University of Hradec Králové

Faculty of Informatics and Management

Department of Information Technologies

Modelling a concept for Digital Shadows as a foundation for Digital Twins

Bachelor Thesis

Author: Aleksandra Belousova
Study programme: Applied Informatics

Supervisor: RNDr. Petr Tučník, Ph.D.

Expert consultant:. Andreas Brenner, M.Sc.

Hradec Králové April 2019

Academic Honesty Declaration:

I declare that I have worked on this thesis independently, using only the primary and secondary sources listed in the References.

Hradec Králové, 30.04.2019 Aleksandra Belousova

Aknowledgment:

I wish to thank various people for their contributions to this work. Thank you to Andreas Müller, Christof Küstner and Sonja Dirnberger for their valuable advices and the knowledge they shared. Thank you to both of my interviewees for helping me to gain knowledge about the engineering field, which was new to me.

I would like to give special thanks to Professor Petr Tučník, my university supervisor, for the time he took to patiently answer all of my questions, and to Andreas Brenner, my supervisor from the Schaeffler Group, for giving me the freedom to conduct my research while always providing me with help when I needed it.

Annotation

The purpose of this paper is to demonstrate how semantic technologies can be used to create a base of interoperable data in an engineering domain on which Digital Twin functionality can rely. In this work, the concept of Digital Twins, including its various definitions and types, was analysed. The concept of Digital Shadows was proven to be an essential element of any type of Digital Twin. Semantic technologies have been applied in order to make heterogenous data within a Digital Shadow work together.

Table of contents

List of Figures

List of Tables

1 Introduction and Motivation

Computers are being given increasing number of functions in the modern world. Digitalisation has become a new trend that is pursued by multiple companies across various industries, from medicine to education. The manufacturing industry was introduced early to the business value of digitalisation and took advantage of it. Constantly evolving technologies in the areas of data assimilation, data-driven modelling, machine learning, et cetera led to the opportunity for further modification of the manufacturing industry (1). This change will affect the way companies design, produce and evaluate their products. Unlike in other industries, embedding new technologies in production often immediately improves efficiency and productivity (2). This is why digitalisation continues to influence all manufacturers in one way or another. To effectively deal with the increasing number of new technologies, concepts, and tools, companies need to switch their focus from integrating isolated solutions for specific problems to a global vision of their digitalisation goals. In response to this need, the vision of the further transformation of the manufacturing process through computerization has been introduced. It is called Industry 4.0 (I4). The name Industry 4.0 is a reference to the previous three industrial revolutions. This reference reflects the complexity of changes and the importance of the impact they bring. Industry 4.0 focuses on the digitisation of products, service offerings and new market models (3). One of many goals of this strategy is to enable a shorter production cycle while minimising the number of defect products as much as possible. This aim is a reaction to the increasing demand for individualized products. Industry 4.0 aims to support mass customisation by uniting the advantages of large-scale production with individualization (4).

The vision of Industry 4.0 is based on the idea of merging physical and digital worlds within one cyber-physical production system (CPS). Such a union is possible by means of a dynamic network connecting people, machines, objects, and systems that can be optimised for different needs, as demonstrated in Fig. 1.

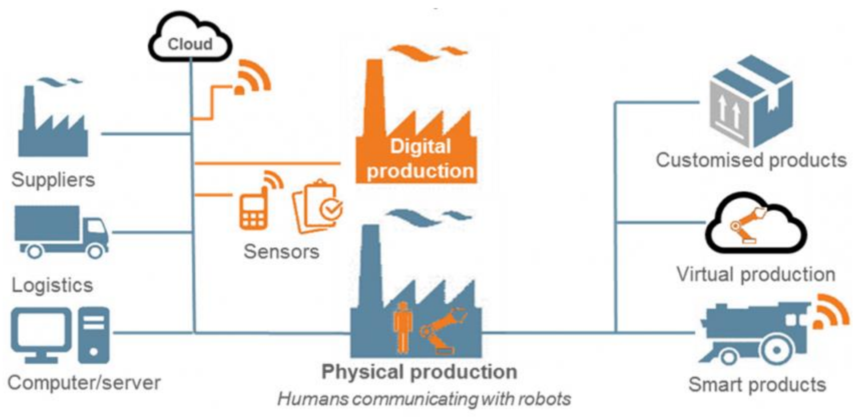


Fig. 1: Physical and digital elements of Industry 4.0
Source: (5)

A Digital Twin (also called an Administration Shell in I4) is an embodiment of such a system. This term refers to the consolidation of two subsystems: a physical one and the digital one that contains all the information about the physical subsystem. These two subsystems are tightly connected throughout the lifecycle of the entire system. However, the idea needs time and means to be implemented. Ubiquitous connectivity, data interoperability, and near-real-time management are hardly achievable when data are fragmented and product models are not made to work together, which is the reality for many companies. In this thesis, the Digital Shadow is promoted as a foundational concept that makes the creation of a Digital Twin possible. The aim of this paper is not only to prove the dependence of the concept of the Digital Twin on the Digital Shadow but also to apply the tools of the Semantic Web to ensure data interoperability for digital models. Therefore, it is expected, that a semantically annotated Digital Shadow can be a foundation for the realisation of a Digital Twin.

2 Literature Research

This section will explore state-of-the-art digital models and semantic tools. The purpose of this section is to understand how the data collected by digital models aid in the task knowledge management.

2.1 Digital Models

Automated and continuous data exchange between the physical and digital worlds opens up many further possibilities for production. One such possibility is the partial or complete autonomy of machines, where a digital instrument can make decisions about the system itself based on the analysis of available data. An equally valuable opportunity is real-time system management based on a continuous flow of data. This level of production control is not possible with a classic approach where all control is manual. The Digital Twin is considered to be a key technology to realise cyber-physical systems (6). In this chapter, digital models that perform such functionality, their value, their types, and their methods of implementation will be analysed.

2.1.1 The Digital Twin

As mentioned above, one of the key features of Industry 4.0 is constant connectivity between digital and real space. This connectivity is enabled by a network of sensors and devices that are interconnected and working together. Digital profiles of physical assets can therefore be merged in cyberspace within an intelligent system that can use data in order to make decisions about the system‘s behaviour. A Digital Twin is an embodiment of such a system.

2.1.1.1 History of the Digital Twin

The concept of the Digital Twin was originally introduced at the University of Michigan in 2002 during a presentation on the formation of product lifecycle management (PLM). The concept of the ‘Conceptual Ideal for PLM’ proposed a new architecture, where one system would consist out of two subsystems: a physical one and one containing all the information about the physical subsystem . These two subsystems would be tightly connected throughout the lifecycle of the entire system (7), as shown on Fig. 2.

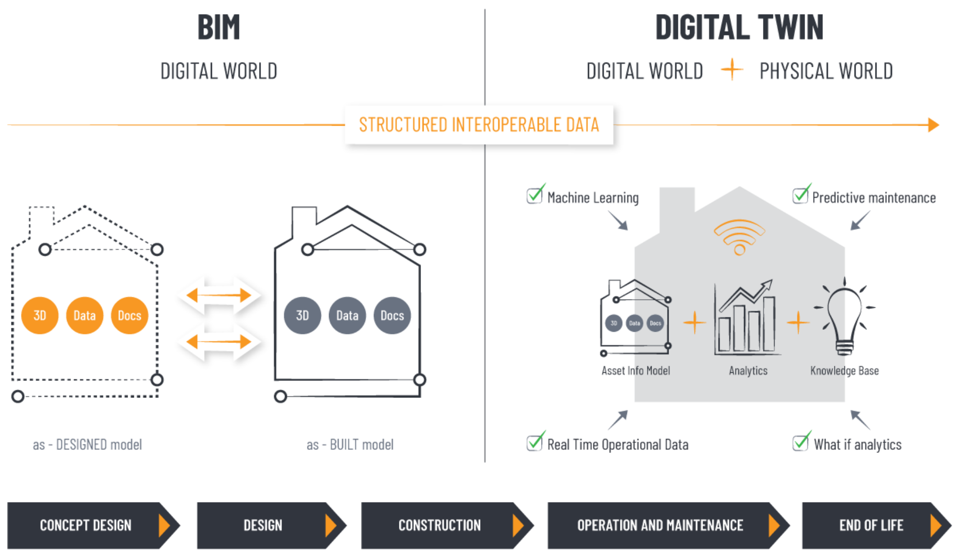


Fig. 2: Digital Twin throughout the lifecycle of its physical counterpart (8)

Such a system would comprise a real space, a virtual space, a connection for data-flow from virtual to real space, and vice versa (9). The new concept was discussed in greater detail at multiple conferences. Later, the Digital Twin was adopted by NASA as a conceptual basis for a new generation of aircrafts (7). In the aerospace industry, the unpredictable and undesirable behaviour of a system has disastrous consequences and may result in fatalities, a damaged reputation, and financial loss. The aerospace industry sees the Digital Twin as a solution to minimise the occurance of such sytem’s behaviour. (7)

2.1.1.2 Definition of the Digital Twin

In the last years, the concept of Digital Twins has become a subject of particular interest among numerous researchers and organisations, not least because of its role in Industry 4.0. For a short time after the concept was introduced to the public, the Digital Twin became a popular topic for research, much of which had a theoretical or even a philosophical nature. However, accompanying the rise of digital models and increased awareness of the capabilities of such an approach, more researches started to closely investigate the ways to implement Digital Twins. On its way from a philosophical idea to a real technology, the definition of a Digital Twin also evolved, transforming from a merely descriptive model to an actionable one. As a result, a high percentage of the definitions for Digital Twins that exist in today‘s literature now look incomplete or too abstract. Table 1: Evolution of definition of a Digital Twin, below, summarises the evolution of Digital Twin definitions and is a result of the research efforts of Negri et al. (10). These scientists were aiming to define the role of the Digital Twin in Industry 4.0 by analysing existing scientific definitions in chronological order.

№	Year	Definition
1	2010 and 2012	An integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin. The digital twin is ultra-realistic and may consider one or more important and interdependent vehicle systems.
2	2012	A cradle-to-grave model of an aircraft structure’s ability to meet mission requirements, including submodels of the electronics, the flight controls, the propulsion system, and other subsystems
3	2012	Ultra-realistic, cradle-to-grave computer model of an aircraft structure that is used to assess the aircraft’s ability to meet mission requirements
4	2013	Coupled model of the real machine that operates in the cloud platform and simulates the health condition with an integrated knowledge from both data driven analytical algorithms as well as other available physical knowledge
5	2013	Ultra-high fidelity physical models of the materials and structures that control the life of a vehicle
6	2013	Structural model which will include quantitative data of material level characteristics with high sensitivity

7	2015	Very realistic models of the process current state and its behavior in interaction with the environment in the real world
8	2015	Product digital counterpart of a physical product
9	2015	Ultra-realistic multi-physical computational models associated with each unique aircraft and combined with known flight histories
10	2015	High- fidelity structural model that incorporates fatigue damage and presents a fairly complete digital counterpart of the actual structural system of interest
11	2016	Virtual substitutes of real world objects consisting of virtual representations and communication capabilities making up smart objects acting as intelligent nodes inside the internet of things and services
12	2016	Digital representation of a real world object with focus on the object itself
13	2016	The simulation of the physical object itself to predict future states of the system
14	2016	Virtual representation of a real product in the context of Cyber-Physical Systems
15	2016	An integrated multi-physics, multi-scale, probabilistic simulation of an as-built system, enabled by Digital Thread, that uses the best available models, sensor information, and input data to mirror and predict activities/performance over the life of its corresponding physical twin
16	2016	A unified system model that can coordinate architecture, mechanical, electrical, software, verification, and other discipline-specific models across the system lifecycle, federating models in multiple vendor tools and configuration-controlled repositories

Table 1: Evolution of definition of a Digital Twin
(10)

The last definition in Table 1 is from 2016. A more recent definition was found in the research of Zheng et al. (11), and it defines a Digital Twin as an '*integrated system that can simulate, monitor, calculate, regulate, and control the system status and process... [and] has the characteristics of individualization, high efficiency and highly* *quasi-real... [and] is developed by data acquisition, virtual manufacturing technologies, based on the control, computation and communication units*'.

Digital Twins can be seen not only as a concept, but also as software or technology. Even though companies such as Microsoft (12) and Vantiq (13) offer their Digital Twin solutions to the industry, there is no finalised or definitive architecture or even scope of functionality. Since the boundaries of Digital Twin definitions are unclear, there are different types of Digital Twins distinguished in the literature, such as those described in Table 2: Types of a Digital Twin.

By generality	A digital model for wide range analyses, such as optimisation of the value stream A digital model for mid-range analyses, such as optimisation of machine processing A digital model for in-depth analyses, such as optimisation of the cutting tool
By purpose (14)	Digital Twins based on data-driven models that capture the structure and dynamics of the targeted plant. Such systems work as black boxes. Simulation-based Digital Twins that have all the needed knowledge (e.g. engineering, physical, chemical) to represent the behaviour of its counterpart
By lifecycle phase (9)	A Digital Twin prototype, or a Digital Twin in the early stages of its existence, when the physical counterpart is not manufactured yet

	A Digital Twin instance that exists and acts in parallel with the life of its manufactured physical twin
By level of aggregation[1] (7)	A Digital Twin prototype describing all of the information needed to build a physical counterpart A Digital Twin instance representing a concrete physical instance of an asset A Digital Twin aggregate aggregating multiple Digital Twin instances
By the scale of an asset (16)	Digital Twins of different functionality, since the Digital Twin is applicable to different types of assets, such as sensors, vehicle subsystems, and vehicle and environment representations.

Table 2: Types of a Digital Twin

Considering the many different types of Digital Twins, each company might create its own definition, that best fits for their technology and its unique purpose, functionality, and complexity.

2.1.1.3 Digital Twins in the Vision of Industry 4.0

Ever since the concept of Digital Twins was first introduced to the public, researchers have done much work to comprehensively study the subject. The high potential for Digital Twin usage has been found in multiple areas, including healthcare (17), smart cities (18), and education (19), among others. However, much of thise research is based on the suggestion that the studied Digital Twin is a complex system made by combining a data-integration module, simulation modules,

[1] *In their work 'Digital twin – a key software component of Industry 4.0', Malakuti at al. (15) call instantiation the third phase of a digital twin lifecycle. According to their research, an 'aggregate' would be a Digital Twin with an implemented mechanism of summarising information from different sources, i.e., a Digital Twin in all phases of its lifecycle.*

and modules of ‘intelligence’; a Digital Twin therefore is able to draw conclusions, make decisions based on data, directly influence the behaviour of its physical counterpart, and simulate physical properties and possible states of a mirrored counterpart and its environment. Such complexity is hard to implement all at once. Manufacturing is an area, where the Digital Twin generates profit throughout all stages of its implementation, from a relatively simple system connecting data to a sophisticated imitation of a manufacturing shop floor. Many manufacturing companes are already using Digital Twins. One of most well-known examples is a Rolls Royce Digital Twin. The company offers its products as a service. The customer not only buys a physical product but also a maintenance service that lasts for the duration of the product’s lifetime (20). Another popular Digital Twin is that of Tesla; every manufactured car has a Digital Twin, which is provided with instant service (for example, a software-based door fix). This service is delivered by constantly transmitting data between the car and the factory (21). Many other companies start by implementing a digital model of a basic functionality and extend it over time. Research (22) based on the results of interviews with anonymised representatives of the German manufacturing industry found that most of the interviewed companies already use this technology for simulation and data-gathering purpose, and all of these companies are planning to further enhance their Digital Twins in the near future.

In the context of Industry 4.0, the Digital Twin concept adopts the form of an administration shell of I4 components (3). Therefore, this research will study the Digital Twin concept in connection with cyber physical systems of Industry 4.0, that is, the system merging physical and cyber spaces into one network, where the multiple physical entities exchange data with a virtual space using their digital models (23). The Digital Twin provides means to organise and manage such data. However, the data collected by digital models is not limited to sensor data. According to many researchers, a Digital Twin is, first of all, a PLM tool that provides a way to merge data produced in different phases of a lifecycle into one standardised model, thereby facilitating a seamless integration of different lifecycle phases (24). During the asset lifecycle, a large amount of non-dynamic data is produced. Business characteristics, geometric and spatial features, and bills of materials are just a few examples (see Table 3: Data produced in different stages of a lifecycle). Traditional approaches do not provide efficient means to manage such data, so this information stays fragmented and isolated (25). Such an issue makes a global analysis or a complex understanding of the information described by this data a very complicated, sometimes nearly impossible process.

Conceptual phase	Construction phase	Utilization phase
Function and aesthetic design data, market competition data, investment strategy data, model data, historical data, data of customer reviews and feedbacks, et cetera	Structural data, color data, mechanic data, size data, material data, energy data, configuration and parameter optimization data, historical data, customer review data, et cetera	Manufacture data, process data, manual operation data, environment data, fault data, behavioral data, simulation data, prediction data, environmental influence data, et cetera

Table 3: Data produced in different stages of a lifecycle
(25)

Abramovici et al. (24) proposed the following use case to demonstrate the possible business value that digital models can provide. The traditional approach of car engine prototyping requires the manufacturer to wait for all the physical components to be manufactured and assembled together during the early phases of the engine's lifecycle. Consequently, a quality check of components, and recognition of construction mistakes can be completed only when the components are actually produced. This solution can be time- and cost-consuming, when an error occurs. By implementing a Digital Twin, the engine's quality could be evaluated long before the components are manufactured. This evaluation could be achieved if the car engine were modelled in a virtual space, where the spatial and geometrical properties as well as product structure and a basic description of the build process would be connected together, and the build process would be simulated, enabling the detection of possible problems. Therefore, the Digital Twin helps to optimise resources by assuring that they are not used to build a defective assembly. Many other use cases can be found in the literature and in industry.

A significant contribution to the understanding of requirements for the Digital Twin in context of Industry 4.0 was done by Chiabert et al. (22). The authors conducted a literature review of a qualitative and quantitative nature in order to discover requirements set by the industry. Based on their research, the authors defined 26 possible characteristics that can be expected from a Digital Twin. The most popular requirement is real-time data, which is used to optimize business processes using knowledge about the current status of the product. Another requirement mentioned multiple times is data integration. The final one of most frequently mentioned requirement is fidelity, since a high-fidelity model dramatically increases the number of potential use cases where a digital model can be used, especially use cases that require different kinds of simulations.

2.1.1.4 Implementation of the Digital Twin

Parrot & Warshaw (26) described how the Digital Twin enables interaction between the physical and digital world. They described the process as '*thousands of sensors taking continuous, nontrivial measurements that are streamed to a digital platform, which, in turn, performs near-real-time analysis to optimise a business process in a transparent manner.*' Parrot and Warshaw used the model in Fig. 3 to express the dual nature of a Digital Twin and the loop between digital and physical world that can be enabled by this technology.

According to the company's needs, any Digital Twin should be implemented step-by-step using agile techniques. '*Digital Twin of an asset matures during the lifecycle of a plant*' (27). On a functional level this means starting with the most important components that fit the current digitalisation level of the company and add immediate value, then slowly expanding to the scope that would fully satisfy company's expectations of a Digital Twin.

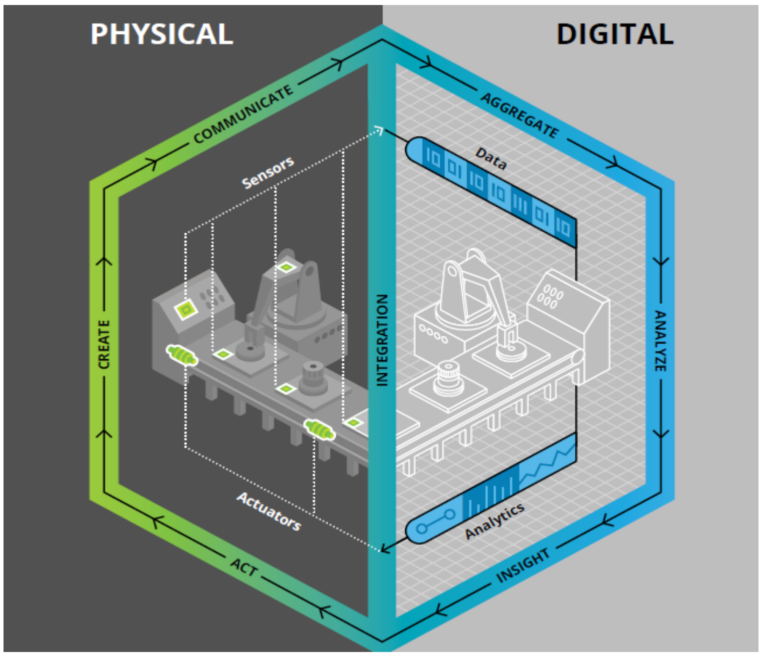


Fig. 3: The loop between physical and digital worlds
(26)

Alam and El Saddik (28) propose the following principle of work in their research. Independent systems aiming to achieve a common goal should be connected. Assuming that data is generated by Digital Twins and every physical asset has its own Digital Twin, these Digital Twins will get an update whenever the physical world changes. An important characteristic of the architecture is ubiquitous Internet connection. In order to interact, every part of such a system needs to be aware of its own existence and have a unique identification (i.e. ipV6, UPC, EPC, RFID, etc.). Every physical asset and its corresponding Digital Twin manage a data store (29). Data that need to be stored include not only dynamic, real-time data but also expert knowledge, historical data, inferred data, and data integrated from other enterprise systems (30). When needed, data are extracted from data stored, and a special submodule retrieves the required information. Finally, in order to be able to handle such large amounts of data, human-computer interfaces must be implemented. Through this interface, a Digital Twin can filter and present human-readable information in accordance with security or privacy limitations (29). Deloitte research (28) also proposes a method of developing the Digital Twin. Their approach will be briefly described below. According to this research, the process of creating a Digital Twin should start by determining the appropriate level of detail and deciding which processes and integration points between different Digital Twins will be modelled. Each process must then be designed in order to demonstrate the sequential model of interaction between people, physical assets, information, and applications.

The first step towards implementing a Digital Twin is to outfit the physical asset with sensors that will take operational and environmental measurements. To extend the data profile of a Digital Twin, sources containing process-based information, such as CAD models, supply chain systems and so on, can be added. Next, the creation of real-time connectivity between the physical process and a digital model and vice versa must be implemented. This connection is comprised of edge-processing, edge-security, and communication interface components. The creation of the connection is followed by data integration. This can be done either in the cloud or on a local server. In this step, data are prepared for analysis and then saved in a common repository. These data will be used by specialists to analyse and visualise the performance of the Digital Twin. When the derived information becomes a resource for optimizing a physical asset, it will be transformed such that is can be fed into the actuators of the asset process. In this step, the loop between the physical and digital systems is closed.

2.1.2 **Digital Shadow**

As was demonstrated, the core of Digital Twin architecture is presented by a data structure model. One of the names used for this model in literature is the Digital Shadow. The Digital Shadow can be called a data container; it creates a means for the data from different sources to be stored together in a way that makes them compatible with each other (27).

The term Digital Shadow appears to be less popular than the term Digital Twin[2]. Therefore, in some sources the concept of merging real-time data and historical data can be found under the name Digital Twin[3]. In this research the two terms are distinct, despite the fact that some researchers see a Digital Shadow as a simplified Digital Twin. The reason for distiguishing these two terms is that the Digital Shadow can be used autonomously or as an early phase of a Digital Twin implementation. Since the Digital Twin cannot exist without a proper data structure model, the Digital Shadow (which is a data profile) must be implemented first and further enriched by actuators to orchestrate physical systems, as shown on Fig. 4: Relationship between the Digital Twin and the Digital Shadow.

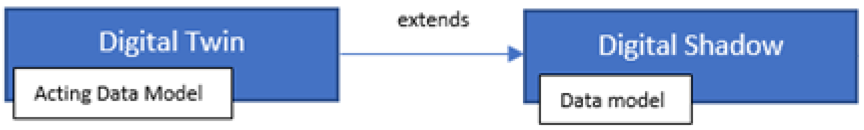


Fig. 4: Relationship between the Digital Twin and the Digital Shadow

Similar approaches can be found in other researches. For example 'Digital Shadow in the Internet of Production' (16). Jarke et al. write '*The "digital twin" is an active simulation aiming to run in parallel, and interact with, a "real" physical, technical, socio-technical, or business system. In engineering, digital twins are often executions of very rich and powerful, multi-parameter models – in the continuous case complex differential equation systems, in the discrete case, they might represent an entity like a business process model instantiation... Digital shadows are abstracted traces captured by sensors of the "real" system*'. According to Wahlster (31), Digital Shadows enhance products by creating the ability to capture and interpret ambient conditions and user actions, therefore providing a basis for perceiving and controlling the environment, observation analysis, and communication with other smart objects or humans.

[2] This finding is based on the results of a 0,08 second search in Google Scholar on the 15th of January 2019 using the search words 'Digital Shadow' and 'Industry 4.0' and then using 'Digital Twin' and 'Industry 4.0', which resulted in 104 against 1100 search results.
[3] Sometimes the combination of a real-time data and historical data within the digital model is referred to as a replication mode of a Digital Twin (1).

The suggestion that a Digital Shadow is a foundation for a Digital Twin is supported by the research of Olivotti (32) . The author proposes that the architecture of data foundations for a Digital Twin, which would include data from existing IT systems, data describing installed components, location data, maintenance protocols, and real-time data, should use sensors and databases as sources. The data foundation do not only combines data in one container but also creates means for data to be used together. A system built using this foundation would be able to store, process, and analyse the data. Predictive maintenance, learning from the stored data, and providing understandable knowledge at the right time and at the right place are few of the tasks defined for such a system. A detailed explanation of the proposed architecture can be found in the appendix A 1**.**

A Digital Twin and a Digital Shadow share many commonalities, and a multitude of Digital Shadow characteristics can be seen as a subset of Digital Twin characteristics. Consequently, in this thesis the characteristics that can be applied to both digital models are refered to as being associated with a Digital Twin/Digital Shadow, whereas the concepts that are unique to Digital Twins will accordingly be referred to as concepts of Digital Twins.

2.2 Knowledge and Knowledge Management

According to Cecchinel et al. (33), approximately 35 zettabytes of the world's data will be produced in this decade. Compared to the last decade, when the produced data amounted to 'only' 0.8 zettabytes, it is clear that the amount of data produced by computers, Internet users, smart products, et cetera is rapidly growing. This volume of information creates much more than a srotage problem. The question of how to effectively find, sort, and use this data becomes inreasingly urgent. While a growing amount of data and therefore an increasingly complicated searching process might not seem very important for an average Internet user, there are organisations that are traditionally deeply dependent on knowledge creation, learning, flexibility, and continuous improvement (34). Since manufacturing companies are a notable example of such organisations, improving knowledge management processes in order to support business operations and decision making is an absolute necessity in a fast-growing industry.

The goal of this chapter is to provide the reader with a general idea of the meaning of knowledge, including ways of expressing it in a formal way, to allow for it to be processed by machines, and an overview of common tools to do so. The first section of the chapter gives a general overview of knowledge and provides the reader with a brief explanation of how knowledge is obtained by humans and, in contrast, by machines. The next two sections describe how and why knowledge gets managed in the engineering domain. The fourth section presents a rather informal description of standard ways of formalising knowledge and is intended for a reader of any background who wants to understand the main principles. Finally, the fifth section describes tools for data annotation that are commonly used in order to provide means for machines to work with information. This step is foundational for automating data manipulation and especially knowledge-retrieval processes.

2.2.1 **Knowledge Definition**

The Oxford dictionary defines knowledge as '*Facts, information, and skills acquired through experience or education; the theoretical or practical understanding of a subjec*t' (35). Increasing one's knowledge of the world appears to be an intuitive and almost effortless task for a human. To mimic this process, a machine must not only have access to data but also be capable of connecting this data in meaningful ways. There are multiple approaches taken, to describe how connecting data serves as a foundation for acquiring knowledge. In computer science, one of the most famous approaches is called the Data, Information, Knowledge, and Wisdom (DIKW) pyramide (36). These concepts are represented as a hierarchy, where every layer is dependent on a previous one, since every new level is built upon a level with a lower level of abstraction. According to Ackoff (36), in order to retrieve any information from data, a relationship between these data must be discovered. Once patterns are recognised, the information becomes knowledge. Finally, when principles of retrieving new knowledge are understood, knowledge becomes wisdom.

The DIKW approach looks rather simplified and does not consider the role of a context and other possibly relevant points. There are both supporters and opponents of this approach. One of the critics of the classic DIKW model is Jennex, who says '*It is posited that the knowledge pyramid is an artifact of KM [knowledge management] processes and not an artifact of reality*' (37). In his work, a new DIKW pyramid is proposed. The new model that can be seen on Fig. 5 demonstrates, that gaining competence using data does not happen in a vacuum but is strongly influenced by other processes, such as filtering, placing information into the context of previous experience, social context, and more.

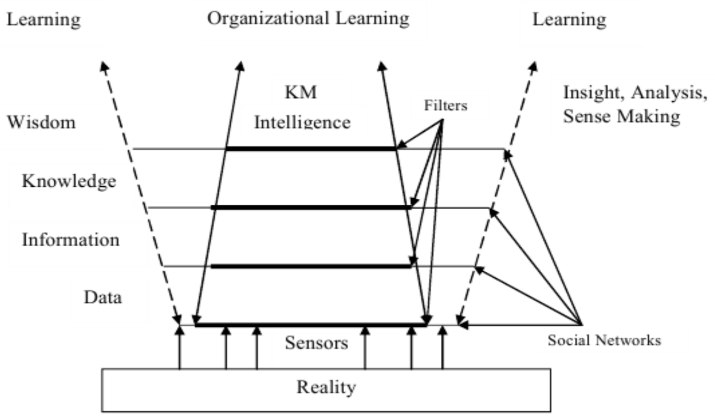


Fig. 5: Revised knowledge pyramid
(37)

According to Jennex (37), data are explicitly expressed facts and answer the questions *who, what, when,* or *where.* If there is context, the data relations can be understood and connection to *who, what, when* or *where* can be made. Culturally understood information becomes knowledge that can answer questions such as *how* and *why*. As Rolstadås (38) say '*knowledge can be viewed as the result of an interaction between intelligence... and situation*'. Finally, wisdom is applied knowledge.

There is another important detail that can be seen in the newer model demonstrated on Fig. 5: Revised knowledge pyramid. The reversed pyramide implies that there is more information than data, more knowledge than information, and more wisdom than knowledge. This hierarchy happens due to the fact that different people have different frames of reference for processing data and different ethical, religious, or cultural beliefs that influence the way people interpret information and generate wisdom (37).

In the context of the described approach, modelling an informational structure of a domain can be seen as formalising knowledge of the domain. In fact, according to the reversed pyramid scheme there can be more models structuring the same information from different points of view, each of which will be correct in the context of its area of application.

It is important to mention that modelling knowledge is a non-trivial task, especially considering multiple possible interpretations, since not all information is easily accessible and formalizable. Conditionally, knowledge can be explicit or tacit[4]. Explicit knowledge, also known as codified knowledge, can be communicated and documented relatively easily, while tacit or unarticulated knowledge is highly personal and is influenced by the one who holds it, his experience, perspectives, and world view. Tacit knowledge is therefore challenging to express (38). As Rolstadås et al. (38) say: "*Even if knowledge is to a large extent tacit in an organisation, it can often also be made more explicit if the contexts are understood by others and it is possible to structure and codify it, if not in absolute and mathematical terms, then perhaps in written text or through indicators.*" Dealing with both tacit and explicit knowledge is a task performed within knowledge management.

2.2.2 Knowledge Management

Knowledge management is a set of activities that contribute to the creation, storage, distribution, and application of knowledge (40). This discipline aims to meet the

[4] Some authors (39) also distinguish an implicit knowledge

growing requirements of the industry by providing tools, techniques and processes for the most effective exploitation of the intellectual assets of organisations. Good management of such assets helps to improve business processes, avoid duplicate efforts, generate new business opportunities, and bring new products and services to the market ahead of competitors (41).

Integrating resources and connecting related information across an organisation are two of the most fundamental tasks of knowledge management. One way to make this connection is to integrate databases and knowledge bases. Ontologies can help to deal with heterogeneous representations of data by adding structure and semantics (41).

2.2.3 Formalisation of Knowledge

One of way to express knowledge is using description logic(DL) to describe ‘things’ and their relationships. DL is a family of languages that uses formal, logic-based semantics to represent knowledge in a structured way (42). Description logic allows one to represent knowledge in a univocal way and provides means for reasoning. There are many languages using DL that differ by their levels of expressivity. High-level world descriptions provided by DLs can be effectively used to build intelligent applications (43). Since formalising all of the knowledge available to humans is an impossible task, formalisation is only applied to the knowledge specific to an application domain and limited by predefined use cases.

The most foundational element of DL is a vocabulary: a set of terms used to describe an application domain. Description logic systems allow for representation of the following components: concepts, roles, and names. The DL vocabulary is a triple of disjointed sets (Nc, Nr, Ni), where the element Nc represents concept name, the element Nr represents role names, and the element Ni represents individual names. ‘Student’, ‘Lesson’, ‘Homework’, ‘Date’ and ‘Online_System’ are few examples of concept names that are used to represent classes in ontology; ‘Tomas_Novak’, ‘MATH2_L4’, ‘11/05/2018’, and ‘Oliva_Blackboard’ represent individuals; and ‘uses’, ‘isUsedBy’, ‘hasDeadline’, and ‘submitted’ indicate role names, that is, binary relationship between concepts or their individuals.

In DL, concepts can be primitive or defined. Primitive concepts are defined by specifying necessary conditions for the individuals of a concept, while defined concepts specify both necessary and sufficient conditions (44). The concept ‘child’ can be considered a primitive concept, if its definition includes only necessary conditions, such as 'a child must be under the age of 10’. However, if it is described by both necessary and sufficient conditions, such as ‘a child must be under the age of 10 and if there is a human who is under the age of 10 it is a child’ - it is a defined concept. Roles can also be primitive or defined and obey the same principles of sufficient and necessary conditions.

A DL theory is divided into assertional box (ABox) and a terminological box (TBox), where the TBox is a set of axioms describing a specific domain in terms of general properties of concepts and roles, and the ABox is a finite set of facts that define participation of certain individuals in some concepts or roles from the TBox. An ABox has a more dynamic nature than a TBox; the latter consists of stable knowledge that is valid for a longer period of time. A TBox might include statements like ‘childrenBook≡⌐adultBook’, ‘Novel ⊆ Literature’, and ‘PopularNovel ⊆ Novel ⋂ isTraslatedTo.Language’. An ABox contains assertions, for example ‘Novel (UnbearableLightnessOfBeing)’, ‘Author(Kundera)’, and ‘hasAuthor (UnbearableLightnessOfBeing, Kundera)’. In the context of semantic modelling, the TBox is usually stored in ontologies, whereas the ABox is saved in repositories and represents descriptions of certain entities of the world described in a TBox. Together, an ABox and a TBox create a knowledge base.

One commonly used DL is the Attributive Concept Language with Complements (ALC). The ALC allows for the following concept constructors: universal restrictions (∀p.C) that can be pronounced in a natural language as ‘for every p C is true’; unqualified existential quantification (∃p.C) that is pronounced as ‘there exists at least one p for which C is true’; conjunction (C1⋀C2) that can be read as ‘C1 and C2’ and means that if one variable is truth another is also truth; disjunction ($C1 \vee C2$) read as ‘C1 or C2’ which means that either C1 or C2 must be truth, and negation ($\neg C$) that can be read as ‘not C’ (45). The statement ‘course book is the book that is not a literary book and all of its readers are students and at least one of the book’s authors is a scientist’ is an example of statement, where negation, conjunction, existential and universal restrictions are used.

The family of DLs consists of multiple languages that differ in their level of expressivity and ALC provides a foundation for many other DLs. Expressive DLs extend ALC in order to achieve higher accuracy and complexity of descriptions. Lightweight DLs that are based on fragments of ALC restrict basic expressivity in order to enable the realisation of efficient algorithms (45). Since DLs provide means to represent information with different levels of detail, they are applied in multiple areas. These areas include data integration, natural language processing, life science, and ontology-based data access (46). In this work, DL will be viewed as a formalism that serves as a foundation for OWL language, which will be explained in the following chapters.

2.2.4 **Ontologies**

The term ontology comes from philosophy and is defined as the systematic explanation of being (44). Nowadays, ontology research has changed from a philosophical to an interdisciplinary subject, with a strong focus on computer science. A very popular definition describes ontology as an ‘explicit specification of a conceptualization’ (47), where the level of detail of conceptualization is dependent on application goals. Another definition given by (48) describes ontologies as ‘*an agreed understanding (i.e. semantics) of a certain domain, axiomatized and represented formally as logical theory in the form of a computer-based resource*’. A part of the world is represented in the ontology by means of organising concepts into a hierarchy, defining its properties and setting restrictions on these properties. The specification is applied using formal languages based on descriptive logic. Since application areas of ontologies broadly vary, there is no standard component list which every ontology should match. According to Jasper (49) ‘*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'*.

Ontologies provide means to organise and transfer knowledge, which allows autonomous and distributed application to communicate in a meaningful way (48). This property makes ontologies a core element of knowledge management systems, whose architecture is based on semantic technologies. Ontologies also provide means for reasoning mechanisms to derive new knowledge from existing knowledge, which corresponds to the DIKW wisdom. However, the subject of reasoning mechanisms is complex and is beyond the scope of the thesis. According to Toro et al. (50), ontologies in knowledge management can be used '*to separate domain knowledge from the operational knowledge, to analyze domain knowledge, to share common understanding of the structure of information among people or software agents... to enable reuse of domain knowledge, and... to make domain assumptions explicit*'.

The main idea behind using ontologies is to enable information exchange between humans and machines by developing a common understanding of a domain. The functionality of complex cyber-physical systems dealing with a large amount of data, such as a Digital Twin, is heavily reliant on semantic technologies that enable the interoperation of entities that are not designed to work together. Ontologies make this possible by creating sharable ontology-based context models to support context awareness of Cyber-Physical systems (51).

Since the application goals of ontologies vary widely, scopes of ontologies vary too. According to Weinert (38), there are two groups of knowledge that can be specified in ontology: general knowledge (which is intuitively the general knowledge about the world) and arbitrary knowledge (which corresponds to narrower professional domains). Arbitrary knowledge is important in manufacturing, especially for a product that is built to match customers' changing requirements. According to

Rolstadås (38), arbitrary knowledge is essential for organisations, but is also difficult to capture and distribute, since it is often a tacit knowledge with a specific context. Codifying arbitrary knowledge is a non-trivial task, but it is possible. Artificial intelligence and Semantic Web are two examples of attempts to enable formal description, structuring and sharing of such knowledge (38). There are names for ontologies of different scopes that deal with both types of knowledge: upper-level ontologies for specifying general knowledge and lower-level ontologies for describing knowledge specific to a domain of application. The scientific community dedicated much effort to developing sophisticated upper-level ontologies and attempted to suggest a way to systematise general knowledge so that any concept could be categorised in terms of these ontologies. Some of these ontologies are opensource and available online. One of the well-known examples of such ontologies is the Dublin Core ontology and FOAF. It is considered good practice to build new ontologies above already-existing ontologies of a higher level of abstraction. It is also possible, and in some cases recommendable, to merge ontologies by mapping concepts to each other. Using already-existing codifications of knowledge helps to ensure interoperability and reduce duplicated terms.

It is also customary to distinguish between ontologies of different depths of detail. Ontologies that are more so taxonomies are called lightweight ontologies. Lightweight ontologies are primarily designed to capture the hierarchy and other relationships between concepts. Those ontologies are easier to build, use, and maintain, but they are also heavily limited in their expressing capabilities. On the contrary, heavyweight ontologies put more restrictions on domain semantics and describe the domain in greater depth (44). A heavyweight ontology is the result of a expansion of a lightweight ontology. As a result of the increasing complexity of the modeled world, lightweight ontologies are gradually growing into heavyweight ontologies. Both lightweight and heavyweight ontologies have strengths and weaknesses, and it is advisable to start by making a lightweight ontology that can be extended as needed.

2.2.5 **Semantic Technologies**

For humans, interpreting words and sentences is a matter of experience. Based on possessed knowledge, a brain gathers symbols into words, puts the words into the context and draws conclusions about their meaning. Often, humans are able to make sense of ambiguous words without any additional information, for instance 'Mr. White' is obviously a name, 'he is 18' refers to age, and 'come after 12' is an invitation, probably for the daytime. However, enabling a computer to mimic cognitive processes that come so naturally to humans appears to be a nontrivial task. Accessibility of data and semantic interpretation are two major problems in this setting (45). While data accessibility is a complicated matter, that can only be solved by joint efforts of organisations and individuals, the second issue can be addressed with the Semantic Web.

The term 'Semantic Web' was introduced by Tim Berners-Lee in 2001 (52) as a proposal for the transition from the machine-readable, 'classic' web to a machine-understandable 'web of data'. „*Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data*" (53). According to the vision of the Semantic Web, machine interpretation is possible when data are described and structured by means of special languages created for this purpose. Well-known examples of such languages include RDF or OWL. Nowadays, the Semantic Web project is standardised by W3C.

The semantic-Web stack demonstrated on the Fig. 6 is an illustration of Berners-Lee, and presents a hierarchy of layers that must be implemented in order to build a semantically interpretable data structure above a hypertext web. Fig. 6: Semantic Web stack summarises widely established standards that grant interoperability to various layers of the Semantic Web.

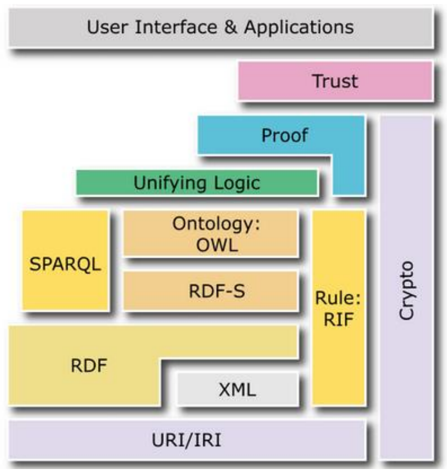


Fig. 6: Semantic Web stack
(54)

To fulfil the idea of Semantic Web, all layers must be covered with corresponding technologies. However, as can be seen in Fig. 6, some layers such as trust, proof, and unifying logic layers do not have a standardised solution, and for now, their implementation remains unclear. Trust is a layer involved in enabling the autonomous communication of parties and scalability; both of these factors are ensured by decentralisation and equality of all participating information sources (55). A proof layer creates the foundation for a trust level by ensuring the correctness of the data found (55). A unifying logic layer represents the need for a common logic to use while reasoning on acquired data. Lower levels, which mostly serve to support the creation of structured and semantically annotated data, are described in greater detail in the following chapters.

2.2.5.1 XML

The XML syntax, which was designed for mark-up in documents of arbitrary structure, was widely used for structured data representation before the rise of the Semantic Web. Nowadays XML is the dominant standard for information exchange on the Web (56). While it shares some concepts of HTML, XML does not provide a fixed set of tags. Instead, XML gives its user freedom to define a tag dictionary specific to concrete cases. Many standards, such as Dublin Core, MPEG-7, METS, and TEI have been specified in XML (56).

Some XML-related technologies include XML itself, XML Schema representing the structure, and XQuery mechanism. These technologies analogous to RDF, OWL, and SPARQL, respectively. However, the XML-related technologies are represented by different data models and have significant differences. Comparisons between XML and RDF, XML Schema and OWL, and Xquery and SPARQL can be found in the article of Bikakis (56).

2.2.5.2 RDF

The Resource Description Framework (RDF) was issued by W3C in 1998 as a recommended metadata model to organise data in a machine-readable format. As implied by its name, RDF is an abstract model that is used to describe resources; these resource can be software, human, love, the universe, or any other abstract, virtual or physical thing. Since the moto of the Semantic Web is 'Anyone can say anything about any topic' (AAA), when 'something' is about to be "said" it gets a URI. Ideally, when more people want to share knowledge about the same thing, they point to the same URI. This process supports decentralisation, extensibility and interoperability design principles. An Example of a URI is '`http://www.example.com/about#john`'.

Resource Description Framework allows users to describe resources using their own vocabulary. The formulation of statements about a resource in RDF has the form of a triplet: 'subject-predicate-object'. Even seemingly complex blocks of information can be expressed in the form of triplets. The statement 'John's friend Mary lives in a big house' can be modelled using the following set of triplets: 'John hasFriend Mary', 'Mary livesIn House', 'House hasSize big'.

On the web, RDF must be 'wrapped' in XML or another mark-up language to make sense. An example of an XML-annotated and RDF-described resource 'page' is demonstrated on Listing 1: XML and RDF.

```
<?xml version="1.0"?>
  <Description

          xmlns="http://www.w3.org/TR/WD-rdf-syntax#"
          xmlns:s="http://docs.r.us.com/bibliography-info/"

                   about="http://www.w3.org/test/page"
                   s:Author ="http://www.w3.org/staff/Ora" />
```

Listing 1: XML and RDF
(57)

It is important to note, that RDF is the simplest building block of the Semantic Web, and neither defines the semantics of the things it describes nor indicates interrelations and the structure of knowledge. In order to create a simple ontology, RDF needs a 'superstructure' in the form of vocabulary.

2.2.5.3 RDFS

The Resource Description Framework Shema (RDFS) is a vocabulary (that is, a set of reserved terms) using normative semantics. This schema, is included in RDF specifications to increase expressivity (45). The RDFS is used by popular RDF vocabularies such as FOAF, Schema.org, and Dublin Core.

The basis of RDFS is formed by keywords describing different kinds of binary relationships between resources. Term rdfs:Class and its instances, which are defined with the term rdfs:type, are constructs related to the concept of having a category with objects that belong to it. Concepts are similar to classes and instances in the object-oriented programming paradigm (56). The RDFS also provides means to restrict properties' usage by putting constraints on a set of objects(rdfs:domain) and the set of subjects(rdfs:range) this property connects. A full list of terms in the RDFS is given in Table 4.

Type	Term	Description	Domain	Range
Class	rdfs:Resource	The class Resource.		
	rdfs:Class	The concept of Class		
	rdf:Property	The concept of a property.		
	rdfs:Literal	The class `rdfs:Literal` represents the set of literal values, eg. textual strings.		
	rdf:Statement	The class of RDF statements.		
	rdfs:Container	This represents the set Containers.		
	rdf:Bag	An unordered collection.		
	rdf:Seq	An ordered collection.		
	rdf:Alt	A collection of alternatives.		
	rdfs:ContainerMembershipProperty	The container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.		
Property	rdfs:isDefinedBy	Indicates the namespace of a resource	rdfs:Resource	rdfs:Resource
	rdf:subject	The subject of an RDF statement.	rdf:Statement	rdfs:Resource
	rdf:predicate	the predicate of an RDF statement.	rdf:Statement	rdf:Property
	rdf:object	The object of an RDF statement.	rdf:Statement	*not specified*
	rdf:type	Indicates membership of a class	rdfs:Resource	rdfs:Class
	rdfs:member	a member of a container	rdfs:Container	*not specified*
	rdfs:subClassOf	Indicates membership of a class	rdfs:Class	rdfs:Class
	rdf:value	Identifies the principal value	rdfs:Resource	*not specified*

		(usually a string) of a property when the property value is a structured resource		
	rdfs:subPropertyOf	Indicates specialization of properties	rdf:Property	rdf:Property
	rdfs:comment	Use this for descriptions	rdfs:Resource	rdfs:Literal
	rdfs:label	Provides a human-readable version of a resource name.	rdfs:Resource	rdfs:Literal
	rdfs:domain	A domain class for a property type	rdf:Property	rdfs:Class
	rdfs:range	A range class for a property type	rdf:Property	rdfs:Class
	rdfs:seeAlso	A resource that provides information about the subject resource	rdfs:Resource	rdfs:Resource

Table 4: Vocabulary of RDFS
(58)

Since there are much more data on the Web wrapped in an XML format than in an RDF(S) format, Semantic Web languages need an integration mechanism in order to use the data models of XML, that are different from their own. The XML data model is inherited from databases and is therefore based on a node-labelled, ordered tree. On the contrary, RDF is derived from a model theory for standard logic and is based on a directed graph with unordered edges that have identifiers. A comprehensive study of the mechanisms enabling integration of XML and RDF(S) data models can be found in (56).

2.2.5.4 OWL

At the beginning of the 1990s, ontologies were built using first-order logic. Later the process shifted towards other knowledge representation techniques based on logic, such as OIL, DAML+OIL, and finally OWL. The latter became popular due to its placement in the Semantic Web stack of languages.

The OWL language is used with RDF(S) to help describe the more complicated logical relationship between resources. OWL is not a single language, but a family of languages. Languages of the OWL family are listed from the least expressive to the most expressive in the following list: OWL Lite, OWL DL and OWL Full. In this thesis, more focus will be placed on OWL DL, since it provides a good balance between expressivity and computational capabilities.

As it is both a descriptive and a logical language, OWL allows users to express expert knowledge in a formal way and to draw conclusions from this knowledge (45).

2.2.5.5 SWRL

The Semantic Web Rule Language was proposed in 2004 by W3C and was based on a combination of OWL DL, OWL Lite, and the sublanguage of RuleM. As the name implies, the SWRL language is made to express rules and logic in the ontology. '*The proposed rules are of the form of implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold*' (59). SWRL rules can be written in XML, RDF/XML or OWL XML, depended on a desired level of expressivity. An example illustrating how to extend OWL RDF/XML syntax with SWRL rules is demonstrated in the Listing 2, where the statement 'x1 hasUncle x3' implies that x1 has a parent x2 and x2 has a male sibling x3 :

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example2"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasSibling">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasSex">
      <ruleml:var>x3</ruleml:var>
      <owlx:Individual owlx:name="#male" />
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Listing 2: Example of RDF/XML annotated SWRL rules
(59)

2.2.5.6 SPARQL

For data to be useful, there should be a way to ask questions about this data. SPARQL is an RDF query language that was introduced to the public by the W3C Semantic Web Activity in 2004. SPARQL allowes its user to '*pull values from structured and semi-structured data; explore data by querying unknown relationships; perform complex joins of disparate databases in a single, simple query; transform RDF data from one vocabulary to another*' (60).

SPARQL works by finding a match between a given pattern and a graph-like structure realised by RDF(S) in the data source of interest. The syntax of SPARQL is quite similar to popular data base query languages like SQL. A SPARQL query may consist of the following parts: dataset definitions, a result clause, pattern matching, solution modifiers, and the output. The goals of each segment are relatively straight forward. First, source graphs that will provide data for queryng are defined. A query starts with the keyword 'FROM' and is followed by an identifiers of graphs that will be queried. The result clause starts with the keyword 'SELECT' and must answer the question of *what* to search. The next part of the query starts with the keyword 'WHERE' and is followed by a pattern that will be matched with a dataset from the chosen sources. The keyword 'ORDER BY' sorts results in a particular order. 'FILTER' denotes a set of constraints on the search result defining *which* search results need to be considered; and finally, the last part of the query allows the user to specify *how* the search results will be displayed. The result of a query is returned via HTTP and can be presented in RDF, HTML, XML or JSON format (60).

3 Methodology for Building Ontologies

In order to build an ontology, a methodology should be specified. As a relatively new field of science, ontology engineering has no widely accepted methodologies. However, there are many proposals for step-by-step activities, which serve as a guideline for developing an ontology. Those methodologies appear either as a result of an author's extensive ontology engineering experience or as an initially proposed, scientifically designed method (46). Examples of well-known methodologies include Methontology, Ontolingua, and Ontology Development 101. Analysis and comparison of these and many other methodologies can be found in the research of Iqbal et al. (46).

The methodology outlined in this thesis is based on the widely accepted work of Noy (61), that already became classic. The authors propose a basis for creating an ontology and define the following activities as part of this process: determine the domain and scope of the ontology; consider the reuse of existing ontologies; enumerate key terms in the ontology, classes and class hierarchy definition; define the classes and the class hierarchy; and define the properties of classes. Since some of these steps are not relevant for this thesis, this guideline was not followed strictly but rather used as a basis; it was modified or extended to meet specific requirements. The specificity of this work is that the described the concept of a Digital Twin is still more of an idea than a technology. The lack of any standards or even an agreed-upon definition makes the identification of ontological entities difficult. Taking into account these difficulties, the process of ontology creation was divided into three phases. In the first phase, the core model for a digital model is created. To do so, the following activities are carried out: boundary identification for the model, step-by-step identification of the key concepts and relationships between them, subsequent expansion of the glossary, and definition of properties. The aim of the second phase is to extend the basic model to prove its validity for solution of a particular problem. This phase involves the following steps: data collection, problem identification, identification of competency questions, taxonomy modelling, and taxonomy refining. To prove a concept, in the third phase taxonomy was modelled in Protégé by means of the Semantic Web. A thesaurus was created, and the resulting ontology was queried. The sequence of steps is demonstrated in Fig. 7.

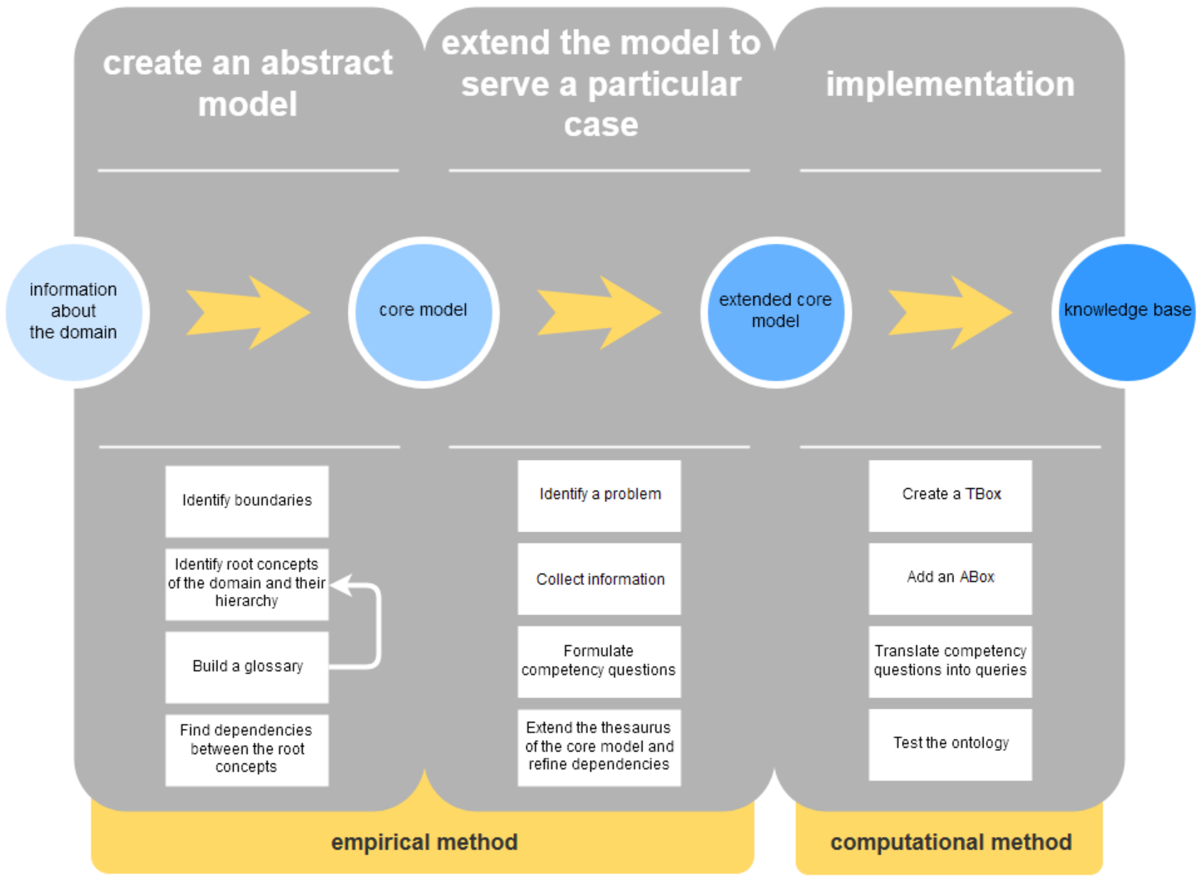

create an abstract model
extend the model to serve a particular case
implementation
information about the domain
core model
extended core model
knowledge base
Identify boundaries
Identify root concepts of the domain and their hierarchy
Build a glossary
Find dependencies between the root concepts
Identify a problem
Collect information
Formulate competency questions
Extend the thesaurus of the core model and refine dependencies
Create a TBox
Add an ABox
Translate competency questions into queries
Test the ontology
empirical method
computational method

Fig. 7: Methodology for creating an ontology

The described sequence of steps is not a typical one and is a consequence of the peculiarities of the pursued goals. At the first phase the scope of the model to be designed is limited by the definition of the Digital Shadow. To prove the validity of this abstract model, it is necessary to have a use case. For this purpose, in the second phase, a problem is identified and data about the problem are collected. It makes it possible to derive the competency questions from the collected information and expand the model according to these questions. Finally, in the third step, the model is checked for compliance with the competency questions.

Step 1.1: Define the Boundaries.

Since the goal of core ontology was to codify a specification of the Digital Twin/Shadow, the ontology engineering process started with a decision on how the sub-models of the digital model would be identified and limited. As previously mentioned, in the case of digital models, there is no explicit specification in the form of a complete and generally accepted definition. Another option would be to access the knowledge of specialists implementing digital models for real cases. However, in this case the obtained specification would very likely be conditioned by the peculiarities of a particular implementation. As was shown in the theoretical section of this thesis, definitions of digital models vary greatly, depending on what is meant by the digital model (for instance, is it a data profile, an autonomous controlling mechanism, or a simulation tool?) and depending on what set of characteristics determines the technology used by each particular company (see Table 2: Types of a Digital Twin). Therefore, in order to define the boundaries of the core model, the various scientific definitions of the concept were analysed and split into specific requirements. A list of criteria for the digital model was created.

Step 1.2: Identify Root concepts, Step 1.3: Build a Glossary and Step 1.4: Identify Properties

The following work is based on the list of criteria for the digital model compiled in the previous step. Identifying the basic entities and making a dictionary is an iterative process in this case. For each iteration, the specification of one Digital Twin/Shadow sub-model is used as an entry; each iteration will result in the hierarchy of root concepts discovered in this sub-model. These concepts will be merged into a glossary after the last iteration is completed. The transition from criteria to the identification of key concept properties requires domain knowledge, which was obtained through the document study (documented in the chapter 4) and interviews with domain experts. The result of the completed first phase is a specification of the studied concept, realised via identification of entities and their relationships, or in other words, the conceptual model of a core ontology.

Step 2.1: Collect Data and Step 2.2: Identify a Problem

To test the degree of usability, extensibility and therefore validity of the core ontology, a proper use case must be identified. The goal of doing so is to answer the following question: ‘Can the core model be easily extended to solve a certain problem in the engineering domain?'. The problem that is being solved to demonstrate the function of the ontology should ideally be easy to understand and also be covered by the scope of digital model specification.

At this stage it is necessary to collect as much data as possible about the problem to be solved. In order to collect information, interviews with domain specialists were conducted, translated, and further analysed. This method was preferred among other methods because domain experts can provide the most complete information about a problem, its origins, its consequences, and expected solutions. The result of data collection should be a clear understanding of the problem, including the questions of 'what causes the problem?', 'what processes, people, and things are involved?'. A suggestion of how the problem can be addressed by using the proposed data structure model should be given.

Step 2.3: Define Competency Questions

Defining competency questions using a natural language is a common method of specifying requirements and therefore limiting the scope of the ontology. This method widely used for cases, in which the questions the ontology should be able to answer are clear. Competency questions must clearly declare the aim of a user and directly correspond to the use case.

Step 2.4: Create a Model of an Extended Taxonomy

Extension of taxonomy happens in a manner similar to the first phase. First, the classes that are directly related to the solution suggested in previous stages should be defined. For this purpose, the results of the interview analysis should be used. Second, the identified classes should be linked in such a way that this connection (i) makes sense, (ii) clearly reflects actual, existing relationships between the two classes, (iii) does not violate the consistency of the taxonomy (iv) connects entities in a way that allows adequate querying. Finally, the new entities should be connected to the entities of a core model. Some of the relationships in the main ontology may be changed (namely, made more precise). In this case, it is worth paying special attention to the preservation of consistency between both models.

Step 3.1: Model a TBox

Since the previous steps must have resulted in complete conceptual model, the discovered entities and their relationships can be modelled as a TBox using Semantic Web tools. In this work, it was realized using Protégé, which is an OWL ontology editor that provides a visual interface for ontology creation and allows the user to convert the result of work into a document, written in one of a series of different languages, including RDF and OWL.

Step 3.2: Model an ABox

Although the ontology can be queried without an ABox, this form is used to represent the structure of the domain rather, than connect a real data. To populate a model, the right set of data is needed. There are multiple ways to get such data, but in this work the sample data were provided by domain experts. Chosen data must be then mapped to the TBox, so the resulting ontology can be tested against the competency questions.

Step 3.3: Test the Ontology

The data-filled ontology can be a rich source for a variety of different queries. The competency questions derived in the previous steps can be translated into query language syntax (for example, SPARQL) and used to determine if the ontology is responding correctly to the questions posed and therefore whether it fulfils its purpose. If the query results in a different answer than expected, effort should be made to analyse the correctness of the query. Otherwise, the consistency of the ontology should be checked, or the previous stages should be revisited to analyse where an error was made.

4 Foundational Work

The goal of this chapter is to summarise all the knowledge needed to further implement an ontology for a Digital Twin/Digital Shadow. In this section, some concepts relevant to this thesis are introduced and discussed in as much detail as necessary for their application in semantic modelling. This overview is important to understand the specifics of a Digital Shadow model intended for use in the engineering domain. Information given in this chapter has a selective nature and neither aims to provide a comprehensive description of the engineering domain, nor to conduct a study on its concepts and terms.

4.1 Product Lifecycle Management

Product lifecycle management(PLM) is now considered to be of high importance since the manufacturing industry has faced the challenge of collecting massively generated heterogenous, product-related data. In (62) PLM is defined as a '*strategic business approach that applies a consistent set of business solutions in support of the collaborative creation, management, dissemination, and use of product definition information across the extended enterprise from concept to end of life, integrating people, processes, business systems, and information*'. In other words, PLM systems help to collect and interconnect all the knowledge related to a product‘s specifications in one place in order to support business processes and create means to provide users with information in the right place at the right time. Product lifecycle management is not a specific technology, but a process of managing a continuous flow of data throughout the product life. To do so, appropriate software is used during a product's lifecycle. Since one of the I4 goals is automatization of processes, PLM should be also concerned with creating a smooth workflow when data is processed, used, and transferred by applications at all stages of production. However, without understanding the context, some applications cannot use shared data due to its heterogenous representations. Adding semantic annotation to the generated data mitigates this issue.

4.2 CAD Modelling

Computer Aided Modelling (CAD) describes a broad scope of software used by engineers, artists, designers, and others to create precise visual models of objects. In engineering, the term CAD refers to the early phases of a product's lifecycle and denotes the use of a special software tool for modelling and documenting the product development process. The development of CAD modelling, in addition to other factors, influenced the creation of PLM. Today CAD software plays a large part in achieving PLM goals. In industrial manufacturing CAD models are very widespread, as they allow users to create, edit, extend, and save sketches and layouts. The virtual representation of a product can be displayed in various ways, including different angles or 2D drawing-like views. Besides geometrical modelling, CAD integrated software allows users to carry out calculations in order to predict the performance of design.

Traditionally, paper and pencil technical drawings were used to model the geometrical properties of a product and to provide visualisation. Computer modelling has several advantages over technical drawings. These include higher precision, calculation of features, and a representation of the information in a format, that can be universally interpreted for further use. During PLM activities, but mostly in the early phases of development, CAD data are generated and shared. From the design phase to the product release, various product-related processes heavily rely on the data that is extracted from CAD and organised in some sort of a database for easier access.

4.3 Standards in the Engineering Domain

According to Toro (50) '*a standard is an agreed, repeatable way of doing something*'. Standards are an essential part of production, since they bring together the experience of all committees and help to increase the reliability of services and goods by providing technical specification or other precise criteria designed to be used consistently as a rule, guideline, or definition (50). Different standards represent different conceptual models that can be applied to ontologies in order to ensure their consistency.

One of the very well-known standards in the engineering domain is an ISO 10303 called the STEP (Standard for the Exchange of Product model data) standard. These series of documents standardises a format for the exchange of technical product data present at different stages of production. Different STEP documents cover a broad scope of product types and lifecycle stages. However, today the most-used part of this standard is concerned with the transfer of CAD data. Most modern CAD software used in manufacturing supports functionality, that ensures compliance between CAD data and the STEP format (63). In this thesis, STEP-defined vocabulary will be used to define concepts for structural representation in the model, proposed in the following chapters.

Another freely available data and information-exchange standard is called MTConnect. The connectivity provided by this standard offers more ways to collect data from production equipment. The software developed using this standard assumes the presence of three modules: *equipment*, *client*, *agent*, where *equipment* denotes any physical unit on the manufacturing floor that publishes data, for example machine tools, sensor units, workstations, and software applications. To ensure consistency in interpreting data, the MTConnect standard includes a semantic dictionary that allows the user to easily interpret data received from different pieces of equipment. The data published by *equipment* is collected and structured by a special software called an *agent*, which can also provide a structured response to queries sent by the *client* software application. To address the diversity of functions using data in the manufacturing domain, MTConnect also presents different semantic informational models, for example, an asset information model annotates data related to an asset that is produced by different equipment pieces. Every informational model of the standard can be extended when needed and the procedure of extension is described in the text of the standard. (64)

Another standard to consider when creating a semantic model for a Digital Twin/Shadow in the engineering domain is IEC 61360: ‘Standard data element types with associated classification scheme’. This standard defines a dictionary for building ontologies in certain domains, including the one that is of concern in this thesis. The platform Industry 4.0 (65) defines IEC 61360 as a standard for describing product properties of the Administration Shell (that is, the analogue of a Digital Twin in Industry 4.0). The standard defines fundamental information units, such as item classes or data element types (properties). The latter includes four groups: identifying attribute (such as code, version number, revision number, and various associated identificators), semantic attribute, value attribute and relational attribute that defines the connections between the entities. The properties of IEC 61360 can be easily mapped to semantic statements. (65)

The ‘Web Thing API’ from Mozilla Corporation (66) is not officially a standard, but rather a proposal document. The document addresses the need for a common data model and an API to promote the interoperability of connected devices on the Web. The Internet of Things is an emerging concept whose main task is to find a way to connect real-world objects to each other using standard Web tools. The vision of the Internet of Things is a network of interconnected physical devices that can communicate and be accessed via the Internet. One of the biggest issues of Web of Things is the difficulty of enabling the interoperability of data models generated by different devices. Hypermedia APIs such as the Web Thing API, the JSON Hypertext Application Language, and the Constrained RESTful Application Language aim to solve this issue by creating a bridge between physical and virtual things and making their properties and functionality exposable to others (67). Since the Digital Twin concept largely overlaps with concepts of the Internet of Things, and in some cases even relies on them, it might be beneficial to consider the existance of common information models in this field. The vocabulary of ‘Thing Description’ introduced in the ‘Web Thing API’ includes objects such as ‘Property’, ‘Action’, ‘Event’, ‘Link’ and their members ‘properties’, ‘actions’, ’events’ and ‘links’, respectively. It also includes the terms ‘name’, ‘description’ as well as terms for optional annotation. The Listing 7 in the appendix A 2 illustrates the suggested way of describing *things* using the example of a Web-connected lamp. The Web Thing API provides an easy-to-read description of connected devices; it also takes into consideration events which have occured and actions followed, which corresponds to the ideas of a Digital Twin and its data model. A semantic model of a Digital Twin proposed in this thesis does not fully satisfy terms introduced in ‘Thing Description’, but it follows the same principles and can be easily extended or edited to comply with Mozilla’s proposal.

There are many other standards that can be found and analysed in order to gain a deeper understanding of standartised or at least widely accepted approaches for data structuring. For instance, the Web of Things from W3C and the Web Thing Model from EVRYTHNG are two other proposals defining semantic annotation for connected devices on the Web of Things. Descriptions of all three proposals (from W3C, EVTYTHNG and Mozilla) can be found in the paper Hypermedia APIs for the Web of Things (67).

4.4 Approaches for Data Structuring

In this chapter, the reader will be introduced to the common practices of the domain, that have been used as patterns in the process of designing the Digital Shadow semantic model.

4.4.1 **RAMI 4.0**

The following text is based on (68). The reference architecture model of Industry (RAMI) 4.0 demonstrated in Fig. 8 is a three-dimensional model that provides a basis to sistematically classify I4 technologies. It allows for a step-by-step transition to Industry 4.0. The model integrates different user perspectives and provides a common understanding of I4 technologies. Every axis in this model aggregates some important elements of Industry 4.0 into a hierarchy. It is implied that the data related to one asset is maintained at various positions in the Information Layer, and can be extended during the lifecycle of the asset.

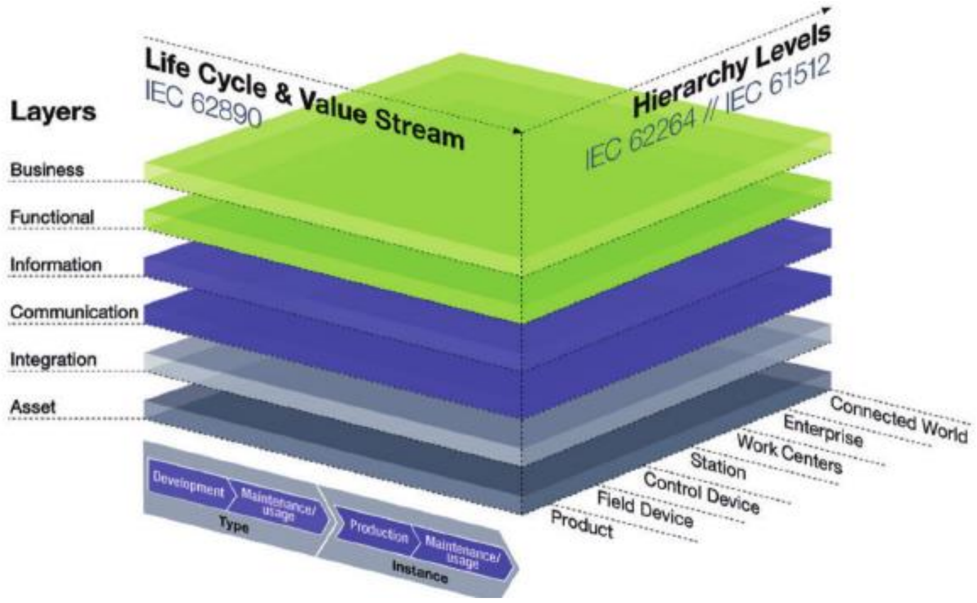


Fig. 8: RAMI model
(69)

The right horizontal axis represents different functionalities within a factory. These hierarchy levels correspond to the international standard IEC 62264 for control systems and enterprise IT. The vertical axis separates a machine into its properties. The main idea of these two layers is to break a complex system into levels.

Finally, the life cycle and value stream axis represent the life cycle of facilities and products. This last axis is of the greatest interest, because it represents the duality of a product that is represented in both physical and digital worlds. This axis is based on the EIC 62890 standard for lifecycle management, the central idea of which is the distinction between type and instance, where the type plays the role of a blueprint for creating instances. According to RAMI 4.0, a type is created as soon as product development begins. A type becomes an instance when the development, prototype validation and testing processes are completed and the actual product is moved into serial production. Since several instances may be produced on one production line, each instance is assigned a distinctive parameter. However, since Industry 4.0 is characterised by continuous improvement of the product, the transition from type to instance is not permanent. As soon as the asset instance is available for sale, feedback and new data on the Asset can lead to modifications (adaptations) of the type, and such 'jumps' between the Instance and Type phases can occur more than once.

4.4.2 Property-Characteristic Product Description

Systematic design is always founded on models. One model used to describe product-representing objects, processes, and the relations between them is the theory of technical systems(TTS). The TTS developed by Hubka aims to describe commonalities between devices, independent of their physical principles of action. One of the variants of using TTS models is the analysis of existing technical systems. (70)

According to Hubka, identification and description are two ways to capture a product. Description involves characterising a product's properties. Identification is realised by formal code linked to the product, in a manner that only allows experts to derive information about the product. Within this theory the author specifies properties of a technical system, which are classified as external and internal processes. External properties are observable and are caused by internal properties. Properties describe, or even define, a product, as certain attributes give value to the product. (70)

Weber (71) was one of authors who contributed significantly to Hubka's theory. His proposal is to distinguish between characteristics (which are similar to internal properties of TTS) and properties of a product. According to Weber, characteristics describe the product itself, such as the structure, shape, and material consistency and can be influenced by a designer. On the contrary, properties represent the behaviour of a product, and cannot be directly determined by a designer. Usually it is the properties that interest the end user, and it is properties that add value to the asset. For instance, material or dimensional characteristics can be modelled in a CAD system in early phases of a product's lifecycle, but weight, safety, aesthetic properties, and environmental friendliness can only be predicted, but not determined, until after the product is manufactured.

According to Weber, characteristics and properties are dependent on each other. This dependence happens in one of two ways: (i) product properties are either determined or predicted based on known or given characteristics, or (ii) characteristics get assigned to a product based on given or required properties. Since the properties are of interest to the client, product development starts by determining the desired properties of a product. Next, characteristics are assigned in such a way that the required properties are met.

The approach described by Weber is well known in engineering and, among other things, supports interoperability, by providing means to integrate multiple existing models into one framework. In this thesis, Weber's approach allows one to not only to divide the attribute values of a product into two different groups according to the type of their value but also to express the logic of their dependence. With that being said, in the context of characteristic-property modelling approach, a product-related knowledge 'consists of (relevant) characteristics and properties of a product with known relations between the two' (71).

5 Practical Implementation

The modelling effort was divided into two phases. According to the proposed methodology, during the first phase requirements for a general Digital Twin model were analysed, and the concept was developed according to those requirements. During the second phase a use case was found and described, and an extension model concept was created in order to meet requirements set by the use case. Finally, in the third phase, both concepts were merged into one ontology and then tested.

General requirements of the model include interoperability (which was partly achieved using design patterns), and usage of known data-structuring paradigms, and a higher level of abstraction (which allows for easier expansion of the model).

5.1 Development of a Concept Model

According to the research described in the first part of this thesis, a Digital Twin has many partially overlapping definitions. Those definitions, listed in Table 1: Evolution of definition of a Digital Twin can be summarised as a set of specifications for models of a Digital Twin and its data profile (that is, a Digital Shadow):

S1: reflects the interaction between a *thing* and its environment
S2: mirrors the life of a corresponding *thing*
S3: represents the structure of a *thing*
S4: describes the properties of a *thing*
S5: represents historical information, as well as information about a current state of the „thing“
S6: regulates and controls the system status and process

S1: Reflects Interaction Between a *Thing* and its Environment

A Digital Twin has information about the current state of a *thing* (provided by a Digital Shadow) and is also aware of environmental influence that may, in extreme cases, cause an update of the state of the *thing*. Examples of environmental factors that affect a *thing* can be time or temperature, but these factors may also include other physical mechanisms or other Digital Twins. In other words, a Digital Twin reflects how the data-driven description of a *thing* is changing depending on the context in which it is placed. This context does not necessarily get modelled for the Digital Twin which is supposedly acting inside of it; it can simply consist of already existing and interacting elements. Also, not every digital model is aware of its environment, especially in earlier stages of implementation or in simpler use cases. The identified concepts are collected in the Table 5.

Entity	Description
Digital Twin	Entity referring to the Digital Twin technology; an aggregation of different data-carrying entities related to a particular asset, and (optionally) agents that work with this data.
Digital Shadow	Entity referring to a mirror state of an asset; plays the role of a „container“ for both historical and real-time data in order to give the most accurate, up-to-date data profile of an asset.
Context	External conditions that can cause a change of state in an asset

Table 5: Glossary extension

S2: Mirrors the Life of a Corresponding *Thing*

One data profile gives a complete (from the perspective of a use case) and unambiguous definition of one *thing*. However, one *thing* can be mirrored by more Digital Shadows as demonstrated on the Fig. 9, for example. when different data profiles are generated for different use cases or Digital Twins of different functionalities.

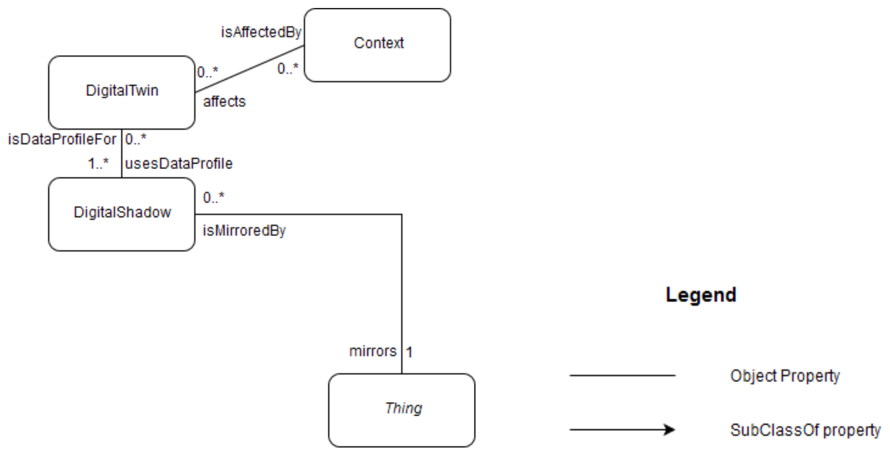


Fig. 9: Representation of the Digital Shadow and the Digital Twin

S3: Represents the Structure of a *Thing*

The structure of a *thing* can take multiple forms, therefore it is important to define what *thing* is being modelled in order to be able to define what this *thing* consists of. One of the most intuitive approaches would be to call *thing* a 'product'. The Oxford Dictionary (35) gives the following definition for this word: '*an article or substance that is manufactured or refined for sale*', however *things* that can be modelled in the ontology do not fit thise definition. *Things* that cannot be considered traditionally defined products include digital or physical components of a final product that are themselves not subject to sale. A common term for a *thing* in the engineering domain is an 'asset', which exchanged the provisional term *thing* in the model (see Table 6). According to Adolphs et al. (72) an '*asset is understood as a physical or logical object which is owned or managed by an organisation and which has an actual or perceived value for the organisation*', e.g. a car, a car's engine, a CAD model of the car, et cetera. In fact, any data generated by a company is somehow related to the assets, since value brought by assets is a main driving factor for a company. Therefore, on the one hand the term limits multitude of *things* to those of economic value, but on the other hand it is a broad enough definition to include many different objects that may be of interest to model.

Entity	Description
Asset	Anything that has value and can be modelled

Table 6: Glossary extension

Since engineering deals to a large extent with physical assets, it is intuitive to describe a structure of an asset in terms of parts and a whole. Terms widely adopted in the engineering domain to describe structure are an 'assembly' and a 'component'. The ICO-10303-1 standard defines component as a '*product that is not subject to decomposition from the perspective of a specific application*", and an assembly as a „*product that is decomposable into a set of components or other assemblies from the perspective of a specific application*' (73). Both terms are used for a structural representation of an asset in the model, as shown in Table 7

Entity	Description
Component	An asset that is used as a *part* to build *the whole.* Is not subject to further decomposition.
Assembly	An asset, aggregating components or other assemblies. Any asset can be seen as *a whol*e in a particular context.

Table 7: Glossary extension

To represent the relationship between assemblies and components, the modelling pattern 'hasPart' was chosen (see Fig. 10). Patterns of semantic modelling offer a solution for commonly occurring problems such as structural representations, representation of roles and et cetera. In other words, these patterns are useful for representing specific types of informational structures, thus they have proven their effectivity and have been adopted by other ontology engineers. The use of patterns helps to better understand the ontology, to expand it more rapidly, to increase its interoperability, and to reduce the risk of incorrect or incomplete relationships. The patterns for structure representation used in this thesis were proposed in the W3C Working Draft (74) . The patterns 'hasPart' and 'partOf' are widely used to represent structure. Both kinds of relationships are transitive, for instance, when a car has four doors, and each of the doors has a door handle, it can be said that four door handlers are also parts of the car. In order to describe a direct part of a whole while avoiding transitive properties, 'partOf_directly' and 'hasPart_directly' patterns were used (74). This kind of relationship between an assembly and its components specifies that components are located on a final stage of an assembly decomposition.

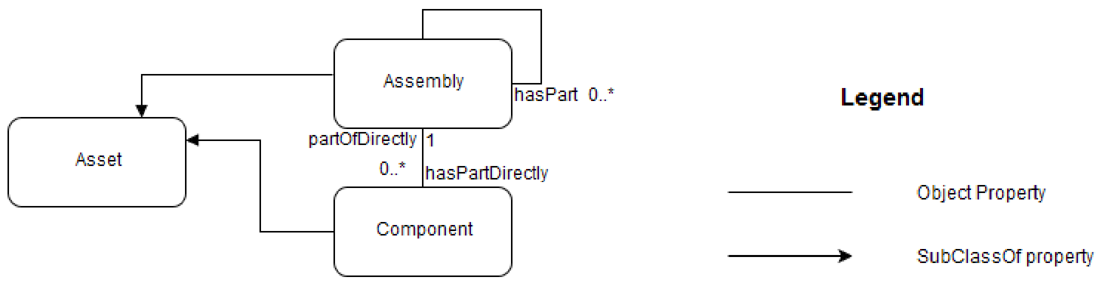


Fig. 10: Structural representation of an asset

S4: Describes the Properties of a *Thing*

Most objects, have a set of descriptive characteristics. For instance, a physical object has a colour, shape, weight, and material. Often, the object properties that are of interest to the consumer, are related to the functionality of this object. As such, aesthetic characteristics are important for art objects; while speed, durability and safety are important for a plane. According to Weber (71) each object has unique properties that can only be indirectly influenced by the designer. Since properties are influenced by characteristics of a modelled object, that is, features of an asset that are related more to how it will be made, than how it will be used, these properties can be influenced through characteristics, too (see Fig. 11). Therefore, properties of the product are of higher importance to the consumer of the product, while characteristics are more important to the manufacturer. The terms 'characteristic' and 'property' were used to extend a Digital Shadow model glossary, as shown in Table 8.

Entity	Description
Characteristic	A feature of an object that can be directly affected by designer (such as dimensions, tolerances, et cetera)
Property	A feature of an object that can not be directly affected (such as weight, material, safety, logistic, etc.), but often can be accessed through modelling of characteristics. Often seen as a value-driver.

Table 8: Glossary extension

Distinguishing between characteristics and properties allow one to isolate different types of information in different lifecycle phases and to capture dependencies between properties and the characteristic that influence them(see Fig. 11).

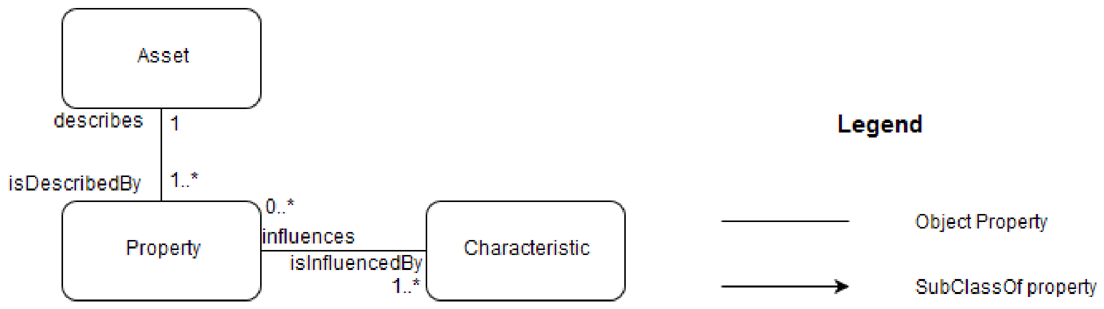


Fig. 11: Property-Characteristic description of an asset

S5: Represents Historical Information, as well as Information About a Current State of a *Thing*

A Digital Shadow should represent not only static but also dynamic information about its counterpart. So-called historical information, such as information related to the production of an asset (that is, its model), stays valid for a longer time period and is relevant to a higher number of manufactured assets. Conversely, data produced by each unique manufactured asset matters only during the lifecycle of its asset and is only relevant to this particular asset, in most of cases. One of the requirements for modelling a Digital Shadow or a Digital Twin is a representation of its duality, when the Digital Shadow mirrors the physical version of an asset, as well as the one that exists in a digital world. For a semantic model, it means that two types of data are required. One has a more static nature, is valid for a longer time and rarely gets changed (for example, manufacturing information). The other type of data has a more dynamic nature (for example, information related to usage of an asset). This difference between data that belongs to different states of an asset was captured by the RAMI 4.0 model (see chapter 4.4.1). The asset has two main phases in its lifecycle: 'type' (that is, the model of an asset, which existing mostly in a digital form) and 'instance' (that is, a stage that begins when the product has been manufactured). A similar idea can be found in the object-oriented programming paradigm, where 'type' refers to the class of similar objects and would get instantiated with a concrete entity that uses its type as a blueprint. These terms are used to extend the model glossary (see Table 9).

Entity	**Description**
TypeOfAsset	Comprises data related to the virtual blueprint of an asset that is used for mass production of an asset, that is data generated during development and improvement of an asset.
EntityOfAsset	Comprises data about each individual asset, manufactured according to its type.

Table 9: Glossary extension

The connection between these two terms is captured in the Fig. 12.

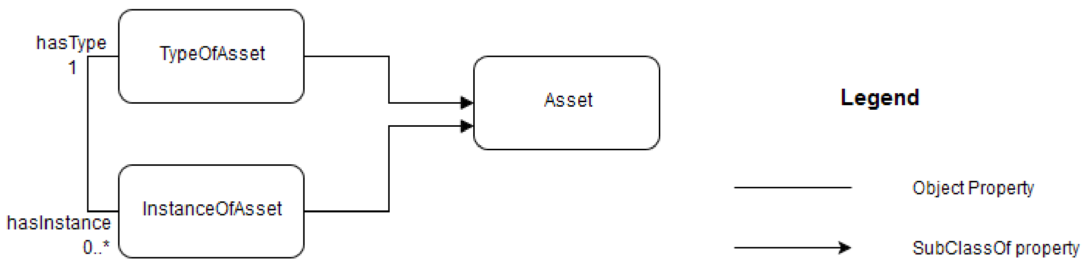


Fig. 12: Representation of an asset on different stages of its lifecycle

S6: Regulates and Controls the System Status and Process

Since this work is focused on modelling a Digital Shadow related to a Digital Twin, modelling of regulation and controlling agents will be ignored, as this practice is typical only for Digital Twin functionality.

All of the relationships discovered during the definition of a Digital Twin/Shadow models are summarized in the Table 10.

Object property	Inverse property	Type	Domains	Ranges
isAffectedBy	Affects		DigitalTwin	Context
Affects	isAffectedBy		Context	DigitalTwin
usesDataProfile	isDataProfileFor		DigitalTwin	DigitalShadow

isDataProfileFor	usesDataProfile		DigitalShadow	DigitalTwin
Mirrors	isMirroredBy		DigitalShadow	Asset
isMirroredBy	Mirrors		Asset	DigitalShadow
partOf	hasPart	Transitive	Assembly, Component	Assembly
hasPart	part of	Transitive	Assembly	Assembly, Component
partOfDirectly	hasPartDirectly		Component, Assembly	Assembly
hasPartDirectly	partOfDirectly		Assembly	Component, Assembly
hasType	hasInstance		InstanceOfAsset	TypeOfAsset
hasInstance	hasType		TypeOfAsset	InstanceOfAsset
isDescribedBy	Describes		Asset	Property
Describes	isDescribedBy		Property	Asset
isInfluencedBy			Property	Characteristic

Table 10: List of relationship for a Digital Shadow model

5.2 Use Case

In this section, the process of the core ontology extension will be demonstrated. To do so, the use case will be identified, described, and separated into a set of requirements. Finally, the extension of the core model will be made.

In order to identify a problem that can be solved by the Digital Shadow ontology, data was collected through interviews. Interviewing is a flexible, qualitative research method that is used when seeking the perspective or experience of an individual.

5.2.1 Results of the Interviews

Since Digital Shadow technology is a tool for knowledge management, two experts from this field were interviewed in order to identify a suitable use case.

Transcription of both interviews can be found in appendix A 4. In order to reduce the search field, the issue of data quality (that is, missing, incorrect, or inconsistent data) was selected as a focus, as it is one of the most appropriate issues the Digital Shadow can address. Narrowing the scope of potential use cases made it possible to select more specific questions for interviews and thus collect more relevant information.

A semi-structured qualitative approach was chosen for the face-to-face interview, as it best meets the needs of this thesis. As both interviewees wished to remain anonymous, they will be designated as Informant 1(I1) and Informant 2(I2) in the following text.

The interview with I1 took place on the 31st January 2019. Informant 1 is a specialist engaged in cost-value analysis. This informant was chosen because in this specialty, the result of any work directly depends on the quality of the data provided, so it was assumed that as an end-user, this informant will be well-aware of possible problems in this area. For this interview, questionnaire (see appendix A 3) was designed to:

1) Understand the interviewee's work and his work processes
2) Obtain information about his opinion on the data quality and, if a problem were identified, to learn more about it

The first interview revealed a problem with data quality: inconsistency or lack of all the necessary data for value analysis. According to I1, incorrect data is often difficult or impossible to identify. It affects the outcome of work, which in turn can cost his company money and contracts. Weight and dimensional data are particularly problematic as they are one of the basic value-drivers (for the group of products I1 analyses) and, according to I1, these parameters are often either missing in the enterprise resource planning system (which is software used to gain access to the information about an asset) his company uses or represented by an incorrect number. One of the examples given by I1 is the use of grams instead of kilograms and vice-versa, or an incorrect semicolon position in a fractional number. In some cases, incorrect data is identified by I1 during the value-analysis process: '*If you...*

have ring of size 1000 mm it cannot weight 2000 kg' (appendix A 4). One of the standard solutions when identifying incorrect data is to refer to engineering drawings from where the information has been transferred to the enterprise resource planning system, but this is a long and complicated process because of the confidentiality policy, which does not allow free access to any necessary information.

The second interview was conducted on the 8th February 2019. Since the first interview identified a problem (incorrect data about weight in the system), the purpose of this interview was, to study the problem at its origins and thereby determine the cause of its occurrence. The second interviewee was chosen with this goal in mind. Informant 2 is a head of engineering methods in the company with many years of experience. This interview had an unstructured nature. It started by the general CAD modelling questions and continued with the discussion about the use case.

Over the course of the second interview, the occurrence of incorrect data was examined from another perspective. According to I2, low quality of weight-related data is not specific only to a cost-value analysis: if the weight specified in the system does not correspond to the weight of the product randomly chosen for inspection, the batch cannot be shipped to the customer until the weight in the system is changed. However, there is a whole process behind this step that significantly slows down the product's shipment.

During the interview, a possible cause of incorrect weight-related data in the enterprise resource planning system was identified. Small errors for specific products are systematic in the engineering domain and according to I2, some products have a grease that is not included in CAD model and is therefore not used for weight calculation. The heavier the grease, the bigger the difference between the calculated weight and the real weight of an item randomly taken for inspection.

The results of the interview are summarised in Table 11: Results of the interviews.

Problem	Consequence	Possible reasons
Incorrect data about the weight of an asset	1) Incorrect weight data affects the final result of cost-value analysis 2) Inconsistency between the weight in a system and the actual weight of the manufactured product hinders shipment until the system weight is changed.	1) Input mistakes, such as improper comma position in fractional numbers 2) Confusion of units of measures 3) Adding initially incorrect data to the system 4) Grease is not included in the weight calculation.

Table 11: Results of the interviews

The following competency questions can be deduced from the Table 11:

Q1: Does the calculated weight correspond to the actual weight?
Q2: Which units of measure are used to describe weight?
Q3: What are the characteristics that affect the incorrect property?
Q4: Where do the incorrect data come from?
Q5: How much does the grease weigh (if we assume that the weight without grease was calculated correctly)?

5.2.2 Extending the Core Model for the Use Case

The competency questions provide a framework for expanding the model and clearly indicate the motivation for such an expansion. Following these questions, 'narrower' (or less abstract) subclasses and relationships can be identified. The aims of this process are to establish a link between the measured and real properties of the object (and more specifically, the weight), and to expand the basic model in order to facilitate the search for errors and identify their source. In the context of the studied use case, this approach has the potential to greatly facilitate the search for objects that do not match the predicted parameters.

As written in the chapter 4.4.2, the characteristics are attributes of an asset on which the designer has a direct influence. Accordingly, the characteristics of an asset are related to the early phases of its lifecycle (or type). It would be logical to assume that the properties refer only to the instance, based on the fact that they are a product of a characteristic's design. However, sending a product into production without roughly calculating its value-driving properties would be acting blindly. Therefore, an inseparable part of modelling is the calculation of properties based on CAD data. Thus, an asset has not only measured property values of the already manufactured assemblies but also the predicted property value, where both of those values should ideally coincide. Therefore, continuing the idea of representing physical objects as instances of a type, the concept of property has been extended to the property of type and the property of instance (see Table 12). For example, in this case study the weight in the enterprise resource planning system is the calculated weight, that is the property of the type; and the weight of the randomly selected product from the batch is a property of the instance.

Entity	Description
PropertyOfType	Refers to the predicted value of a property. Also limits the range of values to which the instance properties must fall in order to satisfy quality requirements.
PropertyOfInstance	Refers to the measured value of a unique instance.

Table 12: Glossary extension

Since the weight of a manufactured instance and the predicted weight of its type both reference to the same concept of weight, it would be logical to suggest that both classes refer to one class representing a type of property (see Table 13). Additionally, this practice can help to avoid redundancies in ABox, since it is enough to say once that property X is dependent on characteristics A, B and C; and this property will apply to any asset property referring to this type X.

Entity	Description
TypeOfProperty	Describes general types of existing properties, including their dependencies on characteristics, if there are any. Is not associated with any particular instance, but every property of an instance must refer to its type.

Table 13: Glossary extension

While studying a multitude of type-characteristics in the manufacturing domain one might realise that those characteristics belong to different logical groups. In this work, Weber's categorisation into material, shape, and structural characteristics (see the chapter 4.4.2) will be used, as demonstrated in the Table 14.

Entity	Description
MaterialCharacteristic	Includes characteristics related to the material of an asset
ShapeCharacteristic	Includes characteristics related to the shape of an asset
StructuralCharacteristic	Characteristics related to the structure of an asset

Table 14: Glossary extension

In many cases, in order to make sense of a value number, the unit of measure should be known. Two different numbers represented in different units can refer to the same value, so having various units of measure as instances of one class (see Table 15) contributes to the task of comparing or converting them. Some examples of measure units in the studied use case are grams and kilograms. It can also be extended to sub-classes representing different logical groups of units, if needed.

Entity	Description
UnitOfMeasure	Comprises different units of measure, such as grams and kilograms

Table 15: Glossary extension

In the case of incorrect data it can be also useful to determine where the data came from, as demonstrated in Table 16.

Entity	Description
DataSource	Points on to a source, such as a document, model, number of a quality test, et cetera, where the data was taken from.

Table 16: Glossary extension

Two terms represented in Table 15 and Table 16 are introduced to capture the values, describing a particular property. However, some use cases might require many more such values, therefore a parent class for all descriptive entities was introduced, as demonstrated in Table 17.

Entity	Description
DescriptiveEntity	Comprises values that add an additional information to the property

Table 17: Glossary extension

The mentioned classes are the minimum requirement to address the competency questions. New connections are summarised in the Table 18.

Object property	Inverse property	Type	Domains	Ranges
hasTypeOfProperty			PropertyOfInstance or PropertyOfType	TypeOfProperty
describes [3]	isDescribedBy		PropertyOfInstance or PropertyOfType	InstanceOfAsset or TypeOfAsset
isDescribedBy [5]	Describes		InstanceOfAsset or TypeOfAsset	PropertyOfInstance or PropertyOfType
influences [3]	isInfluencedBy		Characteristic	TypeOfProperty
isInfluencedBy [3]	Influences		TypeOfProperty	Characteristic
has			Property	DescriptiveEntity

Table 18: List of object properties

The full extended Digital Shadow model is demonstrated on the Fig. 13.

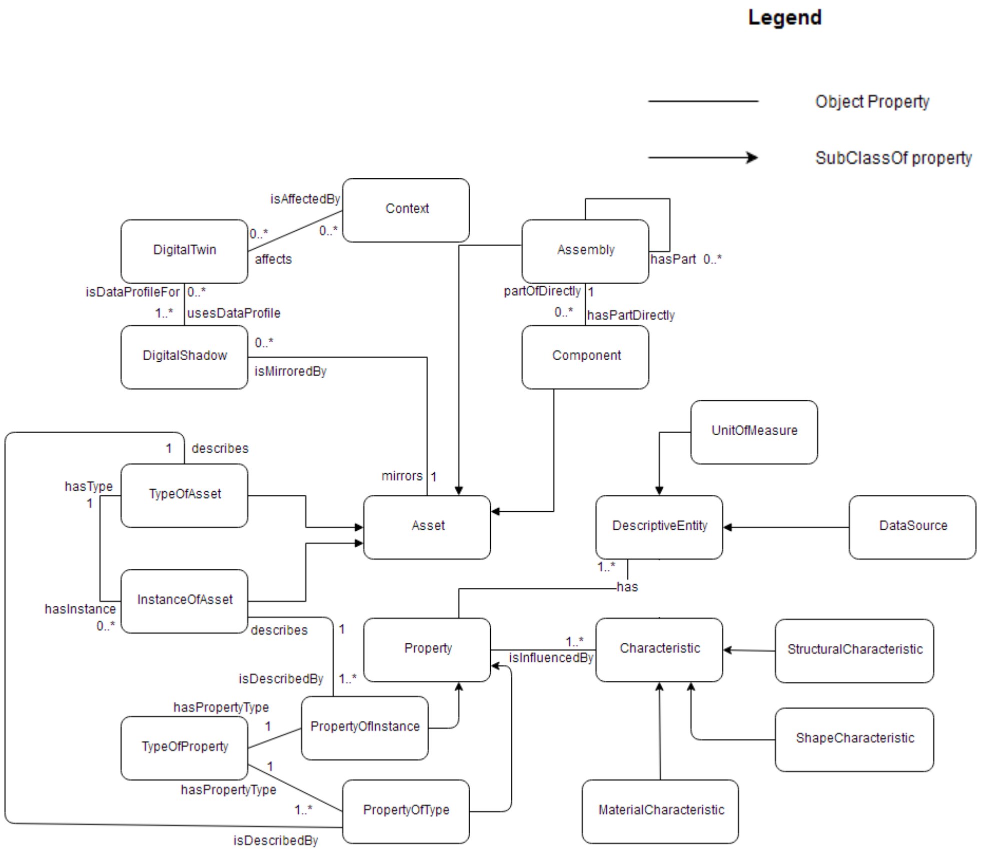


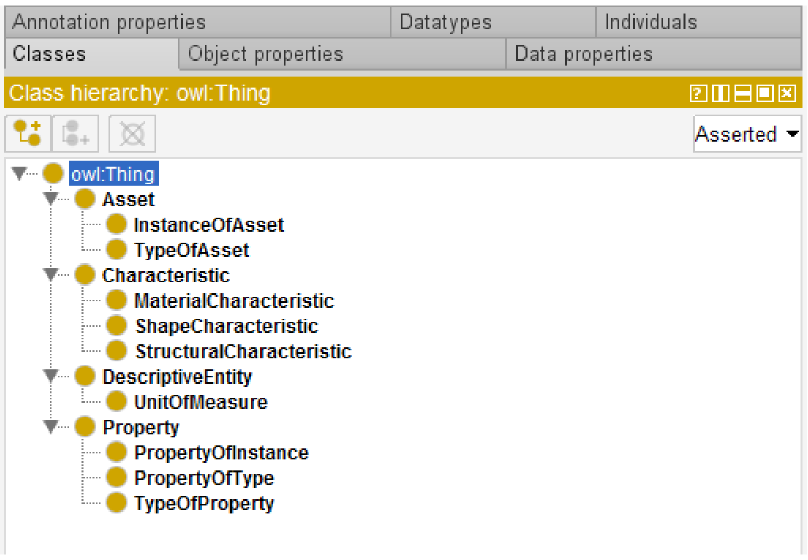
Fig. 13: Extended Digital Shadow ontology

[5] Updated attributes of a higher precision. In the core model belonged to classes of higher abstraction level

5.2.3 Creating the Ontology

The goal of the chapter is to describe the process of modelling the ontology extension proposed in the previous chapters. To create an ontology the Protégé softare was used. Protégé is a free and open-source ontology editor, that supports RDF/XML, OWL/XML, and other formats for codifying the ontology. Protégé provides a simple and customisable graphic interface for developing a knowledge base. The ontology editor supports functionality such as creating, editing and exporting ontologies in several formats, visualising the ontology and using a reasoner on the data.

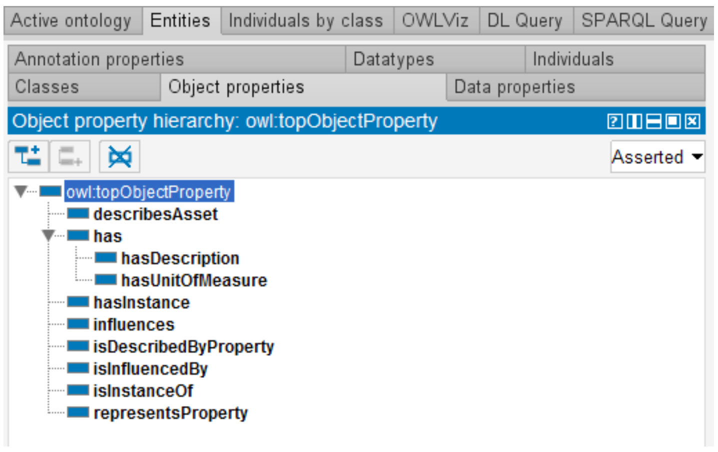
Based on Table 5, Table 7, Table 8 and Table 9, the classes were modelled and put into hierarchy, as shown on Fig. 13.



- owl:Thing
  - Asset
    - InstanceOfAsset
    - TypeOfAsset
  - Characteristic
    - MaterialCharacteristic
    - ShapeCharacteristic
    - StructuralCharacteristic
  - DescriptiveEntity
    - UnitOfMeasure
  - Property
    - PropertyOfInstance
    - PropertyOfType
    - TypeOfProperty

Fig. 13: Creating a hierarchy of classes in Protégé

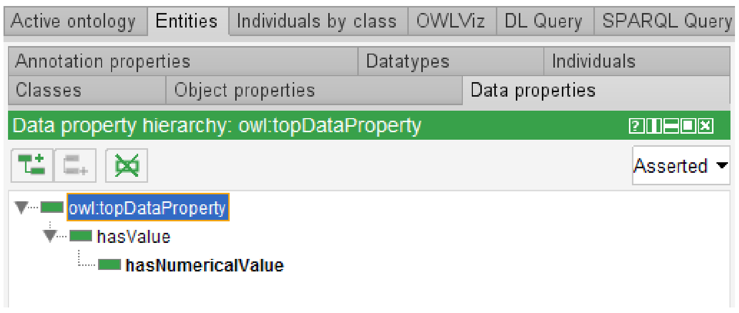
The language OWL distinguishes between object properties and data properties; an object property links individuals, and a data property links individuals to values. Object properties are summarized in the previous chapters. The object properties that belong to the use case were modelled. The Protégé interface permits the definition of additional characteristics of properties supported by RDFS and OWL, such as inverse, transitive, or functional properties. As indicated in the Table 10 and Table 18 object properties were created in Protégé, as shown on Fig. 14.



owl:topObjectProperty
- describesAsset
- has
  - hasDescription
  - hasUnitOfMeasure
- hasInstance
- influences
- isDescribedByProperty
- isInfluencedBy
- isInstanceOf
- representsProperty

Fig. 14: creating object properties in Protégé

There is only one data property that should be modelled in the context of the proposed model. The data property hasNumericalValue, as shown on Fig. 15, is a data property that gives value to the DescriptiveEntity, e.g. 'Age hasNumericalValue 20'.



owl:topDataProperty
- hasValue
  - hasNumericalValue

Fig. 15: creating data property in Protégé

Once the ontology is finished, it can be populated. This step can be done either manually, by creating every instance separately, or by using software modules created for automatization of the data import process, for example, the default-plugin of Protégé called Celfie. Celfie provides a means to import data from Excel tables by defining rules for how the data will be mapped to ontology. Celfie can be used to create classes and properties, as well as instances. Excel table used as a source of data and the JSON file containing code written in Manchester syntax to populate a created ontology can be found in the appendix A 5. Finally, an ontology can be saved, shared or exported in different formats, including OWL/XML, RDF/XML, Turtle, JSON NL and others. The full ontology saved in OWL/XML format can be found in the appendix CD.

5.2.4 Testing the Ontology

To test the ontology, competency questions defined in Chapter 5.2.1 were translated into SPARQL query language (for the SPARQL syntax description see (75)), as is demonstrated in Listing 3, Listing 4, Listing 5, and Listing 6. Proposed SPARQL queries serve a demonstrative purpose and can be modified or written differently in order to satisfy goals and requirements laid up by a use case.

Q1: Does the calculated weight correspond to the actual weight?

```
SELECT ?type ?instance ?valueT  ?valueI
WHERE {
?instance dsh:isInstanceOf ?type.
?instance dsh:isDescribedByProperty ?weightI.
?type dsh:isDescribedByProperty ?propertyT.
?instance dsh:isDescribedByProperty ?propertyI.
?propertyT dsh:representsProperty ?property.
?propertyI dsh:representsProperty ?property.
?propertyI dsh:hasNumericalValue ?valueI.
?propertyT dsh:hasNumericalValue ?valueT. }
```

Listing 3: SPARQL query to compare 'predicted' and real properties

Q2: Which units of measure are used to describe weight?

```
SELECT ?type ?instance ?unitT ?unitI
WHERE {
?instance dsh:isInstanceOf ?type.
?instance dsh:isDescribedByProperty ?weightI.
?type dsh:isDescribedByProperty ?propertyT.
?instance dsh:isDescribedByProperty ?propertyI.
?propertyT dsh:representsProperty ?property.
?propertyI dsh:representsProperty ?property.
?propertyI dsh:hasUnitOfMeasure ?unitI.
?propertyT dsh:hasUnitOfMeasure ?unitT.
}
```

Listing 4: SPARQL query for discovering a unit of measure

Q3: What are the characteristics that affect the incorrect property?

```
SELECT ?property ?characteristic
WHERE {
?instance dsh:isInstanceOf ?type.
?instance dsh:isDescribedByProperty ?weightI.
?type dsh:isDescribedByProperty ?propertyT.
?instance dsh:isDescribedByProperty ?propertyI.
?propertyT dsh:representsProperty ?property.
?propertyI dsh:representsProperty ?property.
?propertyI dsh:hasUnitOfMeasure ?valueI.
?propertyT dsh:hasUnitOfMeasure ?valueT.
FILTER (?valueI != ?valueT).
?property dsh:isInfluencedBy ?characteristic.
}
```

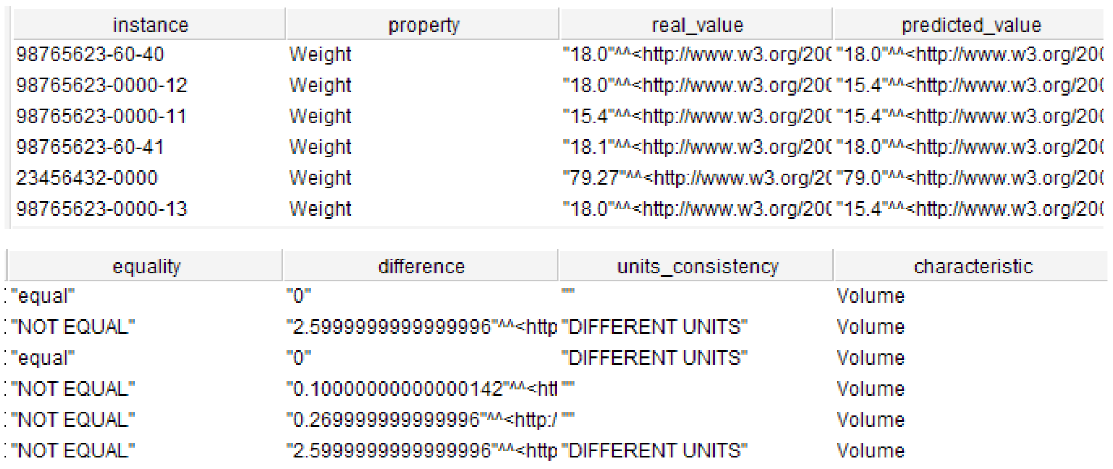
Listing 5: SPARQL query for finding the possible source of a wrong property value

Q4: How much does the grease weight if we assume that the weight without grease was calculated correctly?

```
SELECT  ?instance  ?property  ?real_value  ?predicted_value  ?equality
?difference ?units_consistency
WHERE {
?instance a :InstanceOfAsset.
?instance :isDescribedByProperty ?instance_property.
?instance_property :representsProperty ?property.
?type a :TypeOfAsset.
?type :isDescribedByProperty ?type_property.
?type_property :representsProperty ?property.
?instance :isInstanceOf ?type.
?instance_property :hasNumericalValue ?real_value.
?type_property :hasNumericalValue ?predicted_value.
?type_property :hasUnitOfMeasure ?unit1.
?instance_property:hasUnitOfMeasure ?unit2.
bind  ((if  (?real_value=?predicted_value,  "equal",  "NOT  EQUAL"))  AS
?equality)
bind   ((if   (?real_value=?predicted_value,   "0",   (?real_value   -
?predicted_value))) AS ?difference )
bind ((if (?unit1=?unit2, "","DIFFERENT UNITS")) AS ?units_consistency
)}
```

Listing 6: SPARQL query for counting a difference between measured and real value of a property

The results proving that the ontology satisfies the competency questions are presented in the Fig. 16. The full code of the query can be found in the appendix A 6.



instance	property	real_value	predicted_value	equality	difference	units_consistency	characteristic
98765623-60-40	Weight	"18.0"^^<http://www.w3.org/20(	"18.0"^^<http://www.w3.org/20(	:"equal"	"0"	""	Volume
98765623-0000-12	Weight	"18.0"^^<http://www.w3.org/20(	"15.4"^^<http://www.w3.org/20(	:"NOT EQUAL"	"2.5999999999999996"^^<http	"DIFFERENT UNITS"	Volume
98765623-0000-11	Weight	"15.4"^^<http://www.w3.org/20(	"15.4"^^<http://www.w3.org/20(	:"equal"	"0"	"DIFFERENT UNITS"	Volume
98765623-60-41	Weight	"18.1"^^<http://www.w3.org/20(	"18.0"^^<http://www.w3.org/20(	:"NOT EQUAL"	"0.10000000000000142"^^<htl	""	Volume
23456432-0000	Weight	"79.27"^^<http://www.w3.org/2(	"79.0"^^<http://www.w3.org/20(	:"NOT EQUAL"	"0.2699999999999996"^^<http:/	""	Volume
98765623-0000-13	Weight	"18.0"^^<http://www.w3.org/20(	"15.4"^^<http://www.w3.org/20(	:"NOT EQUAL"	"2.5999999999999996"^^<http	"DIFFERENT UNITS"	Volume

Fig. 16: Result of the query

Many other requests can be made to further explore the possibilities of this data structuring approach. Thus, the proposed model can be expanded and used to address various cases.

6 Conclusion

The first goal of this thesis was to explore the concept of Digital Twin. In the process of searching for a universal definition, a variety of definitions were found, many of which describe concepts with different purposes, different scales, and different scopes of functionality. Having sorted the definitions by the date of their creation, the author noticed that this inconsistency in definitions is the result of the evolution of the concept in the absence of any standards. The concept that was created to support the product lifecycle management process evolved in '*integrated system that can simulate, monitor, calculate, regulate, and control the system status and process... [and] has the characteristics of individualization, high efficiency and highly quasi-real... [and] is developed by data acquisition, virtual manufacturing technologies, based on the control, computation and communication units*' (11). The different interpretations of a Digital Twin which have arisen over the course of this evolution have been found in the scientific literature and summarised in the Table 2.

Another contribution of this thesis is the discovery of the term Digital Shadow in connection with the Digital Twin concept. The Digital Shadow can be seen as a container of interoperable data which represents a reflection of the state of an object at a certain point in time. The term 'Digital Shadow' describes a concept, that is very close to the oldest definitions of the Digital Twin, but the Digital Twins in modern interpretations do not only collect data, but also react to information. However, all of the definitions independent of the time of their origin, have one thing in common: a Digital Twin stores a data profile of its physical counterpart. In this thesis, Digital Shadow is proposed as a term to define this fundamental and essential aspect of every Digital Twin. Thereby, it was shown in this thesis that when developing a Digital Twin, the first major effort should be to locate, bind, and ensure the interoperability of data. Interconnected, interoperable and easy-to-access data strongly support many business processes, since they provide a foundation for deriving knowledge about a subject. This can be partially accomplished using Semantic Web tools. Semantic Web technology provides tools to communicate data, by adding an annotation about its meaning. Because of Semantic Web technologies, the creation of large-scale Digital Shadows gets possible. The application of Semantic Web tools in the context of creating Digital Shadows as a foundation for Digital Twins was demonstrated by creating a conceptualisation of a Digital Shadow in the form of an ontology. Filling the ontology with realistic engineering data demonstrated how heterogenous data from different sources can be connected in a meaningful way. The proposed ontology can be used as a foundation for structuring data within the Digital Shadow/Digital Twin. Identification of a characteristic-property modelling patterns (which differentiates two logically different groups of product's features and enables an explicit definition of dependecy between them) and type-instance pattern (which enables representation of the Digital Twin on different phases of its lifecycle, as well as a Digital Twin of different levels of aggregation) was considered the most valuable contribution of the ontology model, since it can enable a better structuring of the product's data within its digital model (see appendix A 7). In accordance with the objectives pursued, the proposed semantic model can be extended or modified and then filled with data from real data bases. In the long term, when the creation of digital entities with all their measured properties becomes an automated process, this approach can also be used for the partial automation of quality control processes.

The proposed model can meet the requirements of use cases of a similar nature. However, reusing the proposed model for situations, in which the ontology is used by various applications of a heterogeneous functionality may be difficult. The trade-off between usability and reusability is a common issue for large companies. When application perspectives are not taken into consideration an ontology becomes less usable, whereas concentration on specific application requirements makes an ontology less reusable (and also makes it similar to a conceptual data schema) (48). A partial solution can be achieved by using methodologies focused on this problem. An example of such methodology is so called DOGMA methodology (48). The DOGMA approach expands beyond this tradeoff by making an ontology that is 'doubly articulated into a domain axiomatisation and application axiomatisation' (48), where axiomatisation is used in the meaning of knowledge specification (that is, a set of axioms about a certain subject-matter). Transferring the proposed model to two different models: domain axiomatisation and a use case axiomatisation, would help to avoid the use of multiple subclasses, therefore making the taxonomy smaller and more intuitive. Reusability issues would also be avoided.

Since the study of existing ontologies is beyond the scope of this thesis, the proposed ontology exists on its own. However, in practice, new ontologies are often associated with existing ones in order to speed up the development process, avoid duplicate efforts, and ensure interoperability of the new model. With that being said, in the future this model can be expanded by mapping it to other ontologies, and the classes it contains can be combined with the corresponding classes of already existing ontologies. Among other things, there is a high probability that an overview of related ontologies will help to detect connections and dependencies in the domain, that have been overlooked or not taken into consideration in this research.

References

[1] EDGAR WEIPPL, Benjamin Sanderse. Digital Twins [online]. 2018, (115), 6-7. Available from: https://ercim-news.ercim.eu/en115/special/2091-digital-twins-introduction-to-the-special-theme.

[2] STĂNCIOIU, Alin. THE FOURTH INDUSTRIAL REVOLUTION „INDUSTRY 4.0". *Fiability & Durability/Fiabilitate si Durabilitate,(1).* 2017.

[3] ZEZULKA, F., P. MARCON, I. VESELY, and O. SAJDL. Industry 4.0 – An Introduction in the phenomenon [online]. *IFAC-PapersOnLine.* 2016, **49**(25), 8-12. Available from: 10.1016/j.ifacol.2016.12.002.

[4] RENNUNG, Frank, Caius Tudor LUMINOSU, and Anca DRAGHICI. Service Provision in the Framework of Industry 4.0 [online]. *Procedia - Social and Behavioral Sciences.* 2016, **221**, 372-377. Available from: 10.1016/j.sbspro.2016.05.127.

[5] *What is industry 4.0 and how it increases machine efficiency? - ThingTrax* [viewed 1 April 2019]. Available from: https://thingtrax.com/2017/10/05/industry-4-0-increases-machine-efficiency/.

[6] ZHUANG, Cunbo, Jianhua LIU, and Hui XIONG. Digital twin-based smart production management and control framework for the complex product assembly shop-floor [online]. *The International Journal of Advanced Manufacturing Technology.* 2018, **96**(1-4), 1149-1163. Available from: 10.1007/s00170-018-1617-6.

[7] GRIEVES, Michael, and John VICKERS. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: Franz-Josef Kahlen, Shannon Flumerfelt, and Anabela Alves, eds. *Transdisciplinary Perspectives on Complex Systems. New Findings and Approaches.* Cham: Springer International Publishing; Imprint: Springer, 2017, pp. 85-113.

[8] HTTP://WWW.FACEBOOK.COM/COBUILDER.INTERNATIONAL. *The 'digital twin' – a bridge between the physical and the digital world* [online]. 2 May 2019, 12:00 [viewed 2 May 2019]. Available from: https://cobuilder.com/en/the-digital-twin-a-bridge-between-the-physical-and-the-digital-world/.

[9] STEFAN BOSCHERT, Roland Rosen. Digital Twin: a Second Life for Engineering Models [online]. 2018, (115), 8-9. Available from: https://ercim-news.ercim.eu/en115/special/2091-digital-twins-introduction-to-the-special-theme.

[10] NEGRI, Elisa, Luca FUMAGALLI, and Marco MACCHI. A Review of the Roles of Digital Twin in CPS-based Production Systems [online]. *Procedia Manufacturing.* 2017, **11**, 939-948. Available from: 10.1016/j.promfg.2017.07.198.

[11] ZHENG, Yu, Sen YANG, and Huanchong CHENG. An application framework of digital twin and its case study [online]. *Journal of Ambient Intelligence and Humanized Computing.* 2019, **10**(3), 1141-1153. Available from: 10.1007/s12652-018-0911-3.

[12] JULIESETO. *Übersicht über Azure Digital Twins* [online] [viewed 2 May 2019]. Available from: https://docs.microsoft.com/de-de/azure/digital-twins/about-digital-twins.

[13] PAUL BUTTERWORTH. *White Paper: VANTIQ Digital Twin Architecture,* November 2017.

[14] GERARDO SANTILLAN MARTINEZ, TOMMI KARHELA, REINO RUUSU, JIHA KORTELAINEN. Automatic Generation of Simulation-based Digital Twins of Industrial Process Plants [online]. 2018, (115), 8-9. Available from: https://ercim-news.ercim.eu/en115/special/2091-digital-twins-introduction-to-the-special-theme.

[15] SOMAYEH MALAKUTI, JAN SCHLAKE, STEN GRÜNER, DIRK SCHULZ, RALF GITZEL, JOHANNES SCHMITT, MARIE PLATENIUS-MOHR, PHILIPP VORST. Digital twin – a key software component of Industry 4.0 [online]. Available from: https://new.abb.com/news/detail/11242/digital-twin-a-key-software-component-of-industry-40.

[16] MATTHIAS JARKE, GUENTHER SCHUH, CHRISTIAN BRECHER, MATTHIAS BROCKMANN AND JAN-PHILIPPE PROTE. Digital Shadow in the Internet of Production [online]. 2018, (115), 26-29. Available from: https://ercim-news.ercim.eu/en115/special/2091-digital-twins-introduction-to-the-special-theme.

[17] BRUYNSEELS, Koen, Filippo SANTONI DE SIO, and Jeroen VAN DEN HOVEN. Digital Twins in Health Care: Ethical Implications of an Emerging Engineering Paradigm [online]. *Frontiers in Genetics.* 2018, **9**, 31. Available from: 10.3389/fgene.2018.00031.

[18] MOHAMMADI, Neda, and John E. TAYLOR. Smart city digital twins. *SSCI. 2017 IEEE Symposium Series on Computational Intelligence : November 27, 2017-December 1, 2017.* New York: IEEE, 2018, pp. 1-5.

[19] NIKOLAEV, S., M. GUSEV, D. PADALITSA, E. MOZHENKOV, S. MISHIN, and I. UZHINSKY. Implementation of "Digital Twin" Concept for Modern Project-Based Engineering Education. In: Paolo Chiabert, Abdelaziz Bouras, Frédéric Noël, and José Ríos, eds. *Product lifecycle management to Support Industry 4.0. 15th IFIP WG 5.1 International Conference, PLM 2018, Turin, Italy, July 2-4, 2018, Proceedings.* Cham, Switzerland: Springer, 2018, pp. 193-203.

[20] GILCHRIST, A. *Industry 4.0: The Industrial Internet of Things:* Apress, 2016. 9781484220474.

[21] OVERTON, Jerry, and JC. BRIGHAM. The digital twin data-driven simulation innovate the manufacturing process.

[22] CHIABERT, Paolo, Abdelaziz BOURAS, Frédéric NOËL, José RÍOS, DURÃO, LUIZ FERNANDO C. S., Sebastian HAAG, Reiner ANDERL, Klaus SCHÜTZER, and Eduardo ZANCUL, eds. *Digital Twin Requirements in the Context of Industry 4.0. Product Lifecycle Management to Support Industry 4.0:* Springer International Publishing, 2018. 978-3-030-01614-2.

[23] PROFESSOR DR.-ING. WILHELM BAUER, DR.-ING. SEBASTIAN SCHLUND, DR.-ING. DIRK MARRENBACH. Industrie 4.0 – Volkswirtschaftliches Potenzial fьr Deutschland, 20.

[24] HARIK, Ramy, Louis RIVEST, Alain BERNARD, Benoit EYNARD, Abdelaziz BOURAS, Michael ABRAMOVICI, Jens Christian GÖBEL, and Philipp SAVARINO, eds. *Virtual Twins as Integrative Components of Smart Products. Product Lifecycle Management for Digital Transformation of Industries:* Springer International Publishing, 2016. 978-3-319-54660-5.

[25] TAO, Fei, Jiangfeng CHENG, Qinglin QI, Meng ZHANG, He ZHANG, and Fangyuan SUI. Digital twin-driven product design, manufacturing and service with big data [online]. *The International Journal of Advanced Manufacturing Technology.* 2018, **94**(9), 3563-3576. Available from: 10.1007/s00170-017-0233-1.

[26] *Industry 4.0 and the digital twin technology | Deloitte Insights* [viewed 29 March 2019]. Available from: https://www2.deloitte.com/insights/us/en/focus/industry-4-0/digital-twin-technology-smart-factory.html.

[27] SOMAYEH MALAKUTI, JAN SCHLAKE, STEN GRÜNER, DIRK SCHULZ, RALF GITZEL, JOHANNES SCHMITT, MARIE PLATENIUS-MOHR, PHILIPP VORST. *Digital twin – a key software component of Industry 4.0* [online]. 4 December 2018, 12:00 [viewed 15 January 2019]. Available from: https://new.abb.com/news/detail/11242/digital-twin-a-key-software-component-of-industry-40.

[28] ALAM, Kazi Masudul, and Abdulmotaleb EL SADDIK. C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems [online]. *IEEE Access.* 2017, **5**, 2050-2062. Available from: 10.1109/ACCESS.2017.2657006.

[29] SCHROEDER, Greyce N., Charles STEINMETZ, Carlos E. PEREIRA, and Danubia B. ESPINDOLA. Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange [online]. *IFAC-PapersOnLine.* 2016, **49**(30), 12-17. Available from: 10.1016/j.ifacol.2016.11.115.

[30] VIOLETA DAMJANOVIC-BEHRENDT. A Digital Twin Architecture for Security, Privacy and Safety [online]. 2018, (115), 25-26. Available from: https://ercim-news.ercim.eu/en115/special/2091-digital-twins-introduction-to-the-special-theme.

[31] WAHLSTER, Wolfgang. Semantic Technologies for Mass Customization. In: Wolfgang Wahlster, ed. *Towards the Internet of Services: the THESEUS Research Program.* Cham, Heidelberg: Springer, 2014, pp. 3-13.

[32] OLIVOTTI, Daniel, Sonja DREYER, Benedikt LEBEK, and Michael H. BREITNER. Creating the foundation for digital twins in the manufacturing industry: an integrated installed base management system [online]. *Information Systems and e-Business Management.* 2018. Available from: 10.1007/s10257-018-0376-0.

[33] CYRIL CECCHINEL, MATTHIEU JIMENEZ, SЙBASTIEN MOSSER, MICHEL RIVEILL. An Architecture to Support the Collection of Big Data in the Internet of Things.

[34] ROLSTADÅS, A., B. HENRIKSEN, and D. O'SULLIVAN. *Manufacturing Outsourcing: A Knowledge Perspective:* Springer London, 2012. 9781447129547.

[35] SIMPSON, John Andrew. *The Oxford English dictionary.* 2. ed. Oxford: Clarendon Press, 1989. 0198611862.

[36] ACKOFF, R. L. From data to wisdom. 1989, **16**, 3-9.

[37] JENNEX, Murray E. Re-Visiting the Knowledge Pyramid. In: Ralph H. Sprague, ed. *Proceedings of the 42nd Annual International Conference on System Sciences. Abstracts and CD-ROM of full papers.* Los Alamitos: IEEE Computer Society, 2009, pp. 1-7.

[38] ROLSTADÅS, Asbjørn, Bjørnar HENRIKSEN, and David O'SULLIVAN. What is Knowledge? In: A. Rolstadås, Bjønar Henriksen, and David O'Sullivan, eds. *Manufacturing outsourcing. A knowledge perspective.* London, New York: Springer, 2012, pp. 145-155.

[39] ELLIS, R., S. LOEWEN, P. C. ELDER, H. REINDERS, R. ERLAM, and J. PHILP. *Implicit and Explicit Knowledge in Second Language Learning, Testing and Teaching:* Channel View Publications, 2009. 9781847698858.

[40] CHOW, H. K.H., K. L. CHOY, W. B. LEE, and F. T.S. CHAN. Design of a knowledge-based logistics strategy system. *0957-4174.* 2005.

[41] DAVIES, J., D. FENSEL, and F. VAN HARMELEN. *Towards the Semantic Web: Ontology-driven Knowledge Management:* Wiley, 2003. 9780470858073.

[42] BAADER, Franz, Ian HORROCKS, and Ulrike SATTLER. Chapter 3 Description Logics. *Handbook of Knowledge Representation:* Elsevier, 2008, pp. 135-179.

[43] BRACHMAN, R.J., NARDI, D. An introduction to description logics, pp. 1-40.

[44] GÓMEZ-PÉREZ, Asunción, Mariano FERNÁNDEZ-LÓPEZ, and Oscar CORCHO. *Ontological engineering. With examples from the areas of knowledge management, e-commerce and the semantic Web.* [Nachdr.]. London, Berlin, Heidelberg: Springer, 2010. Advanced information and knowledge processing. 1852338407.

[45] EITER, Thomas and Thomas KRENNWALLNER. *Reasoning Web. Semantic Technologies for Advanced Query Answering.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. 7487. 978-3-642-33157-2.

[46] IQBAL, Rizwan, Masrah Azrifah Azmi MURAD, Aida MUSTAPHA, and Nurfadhlina Mohd SHAREF. An Analysis of Ontology Engineering Methodologies: A Literature Review [online]. *Research Journal of Applied Sciences, Engineering and Technology.* 2013, **6**(16), 2993-3000. Available from: 10.19026/rjaset.6.3684.

[47] THOMAS GRUBER. A Translation Approach to Portable Ontology Specifications. 1993, 199-220.

[48] JARRAR, Mustafa, and Robert MEERSMAN. Ontology Engineering – The DOGMA Approach. In: Tharam S. Dillon, Elizabeth Chang, Robert Meersman, and Katia Sycara, eds. *Advances in web semantics.* Berlin [etc.]: Springer, 2009-, pp. 7-34.

[49] JASPER, ROBERT, AND MIKE USCHOLD, ed. *A framework for understanding and classifying ontology applications,* 1999.

[50] TORO, Carlos, Manuel GRAÑA, Jorge POSADA, Javier VAQUERO, Cesar SANÍN, and Edward SZCZERBICKI. Domain Modeling Based on Engineering Standards.

[51] KLINOV, Pavel and Dmitry MOUROMTSEV. *Knowledge Engineering and Semantic Web.* Cham: Springer International Publishing, 2015. 518. 978-3-319-24542-3.

[52] TIM BERNERS-LEE, JAMES HENDLER AND ORA LASSILA. The Semantic Web. *Scientific American.* 2001, (May), 29-37.

[53] *Semantic Web - W3C* [viewed 30 March 2019]. Available from: https://www.w3.org/standards/semanticweb/.

[54] *Semantic Web, and Other Technologies to Watch: January 2007 (24)* [viewed 7 May 2019]. Available from: https://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(24).

[55] N.HENZE. *Semantic Web. The Proof And Trust Layers Of The Semantic Web,* 17 Jan. 2008. IVS Semantic Web Group.

[56] BIKAKIS, Nikos, Chrisa TSINARAKI, Nektarios GIOLDASIS, Ioannis STAVRAKANTONAKIS, and Stavros CHRISTODOULAKIS. The XML and Semantic Web Worlds: Technologies, Interoperability and Integration: A Survey of the State of the Art. In: Ioannis E. Anagnostopoulos, ed. *Semantic hyper/multimedia adaptation. Schemes and applications.* Heidelberg: Springer, 2013, pp. 319-360.

[57] *Semantic Web: Why RDF is more than XML* [viewed 1 April 2019]. Available from: https://www.w3.org/DesignIssues/RDF-XML.html.

[58] *RDF Vocabulary Description Language 1.0: RDF Schema* [viewed 1 April 2019]. Available from: https://www.w3.org/2001/sw/RDFCore/Schema/200203/.

[59] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML* [viewed 30 March 2019]. Available from: https://www.w3.org/Submission/SWRL/.

[60] FEIGENBAUM, Lee. *SPARQL By Example* [online]. 22 June 2009, 12:00 [viewed 30 March 2019]. Available from: https://www.w3.org/2009/Talks/0615-qbe/.

[61] N. F. NOY, D.L. MCGUINNESS. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford Knowledge Systems Laboratory, March 2001.

[62] GARETTI, Marco, Sergio TERZI, Norma BERTACCI, and Maurizio BRIANZA. Organisational change and knowledge management in PLM implementation [online]. *International Journal of Product Lifecycle Management.* 2005, **1**(1), 43. Available from: 10.1504/IJPLM.2005.007344.

[63] PRATT, Michael J. Introduction to ISO 10303—the STEP Standard for Product Data Exchange [online]. *Journal of Computing and Information Science in Engineering.* 2001, **1**(1), 102. Available from: 10.1115/1.1354995.

[64] MTConnect Institute, *MTConnect. Available from:* http://mtconnect.org/index.php?option=com_content&task=category&sectionid=4&id=56&Itemid=154.

[65] Details of the Administration Shell - from idea to implementation [UPDATE]. 19 March 2019, 12:00 [viewed 20 April 2019]. Available from: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/vws-in-detail-presentation.html.

[66] *Web Thing API* [viewed 2 April 2019]. Available from: https://iot.mozilla.org/wot/.

[67] MARTINS, Jaime A., Andriy MAZAYEV, and Noelia CORREIA. Hypermedia APIs for the Web of Things [online]. *IEEE Access.* 2017, **5**, 20058-20067. Available from: 10.1109/ACCESS.2017.2755259.

[68] ADOLPHS, Peter, Sören AUER, Heinz BEDENBENDER, Maik BILLMANN, Bill SE, Gökhan COSKUN, and Martin EHLICH. *Fortentwicklung des Referenzarchitekturmodells für die Industrie 4.0-Komponente,* 27 Apr. 2016.

[69] *Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)* [viewed 2 April 2019]. Available from: https://www.zvei.org/presse-medien/publikationen/das-referenzarchitekturmodell-industrie-40-rami-40/.

[70] BIRKHOFER, Herbert and Martin WAELDELE. Properties and characteristics and Attributes and…-an approach on structuring the description of technical systems.

[71] WEBER, Christian, ed. *CPM/PDD–an extended theoretical approach to modelling products and product development processes.* 10th ed.: Fraunhofer-IRB-Verlag Stuttgart, 2005. 6.

[72] DR.-ING. PETER ADOLPHS, PROF. DR. SÖREN AUER, DR. HEINZ BEDENBENDER, MEIK HANKEL, ROLAND HEIDEL, DR. -ING. MICHAEL HOFFMEISTER, HAIMO HUHLE, MICHAEL JOCHEM, MARKUS KIELE-DUNSCHE, GUNTHER KOSCHNICK, DR. HEIKO KOZIOLEK, LUKAS LINKE, REINHOLD PICHLER, FRANK SCHEWE, KARSTEN SCHNEIDER, BERND WASER. Structure of the Administration Shell. June 2015.

[73] VAUGHAN, C. and NATIONAL AGENCY FOR FINITE ELEMENT METHODS & STANDARDS (Great Britain). *Product data management and the engineering analysis environment:* NAFEMS, 2001.

[74] NOY, Natasha. *Simple part-whole relations in OWL Ontologies,* 13 Aug. 2005.

[75] *SPARQL 1.1 Query Language* [viewed 8 May 2019]. Available from: https://www.w3.org/TR/sparql11-query/#grammar.

[76] *Digital twin – a key software component of Industry 4.0* [viewed 29 March 2019]. Available from: https://new.abb.com/news/detail/11242/digital-twin-a-key-software-component-of-industry-40.

[77] GERHARD, Detlef. Product Lifecycle Management Challenges of CPPS. In: Stefan Biffl, Arndt Lüder, and Detlef Gerhard, eds. *Multi-Disciplinary Engineering for Cyber-Physical Production Systems. Data Models and Software Solutions for Handling Complex Engineering Projects.* Cham: Springer International Publishing; Imprint: Springer, 2017, pp. 89-110.

Appendix

A 1 Architecture of a Digital Shadow

S.Malakuti et al. (27) describe the usage of digital models in the aerospace industry. They give a comprehensive description of the attributes characterizing the Digital Twins. Since the Digital Shadow has a lot in common with the Digital Twin concept, it can also be described as:

- Being a high-fidelity model of an as-manufactured component or a set of components. Such a model includes important physical characteristics, such as material microstructure, defects, etc. Such a model of a physical object provides the means to predict future conditions of a vehicle.
- Relating to a high extent on constantly updating data describing a current state of characteristics of interest with a high frequency. Such ability lets the model always be up-to-date about the current state of the physical asset and therefore enables a function for keeping track for the health and performance of a system.
- Being created uniquely for its counterpart in order to monitor its health and to predict possible risks for this specific system before, during and after its usage.
- Integrating historical information in order to build the most accurate data-profile of the vehicle. That can be done by computers that could analyze and proceed the text of reports.
- Reflecting comprehensively a history of usage of this specific asset. Therefore the highly accurate prediction for the moment to change components of the system can be done.

The digital model can be called a data container, that creates a means for the data from different sources to be stored together in a way that makes them compatible with each other (27) . Such a container would "*deliver previously unseen interoperability out of the box*" (76). Since the Digital Twin can be seen as a Digital

Shadow enriched by various sub-modules pursuing different roles, it offers richer functionality, but also much higher implementation cost. The digital models of different scales might not only change the performance of the asset, but business performance in general. "*The development of products that are networked within their operational environment and the support of service-oriented business models requires the linkage of traditional product data with the digital shadow of the delivered product configuration... as well as the use and association with data of production and operation. A shift from divided designs of physical systems, control subsystems and software architecture to integrated and optimized design can be observed with respect to the process of product and production systems engineering.*" (77). When deciding about complexity and the right architecture for a digital model, economical balance between added business value and money investigations must be found by companies in every unique case. Different factors that might be affected by a Digital Shadow must be taken into account, such as quality improvements, warranty costs optimization, new level of service quality, better management of quality issues through recording of serialized parts, reducing time and therefore costs needed to introduce a new product, revenue growth opportunities and reducing operational costs through improvement of equipment performance, product design and reducing operations variability.

As it was mentioned before, the foundation for a Digital Twin was proposed by Olivotti (32). Since the management system described by the author corresponds with an idea of a Digital Shadow, proposed architecture (see Fig. 17) of the management system can also be an architecture of a Digital Shadow.

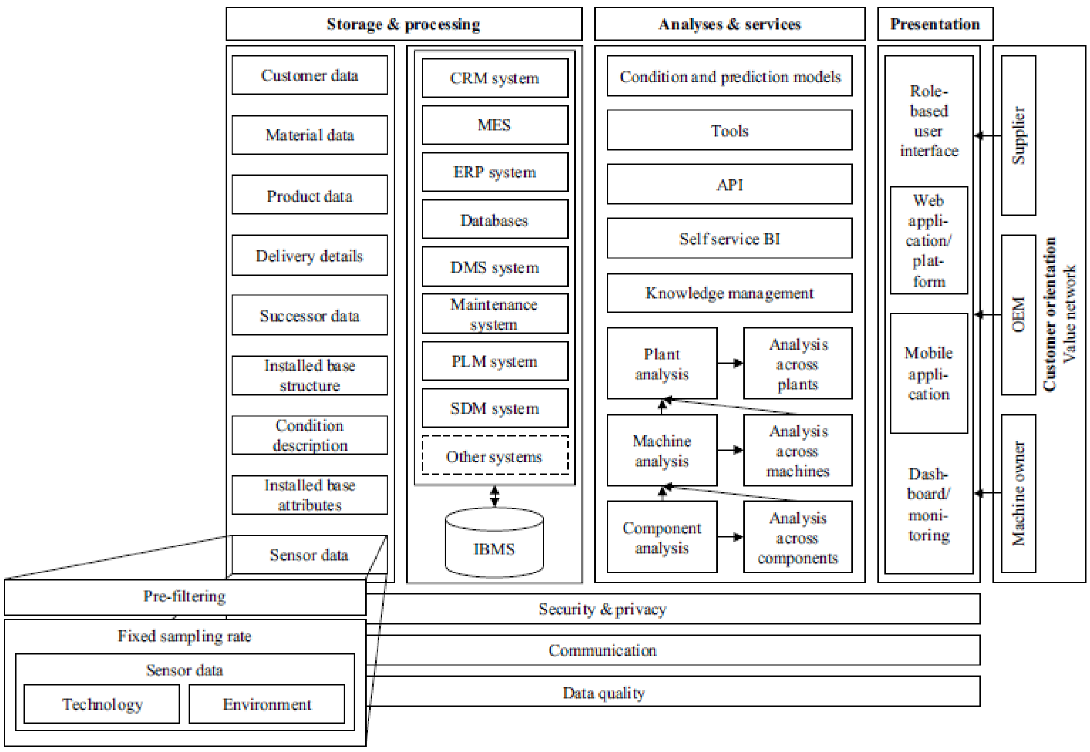


Fig. 17: Integrated base management system
(32)

The first layer is describing data storage and processing. Cells of the first left column present different data collection points. Sensor data require an additional module. Due to the high speed and volume of incoming data, means to correctly process it are pre-filtering and an adequate sampling rate. Since the source of sensor data can be either machine itself or additional sensors of a machine, every technology needs to be aware of what environmental sensors are relevant to it. The second column shows the systems involved in retrieving the information. Those systems are not linked to the data points in the first column since every company realized those connections in its own way. The second layer integrates services, addressing Digital Twin. This layer is necessary to provide means for the analysis and comparing of the data across plants, machines, components. Another very important task of this layer is managing knowledge, that can be done through correlating actual sensor data with existing knowledge. Different solutions to present the information are concerned in the third layer. Choice of a presentation tool must be based on the roles of the partners in the value network. Finally, security, privacy, communication, and data quality are important and must be of concern across all layers. The author emphasizes things, that must be concerned on different levels of the architecture, such as intrusion detecting and preventing systems, device identity access management, vulnerability assessment, and others.

A 2 Web Thing Description Example

```
{ "@context": "https://iot.mozilla.org/schemas/",
  "@type": [
    "Light",
    "OnOffSwitch"
  ],
  "name": "My Lamp",
  "description": "A web connected lamp",
  "properties": {
    "on": {
      "@type": "OnOffProperty",
      "type": "boolean",
      "title": "On/Off",
      "description": "Whether the lamp is turned on",
      "links": [
        {
        "href": "/things/lamp/properties/on"}
      ]
    },
    "brightness": {
      "@type": "BrightnessProperty",
      "type": "integer",
      "title": "Brightness",
      "description": "The level of light from 0-100",
      "minimum": 0,
      "maximum": 100,
      "links": [
        {
          "href": "/things/lamp/properties/brightness"
        }
      ]
    }},
  "actions": {
    "fade": {
      "@type": "FadeAction",
      "title": "Fade",
      "description": "Fade the lamp to a given level",
      "input": {
        "type": "object",
        "properties": {
          "level": {
            "type": "integer",
            "minimum": 0,
            "maximum": 100
          },
          "duration": {
            "type": "integer",
            "minimum": 0,
            "unit": "milliseconds"
          }
        }
      },
      "links": [
        {
          "href": "/things/lamp/actions/fade"
        }
      ]
    }
  },
  "events": {
    "overheated": {
      "title": "Overheated",
      "@type": "OverheatedEvent",
      "type": "number",
      "unit": "degree celsius",
      "description":  "The  lamp  has  exceeded  its  safe  operating
temperature",
      "links": [
        {
          "href": "/things/lamp/events/overheated"
        }
      ]
    }
  },
  "links": [
    {
      "rel": "properties",
      "href": "/things/lamp/properties"
    },
    {
      "rel": "actions",
      "href": "/things/lamp/actions"
    },
    {
      "rel": "events",
      "href": "/things/lamp/events"
    },
    {
      "rel": "alternate",
      "href": "wss://mywebthingserver.com/things/lamp"
    },
    {
      "rel": "alternate",
      "mediaType": "text/html",
      "href": "/things/lamp"
    }]}
```

Listing 7: Description of a Web-connected lamp in terms of Web Thing

A 3 Questionary for the Interview №1

Dictionary

1. Could you define value-drivers?
2. Could you define cost-drivers?

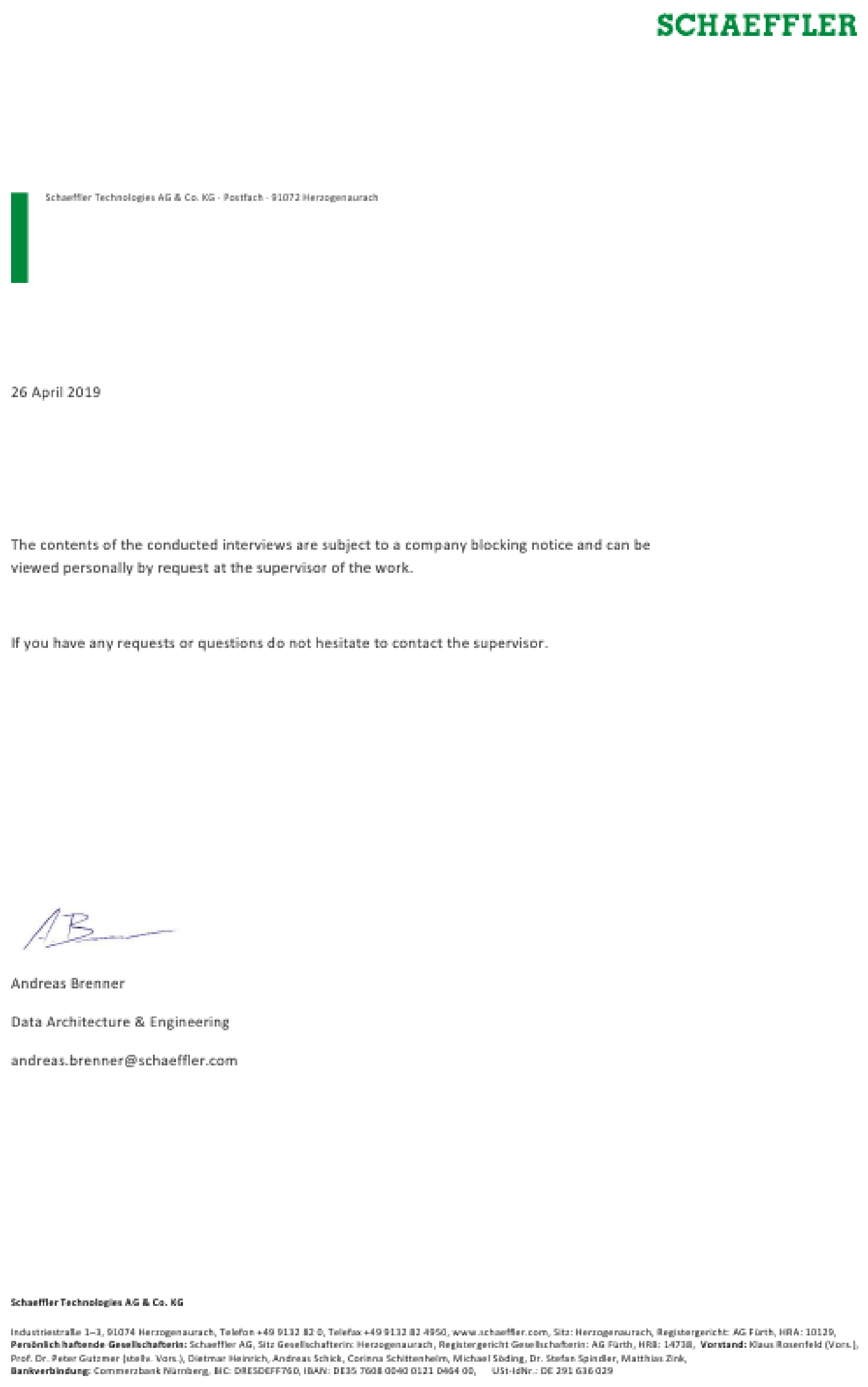
Work process

1. Could you give a few examples of product you analyze on your daily basis?
2. Could you describe your day to day job and responsibilities?
3. Could you describe the process of value-driver identification?
4. Could you describe the process of cost-driver identification?
5. Why do you define value and cost? What value this analysis brings?
6. How this information is getting used after the analysis is completed?
7. Do you define value/cost drivers of the same product more than once?

Data Quality

1. What kind of data is available for making an analysis? Can you give examples?
2. Could you name the sources of data you use?
3. What formats the data you use have (structured/unstructured)?
4. Are there properties that are taken into account more frequently? With other words, are there usual suspects that you use on a regular basis for your analysis?
5. How often do you face the problem of having incomplete/incorrect data?
6. How often incomplete/incorrect data is revealed post-factum? What is the consequence?
7. On what phase or at what moment of time do you recognize incomplete/incorrect data?
8. In case you have recognized incorrect data, what are your actions?
9. Could you say that some sort of incorrect data might cause more troubles than another?
10. Could as-manufactured data for each produced item be useful for cost/value analysis?

A 4 Interviews Transcription



SCHAEFFLER

Schaeffler Technologies AG & Co. KG · Postfach · 91072 Herzogenaurach

26 April 2019

The contents of the conducted interviews are subject to a company blocking notice and can be viewed personally by request at the supervisor of the work.

If you have any requests or questions do not hesitate to contact the supervisor.

[illegible]

Andreas Brenner

Data Architecture & Engineering

andreas.brenner@schaeffler.com

Schaeffler Technologies AG & Co. KG

Industriestraße 1–3, 91074 Herzogenaurach, Telefon +49 9132 82 0, Telefax +49 9132 82 4950, www.schaeffler.com, Sitz: Herzogenaurach, Registergericht: AG Fürth, HRA: 10129,
Persönlich haftende Gesellschafterin: Schaeffler AG, Sitz Gesellschafterin: Herzogenaurach, Registergericht Gesellschafterin: AG Fürth, HRB: 14738, **Vorstand:** Klaus Rosenfeld (Vors.),
Prof. Dr. Peter Gutzmer (stellv. Vors.), Dietmar Heinrich, Andreas Schick, Corinna Schittenhelm, Michael Söding, Dr. Stefan Spindler, Matthias Zink,
Bankverbindung: Commerzbank Nürnberg, BIC: DRESDEFF760, IBAN: DE35 7608 0040 0121 0464 00, USt-IdNr.: DE 291 636 029

A 5 Populating an Ontology with Excel Data

At first, the examples of data tables used in the industry were viewed. Based on it, Excel tables with neutral data were created. To populate a Digital Shadow ontology, Excel tables demonstrated in the Table 19 and Table 20 were used as a data source for writing Celfie rules (see Listing 8).

DocumentNumber	Usage	Mass	Unit
P-9876-0000-99	Proposal/Delivery	79	g
P-2347.02-4361	Proposal/Delivery	15,4	kg
P-2347.02-4300	Proposal/Delivery	18	g

Table 19: Excel sheet „Tabelle 1“

id	name	ArticleNumber	Weight(brutto)	Unit
23456432-0000	Part1	P-9876-0000-99	79	g
98765623-0000-11	Part2	P-2347.02-4361	15,4	g
98765623-0000-12	Part 3	P-2347.02-4361	18	G
98765623-0000-13	Part 4	P-2347.02-4361	18	G
98765623-60-40	Part 5	P-2347.02-4300	18	G
98765623-60-41	Part 6	P-2347.02-4300	18,1	G

Table 20: Excel sheet „Sheet 1“

```
{
  "Collections": [
    {
      "sheetName": "Sheet1",
      "startColumn": "A",
      "endColumn": "A",
      "startRow": "2",
      "endRow": "+",
      "comment": "",
      "rule":   "Individual:@D1(mm:printf(\"Weight_%s\",@A*))\nTypes:
PropertyOfInstance\nFacts:      hasNumericalValue      @D*(xsd:double
mm:decimalFormat(\"##0.00\")),\nhasUnitOfMeasure
@E*,\nrepresentsProperty @C1(mm:printf(\"Weight\"))",
      "active": true
    },
    {
      "sheetName": "Tabelle1",
      "startColumn": "A",
      "endColumn": "A",
      "startRow": "2",
      "endRow": "+",
      "comment": "",
      "rule": "Individual: @C1(mm:printf(\"Mass_%s\", @A*))\nTypes:
PropertyOfType\nFacts: hasNumericalValue @C*(xsd:double
mm:decimalFormat(\"##0.00\")),\nhasUnitOfMeasure
@D*,\nrepresentsProperty @C1(mm:printf(\"Weight\"))",
      "active": true
    },
    {
      "sheetName": "Tabelle1",
      "startColumn": "A",
      "endColumn": "A",
      "startRow": "2",
      "endRow": "+",
      "comment": "",
      "rule": "Individual: @D*\nTypes: UnitOfMeasure",
      "active": true
    },
    {
      "sheetName": "Sheet1",
      "startColumn": "A",
      "endColumn": "A",
      "startRow": "2",
      "endRow": "+",
      "comment": "",
      "rule": "Individual: @A*\nTypes: InstanceOfAsset\nFacts:
isDescribedByProperty
@D1(mm:printf(\"Weight_%s\",@A*)),\nisInstanceOf @C*",
      "active": true
    },
    {
      "sheetName": "Tabelle1",
      "startColumn": "A",
      "endColumn": "A",
      "startRow": "2",
      "endRow": "+",
      "comment": "",
      "rule": "Individual: @A*\nTypes: TypeOfAsset\nFacts:
isDescribedByProperty @C1(mm:printf(\"Mass_%s\", @A*))",
      "active": true
    },
    {
      "sheetName": "Tabelle1",
      "startColumn": "A",
      "endColumn": "A",
      "startRow": "1",
      "endRow": "+",
      "comment": "",
      "rule":     "Individual:     @C1(mm:printf(\"Weight\"))\nTypes:
TypeOfProperty",
      "active": true
    }
  ]
```

Listing 8: JSON file for mapping data from Excel sheets

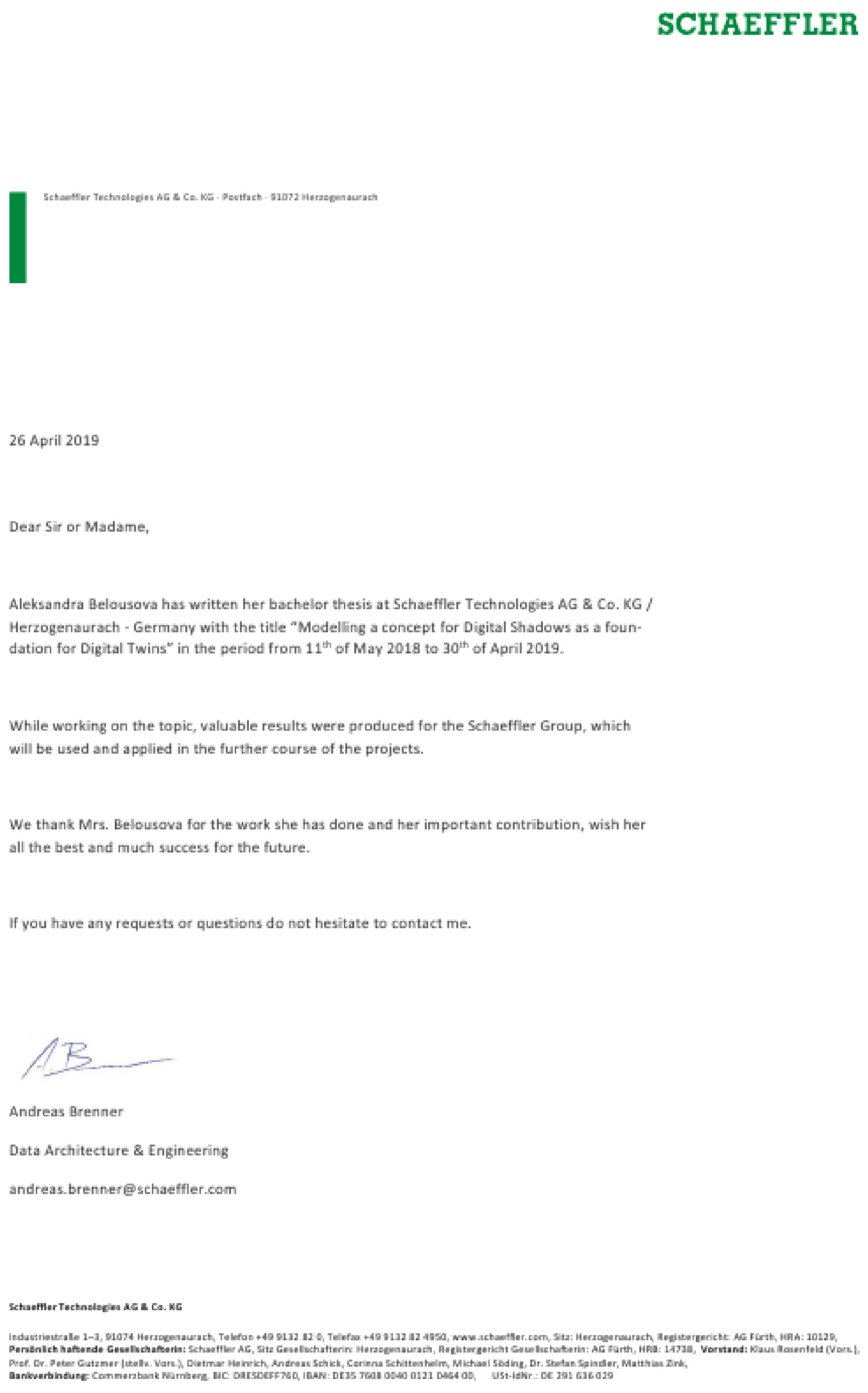
A 6 Full SPARQL Query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX                                                     dsh:
<http://www.semanticweb.org/belouaek/ontologies/2019/3/untitle
d-ontology-19#>
SELECT   ?instance   ?property   ?real_value   ?predicted_value
?equality ?difference ?units_consistency
     WHERE {
     ?instance a dsh:InstanceOfAsset.
     ?instance dsh:isDescribedByProperty ?instance_property.
     ?instance_property dsh:representsProperty ?property.
     ?type a dsh:TypeOfAsset.
     ?type dsh:isDescribedByProperty ?type_property.
     ?type_property dsh:representsProperty ?property.
     ?instance dsh:isInstanceOf ?type.
     ?instance_property dsh:hasNumericalValue ?real_value.
     ?type_property dsh:hasNumericalValue ?predicted_value.
     ?type_property dsh:hasUnitOfMeasure ?unit1.
     ?instance_property dsh:hasUnitOfMeasure ?unit2.


     bind  ((if  (?real_value=?predicted_value,  "equal",  "NOT
EQUAL")) AS ?equality)
     bind ((if (?real_value=?predicted_value, "0", (?real_value
- ?predicted_value))) AS ?difference )
     bind  ((if  (?unit1=?unit2,  "","DIFFERENT  UNITS"))  AS
?units_consistency )
}
```

Listing 9: Full SPARQL query for testing the ontology agains competency questions

A 7 Note about the Achieved Results



SCHAEFFLER

Schaeffler Technologies AG & Co. KG · Postfach · 91072 Herzogenaurach

26 April 2019

Dear Sir or Madame,

Aleksandra Belousova has written her bachelor thesis at Schaeffler Technologies AG & Co. KG / Herzogenaurach - Germany with the title "Modelling a concept for Digital Shadows as a foundation for Digital Twins" in the period from 11th of May 2018 to 30th of April 2019.

While working on the topic, valuable results were produced for the Schaeffler Group, which will be used and applied in the further course of the projects.

We thank Mrs. Belousova for the work she has done and her important contribution, wish her all the best and much success for the future.

If you have any requests or questions do not hesitate to contact me.

[illegible]

Andreas Brenner

Data Architecture & Engineering

andreas.brenner@schaeffler.com

Schaeffler Technologies AG & Co. KG

Industriestraße 1–3, 91074 Herzogenaurach, Telefon +49 9132 82-0, Telefax +49 9132 82 4950, www.schaeffler.com, Sitz: Herzogenaurach, Registergericht: AG Fürth, HRA: 10129,
Persönlich haftende Gesellschafterin: Schaeffler AG, Sitz Gesellschafterin: Herzogenaurach, Registergericht Gesellschafterin: AG Fürth, HRB: 14738, **Vorstand:** Klaus Rosenfeld (Vors.),
Prof. Dr. Peter Gutzmer (stellv. Vors.), Dietmar Heinrich, Andreas Schick, Corinna Schittenhelm, Michael Söding, Dr. Stefan Spindler, Matthias Zink,
Bankverbindung: Commerzbank Nürnberg, BIC: DRESDEFF760, IBAN: DE35 7608 0040 0121 0464 00, USt-IdNr.: DE 291 636 029

Fig. 18: Declaration of results