAT).

Wortmann, F., & Flüchter, K. (2015). Internet of things. *Business & Information Systems Engineering, 57*(3), 221-224.

Wu, X., Zhu, X., Wu, G.-Q., & Ding, W. (2014). Data mining with big data. *Knowledge and Data Engineering, IEEE Transactions on, 26*(1), 97-107.

Yu, M.-C., Wu, Y.-C. J., Alhalabi, W., Kao, H.-Y., & Wu, W.-H. (2016). ResearchGate: An effective altmetric indicator for active researchers? *Computers in human behavior, 55*, 1001-1006.

Zur Muehlen, M., Nickerson, J. V., & Swenson, K. D. (2005). Developing web services choreography standards—the case of REST vs. SOAP. *Decision Support Systems, 40*(1), 9-29.

ZVEI. (2016). Structure of the Administration Shell - Continuation of the Development of the Reference Model for the Industrie 4.0 Component. *Federal Ministry for Economic Affairs and Energy (BMWi),*.

9

APPENDIX

9.1 Prototype source code

The source code of the prototype and the semantic model are accessible in the following GitLab project.

Link: https://gitlab.com/kevin.nagorny/semantic-model-based-data-pipelining

Request access and become a project member!

9.2 Demonstrator video

Video in IEEE Xplore library

A video which is summarising the outcomes of this work was part of a Journal publication and is accessible in the IEEE Xplore library.

Link: https://ieeexplore.ieee.org/abstract/document/9296293

Video on YouTube (1080p60)

The same video in a higher resolution (select "1080p60" in the settings) is available on YouTube. The following link and QR code jump directly to the prototype demonstration part.

Link: https://www.youtube.com/watch?v=0maaf6RUzR8&t=1655s

The following timestamps enable a fast navigation:

- 00:20 - Motivation (problems and needs)
- 01:55 - Requirements
- 02:30 - Our approach
- 19:53 - Use case
- 27:35 - Prototype demonstration
- 37:49 - Exploitation potential
- 39:45 - Outlook

9.3 Semantic model documentation

A documentation of the implemented semantic model was generated with the tool "WIzard for DOCumenting Ontologies (WIDOCO)" (Garijo, 2017). The documentation was published online.

Link: http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/documentation/index-en.html

9.4 Communication, Dissemination and Research Interactions

The research, needed for the elaboration of this doctoral work, required several knowledge exchange activities and feedbacks to identify gaps, needs, to stay on track and to confirm frequently the novelty.

In general, this was done by activities like:

- Continuously checking project outcomes/progress with supervisors.
- Continuous discussions with well-informed researchers about the project field.
- Integration of the project in other projects (e.g. EU projects; s. section 9.4.2)
- Continuous publishing of project results and making presentations on conferences with high impact-factors and experts participation.
- Using collaboration supporting tools and platforms as ResearchGate (Yu, Wu, Alhalabi, Kao, & Wu, 2016) to get in contact with researchers in own fields of interest.
- Continuous reading in digital scientific libraries as the IEEE Xplore Digital Library to be up to date.
- Attending international workshops, conferences, etc. to stay up to date.

- Continuously check the research questions and the hypotheses of the project and compare it with the current project process and new knowledge to ensure a positive project progress.

The following subsections contain a collection of material that supports and strengthens the evidence of this doctoral work.

9.4.1 Generated papers

Outcomes of this doctoral work were continuously published to share the results with research groups and also with interested industrial contacts. This was also important to establish a feedback loop which improved the process of the project. A range of publications have been published in scientific conferences and Journals with peer-review procedure.

- **Nagorny, Kevin**; Colombo, Armando Walter; Barata, José; A survey of service-based systems-of-systems manufacturing systems related to product life-cycle support and energy efficiency,2014 12th IEEE International Conference on Industrial Informatics (INDIN),582-587,2014, IEEE
    - **Relation to this work**: Entering data management domain in smart manufacturing systems. Creation of a SotA overview and identification of potential gaps.
    - **Type:** Conference
- **Nagorny, Kevin**; Scholze, Sebastian; Barata, José; Colombo, Armando Walter; *"An approach for implementing ISA 95-compliant big data observation, analysis and diagnosis features in Industrie 4.0 vision following manufacturing systems"*, "Doctoral Conference on Computing, Electrical and Industrial Systems",116-123,2016,"Springer, Cham"
    - **Relation to this work:** Presentation and discussion of first ideas. Obtain expert feedback for possible improvements and avoidance of missteps.
    - **Type:** Conference
- **Nagorny, Kevin**; Lima-Monteiro, Pedro; Barata, Jose; Colombo, Armando Walter; *"Big Data Analysis in Smart Manufacturing: A Review"*, "International Journal of Communications, Network and System Sciences",10,3,31-58,2017, Scientific Research Publishing (SCIRP)
    - **Relation to this work**: Publication of a comprehensive State-of-the-Art analysis.
    - **Type:** Journal

- **Nagorny, K**; Scholze, S; Ruhl, M; Colombo, AW; *"Semantical support for a CPS data marketplace to prepare Big Data analytics in smart manufacturing environments"*,1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS-2018), 2018
    - **Relation to this work**: Presentation and discussion of the detailed concept and approach. Obtain expert feedback for possible improvements.
    - **Type:** Conference
    - **Reviewer Feedback:** It is a nice work with possibly a lot of potential.
- **Nagorny, K**., Scholze, S., Colombo, A. W., & Oliveira, J. B. (2020). *"A DIN Spec 91345 RAMI 4.0 Compliant Data Pipelining Model: An Approach to Support Data Understanding and Data Acquisition in Smart Manufacturing Environments"*. IEEE Access, 8, 223114-223129
    - **Relation to this work**: Presentation of the overall results and the semantic model in detail.
    - **Type:** Journal
    - **Reviewer Feedback:**
        - *"Heterogeneous data processing is a challenge for smart manufacturing. The article focuses on data understanding and data acquisition in smart manufacturing environments, and it is very practical."*
        - *"This paper presented a DIN Spec 91345 RAMI 4.0 compliant data pipelining model. It does an excellent job of highlighting the data understanding and data acquisition in smart manufacturing environments. It is an exciting work. I would like to congratulate the authors for the work presented in this paper."*

9.4.2 Interaction with other research activities

This doctoral work was integrated, linked and influenced in/with/by several research activities which generated several synergies. A few of these synergies are described here.

EU FP7 ProSEco:

ProSEco (Collaborative Environment for Design of AmI enhanced Product-Services Integrating Highly Personalised Innovative Functions with Minimal Ecological Footprint along Life Cycle and of Their Production Processes) (ProSEco Consortium) was a project running from October

2013 – September 2017 and provided a novel methodology and a comprehensive ICT solution for collaborative design of product services (Meta Products) and their production processes.

Interaction of this work with the project: ProSEco was running at the very beginning of this project. It brought first insides in aspects of data management and was suitable for gathering current requirements from industry mainly in collaboration with the participating industrial partner DESMA Schuhmaschinen GmbH, a subsidiary of the Salzgitter AG Group. DESMA offered realistic test beds for data management and analytics tests and showed a first bandwidth of data sources used in industry. The participation in this project generated also a good state of the art overview in the area of data management.

EU H2020 MAESTRI:

MAESTRI (MAESTRI Consortium, 2016) was a Horizon 2020 (H2020)-Project under the Sustainable Process Industries through Resource & Energy Efficiency (SPIRE)-PPP Initiative running from 2015 – 2019 and aimed to provide a resource and energy management system in the form of a flexible and scalable platform, to promote and simplify the implementation of an innovative approach: the Total Efficiency Framework. Based on a holistic approach, which combines different assessment methods and tools, the Framework generates improvement on a continuous basis and increases eco-competitiveness by fostering sustainability in routine operations.

Interaction of this work with the project: Within MAESTRI data analytics methods to identify and improve ecological KPIs in synergy with use case dependent business KPIs were used and elaborated. Also, experimentations with newer communication technologies as an IoT/MQTT based software platform were performed. As part of this project it was identified that heterogenous data with a high veracity and big volumes are a clear challenge in the area of data management. MAESTRI gave therefore first insights in the area of Big Data and was a starting point for the idea of this project synthesised in the research questions and the hypotheses.

EU H2020 SAFIRE:

SAFIRE (SAFIRE Consortium, 2017) was a project running from October 2016 – September 2019 and developed technology and infrastructure to enable reconfiguration as a service for dynamic smart factory systems and manufactured smart products that take advantage of cloud-based services and computing power to continually optimise the performance of manufacturing systems and products with respect to KPIs, including throughput, power consumption, utilisation, maintenance and other factors.

Interaction of this work with the project: SAFIRE was the first project addressing also directly big data challenges. The project brought insights of big data technologies during the collaboration with consortium partners. First parts of the concept of this work could be tested as the data acquisition with Apache NiFi.

EU H2020 BOOST 4.0:

BOOST 4.0 (Big Data Value Spaces for COmpetitiveness of European COnnected Smart FacTories 4.0) (BOOST 4.0 Consortium, 2019a) was a project running from 1 January 2018 until 31 December 2020. The project aimed to demonstrate the need of data-driven connected smart Factories 4.0, by demonstrating in a measurable and replicable way, an open standardised and transformative shared data-driven Factory 4.0 model to meet the Industrie 4.0 challenges (lot size one distributed manufacturing, operation of zero defect processes & products, zero break down sustainable operations, agile customer-driven manufacturing value network management and human centred manufacturing) (BOOST 4.0 Consortium, 2019a).

Interaction of this work with the project: BOOST 4.0 was a good match for this doctoral work because the project had several similar challenges. Some examples are:

- How to deal with the data chaos
- How to describe and classify data
- How to deal with big data environments
- How to exploit the hidden potential in data

BOOST 4.0 which was in 2020 the biggest European initiative in Big Data for Industrie 4.0 with the aim to guide the European manufacturing industry in the introduction of Big Data in the factory, providing the industrial sector with the necessary tools to obtain the maximum benefit of Big Data (Boost 4.0 Consortium, 2019b).

The project provided with 49 European research and industry partners a large collaborative platform for frequent knowledge exchange. Mentionable collaborative partners were:

- Volkswagen AG – As a strong collaborative industry partner who provided industrial and realistic exemplary test environment requirements for this doctoral work.
- Fill Gesellschaft m.b.H. – as machine supplier of the Volkswagen AG and provider of industrial requirements on machine level for this doctoral work.
- University Bonn and Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, because of their expertise on semantic models, vocabularies and registries in the smart manufacturing domain, that was used as SotA knowledge and seed for new ideas.

- International Data Spaces Association – Because its collaborative network brought together key European competences to drive the European data economy.

Collaboration with the University of Applied Sciences Emden Leer, Germany

Co-Supervisor Prof. Armando Walter Colombo is professor at the University of Applied Sciences Emden/Leer and the author of this doctoral work, Kevin Nagorny, studied in this University. Along this project it came to a strong collaboration. This happened in

- frequent virtual and physical meetings with Prof. Armando Walter Colombo,
- collaboration with university students and supervision of bachelor and master theses,
- proposal writing activities, and
- several discussions with colleagues in the university to exchange knowledge

Collaboration with the ATB Institute for Applied Systems Technology Bremen GmbH, Germany

As the author of this doctoral work, Kevin Nagorny, is an employee of the Institute for Applied Systems Technology Bremen GmbH, several synergies were found within the research activities. This happened through

- the generation of new project proposals,
- internal knowledge exchange, and
- contribution of research results generated in this work to projects.

As a well-established and international oriented research institute, ATB was also able to open doors to a very large network of experts in several domains suitable for the personal development and knowledge expansion.

Collaboration with the Universidade Nova de Lisboa, Portugal

As the supervisor of this doctoral work, Prof. José António Barata de Oliveira, is a professor at the Universidade Nova de Lisboa, this work was also integrated with research activities in the university through collaboration with local students through videoconferences or with joint papers as the Journal Paper (Nagorny et al., 2017) together with the PhD Student Pedro Lima-Monteiro.

Other interactions related to this work

Next to the main interactions should be mentioned here, that many other interactions happened during the elaboration of this work. This was happening in

- talks with Prof. Alexander Fay from the Helmut-Schmidt-Universität on the Hannover fair 2017, who has a strong influence in the semantics and meta-models in industrial systems related to Industrie 4.0,

- talks with Prof. Dr.-Ing. habil. Arndt Lüder from the Otto-von-Guericke Universität who is one of the key drivers in AutomationML which is a strong candidate for modelling future smart manufacturing systems,
- talks and collaboration with a group around Prof. Dr. Ulrich Epple from the RWTH Aachen University who is pushing the development of AAS in context of the Industrie 4.0,
- discussions on platforms like ResearchGate or LinkedIn,
- discussions on conferences, workshops and business meetings, and
- discussions with technical oriented friends working in the IT area.

All these interactions have supported this work over the time and should therefore be mentioned.

9.5 Testing Data

This section includes the data used for the testing and result collection mainly referenced in section 5.6.

9.5.1 Configured I40 components in the semantic model

9.5.1.1 *Volkswagen Enterprise ID1*

Table 9-1: Enterprise ID1 - Instance

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_Enterprise1_Instance1*
Type	*Instance*
Object Properties	*hasManifest: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_Enterprise1_Instance1_Manifest*
Data Properties	-

Table 9-2: Enterprise ID1 - Manifest

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_Enterprise1_Instance1_Manifest*
Type	*Manifest*
Object Properties	-
Data Properties	*Irdi: "0174-Volkswagen-1#02-AFY497004#001"^^xsd:string* *version_number: "001"^^xsd:int* *gps_latitude: 54.518612f* *gps_longitude: 12.376111f* *definition: "An example Volkswagen manufacturing plant."^^xsd:string* *preferred_name: "Volkswagen Manufacturing Plant"^^xsd:string* *version_initiated_on: "2012-12-31T23:57:00.000"^^xsd:dateTime* *labels: "{Enterprise, Volkswagen, manufacturing, plant}"^^xsd:string*

9.5.1.2 Work Station ID1

Table 9-3: Work Station ID1 - Instance

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-workstation-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1*
Type	*Instance*
Object Properties	*hasManifest: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_Manifest*

	hasSubModelPropertyCharacterisation: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_SMP_HistoricalStatusMySQL *hasSubModelPropertyCharacterisation: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_SMP_CurrentStatusAsREST*
Data Properties	-

Table 9-4: Work Station ID1 - Manifest

Type	***Description***
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-workstation-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_Manifest*
Type	*Manifest*
Object Properties	-
Data Properties	*labels: "{Work Station, Volkswagen, manufacturing, plant}"^^xsd:string* *preferred_name: "Volkswagen Work Station"^^xsd:string* *definition: "An example Volkswagen work cell in a production line."^^xsd:string* *gps_longitude: 12.376111f* *version_initiated_on: "2013-04-01T23:57:00.000"^^xsd:dateTime* *version_number: "001"^^xsd:int* *gps_latitude: 54.518612f* *irdi: "0174-Volkswagen-1#02-AFY497004#001"^^xsd:string*

Table 9-5: Work Station ID1 – SMPC Current Status REST

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-workstation-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_SMP_CurrentStatusAsREST*
Type	*SubModelProperyCharacterisation*
Object Properties	*hasLifeCycleStage: Generic_LCStage_Instance_MaintenanceAndUsage_rev01* *isSubModelPropertyCharacterisationOf: GenericSubModel_WorkStationStatus_rev01* *hasDataAccessView: GenericView_FunctionalView_rev01* *isRelatedToLayer: GenericLayer_Information_rev01* *usesDataSourceBundle: Test_DataSourceBundle_RESTGet_1*
Data Properties	*value_type: "02"^^xsd:int* *data_type: "String"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AFY497001#001"^^xsd:string* *version_number: "001"^^xsd:string* *version_released_on: "2013-04-01T23:57:00.000"^^xsd:dateTime* *preferred_name: "Current Status"^^xsd:string* *structural_element: "02"^^xsd:string* *labels: "{SMP, current status, work station, REST}"^^xsd:string* *definition: "Provides the current status of the Work Station as a REST service."^^xsd:string*

Table 9-6: Work Station ID1 – SMPC Current Status REST - DSB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_DataSourceBundle_RESTGet_1*

Type	*DataSourceBundle*
Object Properties	*usesDataSourceProcessor: Test_DSB_Processor_REST_InvokeHTTP_1*
Data Properties	-

Table 9-7: Work Station ID1 – SMPC Current Status REST – DSB - Processor

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_DSB_Processor_REST_InvokeHTTP_1*
Type	*org.apache.nifi.processors.standard.InvokeHTTP*
Object Properties	*usesDataSourceProcessor: Test_DSB_Processor_REST_InvokeHTTP_1*
Data Properties	*automaticallyTerminateRelationshipsParameter: "Original"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "Failure"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "Response"^^xsd:string* *processorParameter: "GET"^^xsd:string -> rdfs:label: HTTP Method* *automaticallyTerminateRelationshipsParameter: "No Retry"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "Retry"^^xsd:string* *processorParameter: "https://bigdata-in-industry.com/rest/workstationid1/status/"^^xsd:string -> rdfs:label: Remote URL*

Table 9-8: Work Station ID1 – SMPC Historical Status MySQL

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-workstation-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_SMP_HistoricalStatusMySQL*
Type	*SubModelProperyCharacterisation*

Object Properties	*hasDataAccessView: GenericView_FunctionalView_rev01* *isSubModelPropertyCharacterisationOf: GenericSubModel_WorkStationStatus_rev01* *isRelatedToLayer: GenericLayer_Information_rev01* *hasLifeCycleStage: Generic_LCStage_Instance_MaintenanceAndUsage_rev01* *usesDataSourceBundle: Test_DataSourceBundle_SQLMySQL_1*
Data Properties	*Irdi: "0174-Volkswagen-1#02-AFY497014#001"^^xsd:string* *preferred_name: "Historical status overview"^^xsd:string* *structural_element: "02"^^xsd:string* *labels: "{SMP, historical status, work station}"^^xsd:string* *data_type: "String"^^xsd:string* *version_released_on: "2013-04-01T23:57:00.000"^^xsd:dateTime* *value_format: "String"^^xsd:string* *version_number: "001"^^xsd:string* *definition: "Provides the historical status of the Work Station as in a MySQL DB table."^^xsd:string* *value_type: "02"^^xsd:int*

Table 9-9: Work Station ID1 – SMPC Historical Status MySQL - DSB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-environment-i40comp-dataselector*
Individual	*Test_DataSourceBundle_SQLMySQL_1*
Type	*DataSourceBundle*
Object Properties	*usesPropertyBuilder: Test_PropertyBuilder_SimpleMySQLQuery_1* *usesDataSource: I40Env_SQLDataStore* *usesDataSourceProcessor: Test_DSB_Processor_MySQL_QueryDatabaseTable_1*

Data Properties	-

Table 9-10: Work Station ID1 – SMPC Historical Status MySQL – DSB - Processor

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_DSB_Processor_MySQL_QueryDatabaseTable_1*
Type	*org.apache.nifi.processors.standard.QueryDatabaseTable*
Object Properties	-
Data Properties	*processorParameter: "myTableNAme"^^xsd:string -> rdfs:label: Table Name* *processorParameter: "MySQL"^^xsd:string -> rdfs:label: Database Type* *processorParameter: "DS(DBCPConnectionPool)"^^xsd:string -> rdfs:label: Database Connection Pooling Service* *processorParameter: "PB(Test_PropertyBuilder_Simple-MySQLQuery_1)"^^xsd:string -> rdfs:label: Custom Query* *automaticallyTerminateRelationshipsParameter: "success"^^xsd:string*

Table 9-11: Work Station ID1 – SMPC Historical Status MySQL – DSB – Data Source

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01*
Individual	*I40Env_SQLDataStore*
Type	*DataStore*
Object Properties	-
Data Properties	*port: "3306"^^xsd:int* *ip: "xxx.xxx.102.83"^^xsd:string*

	driver_location: "file:///var/tmp/mysql-connector-java-8.0.15.jar"^^xsd:string preferred_name: "Cassandra NoSQL Data Store"^^xsd:string password: "thisIsNotThePwd"^^xsd:string uri: "jdbc:mysql://xxx.xxx.102.83:3306/mydatabase"^^xsd:string connection_polling_service: "DBCPConnectionPool"^^xsd:string irdi: "0174-Volkswagen-1#02-AGJ284104#001"^^xsd:string driver_class: "com.mysql.jdbc.Driver"^^xsd:string definition: "The Cassandra NoSQL Data Store of the I40 Environment to store documents, log files or time series. Used is a Apache Cassandra Database"^^xsd:string version_initiated_on: "2016-04-01T23:57:00.000"^^xsd:dateTime username: "user"^^xsd:string labels: "{Cassandra, database, documents, Data Store}"^^xsd:string

Table 9-12: Work Station ID1 – SMPC Hist. Status MySQL – DSB – Property Builder

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_PropertyBuilder_SimpleMySQLQuery_1*
Type	*I40SimpleMySQLQuery*
Object Properties	-
Data Properties	*Column: "irdi"^^xsd:string* *column_key: "myIrdi"^^xsd:string* *table_name: "statusWorkstation"^^xsd:string*

9.5.1.3 KUKA Industry robot Type

Table 9-13: KUKA Industry robot - Type

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1*
Type	*Type*
Object Properties	*hasManifest: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1_Manifest* *hasSubModelPropertyCharacterisation: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1_SMP_DefaultConfigurationNoSQL*
Data Properties	-

Table 9-14: KUKA Ind. robot – Manifest (Type)

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1_Manifest*
Type	*Manifest*
Object Properties	-
Data Properties	*definition: "An example Industry Robot Type."^^xsd:string* *preferred_name: "KUKA Industry Robot Type"^^xsd:string* *labels: "{Robot, KUKA, Type, manufacturing}"^^xsd:string* *version_initiated_on: "2011-01-01T23:57:00.000"^^xsd:dateTime* *version_number: "001"^^xsd:int* *irdi: "0161-KUKA-1#02-ASY497204#001"^^xsd:string*

Table 9-15: KUKA Ind. robot (Type) – Default Configuration NoSQL

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1_SMP_DefaultConfigurationNoSQL*
Type	*SubModelProperyCharacterisation*
Object Properties	*usesDataSourceBundle: Test_DataSourceBundle_NoSQLCassandra_1* *isSubModelPropertyCharacterisationOf: GenericSubModel_RobotDefaultConfiguration_rev01* *hasDataAccessView: GenericView_ConstructiveView_rev01* *isRelatedToLayer: GenericLayer_Information_rev01* *hasLifeCycleStage: Generic_LCStage_Type_Development_rev01*
Data Properties	*data_type: "Document"^^xsd:string* *definition: "The default configuration of the KUKA industry robot"^^xsd:string* *labels: "{KUKA, configuration, default}"^^xsd:string* *value_type: "02"^^xsd:string* *irdi: "0161-KUKA-1#02-AGF597204#001"^^xsd:string* *structural_element: "02"^^xsd:string* *preferred_name: "Default configuration"^^xsd:string* *version_number: "001"^^xsd:string* *version_initiated_on: "2013-04-01T23:57:00.000"^^xsd:dateTime*

Table 9-16: KUKA Ind. robot (Type) – Default Configuration NoSQL - DSB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-environment-i40comp-dataselector*
Individual	*Test_DataSourceBundle_NoSQLCassandra_1*

Type	*DataSourceBundle*
Object Properties	*usesPropertyBuilder: Test_PropertyBuilder_SimpleCassandraQuery_1* *usesDataSource: I40Env_NoSQLDataStore* *usesDataSourceProcessor: Test_DSB_Processor_Cassandra_QueryCassandra_1*
Data Properties	-

Table 9-17: KUKA Ind. robot (Type) – Default Config NoSQL – DSB - Processor

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_DSB_Processor_Cassandra_QueryCassandra_1*
Type	*org.apache.nifi.processors.cassandra.QueryCassandra*
Object Properties	-
Data Properties	*processorParameter: "DS(ip):DS(port)"^^xsd:string -> rdfs:label: Cassandra Contact Points* *processorParameter: "PB(Test_PropertyBuilder_SimpleCassandraQuery_1)"^^xsd:string -> rdfs:label: CQL select query* *processorParameter: "DS(password)"^^xsd:string -> rdfs:label: Password* *processorParameter: "i40dataspace"^^xsd:string -> rdfs:label: Keyspace* *processorParameter: "DS(user)"^^xsd:string -> rdfs:label: Username* *processorParameter: "DS(connection_provider)"^^xsd:string -> rdfs:label: Cassandra Connection Provider* *automaticallyTerminateRelationshipsParameter: "failure"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "success"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "retry"^^xsd:string*

Table 9-18: KUKA Ind. robot (Type) – Default Config NoSQL – DSB – Data Source

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01*
Individual	*I40Env_NoSQLDataStore*
Type	*DataStore*
Object Properties	-
Data Properties	*password: "thisIsNotThePwd"^^xsd:string* *preferred_name: "Cassandra NoSQL Data Store"^^xsd:string* *labels: "{Cassandra, database, documents, Data Store}"^^xsd:string* *port: "9042"^^xsd:int* *ip: "xxx.xxx.102.83"^^xsd:string* *username: "cassandra"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AGJ284104#001"^^xsd:string* *version_initiated_on: "2016-04-01T23:57:00.000"^^xsd:dateTime* *definition: "The Cassandra NoSQL Data Store of the I40 Environment to store documents, log files or time series. Used is a Apache Cassandra Database"^^xsd:string*

Table 9-19: KUKA Ind. robot (Type) – Default Config NoSQL – DSB – Property Builder

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_PropertyBuilder_SimpleCassandraQuery_1*
Type	*I40CassandraSimpleKeyspaceTableKeyQuery*
Object Properties	-
Data Properties	*keyspace: "i40dataspace"^^xsd:string*

column_key: "0161-KUKA-1#02-ASY497204#001"^^xsd:string
table_name: "defaultconfigurations"^^xsd:string
column: "typeirdi"^^xsd:string

9.5.1.4 KUKA Industry robot Instance

Table 9-20: KUKA Industry robot - Instance

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1*
Type	*Instance*
Object Properties	*hasSubModelPropertyCharacterisation: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1_SMP_EnergyConsumptionTSDB* *hasManifest: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1_Manifest* *instanceIsbasedOnType: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1*
Data Properties	-

Table 9-21: KUKA Industry robot – Manifest (Instance)

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1_Manifest*
Type	*Manifest*
Object Properties	-

Data Properties	*irdi: "0161-KUKA-1#02-AGY497204#001"^^xsd:string* *version_initiated_on: "2013-04-01T23:57:00.000"^^xsd:dateTime* *definition: "An example Industry Robot Instance."^^xsd:string* *preferred_name: "KUKA Industry Robot Instance"^^xsd:string* *labels: "{Robot, KUKA, manufacturing}"^^xsd:string* *version_number: "001"^^xsd:int*

Table 9-22: KUKA Industry robot (Instance) – SMPC Energy Consumption TSDB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-industryrobot-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1_SMP_EnergyConsumptionTSDB*
Type	*SubModelProperyCharacterisation*
Object Properties	*hasDataAccessView: GenericView_PerformanceView_rev01* *hasLifeCycleStage: Generic_LCStage_Instance_MaintenanceAndUsage_rev01* *isSubModelPropertyCharacterisationOf: GenericSubModel_EnergyEfficiency_ElectricalEnergy_ElectricalConsumptionActual_rev01* *isRelatedToLayer: GenericLayer_Information_rev01* *usesDataSourceBundle: Test_DataSourceBundle_TSDBInfluxDB_1*
Data Properties	*value_format: "integer"^^xsd:string* *version_number: "001"^^xsd:string* *preferred_name: "0161-KUKA-1#02-AGY597204#001"^^xsd:string* *version_initiated_on: "2013-04-01T23:57:00.000"^^xsd:dateTime* *labels: "{SMP, energy consumption, energy}"^^xsd:string* *structural_element: "02"^^xsd:string* *preferred_name: "Energy Consumption"^^xsd:string* *value_type: "02"^^xsd:string*

definition: "Provides the energy consumption of the KUKA Industry Robot"^^xsd:string

Table 9-23: KUKA Industry robot (Instance) – SMPC Energy Consumption TSDB - DSB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-environment-i40comp-dataselector*
Individual	*Test_DataSourceBundle_TSDBInfluxDB_1*
Type	*DataSourceBundle*
Object Properties	*usesDataSourceProcessor: Test_DSB_Processor_InfluxDB_ExecuteInfluxDBQuery_1* *usesPropertyBuilder: Test_PropertyBuilder_SimpleInfluxDBTimeSeriesQuery_1* *usesDataSource: I40Env_TSDBDataStore*
Data Properties	-

Table 9-24: KUKA robot (Inst.) – SMPC Energy Consumption TSDB – DSB - Processor

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_DSB_Processor_InfluxDB_ExecuteInfluxDBQuery_1*
Type	*org.apache.nifi.processors.influxdb.ExecuteInfluxDBQuery*
Object Properties	-
Data Properties	*processorParameter: "1"^^xsd:string -> rdfs:label: Results chunk size* *processorParameter: "DS(username)"^^xsd:string -> rdfs:label: Username* *processorParameter: "0 seconds"^^xsd:string -> rdfs:label: InfluxDB Max Connection Time Out (seconds)* *processorParameter: "PB(Test_PropertyBuilder_SimpleInfluxDBTimeSeriesQuery_1)"^^xsd:string -> rdfs:label: InfluxDB Query*

processorParameter: "DS(password)"^^xsd:string -> rdfs:label: Password *processorParameter: "http://xxx.xxx.102.83:8086"^^xsd:string -> rdfs:label: InfluxDB connection URL* *processorParameter: "i40EnvTSDB"^^xsd:string -> rdfs:label: Database Name* *automaticallyTerminateRelationshipsParameter: "success"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "failure"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "retry"^^xsd:string*

Table 9-25: KUKA robot (Inst.) – SMPC Energy Consump. TSDB – DSB – DataSource

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01*
Individual	*I40Env_TSDBDataStore*
Type	*DataStore*
Object Properties	-
Data Properties	*version_number: "001"^^xsd:string* *port: "8086"^^xsd:string* *labels: "{Database, TSDB, InfluxDB, Data Source}"^^xsd:string* *definition: "Influx DB, a TSDB to save time series"^^xsd:string* *uri: "http://xxx.xxx.102.83:8086"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AGJ890204#001"^^xsd:string* *ip: "xxx.xxx.102.83"^^xsd:string* *version_initiated_on: "2016-04-01T23:57:00.000"^^xsd:dateTime* *preferred_name: "Influx DB"^^xsd:string*

Table 9-26: KUKA robot (Inst.) – SMPC Energy Consump. TSDB – DSB – PB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_PropertyBuilder_SimpleInfluxDBTimeSeriesQuery_1*
Type	*I40SimpleInfluxDBTimeSeriesQuery*
Object Properties	-
Data Properties	*measurement: "timeseries"^^xsd:string* *keys: "time,asset,smpc,value"^^xsd:string* *WHERE: "smpc='0161-KUKA-1#02-AGY597204#001' AND asset='0161-KUKA-1#02-AGY497204#001'"^^xsd:string*

9.5.1.5 Golf 8 Type

Table 9-27: VW Golf 8 (Type)

Type	***Description***
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type*
Type	*Type*
Object Properties	*hasManifest: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type_Manifest* *hasSubModelPropertyCharacterisation: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type1_SMP_GloveCaseCADObjectStore*
Data Properties	-

Table 9-28: VW Golf 8 (Type) – Manifest (Type)

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1_Manifest*
Type	*Manifest*
Object Properties	-
Data Properties	*preferred_name: "VW Golf 8 Type"^^xsd:string* *definition: "An example Golf 8 car type of the Volkswagen company."^^xsd:string* *labels: "{Golf 8, Product, Type, Volkswagen, Car}"^^xsd:string* *version_number: "001"^^xsd:string* *version_initiated_on: "2014-05-01T23:57:00.000"^^xsd:dateTime* *irdi: "0174-Volkswagen-1#02-AFR797004#001"^^xsd:string*

Table 9-29: VW Golf 8 (Type) – SMPC Glove Case CAD Object Store

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type1_SMP_GloveCaseCADObjectStore*
Type	*SubModelProperyCharacterisation*
Object Properties	*isRelatedToLayer: GenericLayer_Information_rev01* *isSubModelPropertyCharacterisationOf: GenericSubModel_EnergyEfficiency_ElectricalEnergy_ElectricalConsumptionCumulativeEnergy_rev01* *hasDataAccessView: GenericView_PerformanceView_rev01*

	hasLifeCycleStage: Generic_LCStage_Instance_Maintenance-AndUsage_rev01
Data Properties	*preferred_name: "Glove Case CAD File"^^xsd:string* *structural_element: "02"^^xsd:string* *labels: "{SMP, CAD, binary, Glove Case}"^^xsd:string* *value_type: "02"^^xsd:string* *version_initiated_on: "2019-12-01T23:57:00.000"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AFR458104#001"^^xsd:string* *definition: "CAD File of the Glove Case"^^xsd:string* *version_number: "001"^^xsd:string*

Table 9-30: VW Golf 8 (Type) – SMPC Glove Case CAD FTP Store - DSB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/link-age-environment-i40comp-dataselector*
Individual	*Test_DataSourceBundle_FTP_1*
Type	*DataSourceBundle*
Object Proper-ties	*usesDataSource: I40Env_FTP* *usesDataSourceProcessor: Test_DSB_Processor_FTP_GetFTP_1*
Data Properties	-

Table 9-31: VW Golf 8 (Type) – SMPC Glove Case CAD FTP Store – DSB - Processor

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individ-ual-data-selector-example01*
Individual	*Test_DSB_Processor_FTP_GetFTP_1*
Type	*org.apache.nifi.processors.standard.GetFTP*
Object Properties	-

Data Properties	*processorParameter: "/"^^xsd:string* *processorParameter: "DS(port)"^^xsd:string* *processorParameter: "DS(username)"^^xsd:string* *processorParameter: "DS(ip)"^^xsd:string* *processorParameter: "DS(password)"^^xsd:string* *processorParameter: false* *processorParameter: "GloveCaseVWGolf8CAD.SLDPRT"^^xsd:stringautomaticallyTerminateRelationshipsParameter: "failure"^^xsd:string* *automaticallyTerminateRelationshipsParameter: "success"^^xsd:string*

Table 9-32: VW Golf 8 (Type) – SMPC Glove Case CAD FTP Store – DSB – DS

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01*
Individual	*I40Env_FTP*
Type	*DataStore*
Object Properties	*-*
Data Properties	*definition: "The FTP server of the I40 environment"^^xsd:string* *ip: "xxx.xxx.102.83"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AFY951404#001"^^xsd:string* *labels: "{FTP, storage, Data Source}"^^xsd:string* *password: "thisIsNotThePwd"^^xsd:string* *port: "21"^^xsd:string* *preferred_name: "FTP Server"^^xsd:string* *username. "admin"^^xsd:string* *version_initiated_on: "2016-04-01T23:57:00.000"^^xsd:dateTime* *version_number: "001"^^xsd:string*

9.5.1.6 Golf 8 Instance

Table 9-33: VW Golf 8 (Instance)

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1*
Type	*Instance*
Object Properties	*instanceIsbasedOnType: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type* *hasManifest: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1_Manifest* *hasSubModelPropertyCharacterisation: Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1_SMP_GPSPositionMQTT*
Data Properties	-

Table 9-34: VW Golf 8 (Instance)– Manifest

Type	*Description*
Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1_Manifest*
Type	*Manifest*
Object Properties	-
Data Properties	*version_number: "001"^^xsd:int* *version_initiated_on: "2019-12-01T23:57:00.000"^^xsd:dateTime* *definition: "An example Golf 8 car instance of the Volkswagen company."^^xsd:string*

preferred_name: "Volkswagen Golf 8"^^xsd:string *labels: "{Golf 8, Product, Instance, Volkswagen, Car}"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AFR497504#001"^^xsd:string*

Table 9-35: VW Golf 8 (Instance) – SMPC GPS Position MQTT

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/example-i40-components-vwgolf8-01*
Individual	*Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1_SMP_GPSPositionMQTT*
Type	*SubModelProperyCharacterisation*
Object Properties	*isRelatedToLayer: GenericLayer_Information_rev01* *hasDataAccessView: GenericView_PerformanceView_rev01* *hasLifeCycleStage: Generic_LCStage_Instance_MaintenanceAndUsage_rev01* *isSubModelPropertyCharacterisationOf: GenericSubModel_EnergyEfficiency_ElectricalEnergy_ElectricalConsumptionActual_rev01* *usesDataSourceBundle: Test_DataSourceBundle_MQTT_1*
Data Properties	*structural_element: "02"^^xsd:string* *version_initiated_on: "2019-12-01T23:57:00.000"^^xsd:string* *value_type: "02"^^xsd:string* *definition: "Current GPS position of the car."^^xsd:string* *irdi: "0174-Volkswagen-1#02-AFR449504#001"^^xsd:string* *labels: "{SMPC, GPS, position, car, MQTT}"^^xsd:string* *version_number: "001"^^xsd:string* *value_format: "String"^^xsd:string* *data_type: "String"^^xsd:string* *preferred_name: "GPS position"^^xsd:string*

Table 9-36: VW Golf 8 (Instance) – SMPC GPS Position MQTT - DSB

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/linkage-environment-i40comp-dataselector*
Individual	*Test_DataSourceBundle_MQTT_1*
Type	*DataSourceBundle*
Object Properties	*usesDataSource: I40Env_MQTTBroker* *usesDataSourceProcessor: Test_DSB_Processor_MQTT_ConsumeMQTTT_2*
Data Properties	-

Table 9-37: VW Golf 8 (Instance) – SMPC GPS Position MQTT – DSB - Processor

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individual-data-selector-example01*
Individual	*Test_DSB_Processor_MQTT_ConsumeMQTTT_2*
Type	*org.apache.nifi.processors.mqtt.ConsumeMQTT*
Object Properties	-
Data Properties	*processorParameter: "myClientId1234"^^xsd:string -> rdfs:label: Client ID* *processorParameter: "20"^^xsd:string -> rdfs:label: Max Queue Size* *processorParameter: "myMQTTTopic"^^xsd:string -> rdfs:label: Topic Filter* *processorParameter: "0"^^xsd:string -> rdfs:label: Quality of Service(QoS)* *processorParameter: "tcp://xxx.xxx.102.83:1883"^^xsd:string -> rdfs:label: Broker URI* *processorParameter: "Clean Session"^^xsd:string -> rdfs:label: Session state* *automaticallyTerminateRelationshipsParameter: "Message"^^xsd:string*

Table 9-38: VW Golf 8 (Instance) – SMPC GPS Position MQTT – DSB – Data Source

Namespace	*http://www.knagorny.de/semantic-models/2021/i40-data-pipelininig/individuals-i40-environment-example01*
Individual	*I40Env_MQTTBroker*
Type	*MessageBroker*
Object Properties	-
Data Properties	*port: "1883"^^xsd:string* *uri: "tcp://xxx.xxx.102.83:1883"^^xsd:string* *version_initiated_on: "2016-04-01T23:57:00.000"^^xsd:string* *preferred_name: "MQTT Broker"^^xsd:string* *ip: "xxx.xxx.102.83"^^xsd:string* *definition: "MQTT Broker for I40 Environment. Used are HiveMQ or Mosquitto."^^xsd:string* *labels: "{MQTT, Broker, Data Source}"^^xsd:string* *irdi: "0174-Volkswagen-1#02-AFY584104#001"^^xsd:string* *version_number: "001"^^xsd:string*

9.5.2 A generated Data Access Catalogue based on Testing Data


```
{
  "DataProviderConfig": {
    "DataEntries": [
      {
        "I40Component": {
          "SubModelPropertyCharacterisation": {
            "SubModelPropertyCharacterisationProperties": {
              "value_type": "02",
              "structural_element": "02",
              "value_format": "integer",
              "definition": "Provides the energy consumption of the KUKA Industry Robot",
              "version_number": "001",
              "preferred_name": "Energy Consumption",
              "version_initiated_on": "2013-04-01T23:57:00",
              "irdi": "0161-KUKA-1#02-AGY597204#001",
              "labels": "{SMP, energy consumption, energy}"
            },
            "SubModelPropertyCharacterisationIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1_SMP_EnergyConsumptionTSDB"
          },
          "DataSourceBundle": {
            "Connector": {
              "ConnectorProperties": {
                "influxdb-dbname": "i40EnvTSDB",
                "influxdb-username": "admin",
                "automaticallyTerminateRelationshipsParameter": "success",
                "influxdb-url": "http://xxx.xxx.102.83:8086",
                "influxdb-query-chunk-size": "1",
                "InfluxDB Max Connection Time Out (seconds)": "0 seconds",
                "influxdb-query": "SELECT time,asset,smpc,value FROM timeseries WHERE smpc\u003d\u00270161-KUKA-1#02-AGY597204#001\u0027 AND asset\u003d\u00270161-KUKA-1#02-AGY497204#001\u0027",
                "influxdb-password": "thisIsNotThePwd"
              },
              "ConnectorIndividualName": "Test_DataSourceBundle_TSDBInfluxDB_1"
            },
            "DataSourceBundleIndividualName": "Test_DataSourceBundle_TSDBInfluxDB_1",
            "DataSourceBundleProperties": {},
            "DataSource": {
              "DataSourceIndividualName": "Test_DataSourceBundle_TSDBInfluxDB_1",
              "DataSourceProperties": {
                "password": "thisIsNotThePwd",
                "port": "8086",
                "ip": "xxx.xxx.102.83",
                "definition": "Influx DB, a TSDB to save time series",
                "version_number": "001",
                "uri": "http://xxx.xxx.102.83:8086",
                "preferred_name": "Influx DB",
                "version_initiated_on": "2016-04-01T23:57:00",
                "irdi": "0174-Volkswagen-1#02-AGJ890204#001",
                "labels": "{Database, TSDB, InfluxDB, Data Source}",
                "username": "admin"
              }
            }
          },
          "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Instance1",
          "Manifest": {
            "ManifestProperties": {
              "gps_longitude": "12.376111",
              "gps_latitude": "54.518612",
              "definition": "An example Industry Robot Instance.",
              "version_number": "1",
              "preferred_name": "KUKA Industry Robot Instance",
              "version_initiated_on": "2013-04-01T23:57:00",
              "irdi": "0161-KUKA-1#02-AGY497204#001",
              "labels": "{Robot, KUKA, manufacturing}"
            }
          },
          "SubModel": {
            "SubModelProperties": {
              "value_type": "2",
              "revision_number": "001",
              "version_number": "001",
              "value_list": "0...*",
              "labels": "{energy, comsumption, actual value}",
              "structural_element": "2",
              "unit_of_measure": "W",
              "definition": "Current, actual electrical consumption.",
              "short_name": "ECA",
              "preferred_name": "Electrical Consumption Actual",
              "version_initiated_on": "2017-12-01T23:57:00",
              "preferred_letter_symbol": "W",
              "irdi": "0174-nagorny-1#02-AFY497006#001"
            },
            "SubModelIndividualName": "GenericSubModel_EnergyEfficiency_ElectricalEnergy_ElectricalConsumptionActual_rev01"
          }
        }
      },
```

```
            {
             "I40Component": {
               "SubModelPropertyCharacterisation": {
                 "SubModelPropertyCharacterisationProperties": {
                   "value_type": "02",
                   "structural_element": "02",
                   "definition": "CAD File of the Glove Case",
                   "version_number": "001",
                   "preferred_name": "Glove Case CAD File",
                   "version_initiated_on": "2019-12-01T23:57:00.000",
                   "irdi": "0174-Volkswagen-1#02-AFR458104#001",
                   "labels": "{SMP, CAD, binary, Glove Case}"
                 },
                 "SubModelPropertyCharacterisationIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-
b507f8e0fa9b_VWGolf8_Type1_SMP_GloveCaseCADObjectStore"
               },
               "DataSourceBundle": {
                 "Connector": {
                   "ConnectorProperties": {
                     "File Filter Regex": "GloveCaseVWGolf8CAD.SLDPRT",
                     "Username": "admin",
                     "automaticallyTerminateRelationshipsParameter": "success",
                     "Port": "21",
                     "Hostname": "xxx.xxx.102.83",
                     "Remote Path": "/",
                     "Delete Original": "false",
                     "Password": "thisIsNotThePwd"
                   },
                   "ConnectorIndividualName": "Test_DataSourceBundle_FTP_1"
                 },
                 "DataSourceBundleIndividualName": "Test_DataSourceBundle_FTP_1",
                 "DataSourceBundleProperties": {},
                 "DataSource": {
                   "DataSourceIndividualName": "Test_DataSourceBundle_FTP_1",
                   "DataSourceProperties": {
                     "password": "thisIsNotThePwd",
                     "port": "21",
                     "ip": "xxx.xxx.102.83",
                     "definition": "The FTP server of the I40 environment",
                     "version_number": "001",
                     "preferred_name": "FTP Server",
                     "version_initiated_on": "2016-04-01T23:57:00",
                     "irdi": "0174-Volkswagen-1#02-AFY951404#001",
                     "labels": "{FTP, storage, Data Source}",
                     "username": "admin"
                   }
                 }
               },
               "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Type",
               "Manifest": {
                 "ManifestProperties": {}
               },
               "SubModel": {
                 "SubModelProperties": {
                   "coded_name": "44010103",
                   "revision_number": "1",
                   "definition": "Assembly group or element of a motor vehicle",
                   "version_number": "4",
                   "preferred_name": "Glove case",
                   "version_initiated_on": "2018-02-25T00:00:00",
                   "irdi": "0173-1#01-AFY260#004",
                   "labels": "{0173-1,AFY260,en,US,4,0,0}"
                 },
                 "SubModelIndividualName": "0173-1--01-AFY260--004"
               }
             }
           },
```

```
          {
           "I40Component": {
             "SubModelPropertyCharacterisation": {
               "SubModelPropertyCharacterisationProperties": {
                 "value_type": "02",
                 "structural_element": "02",
                 "value_format": "String",
                 "data_type": "String",
                 "definition": "Current GPS position of the car.",
                 "version_number": "001",
                 "preferred_name": "GPS position",
                 "version_initiated_on": "2019-12-01T23:57:00.000",
                 "irdi": "0174-Volkswagen-1#02-AFR449504#001",
                 "labels": "{SMPC, GPS, position, car, MQTT}"
               },
               "SubModelPropertyCharacterisationIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1_SMP_GPSPositionMQTT"
             },
             "DataSourceBundle": {
               "Connector": {
                 "ConnectorProperties": {
                   "Broker URI": "tcp://xxx.xxx.102.83:1883",
                   "Max Queue Size": "20",
                   "Client ID": "myClientId-3059ef85-ce32-44f3-8ccf-b388ad210153",
                   "Session state": "true",
                   "Quality of Service(QoS)": "0",
                   "automaticallyTerminateRelationshipsParameter": "Message",
                   "Topic Filter": "myMQTTTopic"
                 },
                 "ConnectorIndividualName": "Test_DataSourceBundle_MQTT_1"
               },
               "DataSourceBundleIndividualName": "Test_DataSourceBundle_MQTT_1",
               "DataSourceBundleProperties": {},
               "DataSource": {
                 "DataSourceIndividualName": "Test_DataSourceBundle_MQTT_1",
                 "DataSourceProperties": {
                   "port": "1883",
                   "ip": "xxx.xxx.102.83",
                   "definition": "MQTT Broker for I40 Environment. Used are HiveMQ or Mosquitto.",
                   "version_number": "001",
                   "uri": "tcp://xxx.xxx.102.83:1883",
                   "preferred_name": "MQTT Broker",
                   "version_initiated_on": "2016-04-01T23:57:00.000",
                   "irdi": "0174-Volkswagen-1#02-AFY584104#001",
                   "labels": "{MQTT, Broker, Data Source}"
                 }
               }
             },
             "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_VWGolf8_Instance1",
             "Manifest": {
               "ManifestProperties": {
                 "definition": "An example Golf 8 car instance of the Volkswagen company.",
                 "version_number": "1",
                 "preferred_name": "Volkswagen Golf 8",
                 "version_initiated_on": "2019-12-01T23:57:00",
                 "irdi": "0174-Volkswagen-1#02-AFR497504#001",
                 "labels": "{Golf 8, Product, Instance, Volkswagen, Car}"
               }
             },
             "SubModel": {
               "SubModelProperties": {
                 "value_type": "2",
                 "revision_number": "001",
                 "structural_element": "1",
                 "definition": "Shows the current GPS position",
                 "version_number": "001",
                 "preferred_name": "GPS Position",
                 "version_initiated_on": "2017-12-01T23:57:00",
                 "irdi": "0174-nagorny-1#02-AFY655686#001",
                 "labels": "{location, GPS, sub model}"
               },
               "SubModelIndividualName": "GenericSubModel_GPSPosition_rev01"
             }
           }
         },
```

```
          {
            "I40Component": {
              "SubModelPropertyCharacterisation": {
                "SubModelPropertyCharacterisationProperties": {
                  "value_type": "02",
                  "structural_element": "02",
                  "data_type": "Document",
                  "definition": "The default configuration of the KUKA industry robot",
                  "version_number": "001",
                  "preferred_name": "Default configuration",
                  "version_initiated_on": "2013-04-01T23:57:00",
                  "irdi": "0161-KUKA-1#02-AGF597204#001",
                  "labels": "{KUKA, configuration, default}"
                },
                "SubModelPropertyCharacterisationIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1_SMP_DefaultConfigurationNoSQL"
              },
              "DataSourceBundle": {
                "Connector": {
                  "ConnectorProperties": {
                    "CQL select query": "SELECT * FROM i40dataspace.defaultconfigurations WHERE typeirdi\u003d\u00270161-KUKA-1#02-ASY497204#001\u0027",
                    "Username": "cassandra",
                    "automaticallyTerminateRelationshipsParameter": "success",
                    "Keyspace": "i40dataspace",
                    "Cassandra Contact Points": "xxx.xxx.102.83:9042",
                    "Password": "thisIsNotThePwd"
                  },
                  "ConnectorIndividualName": "Test_DataSourceBundle_NoSQLCassandra_1"
                },
                "DataSourceBundleIndividualName": "Test_DataSourceBundle_NoSQLCassandra_1",
                "DataSourceBundleProperties": {},
                "DataSource": {
                  "DataSourceIndividualName": "Test_DataSourceBundle_NoSQLCassandra_1",
                  "DataSourceProperties": {
                    "password": "thisIsNotThePwd",
                    "port": "9042",
                    "ip": "xxx.xxx.102.83",
                    "definition": "The Cassandra NoSQL Data Store of the I40 Environment to store documents, log files or time series. Used is a Apache Cassandra Database",
                    "preferred_name": "Cassandra NoSQL Data Store",
                    "version_initiated_on": "2016-04-01T23:57:00",
                    "irdi": "0174-Volkswagen-1#02-AGJ284104#001",
                    "labels": "{Cassandra, database, documents, Data Store}",
                    "username": "cassandra"
                  }
                }
              },
              "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_IndustryRobot1_Type1",
              "Manifest": {
                "ManifestProperties": {}
              },
              "SubModel": {
                "SubModelProperties": {
                  "value_type": "2",
                  "revision_number": "001",
                  "structural_element": "2",
                  "definition": "Provides the default configuration of an industry robot",
                  "version_number": "001",
                  "preferred_name": "Industry Robot Default Configuration",
                  "version_initiated_on": "2017-12-01T23:57:00",
                  "irdi": "0174-nagorny-1#02-AFY656006#001",
                  "labels": "{configuration, default, industry robot, sub model}"
                },
                "SubModelIndividualName": "GenericSubModel_RobotDefaultConfiguration_rev01"
              }
            }
          },
```

```
          {
           "I40Component": {
             "SubModelPropertyCharacterisation": {
               "SubModelPropertyCharacterisationProperties": {
                 "value_type": "2",
                 "structural_element": "02",
                 "value_format": "String",
                 "data_type": "String",
                 "version_released_on": "2013-04-01T23:57:00",
                 "definition": "Provides the historical status of the Work Station as in a MySQL DB table.",
                 "version_number": "001",
                 "preferred_name": "Historical status overview",
                 "irdi": "0174-Volkswagen-1#02-AFY497014#001",
                 "labels": "{SMP, historical status, work station}"
               },
               "SubModelPropertyCharacterisationIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_SMP_HistoricalStatusMySQL"
             },
             "DataSourceBundle": {
               "Connector": {
                 "ConnectorProperties": {
                   "Table Name": "statusWorkstation",
                   "db-fetch-db-type": "MySQL",
                   "db-fetch-sql-query": "SELECT * FROM statusWorkstation WHERE irdi\u003d\u00270174-Volkswagen-1#02-AFY497014#001\u0027",
                   "automaticallyTerminateRelationshipsParameter": "success",
                   "Database Connection Pooling Service": "51a156f9-016e-1000-c9c3-a6c127b888f0"
                 },
                 "ConnectorIndividualName": "Test_DataSourceBundle_SQLMySQL_1"
               },
               "DataSourceBundleIndividualName": "Test_DataSourceBundle_SQLMySQL_1",
               "DataSourceBundleProperties": {},
               "DataSource": {
                 "DataSourceIndividualName": "Test_DataSourceBundle_SQLMySQL_1",
                 "DataSourceProperties": {
                   "controller_service": "51a156f9-016e-1000-c9c3-a6c127b888f0",
                   "driver_class": "com.mysql.jdbc.Driver",
                   "ip": "xxx.xxx.102.83",
                   "uri": "jdbc:mysql://xxx.xxx.102.83:3306/mydatabase",
                   "labels": "{Cassandra, database, documents, Data Store}",
                   "driver_location": "file:///var/tmp/mysql-connector-java-8.0.15.jar",
                   "password": "thisIsNotThePwd",
                   "port": "3306",
                   "definition": "The Cassandra NoSQL Data Store of the I40 Environment to store documents, log files or time series. Used is a Apache Cassandra Database",
                   "preferred_name": "Cassandra NoSQL Data Store",
                   "version_initiated_on": "2016-04-01T23:57:00",
                   "irdi": "0174-Volkswagen-1#02-AGJ284104#001",
                   "username": "user"
                 }
               }
             },
             "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1",
             "Manifest": {
               "ManifestProperties": {
                 "gps_latitude": "54.518612",
                 "gps_longitude": "12.376111",
                 "definition": "An example Volkswagen work cell in a production line.",
                 "version_number": "1",
                 "preferred_name": "Volkswagen Work Station",
                 "version_initiated_on": "2013-04-01T23:57:00",
                 "irdi": "0174-Volkswagen-1#02-AFY497004#001",
                 "labels": "{Work Station, Volkswagen, manufacturing, plant}"
               }
             },
             "SubModel": {
               "SubModelProperties": {
                 "value_type": "2",
                 "revision_number": "001",
                 "structural_element": "2",
                 "definition": "Provides the status of a Work Station",
                 "version_number": "001",
                 "preferred_name": "Work Station Status",
                 "version_initiated_on": "2017-12-01T23:57:00",
                 "irdi": "0174-nagorny-1#02-AFY682006#001",
                 "labels": "{status, work station, sub model}"
               },
               "SubModelIndividualName": "GenericSubModel_WorkStationStatus_rev01"
             }
           }
         },
```

```
          {
            "I40Component": {
              "SubModelPropertyCharacterisation": {
                "SubModelPropertyCharacterisationProperties": {
                  "value_type": "2",
                  "structural_element": "02",
                  "data_type": "String",
                  "version_released_on": "2013-04-01T23:57:00",
                  "definition": "Provides the current status of the Work Station as a REST service.",
                  "version_number": "001",
                  "preferred_name": "Current Status",
                  "irdi": "0174-Volkswagen-1#02-AFY497001#001",
                  "labels": "{SMP, current status, work station, REST}"
                },
                "SubModelPropertyCharacterisationIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1_SMP_CurrentStatusAsREST"
              },
              "DataSourceBundle": {
                "Connector": {
                  "ConnectorProperties": {
                    "automaticallyTerminateRelationshipsParameter": "Original",
                    "HTTP Method": "GET",
                    "Remote URL": "https://api.predic8.de/shop/products/"
                  },
                  "ConnectorIndividualName": "Test_DataSourceBundle_RESTGet_1"
                },
                "DataSourceBundleIndividualName": "Test_DataSourceBundle_RESTGet_1",
                "DataSourceBundleProperties": {},
                "DataSource": {
                  "DataSourceIndividualName": "Test_DataSourceBundle_RESTGet_1",
                  "DataSourceProperties": {}
                }
              },
              "I40compIndividualName": "Test_I40Component_41eafaec-8182-4c10-8f34-b507f8e0fa9b_WorkStation1_Instance1",
              "Manifest": {
                "ManifestProperties": {
                  "gps_latitude": "54.518612",
                  "gps_longitude": "12.376111",
                  "definition": "An example Volkswagen work cell in a production line.",
                  "version_number": "1",
                  "preferred_name": "Volkswagen Work Station",
                  "version_initiated_on": "2013-04-01T23:57:00",
                  "irdi": "0174-Volkswagen-1#02-AFY497004#001",
                  "labels": "{Work Station, Volkswagen, manufacturing, plant}"
                }
              },
              "SubModel": {
                "SubModelProperties": {
                  "value_type": "2",
                  "revision_number": "001",
                  "structural_element": "2",
                  "definition": "Provides the status of a Work Station",
                  "version_number": "001",
                  "preferred_name": "Work Station Status",
                  "version_initiated_on": "2017-12-01T23:57:00",
                  "irdi": "0174-nagorny-1#02-AFY682006#001",
                  "labels": "{status, work station, sub model}"
                },
                "SubModelIndividualName": "GenericSubModel_WorkStationStatus_rev01"
              }
            }
          }
        ],
        "AccessInformation": {
          "User": "admin",
          "DataCartId": "2c91808d7124fa5c017124fabb140000",
          "AccessToken": "2c91808d7124fa5c017124fabb140000"
        }
      }
    }
```

Code 13: Data Access Catalogue Example

2021

Kevin Pierre Nagorny

DIN Spec 91345 RAMI 4.0 compliant data pipelining: An approach to support data understanding and data acquisition in smart manufacturing environments