Development of an ontology-based Description for increasing the degree of automation in production

Dissertation to obtain the academic degree

Doctor - engineer

(Dr.-Ing.)

at the Media Faculty of the Bauhaus University in Weimar

submitted by

Andrew Colorful

born in Rinteln

Appraiser:

1. Prof Dr Benno Stein

2. Prof Dr Oliver Niggeman

Filed on: April 15, 2020

Date of the disputation: December 11, 2020

thanksgiving

This thesis was created during my work as a research assistant at the Institute for Industrial Information Technology (inIT) at the Ostwestfalen-Lippe University of Applied Sciences. Although I put countless hours into this work, it would not have been possible without the support of others.

First of all, I would like to thank Prof. Dr. I would like to thank Benno Stein for taking over the first review of my work. The intensive exchange with him has always challenged me and his keen eye has enriched the work. I also thank Prof. Dr. Oliver Niggemann for his support, the trust placed in me and the supervision of my work, especially after his move to the Helmut Schmidt University in Hamburg.
His ideas and our numerous discussions have made a significant contribution to the fact that the work is available in this form.

In my six years at inIT I was able to work with many colleagues. The numerous and exciting discussions have always motivated me and will always be fondly remembered. In particular, I thank the research group leader Natalia Moriz for the many discussions and comments on my publications, as well as for the freedom created so that I could carry out this work. Even though the topic of my colleague Peng Li's doctoral thesis is very different, the similar schedule ensured that we were able to exchange ideas and ask for advice.

I would like to thank Alexander Diedrich for the exchange and the discussions on the topic of model-based diagnosis and the always cheerful atmosphere that he spreads at work or at conferences.

For a long time I did not consider the path of a doctorate for myself. prof
dr Volker Feige showed me this possibility for the first time during my studies and significantly influenced my first scientific steps. During my master's thesis, I was given the opportunity to do a doctorate with Dr. Bernd Niemöller pointed out once more and by Dr. Johannes Sonnen supported. I would like to thank you for this encouragement and encouragement, because without it I would not have walked this path.

I would especially like to thank my family, especially my parents, who always supported me throughout my apprenticeship and thus enabled me to pursue my projects.

Andrew Colorful

Lemgo, April 2020

summary

The shorter product lifecycles that can be observed and faster market penetration of product technologies require adaptive and efficient production systems.
Adaptivity allows the production line to be adapted to new products, and
the efficiency of the plant ensures that sufficient products are available in a short time
and can be manufactured at low cost. Adaptivity can be achieved by modularizing the production system. However, nowadays everyone requires
Adaptation manual effort, eg to adapt proprietary signals or to
Adaptation of higher-level functions. This reduces the efficiency of the system
ge.

The aim of this work is to ensure interoperability with regard to the use of information in modular production systems. Information is required for this
described by semantic models. This ensures uniform access to information
and higher-level functions have access to all information on the production modules, regardless of the type, manufacturer and age of the module.
This eliminates the manual effort involved in making adjustments to the modular production system, which increases the performance of the system and reduces downtimes
will.

After determining the requirements for a modeling formalism, potential formalisms were compared with the requirements. OWL DL turned out to be a suitable formalism and was used to create the semantic model in this
work used. A semantic model was created as an example for the three use cases *interaction, orchestration* and *diagnosis .* By comparing the
Modeling elements from different use cases were evaluated for the general validity of the model. It was shown that the achievement of a general
Model for technical use cases is possible and only a few hundred terms
needed.

A versatile production system was used to evaluate the models created
the SmartFactoryOWL used on which the use cases were implemented. To
a runtime environment was created that contains the semantic models of the individual modules combined into an overall model, data from the plant is transferred to the model and a
provides an interface for the services. The services implement higher-level functions
and use the information of the semantic model. In all three use cases, the semantic models were correctly assembled and, with the information contained therein, the task of the respective use case could be carried out without additional
be solved manually.

contents

1 Introduction

Product life cycles have shortened in recent decades [1], while the
Market penetration of new product technologies has progressed faster [2].
This means that more and more products are being manufactured in an ever shorter period of time
Need to become. An additional challenge is the increasing individualization
of the products [3]. All of this means that conventional production facilities are becoming obsolete
Pushing limits, because nowadays an adaptation of the systems often requires a considerable
amount of manual effort [4]. The long-term goal is to create highly adaptable production systems
to realize. Under adaptability, the completely autonomous adaptation of the
Production plant understood to changed environmental conditions, such. B. a changed plant
topology. "This necessary [...] adaptability, agility and interoperability can only be achieved
through the use of semantic technologies" [5]. But
so far it has not been possible to apply a corresponding semantics in production plants
establish. “A scientific foundation is needed for an integrative modeling theory that captures
semantics for production technology in mechanical and plant engineering. Existing theories,
means of description and methods including with it
associated basic technologies from computer science, such as those provided by the model
Based Systems Engineering are not sufficient” [6]. The long-term goal of
Fully autonomous customization is not achievable with today's technologies, they
can, however, already be usefully integrated into production systems. the
This thesis examines the possibilities provided by current semantic technologies and uses
three use cases to show how they can be integrated into production systems
can be used and what benefits result from it.

This chapter first describes the problem in detail. After that the
Relevance of the work clarified. The research questions (FF) follow, which set out the scientific
contribution of the work. In the last section, the structure and the
Outline of this work explained.

1.1 Issue

Modern production plants nowadays typically consist of different ones
Modules that are largely independent of each other, ie each module has its own
Control or control programs in order to fulfill the task intended for it. About ordered functions
such. B. the human-machine interaction
Interface (HMI)) or the diagnosis is taken over by the application level above it, this is shown in
Figure 1.1. The modules are with each other
connected to a so-called fieldbus system (or fieldbus for short) via which they can exchange
data. However, the signals are proprietary, so the associated software is also common

can only be used for one system configuration [3]. For a new system configuration, the higher-level functions and the control programs must be manually
be adapted, which means that the system conversion is expensive and time-consuming [7]. It stands in contradiction to an adaptive production, in which the interoperability of the modules must be guaranteed without having to make manual adjustments [8].
The term interoperability is defined in the IEEE 61012 standard as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [9]. Only if each module provides information about itself and this is used by cross-module functions can these
Adapt functions to changed environmental conditions. Interoperability is therefore the prerequisite for the desired adaptivity. The aim of this work is to enable the interoperability of production modules with regard to the use of information
and to show the resulting adaptivity based on three use cases.
The central element to achieve this goal is the semantic model, which
Brings information into context and enables processing [10].

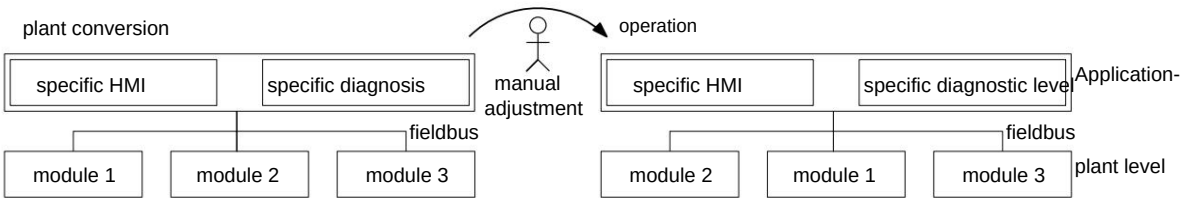

Figure 1.1: In today's production plants, manual work is necessary after a plant conversion, since cross-module functions are created specifically for a plant configuration.

Production systems are usually custom-made, ie different modules are used
compiled and programmed by a system integrator according to the customer's wishes. This is often the fastest method for commissioning such a system
Writing proprietary software, given best practices and standards for a general purpose solution missing. The system integrator himself only has a limited interest in changing this, since it means an increased risk for him during the implementation and a plant conversion brings additional sales. In the future, today's proprietary solutions will not
be more sustainable as the cycles in which assets are rebuilt become shorter
and thus also the financial and time costs increase. To the requirements
However, to meet the demands of the market, the time and effort required for integration must be reduced.

The problem is explained below using a small example: In
An additional module is to be integrated into an existing production system, since this required for a product. For monitoring, the operator1 should have some typical Characteristic values are displayed, including the total power consumption. For the calculation of Power consumption, the software is adjusted manually so that it can match the power signal of the new module and added to the previous value. However, if the signals are described semantically, the power consumption of the individual modules
can be identified, the adjustment would be easy to automate. The total power consumption would only have to be defined as the sum of all individual services and

1For reasons of better legibility, this is used consistently in this work for personal designations generic masculine, which includes persons of all genders.

then the calculation can be performed. This would always be the correct value displayed, regardless of the number of modules and without any manual effort.

Semantic technologies can address the problem described above [5] by Assign semantics to signals, making their values interpretable. Technologically, z. B. through OPC Unified Architecture (OPC UA), the possibility of mapping information models [11]. OPC UA is defined in the higher layers of the ISO-OSI reference model and can therefore use many common communication protocols Use data transfer [12]. But the challenge isn't with the selection solved by a modeling formalism. In order to achieve interoperability, it must be ensured that the data stored in the model can be used, which some brings challenges. Necessarily, the model must be constructed adequately and present the information for the intended application in a complete and unambiguous manner. For example, uniform modeling across manufacturers must be used and not proprietary models. Since each module has its own (partial) model has, the partial models must be combined into an overall model in order to access information about the entire facility. In practice is an efficient creation or reuse of the models is important because modeling has so far been done manually. It is also unclear how application-specific a model is and whether there is perhaps a higher-level "world model" into which all other models fit. Furthermore an inference mechanism is necessary for the information to be used efficiently can become.

The idea of an adaptive production facility is shown in Figure 1.2. after one Plant conversion, the models of the individual modules of the integration level are made available. Due to the defined semantics, these models can automatically become one overall model are aggregated. The necessary information is available in this overall model, e.g. B. of cross-module functions on the application level are needed. Manual adjustment is no longer necessary here.

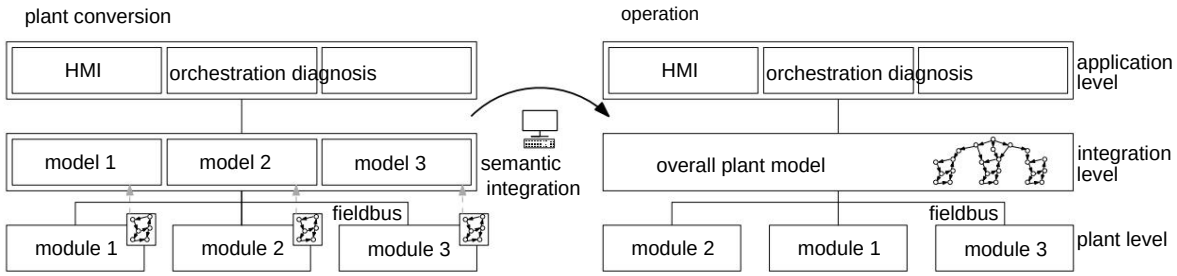

Figure 1.2: If the modules present information in a unified model, a automatic use of the information possible, even after a system conversion.

Due to the different characteristics of production plants and diverse applications [13], it is not possible to provide a generic solution within the scope of this work to work out. For this reason, there are not answers to all the problems addressed, but three typical use cases are defined for which specific solutions are found be worked out. For these use cases, the functionality of the approach and the benefits are presented. In the *Interaction* use case, key figures of the

production plant is automatically and correctly determined, regardless of the plant configuration, in order to make it available to the operator. The *orchestration* describes the mechanical interfaces of the modules. Based on this, an algorithm determines whether the modules are compatible or not. The third use case is *diagnosis* considered, in which each module receives a module description that provides diagnostic information. If an error occurs, a diagnostic algorithm can used to determine possible causes of the error.

1.2 Relevance of the work

The manufacturing industry is facing major challenges because the European Union (EU) wants it to become more economically and ecologically sustainable [3]. But on the market side, there is a trend towards personalized products [3], so that the optimized production plants, which are designed for large quantities, meet this need cannot cover. This can be remedied by adaptive, reconfigurable production modules that are suitable for the manufacture of varied and individualized products are [5]. In addition, these dynamic decentralized production systems enable a more efficient use of resources due to interruptions in a production cycle, varying quality of the input material or bottlenecks are recorded and dealt with decentrally can be [5]. However, modular production systems have to be reconfigured regularly [3] in order to adapt the processes to new products and product types. So far these time-consuming and costly process changes are implemented by engineers and programmers [7]. Therefore, a need for research is seen to establish an automatic Machine integration (this is also called plug-and-produce) [3], since process changes will be possible in the future automatically and without human intervention are [7]. This necessary interoperability can only be achieved by using semantic Technologies can be achieved that dynamically orchestrate the modules and thus implement a plug-and-produce scenario, so that the system is immediately ready for use after a conversion is [5]. But they should not only be able to connect automatically, they should also be able to make decisions themselves [14]. In order to realize such services, all Participants have a common understanding, ie have a common semantics [15]. If the automatic merging of production modules succeeds, then the highest level of intelligent networked systems, as they are a superordinate system (system within the system) [14].

The technologies used in this work are used in many other publications mentioned, but only in connection with the objective and not with a solution [13, 16, 17]. Only in a working group of the Industry 4.0 platform is this topic discussed developed a concrete solution: an Industry 4.0 (I4.0) language. You are with that Realization came that due to the many heterogeneous requirements, there is not one only language, but that application-related adjustments are necessary are [13]. The focus is on the so-called interaction protocols, which e.g. B. specify how a negotiation of an order between two machines proceeds. A concrete one Implementation with the technology does not exist. In particular, there is no semantic Model that can be used to implement multiple use cases, as in this work is presented.

In summary, the relevance and need of adaptive production systems could be presented, for the realization of which semantic technologies are required.
However, concrete implementations of the semantic technologies in the industrial environment are missing so far. This gap is closed in the present work, whereby the specific questions are presented in the next section.

1.3 Research Questions and Scientific Contribution

In order to create interoperability in information processing, information must be modeled in semantic models. All modules have their own model,
which is exchanged and processed between the modules and the integration level can be. Information is processed, among other things, by means of conclusions, which means that implicitly available information is explicitly presented, interpreted and thereby made usable. Whether conclusions are possible depends on that used modeling formalism, so that when choosing a suitable formalism must be observed. Furthermore, the information to be modeled in the industrial environment is very heterogeneous. Nevertheless, they should be represented by a formalism so that only one model is required. In addition, more Boundary conditions, such as B. the clear identification of concepts or versioning options, are met. This results in the first FF:

Research question 1: *Which modeling formalism satisfies for semantic models the requirements for industrial use cases?*

Since there are no semantic models in the industrial environment to date, relevant Use case concepts are identified. The identified concepts must structured and mapped with the formalism selected in FF 1. The ones there Any conflicting goals that arise must be identified and dealt with so that the information is presented in a complete, understandable and compact manner. The aim is to discard the To be able to use information easily and to solve the use cases with it. This results in the FF two:

Research question 2: *How can a semantic model for the use cases of the interaction, orchestration and diagnosis look like?*

When comparing semantic models of different industrial use cases, it is not known how many concepts are used across all use cases
are the same. With a high proportion of the same concepts, the models have a similar Structure and only application-specific extensions have to be added. A Such a semantic model enables easier use of information and
can also be used for many different applications. That offers one significant added value compared to application-specific models. This results in the third FF:

Research question 3: *How similar are the semantic models of the use cases under consideration and can a cross-use case model be created from them?*

Scientific contribution

Twelve scientific articles were published as part of this dissertation. Of these, six contributions were used to create this work, the other six
Contributions were not included in this work to maintain focus. Below is an overview of which publications have contributed to which chapters. This is also shown in Table 1.1.

The problems and the challenges in a somewhat larger context are considered in [8], so that this paper makes a contribution to Chapter 1. The Chapter 2,
3 and 4 define terms, requirements and represent the state of the art
all publications make a small contribution to these chapters.

Three publications contribute to the approach in Chapter 5. In [18] a
presented a new residue-based diagnostic approach for production plants that uses machine learning methods to create the diagnostic model. In this work
the approach used and extended for modular production systems. A first approach, modeling for querying Key Performance Indicators (KPIs) and the
Diagnosis as well as an exemplary implementation can be found in [19]. A similar one The approach is presented in [20], with the focus of this publication being on the integration of machine learning algorithms. In this work, a further developed
approach used; the basic structure is nevertheless very similar.

Two publications were used for the evaluation topics in Chapter 6.
On the one hand, in [21] the connection between Web Ontology Language (OWL), OPC UA and Automation Markup Language (AML) presented and the possibilities to use OWL in display other formats. However, the paper covers the topic even further than is done in this work. On the other hand, a method is presented in [22]
how the results from machine learning processes are converted into a semantic model can become and thus acquire a meaning. This basis is used for creating
a runtime environment and evaluation of the approach.

The six publications not used in this work can be divided into the subject areas of *architecture* and *diagnosis .* For the themes of architecture
in [23] carried out a comparison of different architectures in automation technology and identified a gap for a cognitive architecture. In the works [24], [25]
and [26] each presents solution approaches or architectures for monitoring or diagnosing technical systems. The focus in [24] is on one right
special cognitive architecture to reduce the effort required to adapt after a conversion modular production system. In [25] and [26] the focus is on one
generic approach applicable to various cyber-physical use cases
Systems (CPS) is suitable. Suitability is determined by an evaluation at various use cases. In the field of diagnostics, [27] presented a method for knowledge-based condition detection using wind turbines as an example. There is
So there are definitely overlaps with this work, but no parts of the publication were used directly. In [28] a method for error identification in
Production systems based on machine learning methods are presented. This is for

Table 1.1: Author's publications and their use in this work.

Usage	**Conference**	**Pages**	**Publisher**	**Year**	**reference**
Ch. 1 ETFA [8]	4		IEEE	2019	
	Andreas Bunte, *Paul Wunderlich, Natalia Moriz, Peng Li, Andre Mankowski, Antje Rogalla and Oliver Niggemann. Why Symbolic AI is a key Technology for Self-Adaptation in the Context of CPPS.*				
Cape. 5 AAAI	AAAI Press [18]	9		2019	
	Andreas Bunte, *Benno Stein and Oliver Niggemann. Model-Based Diagnosis for Cyber-Physical Production Systems Based on Machine Learning and Residual-Based Diagnosis Models.*				
	DX online	8th	***Andreas Bunte,*** *Alexander*	2016	[19]
	Diedrich and Oliver Niggemann. semantics Enable Standardized User Interfaces for Diagnosis in Modular Production systems.				
	ETFA	8th	IEEE	2016	[20]
	Andreas Bunte, *Alexander Diedrich and Oliver Niggemann. integrating Semantics for Diagnosis of Manufacturing Systems.* [Best Paper Award]				
Cape. 6 ETFA		8th	IEEE	2018	[21]
	Andreas Bunte, *Oliver Niggemann and Benno Stein. Integrating OWL Ontologies for Smart Services into AutomationML and OPC UA.*				
	ICCART 8		***Andreas*** SCITEPRESS	2018	[22]
	Bunte, *Peng Li and Oliver Niggemann. Mapping data sets to Concepts Using Machine Learning and a Knowledge Based Approach.*				
-	ETFA [23]	8th		IEEE	2019
	Andreas Bunte, *Andreas Fischbach, Jan Strohschein, Thomas Bartz Beielstein, Heide Faeskorn-Woyke and Oliver Niggemann. Evaluation of Cognitive Architectures for Cyber-Physical Production Systems.*				
-	ML4CPS 10 jumpers		*Alexander*	2015	[24]
	Diedrich, ***Andreas Bunte,*** *Alexander Maier and Oliver Niggemann. Cognitive architecture for concept learning in technical systems.*				
-	Book 20	*Oliver*	jumper	2017	[25]
	Niggemann, Gautam Biswas, John S. Kinnebrew, Hamed Khorasgani, Sören Volgmann and ***Andreas Bunte.*** *Data analysis in the intelligent factory.*				
-	DX	8th	on-line	2015	[26]
	Oliver Niggemann, Gautam Biswas, John S. Kinnebrew, Hamed Khorasgani, Sören Volgmann and ***Andreas Bunte.*** *Data-driven monitoring of Cyber-Physical Systems Leveraging on Big Data and the Internet of Things for diagnosis and control.*				
-	ML4CPS	jumpers [27]		2017	
	Andreas Bunte, *Peng Li and Oliver Niggemann. Learned abstraction: Knowledge Based Concept Learning for Cyber Physical Systems.*				
-	DX 8 online		*Oliver Niggemann, Stefan*	2014	[28]
	Windmann, Sören Volgmann, ***Andreas Bunte*** *and Benno Stein. Using Learned Models for the Root Cause Analysis of Cyber-Physical Production Systems.*				

each component creates a power profile, creating virtual power meters. Thus, in the case of an abnormal performance profile, the
faulty component to be closed.

1.4 Outline

The work is structured as follows: In Chapter 2, all the important terms are explained explained and defined. The use cases considered in this work are presented in Chapter 3, from which the requirements for the solution are derived. In
Chapter 4 possible relevant standards from the automation technology and the Semantic Web presented. Also, related work and how to proceed
presented for modelling. The selection of an appropriate formalism (FF 1) that
A semantic model is created for the three use cases (FF 2) and the general validity of the model is analyzed (FF 3) in Chapter 5. For the evaluation
in Chapter 6, the concrete implementation of the solution is first described, followed by a description of the demonstrator and the evaluation for each of the three
Use cases. A critical discussion follows in Chapter 7, in the potentials, however
in particular, gaps are also identified. Chapter 8 rounds off the work with a summary and an outlook.

2 terms

Terms relevant to the work are defined in this chapter, since only a common agreement on the use of the words can ensure a correct exchange of information allows. What this chapter does for this work on a linguistic level is to be transferred to technical systems, in particular production plants, in this work: The uniform use of terms facilitates the exchange of information enabled and the desired interoperability achieved.

For this purpose, the relevant terms from the field of knowledge modeling are defined in the first subchapter. Since models play an important role in this work, these are dealt with in the second subchapter. Different types of models are used considered: Some of the types are used in this work, the others are used for clear demarcation.

2.1 Vocabulary, Syntax, Semantics and Knowledge

In modular production plants, a standardized exchange of information is important, because modules from different manufacturers and different ages should work together, so they have to be interoperable. This means that they should be able to exchange information across modules in order to create the basis for higher-level functions [13]. These higher-level functions, such as B. configuration or diagnostics should can then be provided automatically. Nowadays, data is exchanged between the modules, which is sent to the proprietary signals with a great deal of manual effort need to be adjusted. Specifically, the interpretation of the data in the receiver must be be carried out according to the specifications of the broadcaster [13]. For example, if the systems B. with a new value are to be expanded, both transmitter and receiver are to be adapted to to establish compatibility.

2.1.1 Symbol and Vocabulary

A *symbol* is a recognizable element used as a representative can. Symbols can take many forms, such as B. words, letters, Numbers, images or sounds. But all forms have in common that they can be transferred, ie they can e.g. B. represented and perceived by image or language are therefore recognizable [29]. This makes it possible to place symbols between a Exchange transmitter and receiver. In theoretical computer science one would call a *symbol* a *word .*

Definition 1 *A **symbol** s is a recognizable element, S is the set of all symbols.*

A specific set of symbols is called a *vocabulary .* In theoretical computer science, the term resembles a formal language.

Definition 2 *The **vocabulary** V ÿ S is a set of symbols.*

2.1.2 Syntax, Sentence and Data

The *syntax* determines how the symbols of the vocabulary may be put together.
This is done using a set of rules that specify allowable symbol sequences.
This makes it relatively easy for a machine to check the syntax, since only the conformity with the rules has to be checked. In the case of formal languages in particular, this works well due to the usually simple syntax, but the semantics can only be checked to a very limited extent or not at all. The term syntax is defined as follows based on [30].

Definition 3 *A **syntax** R describes permissible combinations of symbols of a vocabulary V via a set of rules.*

A composite sequence of symbols that is valid according to the syntax is called a *sentence .*
This means that the syntax describes the formal structure of a sentence using rules [30].
However, this does not give the sentence any meaning, and correct syntax does not mean that the sentence makes sense. The meaning of the sentence only comes from the semantics.
The conception of a proposition is similar to a formula in logic.

Definition 4 *A **sentence*** z is a tuple z = (s1, ..., *sn) n* ÿ N *consisting of symbols of a vocabulary , which was formed according to the syntax R. ZR is the set of all sentences over a vocabulary V that can be formed using the syntax R.*

Data are one or more sentences, ie they are symbols arranged according to syntax rules.
Data also have no further meaning [31, 32], ie the composition of the data appears arbitrary.
The following data was selected as an example of a technical process.

0 0 1 B 5 3 8 8 1 4 F 9 3 C 0 7 5 4 2 2 B 2 9 2 0 8 0 0 4 5 0 0 0 0 5 4 1 0 0 4 0 0 0 0 4 0 0 1 .. .

With a technical background, it can be guessed that these are characters from the hexadecimal system. Whether it is a number, the ACSII code or something completely different cannot be recognized. Here it becomes clear that the data without (recognizable) meaning are. Incidentally, this is an Ethernet frame of a ping request to the Heise server.

Definition 5 ***Data*** *d is a tuple consisting of sets d* = (z1, ..., *zn) with n* ÿ *N.*

In the technical environment one can assume that data is available, ie that the symbols are composed according to a syntax. The syntax is usually defined by underlying layers, e.g. B. a network protocol specified. This communication perspective is not considered further in this work.

2.1.3 Concept and Term

The term *concept* is often used in knowledge modeling because people think in terms of concepts. They enable us humans to depict the continuously changing world in discrete categories, even with originals that have never been seen in this form before. People are very good at learning the concepts, that is, in particular, setting the boundaries of the concepts. In addition, not all originals have to be real, even abstract concepts such as e.g. B. alive or love, can be formed. [33]

In the DIN 3242 standard, a concept is defined as “a unit of thought that consists of a set of objects [originals; note. i.e. Author] with identification of these objects [originals; note. i.e. Author] common properties is formed by means of abstraction" [34]. However, the concepts can differ due to the social and cultural background [34], so they are not universally valid. On closer inspection, there are two ways of looking at concepts, extension and intension. The above description refers to the intension, which is also called the meaning content. It defines necessary characteristics/attributes of a concept [34]. The intention is more important for us, since it is able to assign new and unknown originals to a concept. The concept of a car could be defined as an original with the attributes of *having a motor, steering wheel, four wheels,* etc. However, in the future there may be autonomous cars that no longer have a steering wheel, so the definitions of concepts will change over time can change.

In contrast, the extension is also referred to as the scope of meaning and includes all originals of the concept [34]. The extension *university* would include all universities worldwide. This means that the concept does not have to be defined explicitly, ie by specific attributes. From a technical point of view, this has the advantage that some concepts can be easily described without the fact having to be expressed in attributes, e.g. B. with the data type *Boolean* which is defined as Boolean = {true, false}.
However, no new originals can be assigned to these concepts. This principle is known in programming as enumeration. In this work, a *concept* based on [34] is defined as follows.

Definition 6 *A **concept** defines a set of originals, which is formed by means of abstraction, taking into account common features of the originals.*

The *term* and *concept* are closely linked, but strictly speaking they do not have the same meaning1 . A *term* is a concept that has a label, while a

1In English-language literature, it should be noted that the distinction between term and concept is not present, there both words are called *concept* [35, 36].

concept can also exist without an identifier [35, 36]. Under an identifier, a
understood the symbol associated with the concept, which represents the concept. Around
To rule out misunderstandings, a term should have a clear designation. There
one can only talk about concepts that have an identifier, as in Section 2.1.4
is presented, the more specific designation *term* is used in this work.

Definition 7 *A **term** is a concept that has a unique designation.*

2.1.4 Semantics

The word *semantics* comes from the Greek and means “belonging to the sign” [30]. The term is used very differently in different specialist areas [30]. First, the original linguistic perspective is presented here,
which is often used in general language context. The semantics are also
referred to as the theory of meaning and deals with the meaning of symbols, words and phrases [30]. Meaning is important when information is exchanged
should be used, because correct further processing is only possible if the meaning of the information can be correctly interpreted. For a simple case that
If only one original is to be referenced, relevant connections are shown in the so-called semiotic triangle, see Figure 2.1. In addition to the *original* (in the literature
also referred to as a thing) there is an associated *term* and *symbol.* Under *Original* is everything about which one can communicate, e.g. B. objects,
ideas, feelings or fantasies. The *original* has an assignment to a *concept.* Of the *Term ,* in turn, can be represented by an assigned *symbol,* the identifier. A *symbol* can therefore be assigned to an *original* via the *term .* This
Assignment between *original* and *symbol* can be understood as semantics and is shown dashed in Figure 2.1. For a more detailed discussion, see [37]
and [38] referenced.

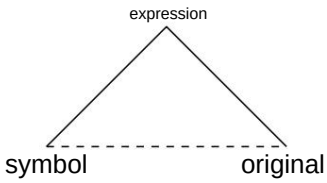

Figure 2.1: Semiotic triangle, represents the relationship between original, concept and icon. [39]

In logic, the semantics are formally defined. This happens through the so-called interpretation or interpretation function, which gives a symbol a truth value *(true* or *wrong)* assigned [40, 41]. The semantics define rules that describe how concatenated symbols are processed so that they too are assigned a logical value
can [42]. This makes it possible to draw logical conclusions. However, these conclusions only apply in a model world, ie only for the current truth values
of symbols. This makes it possible to map a connection between the real world (of an original) in a model world of logic and to draw logical conclusions that are also valid

are valid in the real world. This relationship is shown in Figure 2.2.
The real world is represented by modeling and can be represented by a suitable
interpretation function are processed. The following conclusion also applies to this
Original.

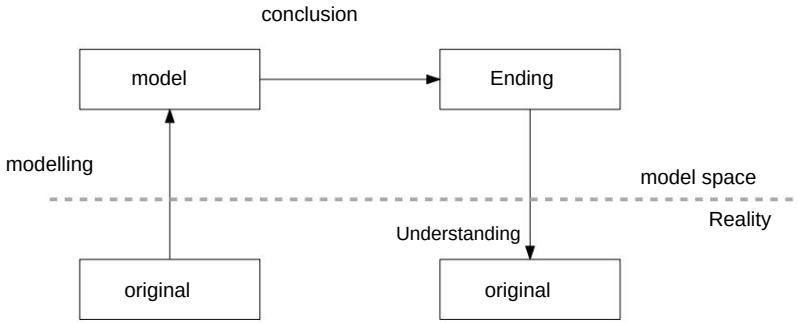

Figure 2.2: Logical models enable conclusions and new insights can be gained about the original. Representation based on [42].

An important difference between the two presented semantics is that the linguistic semantics refers to the original, i.e. to the real world. The semantics of
Logic, on the other hand, is only limited to a model world. A usually desired reference to
the real world must thereby be established by choosing the right model and
is therefore not part of the semantics.

It is sometimes claimed that the *semantics* are brought into machines through modeling techniques. But that is wrong, because the semantics cannot be explicitly stated
represent. As soon as one tries to represent them explicitly, the semantics change to
Syntax that follows certain rules but no longer has any meaning [43]. If you like it
When we say that we represent the semantics, this often means that in the explicit representation shown, capabilities are defined by rules, such as e.g. B. implicit
Developing knowledge or recognizing contradictions [44]. These skills indicate
Understanding and thus to the existence of semantics, although it is
is actually a matter of syntactic processing.

In this work, the semantics are based on the usage in logic as follows
Are defined:

Definition 8 *A **semantics** orders every sentence z* ÿ via an interpretation function I
ZR in a model world a value.

2.1.5 Language and Information

According to the German dictionary, language is, among other things, “a (historically created and developing) system of signs and rules that serves a language community as a means of communication; language system b. System of signs (that of communication or similar serves)” [45]. On the other hand, in theoretical computer science, one understands a language as one

Subset of an alphabet [46, 47]. This is similar to the concept of *vocabulary defined here.*

This work is based on the language term used in the Industry 4.0 platform and which is similar to the general or linguistic understanding [13]. the VDI/VDE Guideline 2193 [48] includes a vocabulary, a syntax and an interaction protocol for *language .* The latter is used to plan the course of a communication. There the aim of this work is the exchange of information and not primarily the interaction between the partners, the interaction protocol can be dispensed with, especially since this is author's point of view, is language-independent. Instead, the semantics are part of the language, for language can only serve as a means of understanding if it is a unified understanding is based.

Definition 9 *Every* ***language*** *L = (V, R,* I) *is defined by a vocabulary V tax R and ， a Syn an interpretation function I of the semantics.*

The larger the vocabulary, the more specifically information can be articulated. This can also be understood as the modeling depth of the language. For languages with different modeling depth, it is not possible to add a suitable word to every word in another language [35], so that the semantics must always be taken into account when translating into another language. In technical systems stands the advantage the specific articulation through a large vocabulary the implementation effort contrary, so that a suitable measure must be found. The need is often from Use case dependent. So would e.g. For example, the vocabulary for energy optimization is more specific than when it is about the interaction between the system and the user in general go.

By giving data a context, information is created, which is also known as facts be designated. *Information* can be processed by a human through the context . The date 27.3 becomes the information *Sensor_Reactor = 27.3°C.* This information can be evaluated and processed by humans. The corresponding context is often implicit in people. In contrast to this, it usually has to be specified explicitly for machines. This can e.g. B. through an information model done or, as with the Ethernet frame, by the associated standard.

The understanding of information is broadly similar in the literature; differences exists above all in the demarcation to knowledge. According to Rowley, information is "contained in descriptions, answers to questions that begin with such words as who, what, when and how many. [...] Information is inferred from data” [49]. Very similar in content the term is defined by [31].2 Other definitions place the context more in the Focus: "Information is data that have been given meaning by way of context" [32]. for this work, the above definitions are somewhat abstracted.

Definition 10 ***Information*** *is data that has a semantic section represent the model world.*

[2] "Information is data that has been given meaning which allows to answer questions like who, what, where, and when. Eg in computer science data stored in a (relational) data base is given meaning by naming a row (attribute) and assigning them to a (named) entity." [31]

2.1.6 Knowledge

Science has not yet succeeded in finding a uniform definition of *knowledge* that covers all facets [50]. First, the term *knowledge* is considered from the perspective of information processing: "Knowledge is information that is connected by some relations" [31]. This definition illustrates how knowledge can be represented, also in the technical environment, but is a restricted view that does not do justice to the complexity of the term [51]. Nevertheless, it reflects important aspects for our applications, as illustrated in the following example. This is based on a knowledge base consisting of two statements. *(i)* If the reactor temperature is below 35°C, the process is disturbed; *(ii)* if the reactor temperature is disturbed, the product can no longer be utilized. Linking *this with the information 'The sensor on the reactor measured 27.3°C'* creates an overall picture. This enables information to be presented that is not explicitly available, ie the information is inferred. It can thus be concluded that the process is disrupted and the product can no longer be recycled. This example illustrates that information alone is of little use, the benefit only arises when it is linked to other information; this is how knowledge is created.

Other definitions focus on added value and decision-making and actions or capabilities derived from it, see Rowley3 [49]. In this way, knowledge does not become an end in itself, but serves a specific goal, in particular the acquisition of new information. The following definition is derived from these two thoughts.

Definition 11 ***Knowledge*** *consists of networked information that makes it possible to conclude new information in a model world.*

This definition reflects the understanding of knowledge in this work, with the awareness that this definition in particular is imperfect. Knowledge has many dimensions that are difficult to bring together. Figure 2.3 shows six dimensions, based in particular on [52]. This figure does not claim to be complete either, but serves to illustrate the complexity of the term.

Knowledge can be assigned a value in each dimension, although the boundaries are often not sharp. In the following, only the dimension for which work is most important is considered: implicit and explicit knowledge.

Explicit *knowledge* is knowledge that can be communicated, ie it is factual knowledge [53]. Man is aware of this knowledge and the knowledge can be cataloged [51]. This means that knowledge can be retrieved in a targeted manner. The simple exchange and use of explicit knowledge makes it well suited for use in technical systems.

[3] "Knowledge is the combination of data and information, to which is added expert opinion, skills, and experience, to result in a valuable asset which can be used to aid decision making [37, p. 223, quoting the European Framework for Knowledge Management]'" [49]

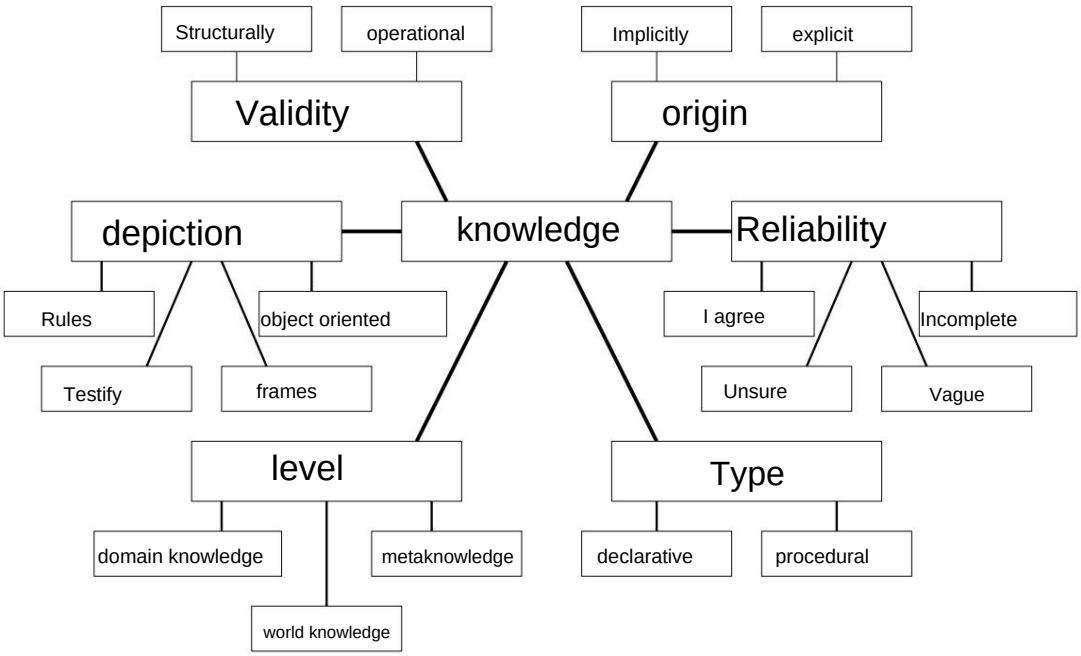

Figure 2.3: Six dimensions of knowledge according to [52]

In contrast, *implicit knowledge* is behavioral knowledge that lies in the subconscious. So
For example, a person who has a good grasp of situations can handle conflict through early
Avoid giving in or another person is good at telling jokes. This form of
Knowledge cannot be formalized in rules or the like, or only to a limited extent [54]. Skills
are part of the implicit knowledge [53], since e.g. B. a problem-solving competence id R. through
the procedure and not through factual knowledge. With a focus on technical
Applications are often interested in these capabilities, e.g. B. the assembly of a product. This form
of implicit knowledge can be formalized and thus made explicit [54]. An expert may explicitly have
the ability to assemble the product
do, e.g. B. in the form of instructions. This often requires several iterations,
until the explicit knowledge is correctly represented.

Tacit knowledge is also omnipresent in technical systems. B. each
Tacit knowledge program. This has the disadvantage that this knowledge is not readily available
can be transferred to another system. However, it is possible to set up a system in which as much
information as possible is explicitly available in a knowledge base.
By passing on the knowledge base or changing the knowledge, with relative
little effort the system behavior can be adjusted. However, the knowledge base must conform to
the processing system. This type of systems is called
knowledge-based systems. [55]

2.2 Model Types

In this section, the general concept of the model is first explained. Below
Various types of models are presented, namely metamodel, conceptual model, semantic model,
information model and ontology.

2.2.1 Model

First, the term *model* is explained in more detail,4 then different model types are distinguished. The roots of the word *model* go to the Latin term *modulus ,* which is the diminutive form of *modus .* Originally he stood for measure, Scale and standard, but also used for manner, form and regulation [57].
You can see how broad the term is. Although models have been used for thousands of years, there has long been no precise definition of the term. The roots of the general model theory published in 1973 date back to back in 1957 [57]. According to general model theory, a model has three characteristics Features [57]:

Representation **feature:** Models are always representations or representations of originals. The originals themselves can be natural, artificial, or models themselves.

Foreshortening feature: Models generally do not capture all attributes of the Originals, but only those that appear relevant to the intention of the model. This also means that a model is created for a specific purpose.

Pragmatic feature: Models cannot always be clearly assigned to their originals. The correct representation can be determined by the time on operations or restricted to specific model users.

A model is therefore a representation of an original of some kind. implicitly begets the foreshortening characteristic of an abstraction that every model entails. Therefore, the model must fit its intended purpose, so that all Attributes of the original relevant to the intended use are represented.5 The pragmatic According to [57], characteristic means that a model is created for a specific model user will. However, the distinction between different model users is not relevant if the following assumptions are made: *(i)* the model is general recognized semantics; *(ii)* the modeling formalism is generally interpretable; *(iii)* all model users have the same skills. These assumptions can can be made for the technical systems under consideration, so that a generic model can in principle be created for these systems. Based on [58], the model defined as follows.

Definition 12 *A **model** is an abstract representation of an original with the aim of presenting relevant aspects for a specific purpose under specific conditions.*

The definition of the term *model* is so general that a further subdivision or classification of the models makes sense. In the literature, models with the same Properties are combined into model types, to which modeling formalisms and

[4] In logic, another model concept has been established. There is an assignment *A* as a model for the Formula *F* denotes if *A* |= F. [56]

5 This assumes that the model creator is aware of all the properties of the original and theirs objectively assess relevance.

guidelines can be assigned. Which models can be used for which applications is independent of this and is not explained in more detail here. the in
The following presented subdivision of the models is not complete, but on the for limited to the model terms relevant to this work.

2.2.2 Metamodel

The *metamodel* is closely related to the *model .* The *meta* prefix comes from the Greek and means that something is "on a higher level, level or
is classified above" [45]. From this derives a frequently used definition of *metamodel ,* which describes it as a model of a model. However, this definition is very generic, so almost any model can be called a metamodel. Is it at all possible to model an original, or is a model of the original modeled,
which one the model builder has in mind? This philosophical question aims in this work remain unanswered, instead the definition of the meta model is specified in more detail. Hesse defines a metamodel as follows: “A model C then becomes a metamodel called if its original itself is a model - here called B - with an associated Original A is and if both modeling relationships are of type modeling type are” [43]. This definition requires a more abstract description of the meta-model of the model or the modeling elements. This is exactly what the intention of a meta model reflects, it should be a kind of guide for the use of modeling elements represent. [59]

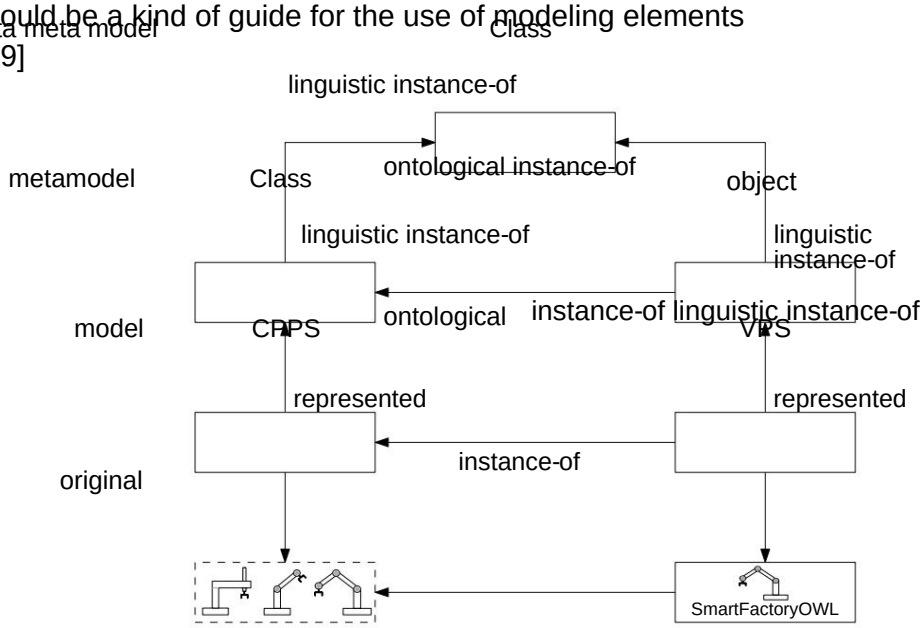

Figure 2.4: Difference between linguistic and ontological metamodel according to [60] and [61]

However, metamodels can be further subdivided into linguistic and ontological metamodels [60]. Figure 2.4 shows an exemplary modeling of originals. Four levels extend vertically (original, model, meta model
and meta-metamodel). The original on the right is a concrete physical one SmartFactoryOWL cyber-physical production system (CPPS) that models will. On the left of the original plane is the concept of CPPSs. the concrete

CPPS on the right is an *instance of* the real CPPS concept, since there is a concrete implementation of the concept. A model is now created from these originals, which is intended to represent the originals. Therefore, between originals and whose model-level counterparts represent the relationship .

Each of the vertical model levels defines a language for the underlying model [60, 61], ie the use of the modeling elements is specified. this will referred to as a linguistic metamodel [60]. A well-known use of linguistic meta-models is the description of the Unified Modeling Language (UML), which is defined by the Meta Object Facility (MOF) is described [62]. A feature of the linguistic metamodel is that the level is changed by instantiation [60] and the modeling takes place on a different level with a different formalism. After [63] Linguistic metamodels are aimed primarily at modeling tool manufacturers, e.g. B. um to implement the formalism correctly.

In contrast, an ontological metamodel moves only within the model domain, i.e. if there is an *instance-of relationship* within a modeling formalism , this represents an ontological metamodel [60]. With such a meta model terms of a domain are defined [60]. In Figure 2.4, *CPPS* is a term defined in this way. The versatile production system System (VPS)) uses the term through an instantiation within a model level. The terms thus defined are only valid in a limited domain and not, like the language of the linguistic metamodel, in general [61], so that this modeling is aimed at domain experts [63]. A detailed discussion on metamodels can be found can be found in [61], [64] and [65].

The following definitions can now be derived from the knowledge gained above.

Definition 13 *A **linguistic metamodel** describes the modeling elements and their use in a model.*

Definition 14 *An **ontological metamodel** describes types that are instantiated by the model and are created in the same modeling language. That means at the Instantiation will not leave the model stage.*

In normal language use, there is often no distinction between these two types of meta models differentiated, due to the higher importance, the meta-model often means the linguistic meta-model [66]. This work is oriented towards the terminology in the literature, so that it is generally referred to as a meta model. An explicit designation of the type is only used if the distinction is not immediately obvious.

In addition to the meta model, a meta-meta model can also be defined, i.e. the model of the meta model or the meta model of the model. In this way, any number of meta levels can be introduced. In practice, however, one rarely finds more than four linguistic meta-levels. For example, the UML is also defined by four levels. [67]

2.2.3 Conceptual Model and Semantic Model

Conceptual models have their origin in the modeling of databases. Peter Chen first succeeded in 1976 with the entity-relationship approach (ER approach) to create an implementation-independent formalism for data modeling, which then became known as the conceptual model [43, 68]. The aim of the model is uniform description of data [68]. In general, a conceptual model should represent a solution-independent description of a problem in a domain with which The aim is for all those involved to have a common understanding of the problem [69]. The identification and description of relevant terms is called conceptualization [70]. The conceptualization is not universal, but requires agreement on one certain conceptualization (ontological commitment) [70]. This is necessary because usually many conceptualizations are possible and the modeler chooses one based on the goal must select suitable ones [71, 72]. This chapter *terms* has the aim of establishing a convention between author and reader about a conceptualization for this work. A specific conceptualization is called a *conceptual model* [71]. Kidawara et al call a conceptual model *an object-relationship representation of term definitions* [70]. The following definition is derived from this.

Definition 15 *A **conceptual model** describes terms in a domain using objects and relationships with the aim of achieving a uniform understanding of the terms.*

The term *semantic model* is not used consistently in the literature. According to [43], conceptual models are referred to as semantic models if they explicitly depict many inner relationships and dependencies. However, this definition leaves a lot of room for interpretation, so that there is no clear distinction between the two terms is possible. Tiurirmin and Massel describe semantic models in the energy sector as cognitive, event-based and probabilistic models that are based on Bayesian belief networks and are intended to represent different situations [73]. This shows how broadly the term is used; however, this definition does not reflect the understanding of semantic models in this work. Describe after Taheriyan et al semantic models terms and relations of a domain and can be used as graphs, with ontology classes as nodes and ontology relations as connections between the nodes [74]. After that, the validity is limited to the domain and the description is given via ontologies. The given description of ontologies unnecessarily restricts the definition, but ontologies allow logical draw conclusions and thus do justice to the term *semantics .*

Due to the heterogeneous use in the literature and the often not clearly defined definitions, the semantic model is defined as follows:

Definition 16 *A **semantic model** is a conceptual model that in a Formalism is created that allows conclusions to be drawn.*

2.2.4 Information Model

The term *information* model is often used imprecisely in the literature, and there is no uniform definition. The following is a brief overview of various definitions. "An information model is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse" [75]. This very general definition differs little from the conceptual model. Kolberg [76] becomes more specific by introducing two new aspects, a technology independence and the purpose of the information model, namely the integration into given structures and processes. "An information model is a technology-independent description of functions, relationships and attributes of entity types in order to realize use cases. The purpose of the information model is to support the integration of ICT [information and communication technology] into specified structures and processes" [76]. Selig [77] further constrains the information model by requiring that it be use case specific. "In an information model, the sum of all information relevant to the application of the objects and facts under consideration is summarized" [77]. Similarly, the information model in OPC UA is defined as "an organizational framework that defines, characterizes and relates information resources of a given system or set of systems" [11].

The model thus describes the data of a data-generating system so that it is given a context and becomes information.

In this work, the definition of the OPC UA *information* model is the guiding principle. This makes particular sense because OPC UA is becoming more widespread in production technology and OPC UA is defined in a standard. The definition shows that there is an overlap with the conceptual models and no sharp distinction is possible. However, the focus of the information models is on the description of concrete data of the use case or certain systems, while the conceptual model aims at a holistic view of the domain.

The information model can thus be viewed as a mapping component that can establish a connection between the system and the terms of the semantic model. The information model is defined as follows.

Definition 17 *An **information model** is an organizational framework that defines, characterizes, and maps information sources of one or more given systems.*

2.2.5 Ontology

In the knowledge modeling literature, one often comes across the notion of *ontology.* The term ontology, which translates as *the theory of beings,* originally comes from philosophy and attempts to describe the basic structures of reality in a concrete and universal way [78]. In the field of computer science, the term is not uniformly defined, as no suitable definition has yet been found [79]. The challenge is

that general definitions are imprecise and specific definitions exclude certain ontologies [79]. There is broad agreement that an ontology describes concepts and relationships in a segment of the world [80, 81, 82], ie it describes them
a concrete conceptualization [71]. To represent ontologies, logical languages can be used, e.g. B. to predicate logic or description logic
are based [82, 71]. As a result, logical conclusions can be drawn automatically and the Descriptions can be checked for consistency [82].

Ontologies are therefore very flexible and can, depending on the formalism and modelling represent meta-models, conceptual models, semantic models or information models. If one of the models defined above is meant in general, this
Work that uses appropriate more specific terms. The concept of ontology becomes on the other hand, used when a general formalism for the creation of corresponding models is meant. For this work, the ontology is defined as follows.

Definition 18 *An **ontology** is a logic-based description of concepts and relations in a domain.*

3 requirements

In this chapter, the requirements for semantic models and their necessary peripherals for use in modular production systems are determined. become in the first subchapter three typical use cases are presented and described, like this one can be simplified by using semantic models. These use cases will later be used to evaluate the approach. In the second subchapter the requirements are derived from the use cases.

3.1 Use Cases

In the three use cases described , *interaction, orchestration* and *diagnosis* complexity of the semantic modeling increases. As a result, on the one hand, Possible applications are presented as comprehensively as possible and on the other hand The benefits of the approach presented here must be easily comprehensible. The benefit is both at comparatively simple as well as complex applications.

3.1.1 User Interaction

The interaction between man and machine will continue to play an important role in the future play [7]. There is even talk of *human-centric production* [3]. This means that the user should be optimally supported by the system in the future and always all relevant information is displayed using appropriate visualization techniques [3, 7].

Static user interfaces that display information once defined are dominant today. To do this, certain values from the controller are connected to the user interface during the engineering phase. The values are either in the controller before or the calculation rules for combined values are hard-coded. There the systems are not or only rarely converted, making the interface more flexible is not necessary and uneconomical.

This changes when the production system is modular and the production facility is regularly rebuilt. Then the user interface must automatically adapt to a new system configuration, since manual adjustment is too time-consuming and would be error-prone. With the regular reconfigurations, e.g. T. the Calculations of the values to be displayed. For example, the total energy consumption the sum of all individual energy consumptions and must also be defined in this way, since the

Number of modules can change. A static determination of the values does not meet the requirements of modular production systems.

In this application, the values for total power consumption, total energy consumption, total production for the day, throughput per hour and overall equipment effectiveness (OEE) should be displayed. The considered values show an increasing complexity
on. The power consumption is an addition of several values. The time must be taken into account for the total energy consumption, as energy consumption is always related to a period of time. The daily energy consumption is used here as an example. For the
Daily production requires a meter, and to determine throughput, in
be counted at a time interval. Complex calculations are necessary for the OEE,
the z. T. based on the previously determined values. There are any number of other values
which can be defined and can also be company-specific. However, the essential modeling patterns are contained in the above values, so that with these
You can also define your own values for patterns.

For the realization of a user interface with a dynamic determination of the values
a semantic model is used. Values can be assigned to a term in this way, e.g. B. power consumption, reactor temperature, ok part or status. display elements
of the user interface are linked to these terms, so that the values associated with the term are displayed. Through this assignment, the semantic model,
which is the same for all configurations, provides a static interface for the user interface. The dynamic is intercepted by the semantic model by
that values that require calculation and are not determined directly in the controller are defined in the semantic model. So the values can be calculated and to
access the user interface. The overall complexity of the system is not reduced by this approach, but the dynamic part is exclusively within the
production system. The dynamics can be handled independently by appropriate models, so that there is no dynamic within the user interface
and therefore no adjustments are necessary.

This use case is implemented and evaluated on the VPS in the SmartFactoryOWL. The VPS consists of four modules: *delivery, storage, dosing* and *production.* It processes corn from which popcorn is made in the production step. the
Production plant is typically operated with all modules, but can be adjusted depending on requirements. if e.g. B. a continuous delivery takes place, can
the *storage* module can be dispensed with. Due to its versatility, it is well suited to the
represent the variability necessary for this application.

3.1.2 Orchestration

An advantage of modular production systems is that they are flexible to events such as
e.g. B. a failure of a module, can react [83]. An automatic process planning
adapts the production process in such a way that the remaining modules continue to
can be used effectively [84]. For process planning, it must be known which
Modules can be combined for a specific product. A static one

Plan does not make sense due to the possible variability of the available modules and the many boundary conditions, especially since the boundary conditions z. T. not in advance are known because they depend, for example, on the product. Instead, the mechanical Interfaces of the modules are described clearly and uniformly with all relevant boundary conditions. Process planning can use an algorithm to check whether two modules are compatible and use this information to create an optimal plan to generate. The check for the orchestration, i.e. the compilation of modules, takes place dynamically. If two modules are compatible, the configuration phase follows, in which production parameters are defined. This phase is not further considered here, as it relates to the independent research area of production planning.

The core of this use case is the manufacturer-independent generic description of the mechanical module interfaces through a semantic model. The interface description must show all relevant boundary conditions. Some of the interfaces are predetermined by the design of the modules and cannot be changed, e.g. B. the mechanical interface or permissible processable materials. In order to two modules can be connected together, these *constructive interface parameters* must be compatible. Situational *interface* parameters summarize influencing variables that can be changed, i.e. that do not rule out compatibility in principle, e.g. B. the orientation of the product. Additionally there is Parameters that can be influenced by the controller, such as B. the throughput or batch sizes. Since these parameters affect performance or efficiency, they will called *performance-related interface* parameters in this work . An algorithm uses the description of the semantic model to check the compatibility of two modules. The constructive parameters are checked for compatibility and the Specified range of performance parameters. In addition to the interface description, e.g. T. also product knowledge, there the product will change as a result of the production processes. So e.g. a product easier when drilling holes in the product and when joining two products the weight and dimensions change. The product knowledge is usually available in the process planning and is made available from there.

The implementation of this use case in this work is also carried out on the VPS SmartFactoryOWL. For this purpose, the modularity required for this application plant used. The interfaces of each module are described so that inquiries regarding the compatibility of two modules can be answered. On this basis one can Production plan and a plant topology are created by the process planning.

3.1.3 Diagnosis

Today's production systems often have an integrated diagnostic system that detects errors and determines possible causes. Errors are caused by deviations of the expected behavior, so-called symptoms. Possible causes of error are shown as diagnoses referred to. The diagnostic systems are often based on heuristics that are based on of the effort involved in adapting to new system configurations are not suitable for modular adaptive systems [18]. Besides the challenge that different modules

have different diagnostic systems, the dependencies within a module can change due to the adaptation of the controller. On the other hand can
symptoms spread across modules [85], ie there is a cause of the error
not necessarily in the module in which the error is detected. That is, even if
each module contains a diagnostic system, errors cannot be diagnosed across modules [86]. This increases the complexity, diagnoses in modular systems
to investigate.

In this application, a cross-module diagnostic system is implemented. Therefore
it is assumed that each module has its own diagnostic system, e.g. B. a
model-based system as described in [18]. In order to enable the system for a module-wide diagnosis, the relevant terms of the diagnosis are defined using a
semantic model described. In addition, the model of the module contains a description of the error propagation. The modular descriptions are combined into an overall model. A higher-level diagnosis engine accesses the information
interprets them and, if necessary, starts the diagnostic process. In this way, diagnostics can be determined across modules. In addition, the diagnoses are uniform and
clearly represented by a user interface (use case 1).

The VPS is also used to evaluate this use case. The VPS
includes both discrete and continuous signals, has cause-and-effect relationships across modules, and is modular, making it a good fit. Through
the semantic model should make it possible to use a hybrid consistency-based diagnostic approach that identifies cross-module causes of error, in which after
manual effort is no longer necessary for the reconfiguration.

3.2 Requirements Analysis

Requirements should be as complete, correct, clear and consistent as possible [87]. Around
To achieve this goal, first meet the requirements from those described above
Use cases derived. Then some general requirements are determined,
which cannot be derived from a single use case, but only from the
combination of several. In particular, the non-functional requirements were only
derived across all use cases and are marked accordingly.

3.2.1 Use case requirements

In this section, the requirements resulting from the use cases are presented and explained. An overview of the requirements is given in Table 3.1
Find.

The task of the semantic model is the definition of uniform terms and
relations (requirement 1). This is a key component in all three use cases, as it creates a common understanding and models relationships
can become. The defined elements must be clearly identifiable, because without

Table 3.1: Requirements from use cases - functional requirements

serial no.	request	description
1	Definition of unique identifiable terms and relations in the domain	A common understanding is necessary for an exchange of information. Therefore relevant terms must be defined for the domain and the application and be clearly identifiable.
2	hierarchical Order the Terms (Term hierarchy) and relations (relation hierarchy)	The hierarchy makes it possible to use terms and relations on different levels. So can, even if a term closer was specified, the generic term can be used due to the hierarchical arrangement, so that a context-specific use of the term is possible.
3	Representation and calculation of numeric values and value ranges	Information from the production facilities contain numeric values or ranges of values. These must be in the semantic model can be displayed. Some values are not output directly from the system, but must be calculated.
4	Representation and handling of time references	Some values have a time reference that must be presented and used, e.g. B. Energy consumption or throughput per Hour.
5	Models with meta and instance level	The model must have an ontological metalevel have on which concepts and general knowledge is modeled and an instance level in order to to describe specific objects.
6	Description of the subsymbolic algorithms	Subsymbolic algorithms used must be defined in the semantic model be. In addition, their specific Characteristics (instances) including the parameterization are described.
7	Dynamic access to the data of the model	The models must provide an interface so that the data from the outside can be reached dynamically with read and write access.

8th	access to model	The modules must make their models available so that they can be accessed externally and an image can be created. From the point of view of the requirements it is not relevant whether the module directly or the models by linking to an external data source.
9	Creation of an overall model	From the partial models of the modules, a Overall model can be generated, which is centrally available. The overall model contains all the information of the system and can provide the information required for the applications.
10	Extensibility of the models	The models should be supplemented with additional knowledge, e.g. B. with process knowledge, can be expanded to improve the diagnoses. Included the extension can be generic or customized by user or manufacturer e.g. B. Company key figures to display in define the user interface.

Unambiguous identification can lead to misunderstandings, e.g. B. if a term has a different meaning in another domain. through the hierarchical order of concepts and relations (requirement 2), a simpler use of the model is achieved. Depending on the context, a term can be used that is more specific or is more generic. This is e.g. This is useful, for example, in user interaction, since the user is often shown more general information in a graphical interface the total power consumption is displayed, resulting in the model from different types of services can be put together, e.g. B. from electrical power and compressed air. This allows the model to be used with deeper levels of detail without the This limits compatibility. In the model must have numerical values and value ranges can be displayed and used (requirement 3). For the The modeling formalism must be able to represent numerical values in a suitable form. It must be possible to represent positive and negative decimal numbers. From these values z. T. other values are calculated, such. Am Application 1 the OEE. These are the four basic arithmetic operations and a grouping of partial terms for the preferred calculation (replacement in parentheses). ranges of values have to e.g. B. for the interface description of use case 2, such as the range of permissible product dimensions.

In contrast to classic knowledge modeling, in which static knowledge is modeled production plants are dynamic systems. This means that some values have a have a time reference, e.g. B. throughput per hour or energy consumption, so that this too must be presented and used (requirement 4). The model must have a meta level and an instance level (requirement 5). On the ontological meta-level are defines the terms and relations that are valid across modules. About it

In addition, concrete objects of the module are described at the instance level. concrete
Objects are in the use cases z. B. Values of the production plant, interfaces
of the modules or diagnoses determined by the diagnostic system. For application 3, sub-symbolic algorithms are required that learn the system behavior. Under
sub-symbolically one understands the evaluation of data with static means, while
symbolic methods process discrete symbols, e.g. B. logic-based systems [88]. the
The available sub-symbolic algorithms must be described, and for the instances there are images of the specific algorithms, including the parameterization, on the
Create instance level of the model (requirement 6). The models must provide an interface for data access that allows both read and write access
(Requirement 7). Access is provided by algorithms (use cases 2 and 3) or by the
User interface (use case 1) required. Information from the
Model loaded and e.g. B. the results of the algorithms are written into the model.
The data from the production plant is also included in the model in this way, so that
the module manufacturers must provide a corresponding interface. Not only the
Data of the model must be provided, but each module must also
provide its own model (requirement 8). This can be called the reflectivity of the
interface, since it can use not only the model, but itself
also aware of its existence. Whether this happens directly by accessing the module or whether the module provides a link is irrelevant. the
Models are required in particular to create an overall model of the system,
when a cross-module perspective is required. This is required for use cases 1 and 3, so the partial models of the modules have to be merged
(Requirement 9). If necessary, the model should be enriched with additional knowledge
can (requirement 10) so that the semantic model is not self-contained
may. This knowledge can be used by algorithms to make a task more specific
to solve, e.g. B. Improved diagnoses based on specific knowledge or additional
Conditions when checking for compatibility of modules. Because of the heterogeneity
of interests, processes and companies, it will not be possible to create a model
develop, which meets all requirements. To take this fact into account
the models should also be individually expandable, so that company-specific or plant-specific specifications can be made.

3.2.2 General Requirements

Some requirements cannot be derived from the specific use cases, but require a higher-level perspective. This is especially true for non-functional ones
Requirements. These more general requirements are shown in Table 3.2.

Versioning (requirement 11) is necessary because the semantic model changes with
of time will change. Versioning is a simple means of preventing or identifying incompatibilities. So it may be that an algorithm
requires or excludes a specific version of the semantic model. If necessary, mapping between the versions is usually possible. In today's globalized
In the world, such a system is not only used in one country. Since each term should be clearly identifiable, the names of the terms must be in different languages

Table 3.2: General requirements

Running No.	Requirement Description	
11	versioning *(non-functional)*	The semantic models used will change over time. In order to be able to understand these changes and to ensure compatibility is versioning required.
12	Multilingual terms *(non-functional)*	In order to have uniform terms with a unique ID for all languages each term designations in different languages are assigned.
13	Height reusability *(non-functional)*	The manual adjustment of the semantic model is complex and error-prone. Therefore, the greatest possible reusability must be guaranteed.
14	Consistency check of model	When expanding the model, the Consistency are checked so that there are no logical contradictions and the correct results using the model delivers.
15	reasoning ability for implicit knowledge	Decreased by inferring implicit knowledge in the model the manual modeling effort. That makes for a smaller and clearer model and simplifies the acquisition of knowledge.

be possible (requirement 12), because unique identifiability can only be guaranteed by using different names for the same term. Because the modeling requires manual effort, the highest possible level of reusability should be achieved (requirement 13). When models are merged or expanded, this entails there is always a potential for error, which means that the model can become ambiguous or contain logical errors. It should therefore be possible to check the model for consistency Identify inconsistencies to ensure the logical consistency of the model (Requirement 14). In addition, implicit knowledge in the model should be inferred automatically (requirement 15), ie this knowledge can be based on logical connections are made explicit. This also reduces the modeling effort and new insights can also be gained.

4 State of the Art

This chapter examines the state of the art in more detail. Norms and standards are of great importance in automation, which is why they are considered in the first sub-chapter. Then the ontology language OWL is presented, which plays an important role in this work. In the third sub-chapter, other works with comparable objectives are presented. Finally, the procedure for creating the model is presented so that proven methods can be used.

4.1 Relevant norms and standards in automation

Six norms or standards are presented below that may be relevant to this work: AML is a data exchange format, OPC UA and oneM2M standardize communication, eCl@ss and DIN 8580 are classification systems for products and services or manufacturing processes and the VDI/VDE 3682 is a guideline for formal process description.

4.1.1 AutomationML

The AML is an open data exchange format for engineering data. The idea behind this is to use one data format for the entire engineering process in order to make it more efficient. The aim of AML, as a universal and manufacturer-independent exchange format, is to completely map the data that is generated, to be based on proven formats, to enable extensions and thus to achieve a high level of market penetration. [89, 90].

These goals can also be seen in the architecture of AML. An AML file consists of a Computer Aided Engineering Exchange (CAEX) file, according to the IEC 62424 standard, which maps the system topology [89]. However, topology in AML is only understood as the hierarchical arrangement of system components [90], but relations can also be inserted in the hierarchy. For a more precise specification, further standards are referenced from the CAEX file. The Collaborative Design Activity (COLLADA) standards for representing geometries and kinematics, and PLCOpen Extensible Markup Language (XML) for representing behavior are planned and available from the start. Since AML is based on the XML standard, other standards can be linked if they can be represented in XML. [89]

AML provides four concepts for modeling in the CAEX file: *(i)* There are classes that map templates, are reusable and further specified through inheritance

can become; *(ii)* The topology of a specific system is determined via an entity hierarchy or subsystem described, so AML is object-oriented; *(iii)* interfaces and relations can be described; *(iv)* roles can be defined and assigned,
representing an abstract function [89]. So that planning can be carried out efficiently, AML offers a class library *(SystemUnitClassLib),* a role library
(RoleClassLib) and an interface library *(InterfaceClassLib)* [89]. About attributes a plant is assigned specific values. There are some attributes that are firmly defined in AML that can be supplemented by the user. A fixed semantics
however, do not have the self-defined attributes.

In summary, it can be stated that AML has achieved many of the goals. So AML offers versatile modeling that can be used to describe systems in the engineering process. By being able to reference other standards and
integrating them in this way, AML is open and can be flexibly expanded. However, must the tools used also support the other standards. The market penetration of AML is still far from the target it has set itself. Although the standard in known to the industry and is also seen as future-proof within the framework of I4.0 however, does not result in consistent use of AML.

Since AML is focused on engineering data, i.e. offline data, and cannot map any semantics, the standard is not for the use cases dealt with in this work suitable. For model creation, however, it would be interesting to convert the information available in AML into the semantic model. So would double the work prevented and semantic models could be easily created. Basically that is possible, but will not be considered further in this work.

4.1.2 OPC UA

OPC UA is a platform-independent standard that aims to transmit data and control signals across different levels of automation technology [11].
OPC UA does not define its own protocol, but is based on others. So can about the protocols TCP, HTTP, UDP, AMQP and MQTT can be used [91]. The data can be exchanged in the classic way via a client-server connection or via a Publish-Subscribe mechanism in which the OPC UA server sends the messages to specific clients [91]. OPC UA offers a range of services that give the user a offer added value. These include information modelling, a security model, profiling, data access to current or historical data, method calls, use of a discovery server and use of alarms [92]. Due to the
Variety of possibilities is presented in this work only on the information model and the Discovery functionality received.

The information modeling follows the principle of object orientation and allows the creation of hierarchies for objects and instances [93]. By pasting
Further information can be modeled from references, so that a fully meshed model is created [93]. Clients can browse the information model and so to the get the desired information, both the instance and the object information. Manufacturers or interest groups can use so-called companion standards to create their own

Define and standardize OPC UA information models [92]. In this way, the information model can be adapted to the specific application, which means that the modeling and
Use is simplified for the user. According to the OPC Foundation website, there are currently 46 companion standards
or working groups that create new standards [94].

In an environment where there are many OPC UA servers and clients, this is not always possible
It must be ensured that a client knows the corresponding server and can address it
can. This problem can be solved with discovery servers that are in different
versions are specified, e.g. B. local or global discovery servers. The principle is
however always the same. There is a discovery server to which all OPC UA servers connect
register and store information that is required for a connection, such as
e.g. B. IP address and security requirements. A client can use this information at the
Request Discover servers and thus select the right server in the network. [93, 95]

In this work, OPC UA is used to communicate with the production plant at the
evaluation used. This is particularly useful in modular production systems
through the discovery function. The data can be described via the information model. In principle, ontologies can also be read in the OPC UA information model
However, the information model does not offer the expressiveness of the ontology language OWL [21].
Therefore, the semantic model via OPC is not suitable
to provide UA.

4.1.3 eCl@ss

eCl@ss is a cross-industry and machine-readable classification system, in
where products and services are clearly described by identifiers. When
a company wants to buy or sell a product or service
these are described with eCl@ss [96]. This makes voting easier and through the
Identifiers can be used to identify new customers or suppliers more easily [97].

The classification in eCl@ss takes place over four levels, divided into *subject areas, main groups, Groups* and *classes* are divided. Each of these four levels is identified by a two-digit number, the levels are separated by a hyphen. That's how it is
the identifier 16-01-01-04 in version 11.0 for corn. Different versions are not
always fully compatible. The subject area 16, to which the corn is assigned, stands for
Food, beverages, tobacco products. The main group 01 contains *cereals and legumes,* group 01 stands for cereals and class 04 for maize. For the corn
23 characteristics are defined in eCl@ss, e.g. B. Manufacturer name, brand, supplier name
or customs tariff number. This example shows that the characteristics for the
administration are created. None of the characteristics specify the corn from a technical point of view, such as B. by density, grain size or moisture content. Technical systems, on the other hand, have significantly more features, since the features define the technical system
specify. For example, a notebook is described in class 19-01-02-02 and
has 103 features, a programmable logic controller (PLC), class 27-24-22-07,
has 80 features. [98, 99, 97]

eCl@ss is international and available in 16 languages, but not in all languages
full scope available [99]. In version 11 there are 45,293 in German and English
Classes and 16,139 characteristics described [99]. Defined for easier use
eCl@ss also keywords that simplify the application [97]. In German
129,606 keywords are defined for this [99].

In addition to the *basic* version described here, there is also eCl@ss *Advanced.* Included
there are some additional features, such as B. Polymorphism, which allows dynamic
assignment of values [100]. However, this will not be discussed further since it is necessary for
this work is not relevant.

eCl@ss offers a comprehensive classification of products and their characteristics. That
the original goal of eCl@ss, purchasing and sales, is clear in many places.
However, it is now being mentioned as an important standard for I4.0 and concrete fields of
action are named in a white paper in order to close existing gaps [100]. the
information stored in eCl@ss cannot be used, or only to a very limited extent
use in operation. The characteristics of the products or production resources, e.g. B. the
The number of inputs and outputs of a PLC are not of great importance in operation.
Information such as B. the dependencies between the production resources,
their input and output values or the specific system topology. However, this information
cannot be displayed in eCl@ss. Still, a reference can be one
production resource on an eCl@ss class, e.g. B. This makes it easier to report a need for
spare parts in purchasing. Such use cases are discussed in
not considered further in this work.

4.1.4 VDI/VDE 3682

The VDI/VDE 3682 is a guideline for the graphical representation of formalized process
descriptions. Process-technically defined states and process operators
connected via directed edges, so this concept is graphically representable, but also
can be transferred to other formalisms. The goal is a simple, industry-neutral, easy-to-
understand process description that can be applied to the entire CV
to realize. [101]

In addition to the concept and the graphic representation in Part 1 of the Directive, includes
sheet 2 the information model [101]. The UML class diagram is used to describe the
formalism [102]. An XML representation is provided
of the guideline, as sheet 3 [101]. So it could z. B. integrated into AML
will.

The policy abstracts the process in a system boundary to a process operator,
that uses a technical resource and the three objects, product (P), energy (E)
and information (I) required and generated by the process [101]. The system limit can
be understood as a balancing area in relation to product and energy. In Figure 4.1
a typical process is shown as an example. A specified product (red
circle) into the system boundary, just like energy (light blue rhombus) and information
(blue hexagon) [101]. The process operator (green square) requires a technical

Resource (grey square with rounded corners) to run the process [101].
After the process, a product, energy and information is available again.

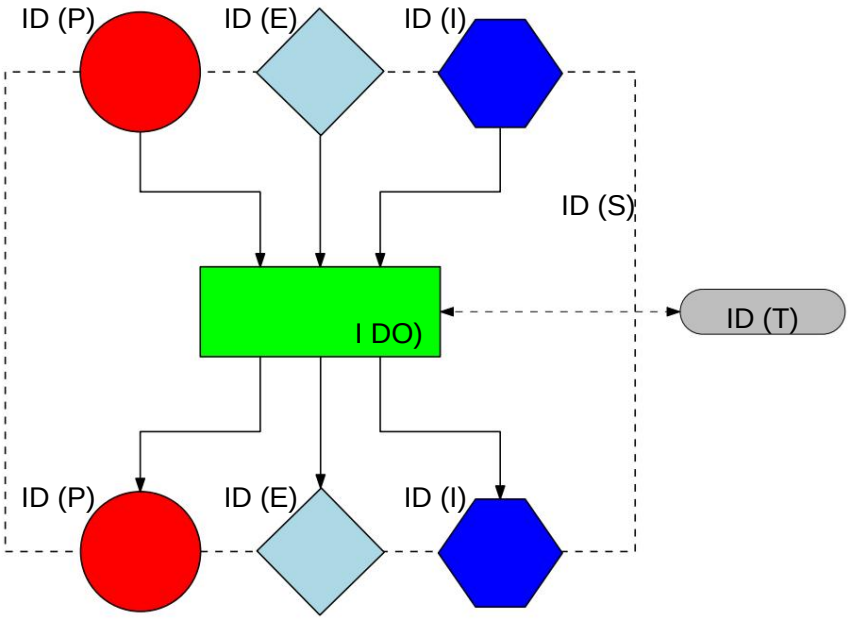

Figure 4.1: Exemplary process representation according to VDI/VDE 3682 [101].

The guideline represents a formal process description of a process. The degree of abstraction can be chosen freely, so the process operator can assemble a whole product or just a small part. In addition, the attributes that specify the products, energies and information are not defined. That's why it is difficult to use the modeled information in a semantic model. That's why this guideline is not suitable for integration into the approach presented here.

4.1.5 DIN 8580

The DIN 8580 standard describes and classifies manufacturing processes in an order system consisting of main groups, groups and subgroups. Manufacturing processes are understood to mean processes for producing a solid body with certain properties. There are six main groups : *archetypes, forming, separating, Joining, coating* and *changing material properties.* Each main group is in five to ten groups and each group into zero to nine subgroups. Similar to eCl@ss are assigned numbers at each level of the classification system, which can be used as a unique identifier. For example, the identifier is 3.1.6 *break* and belongs to group *Dissect,* main group *Dissect.* The standard has 53 references There are other standards in which the individual production steps themselves are standardized, but there is not one for all the production processes defined in DIN 8580 Standard. [103, 104]

The standard provides a generally accepted classification of manufacturing processes. for such a classification is not required for the use cases considered here, since

Production planning in use case 2 is outsourced to a planning system. In addition, the abilities of the production modules are not described. Therefore finds the standard is not used in this work, but would be suitable in principle for the description of skills in the manufacturing industry.

4.1.6 oneM2M

The goal pursued by oneM2M is uniform communication between machines to be specified to ensure interoperability. Among other things, this was developed a basic ontology that defines basic concepts and relations [105]. The basic idea is similar to that in this work, that a basic ontology can be expanded and thus adapted to the specific circumstances. The oneM2M is focused on basic communication functions in the IoT area. Eight partners (so-called Partner Type 1) stand behind oneM2M for regional standardization make, such as B. *Telecommunications Industry Association* (TIA, USA) or *European cal Institute for Telecommunications Standards* (ETSI, EU). In addition, you can also Companies become a member of the organization, so the organization as a whole almost 200 partners and members [106].

There are efforts by the organization to focus on industrial use cases [107]. The use cases described there outline a oneM2M platform and the applications based on it outside the factory, so that the data can be sent via the Internet be transferred [107]. In addition, a cross-factory use case is outlined. Together with the list of members, which contains very few industrial companies, it gives the impression that the use cases are very general and are intended to implement data exchange between different systems. An integration of the work could be interesting in the future, but so far there are no results on this use cases.

4.2 Semantic Web Technologies

In this work, the Web Ontology Language (OWL1) will play an important role, which is why this formalism is presented below. First, resource Description Framework (RDF) and RDF Schema (RDFS) on which OWL based. OWL is then presented in terms of its linguistic meta-model, since it has a gives quick and deep insight. Introducing the Semantic Web Rule Language (SWRL) completes this subsection.

[1] In the literature there are many attempts to explain why the acronym OWL (in German *owl)* and not WOL. In the mailing list of the Web Ontology Working Group, the responsible standardization body, the name *OWL* is suggested by Tim Finin because: *(i)* it is clear and simple can be pronounced; *(ii)* opens up great opportunities for a logo; *(iii)* owls are associated with whiteness; *(iv)* a Mr. Martin from MIT already tried a *One World* in the 70s to develop *language .* [108]

4.2.1 RDF and RDFS

RDF is a formal language created in 1999 by the World Wide Web Consortium (W3C) in standardized in a first version [30]. The aim of RDF is to represent information on the Internet using an abstract syntax [109]. RDF forms a graph structure by subject-predicate-object triples, whereby each of the three elements can also be specified [109]. This makes it possible to display complex structures with RDF while XML is restricted to tree structures [30]. Nevertheless, RDF data can be converted into XML represent, although the specification also defines other syntaxes [30]. The individual elements are identified via an Internationalized Resource Identifier (IRI), so that the elements are clearly described [109]. This often uses the web address of the entity that created the item, e.g. B. *http:// www.w3.org/ 1999/02/22-rdf syntax-ns#langString* is a typical IRI containing a language identifier day) represented.

RDFS extends RDF with schema knowledge, also known as terminological knowledge (TBox) [30]. The extension is RDF compliant, so each valid RDFS document is also a valid RDF document [30]. Schema knowledge means that concepts and relations are specified by a description, e.g. B. with a range of values and goals [110]. This means that RDFS is already an ontology language that is used in the However, compared to other languages, it only has a limited expressiveness [30]. Both the terms and the relations can be ordered hierarchically [110], so that suitable for simple applications. RDFS also has defined semantics, so it is possible to draw conclusions about an RDFS document [30].

4.2.2 OWL

OWL has become one of the most widely used since it was specified in 2004 developed ontology languages. OWL is an extension of RDFS, which has more predefined relations that are assigned meaning [78]. Included an OWL ontology can be represented in the RDF syntax, making it a valid RDF document is [30]. Due to the greater expressiveness, the semantics can no longer are defined in the form of derivation rules, but is defined by the description logic defined [79]. The description logic is a subclass of the first order logic, in which expressive language constructs of certain complexity classes are allowed are so that the logic is decidable and efficient conclusions are drawn can [30]. Therefore, description logics are well suited for knowledge representation.

In 2009 a new OWL version was released under the name OWL 2, also as *W3C Recommendation* [111]. New features were added so that OWL 2 has a higher expressiveness. To make a distinction from the previous one To create a version, e.g. T. the previous version as OWL 1 called. But one uniform designation has not prevailed. The different designations are sometimes confusing, but not decisive in terms of content, since one was created in OWL 1 ontology is also valid in OWL 2. Despite the sometimes ambiguous terminology, it can now be assumed that OWL 2 is meant when OWL is spoken of

will. The common tools support the new features and are therefore OWL 2 compatible. In this work, this terminology is followed and the version information omitted, as only OWL 2 is used.

Modeling in OWL is divided into classes, relations and individuals. OWL can be supplemented with SWRL rules, with SWRL being presented in more detail below will. First, the modeling constructs of OWL based on linguistic presented in metamodels. The graphic notation quickly gives a good overview reached. The metamodels are represented in MOF 2, which is standardized by the Object Management Group (OMG). MOF 2 was developed in close coordination with the UML 2 developed and has taken up many of the formalisms of UML. [67] This is the Formalism easy to understand, so MOF 2 is not presented explicitly.

classes

Figure 4.2 shows the metamodel for class expressions. The class expressions describe the extension of the class, ie which instances belong to the class. Class axioms can be used to combine class expressions and thus describe relations between the class expressions, such as B. a class hierarchy. The specification distinguishes between class expressions and class axioms [112]. This distinction is certainly in the specification sensible, but not relevant for the application. Therefore, this distinction is only made in this chapter. First, the six types of class expressions explained in more detail, the class axioms follow.

1. **Class Name (OWLClass):** The description of an OWL class by a name is special since the other five types produce anonymous classes. This means that only this type of class has a unique naming. Just this type defines a term, the other types define concepts. thats why this type of particular importance. However, the meaning of this term is not described, ie conclusions are not possible Combination with other types necessary. [113]

2. **Enumeration of class instances (EnumeratedClass):** OWL classes can be defined by a full enumeration of class instances. This is very well suited for closed terms, such as e.g. B. *Binary state* of a Class can be modeled with instances *true* and *false .* Because the class Once the enumeration is complete, no more items can be added. Syntactically, these classes are defined by the expressions *owl:oneOf* and *rdf:parseType="Collection"* defined. [113]

3. **Restriction of the class (OWL Restriction):** With this class expression, the property characteristics of an OWL class are restricted in order to so class describe. All individuals whose traits fit the restricted classes can be assigned to this class. There are a variety of options to restrict classes. These are shown in Figure 4.3 and supplemented described. [113]

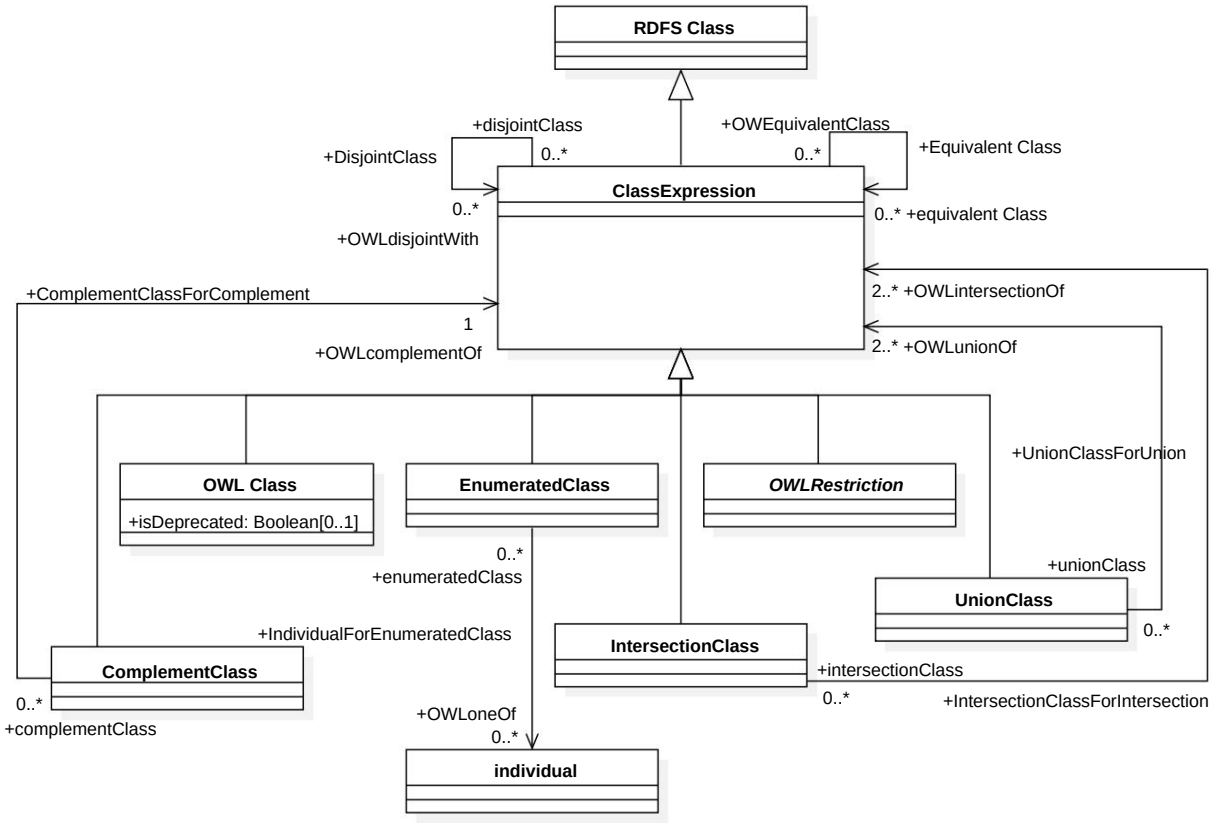

Figure 4.2: Meta model of OWL classes [113]

4. **Intersection of classes (IntersectionClass):** New classes can be defined via an intersection of two or more classes. Exactly those instances belong to the resulting class that belong to all classes of the definition. So the meaning is like a logical conjunction. [113]

5. **Union of classes (UnionClass):** As with the definition of classes via intersections, these can also be defined via unions. The resulting class then contains all the instances in one of the Definiti

   onsclasses occurs, ie the meaning here is equal to the logical disjunction. [113]

6. **Complementary classes (ComplementClass):** A class can be defined by a complementary class. Here all individuals belong to the class that are not in the complementary class, so it has the logical meaning of a negation. [113]

The further specification of the *OWL Restriction* class is shown in Figure 4.3. The restrictions can be divided into two types, value and cardinality restrictions.
First, the value restrictions are considered. The *SomeValuesFromRestriction corresponds* to an existential quantifier for a relation. This quantifier expresses that an individual belonging to this class has at least one corresponding relation. The *AllValuesFromRestriction* represents the all quantifier, so that all used object relations of a type refer to a specific class expression or all data relations of a type are within a defined value range. *HasValueRestriction* can be used to classify a class based on a specific relation to a

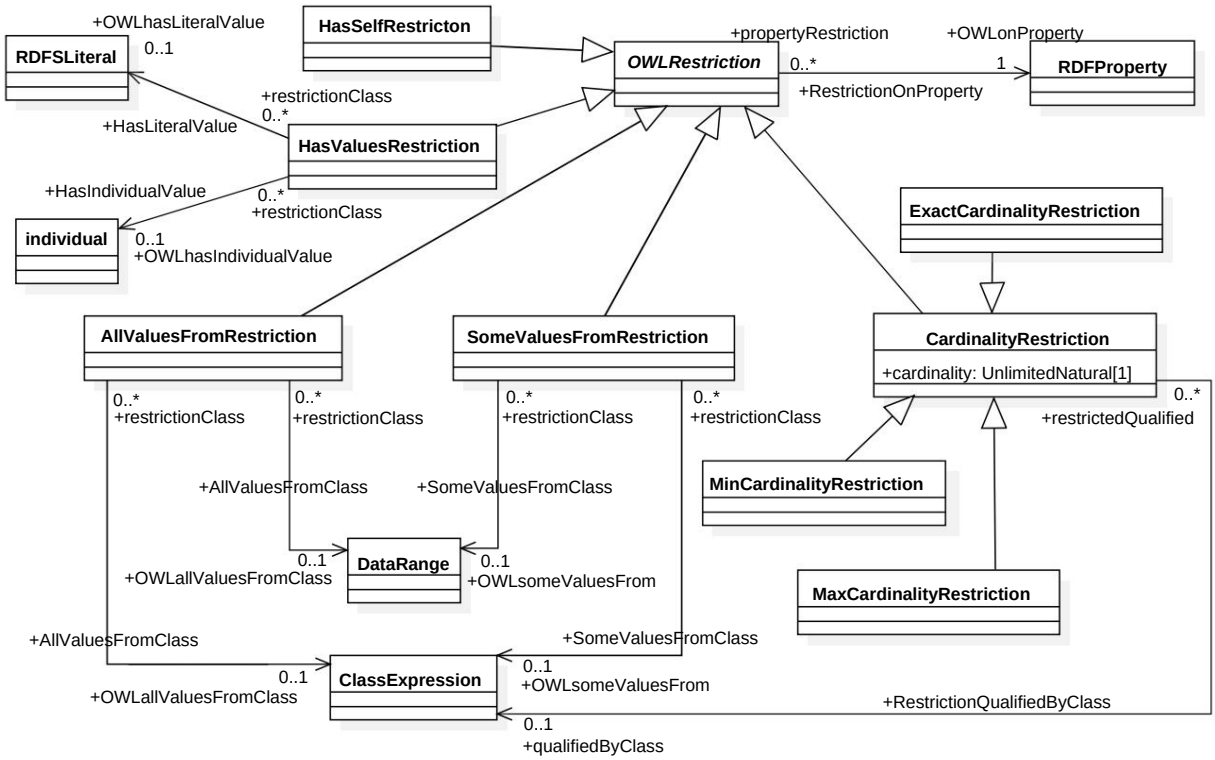

Figure 4.3: Meta model of the OWL class constraints [113]

define an individual, such as B. *StudentTHOWL* ÿ *studiesIn owl:hasValue Lemgo,* so that all individuals studying in Lemgo are summarized as students of the TH OWL. The *HasSelfRestriction* is not used that often because it only relates to itself. This is similar to *HasValueRestriction ,* but it always refers to itself, e.g. B. *Narcissist* ÿ *loves owl:hasSelf "true"^^xsd:boolean,* ie a narcissist is someone who loves himself. In addition to these value restrictions, there are three cardinality restrictions: minimum, maximum, and exact cardinality. This allows classes to be defined that have a specific number of relations. For example *Auto* ÿ *owl:qualifiedCardinality 4 wheels* expresses that a car has exactly four wheels.

The minimum and maximum cardinality can be applied in the same way. [113]

class axioms

The six class expressions form a basis for using OWL. But each construct is not very helpful in itself, for example, conclusions with anonymous classes would be possible, but these are not useful. Therefore they are summarized in class expressions in order to be able to describe such complex classes. The following three class axioms are available for this purpose [114]:

1. rdfs:subClassOf

2. owl:equivalentClass

3. owl:disjointWith2

The axiom *rdfs:subClassOf* defines a hierarchy between the classes, ie all individuals of a subclass are also individuals of the superclass. The subclass can be combined with a class name to express a simple hierarchy, such as B. *Sensor rdfs:subClassOfDevice.* The axiom is a necessary but not a sufficient condition. That is, an individual of the *Sensor* class is always an instance of the *Device class,* but an instance of *Device* is not necessarily an instance of *Sensor.* So the meaning is the same as in RDFS. It is also the same construct, as indicated by the prefix *rdfs* in the example.

The *owl:equivalentClass* axiom is used to express equivalence. This axiom is stronger because it expresses a sufficient condition as well as a necessary one. Therefore, more complex conclusions can be drawn. However, the aim of this axiom is not to define synonyms, but to describe classes, such as B. *Binary* ÿ *{true, false}.* From a modeling perspective, this axiom can also be constructed from two *rdfs:subClassOf* relationships, *A rdfs:subClassOf B* and *B rdfs:subClassOf* A. However, this relationship is usually not immediately apparent and should be avoided.

Two classes are modeled with the *owl:disjointWith* axiom if they share no instances with the other class. The axiom is a necessary but not a sufficient condition. For example, the two classes *sensor* and *actor* can be defined as disjoint, because there is no device that is both a sensor and an actor at the same time. However, the example also illustrates the difficulty of modelling, because the statement can be true or false depending on the depth of the modelling. There can be devices that act as both sensors and actuators. In general, therefore, not too many restrictions should be modeled so that the model remains generic.

The axioms can be used to define complex classes and thus more conclusions can be drawn. In the following, relations are presented that also have a major influence on the possible conclusions.

relations

The relations defined in OWL are based on the *RDFProperty* as shown in Figure 4.4. The *OWLAnnotationProperty* can be used to annotate classes, relations, individuals and the ontology (ie the ontology header). In addition to the four labels predefined in RDFS (rdfs:comment, rdfs:isDefinedBy, rdfs:label, rdfs:seeAlso), five more are defined in OWL: owl:versionInfo, owl:deprecated, owl:- backwardCompatibleWith, owl:incompatibleWith and owl :prior version [114]. Any number of others can be defined, whereby in addition to hierarchies, a domain and a range can also be specified. These are not shown explicitly in Figure 4.4 because they are already included in the RDFProperty.

2After the specification there are two more class axioms. One for pairwise disjoint classes and one for disjoint union sets. Both do not increase the expressiveness, but only simplify the modeling [114]. These constructs are therefore also referred to as *syntactic sugars* and are not considered further here.

The *Property* class is an abstract class that encapsulates the similarities between *OWLData typeProperty* (also known as Attribute) and *OWLObjectProperty* (also known as Relation). On the one hand, there is the possibility of defining an equivalent relation and properties can be marked as obsolete and as functional [113]. When a property is defined as functional, each instance can have at most one outgoing relation of that type. In a network this could e.g. B. the definition of a master. There can only be one master, so the assignment to a master could be modeled using a *functional property ,* so that each slave can only have one relation *hat master .*

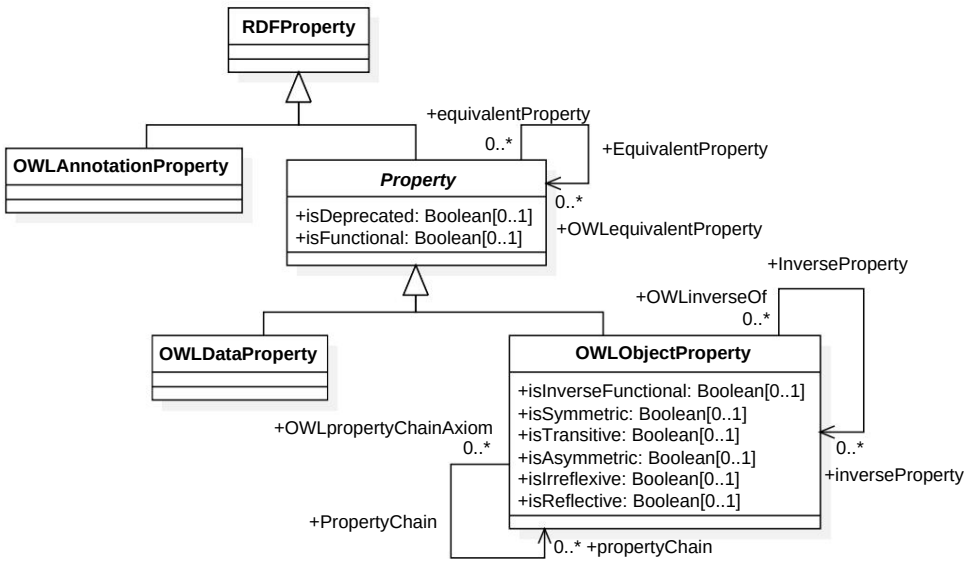

Figure 4.4: Metamodel of the data and object relations based on [113]

The *OWLDataProperty* can be used to assign an attribute of a specific data type to classes or a value (numeric or alphabetic) to individuals.
The attributes and relations can also be assigned a definition and target area, arranged hierarchically and defined as disjoint relations.
The *OWLObjectProperty* defines relations between classes or individuals. They have significantly more characteristics than the *OWLDataProperties* and can therefore be described more precisely. The inverse functional relation is the opposite of the functional relation and states that at most one relation of this type may refer to each instance. With this property z. For example, you can implement access control for a resource that only one device can access at a time. A symmetrical relation is also valid against the modeling direction, while the asymmetrical relation against the modeling direction is explicitly inadmissible. If a relation *PartOf* is defined as transitive, then it can be concluded from *Sensor PartOf Module* and *Module PartOf ProductionLine* that *Sensor* is also PartOf ProductionLine. In a reflexive relation, the relation has the same start and end point, ie the individual refers to himself with the relation. With an irreflective relation, the individual must not refer to himself with the relation, ie the start and end points must be different . [113] *Property chains* represent a kind of extension of transitivity. Relation chains can be defined here, which then themselves result in a new relation [115], e.g. B. *uncle* ÿ *parents* ÿ *brother,* ie the uncle is the brother of the parents.

individuals

Individuals are instances of classes and represent real-world objects. There are two ways in which individuals can represent knowledge. On the one hand through the class membership and on the other hand through relations to other individuals or data. Each relation and each class affiliation represents a fact. The individuals and the relations together represent the assertional knowledge (English: Assertional Knowledge (ABox)). When creating the model manually, all individuals are typically given a name. However, an individual does not have to have a name (IRI), it can also be anonymous. However, only local use of the individual is then possible, since it is addressed globally via the IRI [112]. Anonymous individuals are therefore suitable for internal issues that should not be addressable from the outside, e.g. B. Intermediate results.

Individuals can also obtain facts about their identity. This makes sense because, unlike many other knowledge modeling languages, in OWL there is no assumption that an object has just one name. That is, different names can refer to the same real object. Therefore, the equality or the inequality of individuals can be represented explicitly. Three constructs are available for this: *owl:sameAs, owl:differentFrom* and *owl:AllDifferent* [113]. With a single ontology these will usually not be used, typically only when merging ontologies.

With OWL 2 two new axioms were introduced that make it easier to assign negative relations or negative values to individuals. This means the explicit specification of relations or values that do not apply, e.g. B. *Age* 6= *25.* This is done by the *NegativeObjectPropertyAssertion* and *NegativeDataPropertyAssertion* [112]. Again, this is a fact that is potentially useful for drawing conclusions, but is not relevant to the use cases described here.

profiles

Certain forms of the formalism are referred to as profiles, which are selected in such a way that they are particularly suitable for certain purposes. It is all about the trade-off between expressiveness and the associated reasoning efficiency.
Because the more expressive the formalism, the more computationally intensive the reasoning. This goes so far that some formalisms cannot be decided, so that the conclusions contained cannot be drawn with certainty [116]. Therefore, various profiles of OWL are presented below.

In OWL 1, 3 profiles with different expressiveness are defined, OWL Lite, OWL Description Logic (German: Description Logic) (DL) and OWL Full. OWL DL has the greatest expressiveness of all decidable profiles, OWL Lite has less expressiveness and OWL Full is undecidable [111]. A few issues have been identified when using OWL DL, e.g. For example, frequently used patterns could not be modeled directly and the expressiveness of relations and data types was low, so that a new standard was developed [117]. With the introduction of OWL 2, many new features and three new profiles have been added. The three profiles defined by OWL 2 are presented below, as well as OWL 2 DL:

OWL 2 RL (“Rule Language”) was developed for applications in which good performance is required, but not too much expressiveness should be sacrificed. Significant conclusions are possible in polynomial time depending on the overall ontology size. This profile can be implemented with a rules language. [116]

OWL 2 QL (“Query Language”) is suitable for ontologies that consist of large amounts of instance data, ie with a large ABox, which must be used to draw conclusions in the context of queries. This is possible with a guaranteed memory consumption in relation to the ABox size (NLOGSPACE). [116, 118]

OWL 2 EL (“Existential Quantification”) is suitable for large ontologies with a large “schema” part, ie with a large TBox. Conclusions are possible in polynomial time depending on the TBox. [116]

OWL 2 DL (“Description Logic”) was developed for the greatest possible expressiveness while at the same time being able to decide. As a result, however, the complexity is greatest with N2EXPTIME. [116]

The question may arise as to whether there is an OWL 2 DL profile at all, since it is not defined in a separate section in [116] like the other profiles. Instead, only the introduction states that OWL 2 DL can be regarded as a profile3 . Due to the three new specialized profiles of OWL 2, all of which are below the expressiveness of the DL profile, the Lite profile is practically irrelevant. OWL 2 Full, on the other hand, is not explicitly defined in a specification, but only "by exclusion criterion". It is then an OWL 2 full ontology if it is not a DL ontology.4 This shows that the full profile also has no significant meaning, which is due in particular to the greater expressiveness of the DL profile, which makes the difference between these profiles is low.

merge

In the relevant literature, merging is typically understood to mean the unification of heterogeneous knowledge bases [119, 120]. This understanding is shaped by the Semantic Web, where there are many knowledge bases with different perspectives on a domain. However, the knowledge bases are often not very strongly networked, so that they are relatively simple ontologies. For merging, semantic and syntactic similarities of terms and relations are searched for and assumed to be the same if they are of sufficient quality [121]. The results achieved are very different depending on the application, e.g. Sometimes there are high error rates [122, 123]. An overview and current challenges are presented in detail in [122]. These approaches are not suitable for merging ontologies for industrial applications, since the reliability and correctness of the model are very demanding in the industrial environment

[3] In the introduction to [116] it says: "however, all OWL Lite ontologies are OWL 2 ontologies, so OWL Lite can be viewed as a profile of OWL 2. Similarly, OWL 1 DL can also be viewed as a profile of OWL 2.

[4] In the original: "Ontologies that are not in OWL 2 DL are often said to belong to OWL 2 Full [...]" [115]

will. The industrial requirement is to achieve an accuracy of >99%. However, no algorithm could be found that reliably achieves this.

4.2.3 SWRL

SWRL combines OWL ontologies with DATALOG rules. The consistency with RDF and XML remains unaffected [124], so that SWRL can easily be integrated.
Nevertheless, SWRL is not standardized in the actual sense, but only has the status of a W3C Member Submission [124]. The rules make it easy to express if-then relationships, but the application is limited to the ABox. However, connections can be established between different individuals and new class memberships, relations or attributes can be inferred, which would not be possible without SWRL. A special feature is that the additional expressiveness, even with the DL profile, does not limit the ability to decide [125]. This is the case if the rules are *DL-Safe ,* ie the type of each occurring individual must be specified in the rules [126]. In addition to the classic rules that infer an effect from a cause, so-called built-ins can also be used. Built-ins enable functions such as comparisons, calculations, manipulation of strings or processing of time and date information to be integrated into the rules [124]. This also results in new possibilities, but no reasoner could be found for the built-ins that supports them, so that they are not yet applicable. Overall, SWRL offers an easy way to model further knowledge in OWL and to draw more complex conclusions.

4.2.4 Graphic representation of ontologies

The representation of ontologies is important for explaining the modeling. Due to the large number of modeling elements, not all aspects can be represented appropriately, which means that the entire model is not visible in the graphic representation.
This difficulty is certainly one reason why the representation of ontologies is accomplished in very different ways in the literature. On the one hand there is the textual representation in which the syntax of the ontology is presented. The notation of the description logic or the RDF/XML syntax is often used, e.g. e.g. in [127, 128, 129, 130].
This method is suitable for precisely describing a very small section, but not for larger sections or to give an overview. Alternatively, the ontology can be represented graphically, whereby a large number of formalisms can also be found in the literature. A well-known tool is OntoGraf, a plugin of the well-known editor Protégé. Classes and individuals are represented as nodes (squares), with the class name being preceded by a yellow circle and the individual name being preceded by a purple diamond. The nodes are connected to each other via edges (relations), whereby the edges only differ in color and thus represent the name of the relation. The allocation of the colors to the relations can then be read in a legend, although the allocation is often not clearly possible due to similar colors. Due to the simple creation of the images, this variant can often be found in the literature, e.g. e.g. in [131, 132, 133, 134]. However, since only a little

information is presented, this representation of the model is considered unsuitable in this work. It is also possible to represent ontologies in UML (class and object diagram). As with the other forms of representation, not all aspects of the ontologies can be mapped in UML [135]. By specifying edges more precisely, more information can be displayed than with the OntoGraf. Furthermore, attributes can be represented in classes and individuals. In addition, UML has the advantage that the formalism is widely used and known, making it easily accessible to a large number of readers. This approach is therefore more common in the literature, e.g. e.g. in [136, 137, 138]. As an alternative to UML, you can also define your own notation. These representations are often difficult to understand and confusing, see [139], or focus on a certain aspect of the ontology, see [82], so that this variant is also considered unsuitable for this work.

In summary, it can be stated that there is no unconditionally good form of representation for ontologies that is understandable for humans. After weighing up all the advantages and disadvantages, the UML seems suitable for this work, as it represents a good compromise between clarity and completeness. In addition, this graphical language is formally defined. UML version 2.5.1 is used [62].
Class diagrams are used to represent the TBox and object diagrams are used for the ABox. The UML specification is relatively open, so the representations conform to the standard. However, there may be deviations from the usual notation. For example, in the class diagram For example, the associations are only named so that the two ends that represent the role are not named. Since the name of the association is often only valid in one direction, a small filled triangle is used to indicate the reading order.

4.3 Related Works

In the last two subchapters, the focus was on technologies that can be used to solve the challenge presented in chapter 1.1. For this purpose, related work that pursues similar objectives is presented and discussed here for the application.

4.3.1 Modeling of KPIs

The publication [140] conceptually deals with the design of a system for the automatic calculation of KPIs in production plants. The procedure is strongly based on automation technology and it is examined which information must be available at which levels. On the one hand, conformity to four different standards is required so that the information is available.
On the other hand, a KPI Element Information (KEI) model is required that describes the KPIs. However, the KEI model is not presented, which means that the description remains abstract. In addition, no modular systems are taken into account.

The motivation in [141] is to ensure the quality of the determined KPIs. This should be guaranteed, since important company decisions are made based on the KPIs. Quality assurance includes the process of data collection and processing. A KPI quality model is presented, which has 59 properties includes, which are to be ensured during processing. True, this method will used in software development, but many of the quality properties could be found on industrial KPIs are transferred. Thus, this work represents a possible extension However, the description of the rules for calculating KPIs will be presented in the work not considered.

In [142], work is being done on the goal of automatically recording KPIs from modular production systems. Ontologies should also be used for this purpose. The work handles the use of engineering data as a source of information and the integration into an ontology. The creation of the ontology and thus the calculation rules, as well as the Implementation of the presented approach is designated as future work. So the work [142] is a good addition to this work, since in this work the use of engineering data is not dealt with explicitly.

So that business processes can be offered *as a service ,* a concept is presented in [143]. presented for a KPI system that can be outsourced to a cloud. The goal is to be described with a description using ontologies, relevant characteristic values of the KPIs, e.g. B. for which goals the KPIs are determined or which limit values the KPIs to have. In the approach, a second ontology is defined, the dependencies between different system components. The query language for the results will be SPARQL Protocol And RDF Query Language (SPARQL) used.

The diverse current work shows that KPIs are of great importance. The goals of the work are similar, reliable and correct KPIs should be established be ensured. The approaches vary, with implementation often involving similar technologies. A comparison of different methods used in used in the literature can be found in [143]. It turns out that the ontologies used are well suited. It is conceivable that the different approaches to be united. For the approach pursued in this work, the calculation rules for Modeling modular production systems could not be implemented in the literature being found.

4.3.2 Description and control of modular systems

According to the concept of [144] can be used to describe modular production systems so-called skills are used during the configuration phase and control in the operating phase will. A skill describes the potential of a component to achieve an effect [145]. In other words, skills provide functions that are orchestrated can be used in order to be able to map production processes. The description of Skills are based on a meta model [144]. The system is controlled via OPC UA realized by method calls, which then run a skill on the production module. In connection with the publish-subscribe mechanism and the time-sensitive Networking (TSN) standard, real-time capability can be achieved [144, 146]. But

In order for the various skills to become a process, they must be orchestrated according to a process description. These orchestration challenges will explained in [147]. For this purpose, a four-dimensional classification of skills is presented, which clarifies the different aspects of the skills. OWL ontologies are considered more promising for describing skills and comparing them with the process description approach proposed [147].

In [148] a concept for the automatic adjustment of the production process at a change of environment, e.g. B. by changing hardware presented. In doing so assume that the modules are interconnected via autonomous transport vehicles are. Thanks to the skills described, the system is able to manage the production process to be determined and adjusted if necessary. To determine changes, an OPC UA Discovery Server, a Local Discovery Server with Multicast Extension.

Alternatively, the system can also be controlled by a higher-level controller based on the status, as presented in [149]. The basic requirements that each module has its own controller, the capabilities are described and that OPC UA is used are the same for the two approaches from [144] and [149].

The work [138] describes the capabilities of production modules in order to to then compare this with a recipe (process description) and suitable modules to select. This approach is limited to batch processes in the process industry and does not consider the control of the system. For the description of skills and OWL ontologies are used for the process description. In [150] the process of a discrete production, also via an OWL ontology. In doing so however, only the path of products through the production plant is described and adjusted if necessary, so that this is more of an optimization at runtime. One other work provides a formal process description for discrete manufacturing VDI/VDE 3682 and defines it via a linguistic meta model [151].

In [134] a methodology for the dynamic, context-based orchestration of field device functionalities has been created. For this purpose, the capabilities of the field devices are described, as is the process description for the products. For a specific process to create, the process description is compared with the capability description, to identify a suitable process. context information is used in order to realize a close integration with the production plant, ie this as possible to use efficiently. In the implementation, the capabilities and the process with OWL-S described, with OWL-S being a variant adapted for descriptions of web services from OWL is.

The list of works presented here is not complete, as it is a very active one topic in the research. A distinction was made between the topics of *controlling modular production systems* and *orchestration .* Ultimately, the process flow chart should also be executed so that the two subject areas related. However, since the issues are complex and the challenge is great, the work focuses only on the core issues. Related topics such as B. the Compatibility of the modules in application 2 are not taken into account in this work. However, an intersection can be identified in the technologies which simplifies a possible integration of the works.

4.3.3 Diagnosis

There is a large number of works on the diagnosis of technical systems. In [152]
an ontology is used for fault diagnosis of rotating machines. For this purpose, the classes
and relations for describing the error are defined, e.g. B. possible
Error names and vibration characteristics. About rules defined in SWRL lets
the reason for the respective error can be concluded. [152]

A similar approach is used in the work [82] on monitoring infrastructure networks,
such as B. rail networks or power grids, described by means of semantic modeling.
The semantic description is intended to simplify monitoring and enable linking with other
networks. OWL DL
used. Monitoring can be done directly in OWL by reasoning,
further functions can be realized via SPARQL. Due to the large extent of the infrastructure
networks, the approach was developed in such a way that the evaluation also
can be distributed. [82]

The work [153] aims to create a generic concept for the knowledge-based resource
monitoring of production plants. From the procedure that is
Concept similar to that of [82] and [152], ie OWL ontologies are used,
for which rules for monitoring are created and an OWL Reasoner takes care of them
Conclusion. Alternatively, requests can also be made via SPARQL.

A knowledge-based diagnosis concept is presented in [86], in which a plant structure model
and a process model are used for diagnosis. The necessary plant structure model can be
derived from a process flow diagram, so that
the engineering information is reused. However, additional information is required that
cannot be obtained from the engineering data. the
Process description, which is based on the guideline VDI/VDE 3682, is formalized by a
Process Specification Language (PSL). The plant structure model is
represents the CAEX format, which is also used in AML. Possible causes of errors are then
determined using generic rules. The focus of the work
lay on process engineering systems, for which also an example of a diagnostic procedure
was presented.

In addition to the knowledge-based diagnostic approaches presented so far, there are many
of other procedures that create a model in a special formalism, which
can then be used for diagnosis. In [154] a diagnostic method for hybrid systems is
presented, which is based on a model-based diagnostic approach and which
Satisfiability Modulo Theory (SMT) used. The model of the plant is thereby through
a state space model is shown, for the creation of which a mathematical description of the
system must be created. A diagnostic approach that is also based on SMT
is presented in [155]. The system is divided into different states, but no distinction is made
between continuous and discrete signals,
as in many other approaches. The disadvantage of this method is the long runtime
Calculation of the causes of errors.

A frequently used formalism for hybrid systems are hybrid bond graphs (HBG),
as presented in [156] using the example of a fuel delivery system of an aircraft

the. The HBG is converted into a temporal causal graph
Graph (TCG)), which can then be used for diagnosis. There
but the scalability of this approach is bad, in [157] an extension of the
Approach presented, which is suitable for modular systems. A distributed algorithm is used
to identify locally occurring errors. Only if more than one
local algorithm detects an error, a higher-level diagnostic algorithm is used
executed, whereby runtime advantages are achieved. However, this can lead to this
It can happen that the correct causes are not identified in complex root cause analyses
can be, e.g. B. because they are in another module.

Another focus of research in the field of diagnostics is increasing the performance of
algorithms, since the complexity of diagnostics is often a challenge
is [158]. There are some performant algorithms, such as B. SAFARI [158], SDA [159] or
SATdB [160]. Each of these algorithms is more efficient than the previous algorithms and
often improves performance by more than an order of magnitude. There are also
approaches that parallelize the calculation so that multi-core processors are better utilized
which makes calculations up to 85% faster [161].

The knowledge-based diagnostic approaches often use the ontology as an environment for
the determination of causes, ie the diagnosis is completely mapped in the ontologies.
As the works show, this works in many areas. However, in these cases the models are
specifically geared to the problem and are therefore not generic.
The actual determination of causes is then typically carried out using rules, so that a
relatively simple method is used. In contrast, special models are used in the non-knowledge-
based methods, which are T. complex algorithms can detect possible causes of error. The
challenges are to
one is the creation of the diagnostic model and the runtime of the algorithms [18]. Therefore
a combination of the two methods is developed in this work, the complex
Algorithms used to identify the causes of errors and with the advantages of
semantic modelling.

4.4 Approach to modelling

This chapter first presents a best practice for creating ontologies. The procedure refers to
the entire creation process,
not only about the modelling, and can also be found in the structure of this work.
In the second section, the modeling is discussed more specifically. become
first the general requirements for a model are explained and then rules and
Established conventions for creation.

4.4.1 Procedure

Many methods for creating ontologies can be found in the literature [162, 163, 164, 165].
The best course of action depends on many factors,

such as B. Number of stakeholders, experience, complexity of domain and type of information sources. Thus, the methods provide a guide and can always be referred to be adapted to a specific application. It can e.g. B. individual steps sequentially, be performed in parallel or in a loop. A well-known and generic method is the so-called METHONTOLOGY. It defines the necessary steps, but does not lay them down determines the order in which these are to be processed [162]. The steps of this method are presented below, most of them also in the other methods are to be found.

- **Specification:** First, the ontology must be specified. This includes, among other things the use cases, usage scenarios of the ontologies, scope of the ontology and Requirements are recorded by the users.
- **Knowledge acquisition:** Necessary knowledge is collected. There can be a variety of sources such as B. Expert interviews or text analysis. the goal is to get an overview of the domain.
- **Conceptualization:** Describes the process in which the already known vocabulary is assigned to a modeling construct. This is independent of the formalism, since only classes, relations and instances are assigned.
- **Integration:** This step checks whether existing ontologies are used can become. For this purpose, similarities are sought in order to then map between of conceptualization and to establish an existing ontology.
- **Implementation:** This includes the implementation of an ontology in a specific modeling formalism. This should be done in a suitable tool to avoid errors.
- **Evaluation:** During the evaluation it is checked whether the ontology is complete and correct.
- **Documentation:** The documentation is a task that should be carried out in all phases, so that the information from all phases can be accessed later are.

A similar procedure for creating ontologies for CPS is proposed in [163], this procedure is more formal and structured. There are necessary documents and sequences, which does not reflect the iterative character of ontology creation. In addition, this approach provides for three different stakeholder groups, domain experts, ontology experts and users of the finished system. and is therefore aimed at larger projects. Therefore this approach is not used. Out of The explicit request to use information from the community, such as e.g. B. Norms and standards. These are common and the comprehensibility of the ontology thus increases. Some industry standards have been already modeled in an ontology and published on GitHub [166].

The steps of METHONTOLOGY can be found in a very similar way in the ontology life cycle in [167], which is shown in Figure 4.5. In addition, the life cycle nor the items *use* and *maintenance* or *update* of the ontology. The life cycle shows that ontologies evolve and the work does not

the creation ends. Examples are the Quantities, Units, Dimensions and Types (QUDT) ontology [168] or the time ontology of the W3C [169], which revised and such be adapted to the needs.

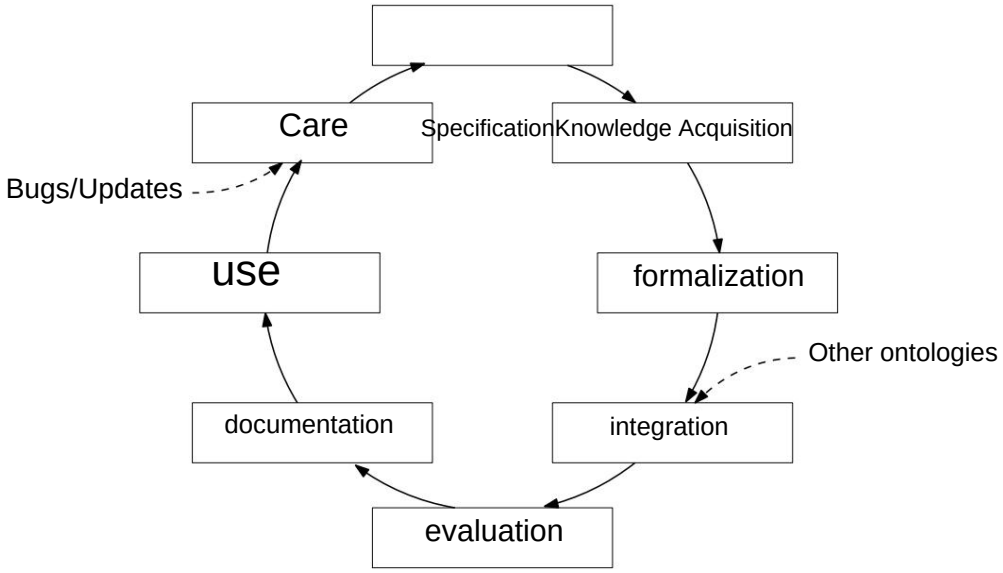

Figure 4.5: Life cycle of an ontology according to [167]

In the Lifecycle Support for Networked Ontologies (NeOn) project, the focus was on an improved handling of networked ontologies in a dynamic environment. Among other things, modeling patterns for ontologies (engl. Ontology Design Pattern (ODP)) identified and made available5 . On this platform you can search for patterns, discuss and post your own patterns, so that the platform can be used as support for modeling problems [170, 171]. More Information on ODP can be found in [172] and [173].

The procedure in this work is based on the METHONTOLOGY method, what is also reflected in the structure of the work. The specification is presented in Chapter 3. Since the unusual case here is that the author has a great deal of prior knowledge brings to the domain, the area of knowledge acquisition is small and has been e.g. Partly documented in publications [18, 19]. The conceptualization and integration done in chapter 5. Implementation and evaluation can be found in chapter 6. the documentation is e.g. T. in the respective chapters and additionally in the ontology itself present. Existing ontologies are used when it makes sense, as well known modeling patterns.

4.4.2 Model Creation Rules

The possibilities for creating ontologies in OWL were presented. But just knowing about it is not enough. The challenge of creating of ontologies is the correct use of the elements. The difficulty is in [174] graphically illustrated where a domain expert creates an ontology with 165 classes

5www.ontologydesignpatterns.org

has, of which 114 classes are unsatisfiable. It is therefore important to familiarize yourself with the correct use of the modeling elements and also to be aware of typical mistakes.

In [175] Gruber proposes five design criteria for the creation of ontologies.

- **Clarity:** An ontology should be clearly defined, ie the intended meaning of the terms and relations should be clear, understandable and objective. If possible, complete definitions (ie necessary and sufficient conditions) should be used and all definitions should also be presented in natural language, ie as a comment.

- **Coherent:** An ontology must be self-consistent. It should not contain any logical contradictions so that the terms and conclusions correspond to the intended concept. She should not have any unattainable concepts about this.

- **Extensibility:** An ontology should be extensible so that it can be adapted to new requirements without having to change the original terms.

- **Minimum implementation influence:** The terms should be influenced as little as possible by the implementation in a specific formalism. if e.g. If, for example, a formalism with too little expressiveness was chosen, the knowledge cannot be represented or can only be represented in a very complicated way, so that there is an implementation influence.

- **Minimal ontological obligation:** An ontology should define terms as openly as possible. This way it can be used by many and the terms can be specified if necessary.

These general design criteria also apply to a large extent to the ontologies to be developed here. Only the last criterion is not entirely correct, because in addition to a large number of desired users, the ontology should above all be unambiguous and this requires more precise specification. Furthermore, the design criteria can serve as a target. However, not all criteria can be fully met, since they are e.g. T. influence each other, so that there are conflicting goals. In addition, an evaluation based on these goals is difficult because they can only be measured to a limited extent. Therefore, they can only serve as an aid, but they are not sufficient to create a model based on them.

However, the criteria listed above are not suitable for the assessment of individual modeling patterns, since e.g. B. a pattern that is correct is also coherent. No criteria could be found in the literature by which certain patterns could be assessed as suitable or unsuitable. Therefore, own criteria were developed, which are used in this work for evaluation:

- **Unambiguity:** The modeling must be unambiguous so that the statement can be reliably recorded and there is no confusion.

- **Comprehensibility:** The modeling must be comprehensible so that the meaning is recognizable and it can be used correctly.

- **Compactness:** In the case of equivalent models, the more compact is to be preferred, since it is created faster and requires fewer resources.
- **Compatibility:** The modeling should be compatible with other ontologies, if applicable, but especially to your own model, so that it can be integrated into the model lets, e.g. B. Attribute ÿ Individual.
- **Extensibility:** The models created should be able to be expanded, e.g. B. to map additional features.
- **Appropriateness:** With all the criteria, however, the modeling must also be proportionate, so that central elements are defined accordingly, but peripheral areas not unnecessarily expanded or complicated.

Modeling constructs can be evaluated with these criteria. But with that
To create a good model, additional rules and conventions are required.
These support comprehensibility and are part of good professional practice. For this work, the following conventions and rules are established.

- **Naming convention:** There is no general convention on how ontologies should be created. For classes, the designation can be singular or plural to get voted. The argument in favor of the plural is that a class is typically a combination of several subclasses or instances. On the other hand, at Considering an axiom, the designation in the singular in the modeling is intuitive. Nevertheless, both variants are common [164]. In this work, classes in Singular referred to, since project-specific adjustments were made in the ABox and this variant is more suitable for this. Furthermore, can be specified how classes, relations and individuals are denoted. In this work classes and individuals always begin with a capital letter and become so no spaces used, since spaces may complicate processing due to incompatibilities. Instead, the lowercase notation is used.
  camel case) used to represent compound words. Relations always start with a lower case letter and also use the lower case letters
  Notation. These naming conventions are also used in the same way by the W3C followed [176]. However, it should be noted that when using external ontology parts, the name of the other ontology must be adopted, even if it contradicts this naming convention.
- **Handling of synonyms:** The handling of synonyms is often not done correctly [177]. It's temptingly easy to think of a new term as a class to represent and to define this as an equivalent to the synonymous term. at Queries can now use both names. However, this approach has the disadvantages that the ontology becomes confusing and large, which increases resource consumption significantly. Also, no languages can be distinguished and it is logically wrong since they are not two different ones concepts but two names for the same concept. Therefore, it is better to use the popular Simple Knowledge Organization System (SKOS).
  SKOS defines two relevant terms for this: *skos:prefLabel* and *skos:altLabel.* This allows a typical name to be defined, but also alternative designations represent. Also, they can be tagged with a language so that

allow these label terms with the same IRI in any language. [178]
In order to use a language, it is therefore only necessary to ensure that
corresponding labels of required terms for the language are defined.

- **Language and documentation:** Any language can be mapped using language tags.
However, the main language is English, ie the IRIs are in
to name in English. In any case, the documentation is in English
to be carried out, other documentation languages are not used to
Ensure consistency of documentation. Each class and relation required
a description in natural language. The description should be precise about the item
define and distinguish from other elements, an example of usage
is also desirable.

- **Integration of ontologies:** If a new ontology is to be created, it is usually checked
that no existing ontology is used for the purpose
can. But even if no existing ontology represents the whole knowledge, so
part of the required knowledge is often already formalized in ontologies. Therefore
it makes sense to include these ontologies in the new one. For one, it simplifies the
Application or application development, if the user is already familiar with parts of
the ontology and, on the other hand, the knowledge does not have to be remodeled, but
can be reused. Typically, the existing ontology is imported to use the terms and
relations. An import can affect a
refer to a specific version or to the ontology in general, which is then the latest
ontology version corresponds. However, it is possible that in a current version
Changes were made, allowing conclusions of their own ontology
are missing or inconsistent. Because the functionality of the new ontology but
depends on the availability of the imported ontology, the availability will increase
a requirement for usability. Even if that is the case for older ontologies in the
Web is not always given, this problem is manageable for industrial applications. A
second negative effect of imports is increasing size
of the ontologies, because all axioms are added to the ontology [179]. As a result,
the ontology becomes more confusing and reasoning takes more time
and resources. Since ontologies mostly cover a very small special part,
the semantic models in this work, however, overlap with many areas
often only a very small part of the ontology is used. Therefore become
the ontologies are not imported in this work, but the IRIs are taken from the relevant
classes. Since the IRIs are unique, it can be seen that
these are the same terms. In addition, depending on the application
the other ontologies are then imported to have full functionality
to use. Equivalent classes and relations do not have to be marked for this
will. This makes the ontology compact, but at the same time easily expandable,
so that it is the appropriate concept for this use case.

- **Class or instance:** A common difficulty is deciding whether to model an abstract
concept as a class or as an instance, e.g. B. *Energy consumption.*
The decision depends on the application. This is how it would be modeled as a class
if there were several instances, ie the energy consumption was specified even
further [164, 177]. If this is not the case, modeling as a class is chosen.

One challenge is translating this into general modeling, since
it is not known whether terms are to be further specified later. Included
the so -called *punning* can help, where classes and instances have the same name
may have and is selected on the basis of the context, which is meant. As a result,
the class hierarchy can be specified and the levels in which none
instance exists are ignored.

This section introduced the basic procedure and rules for creating a model. They serve as guidelines for creating the models
in Chapter 5.

5 solution

In the past, a constant shortening of product life cycles could be observed [1], which is favored by the fact that new technologies are being developed more quickly market [2]. The increasingly shorter product life cycles are for the manufacturing companies a challenge, since the more frequent conversion of the production facilities is necessary [180], but efficiency must be maintained [181]. Modern production plants counter this challenge with, among other things, the Modularization of the production facilities [182]. This allows the plant topology to be changed in order to respond to different product requirements [5]. Often modules from different manufacturers are used within a system because depending on the industry, the manufacturers have specialized in a part of the process. In order to flexibly combine modules from different manufacturers, interoperability is essential be achieved on several levels. The levels are shown in Figure 5.1. the Hardware of the modules must be compatible so that e.g. B. an electrical and mechanical connection can be established. Fieldbus systems need communication and thus enable data exchange, which is necessary for joint operation. Transmitted data must comply with a syntax that has been agreed between the modules have to be. So that the syntactically correct data can be interpreted, semantics are required. If interoperability is achieved at all levels, modules from different manufacturers work together and use shared services.

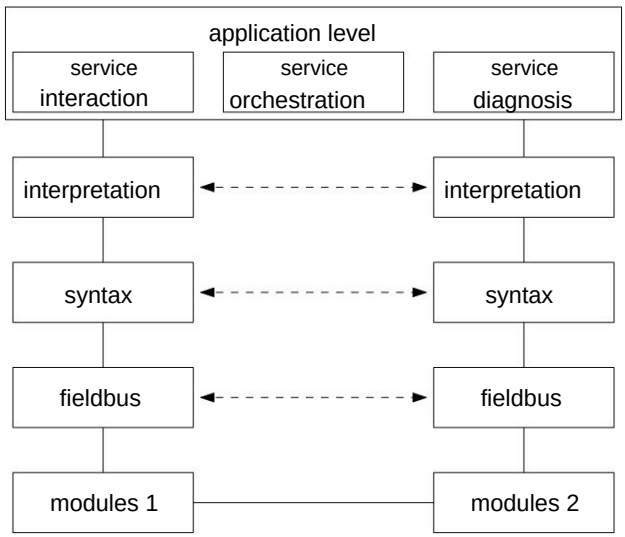

Figure 5.1: Different layers to create interoperability.

In order to achieve interoperability in modular production systems, fixed interfaces are now defined [183]. But when the system is converted, the

Interfaces have to be adapted manually [7], as they are not generic and not standardized at all levels. Due to the necessary manual adjustments, this concept not sustainable, since the conversion always requires time and thus also financial expenditure means [183]. The aim of the manufacturing companies is therefore that production modules can work together after a conversion without manual software adjustments.
This goal can be pursued through different approaches. Go classic in the Automation technology uses many standards or norms. In it z. B. for required and optional signals and parameters for each module type are described. However, this has the disadvantage that the description is inflexible. If a manufacturer want to equip his module with an additional function, this may not be complete covered by a standard. Furthermore, the determination of the signals and parameters unsatisfactory. Other information such as B. the topology or causalities th between the modules must be shown. Standards have legitimately have a firm place in automation technology, but are not sufficient for this task. Alternatively, a semantic model can be used. A semantic Model defines terms and relationships between the terms in a metamodel. With the help of a uniform meta model, production modules can assign their signals and assign a meaning to parameters in the so-called plant model. Another production module can interpret the signals and parameters through the plant model, if both use the same metamodel. The advantage of this approach over a The standard is that the model displays all the necessary information in machine-readable form and thus extensions are easily possible.

A simple expandability of the system description at model level is necessary, in order to do justice to the special features of individual systems or sectors. Because production facilities are typically one-offs built to customer specifications many systems have corresponding special features. Furthermore, must e.g. T. industry-specific peculiarities are observed. The focus of this work is on the interpretation of the data from different modules, see Figure 5.1. The semantic Model was chosen because it can be flexibly extended and all information is explicit and machine-readable. This approach is also used in the literature as a goal for future production facilities are described, see [183]. To realize a semantic A top-down approach is used in this work.

The structure of this chapter is shown in Figure 5.2. In chapter 5.1 first a suitable modeling formalism is selected (FF 1). To choose the appropriate one For formalism, the requirements of Chapter 3 are compared with the capabilities of the formalisms, and then the one that is most appropriate is selected. In subchapter 5.1.2, a suitable modeling language is selected in which the formalism is represented. In particular, it is possible to choose between different levels of expression, with the time required for the calculation increasing as the level of expression increases of conclusions increases. The formalism represents the meta-metamodel, i.e. the Language in which the meta model is created. The meta model defines terms (also called types or classes) and relations, which are then used to create a plant model be used, see [19] and [20]. In Chapter 5.2, a meta model is developed for each of the three use cases presented (FF 2). Furthermore, the similarity of the three Metamodels are checked and it is analyzed whether a cross-application model can be created (FF 3). The application is presented in chapter 6.1, the

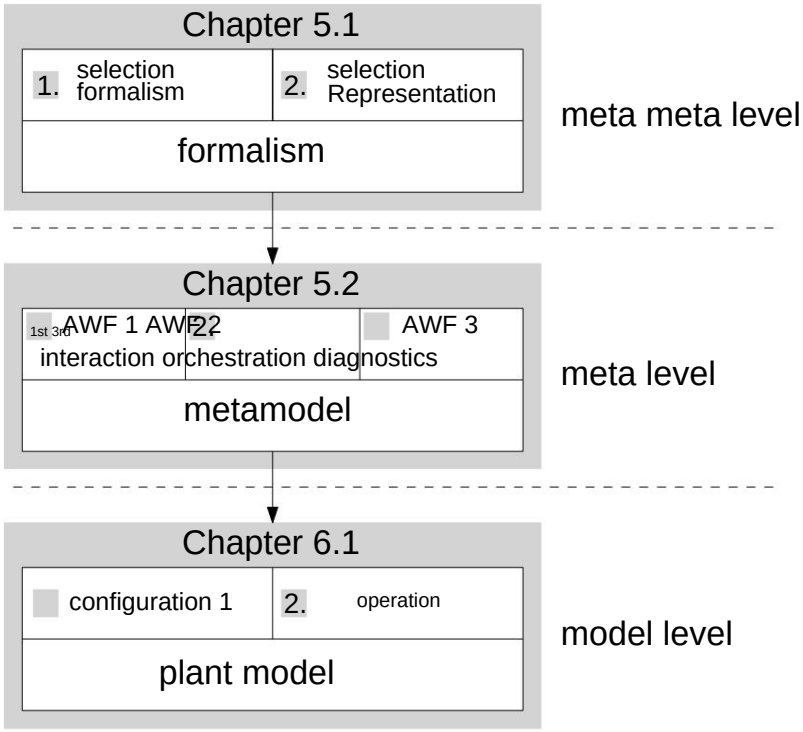

Figure 5.2: Procedure for developing a solution.

shows exemplary necessary steps for using the models in modular production systems.

5.1 Choice of modeling formalism

“Artificial intelligence [...] provides a multitude of formalisms that allow to create semantic models” [78]. From this multitude of formalisms the appropriate one must now be selected. For this purpose, requirements for a formalism were already identified in Chapter 3. This section lists the requirements compared with available formalisms to select an appropriate formalism, so that FF 1 can be answered.

5.1.1 Formalisms for Knowledge Modeling

Figure 5.3 shows some knowledge modeling formalisms with increasing expressiveness. According to [184], expressiveness is understood to mean the range of statements that can be expressed by a formalism. Statements can be made in this be equated with modeling constructs, since the constructs used to represent statements. A formalism *A* is if and only then more expressive than formalism B if formalism *A* can describe everything that *B* describes, but not the other way around [184]. Due to this relatively strong restriction, similarly strong formalisms, no clear order can be established, as will be the case later will be shown.

Of the formalisms considered, a glossary has the lowest expressiveness. A Glossary is a collection of terms aimed at clear naming, avoiding homonyms and synonyms. It is not with a glossary however possible to form a structure with the terms [185]. This allows for a taxonomy in which a hierarchical structuring of terms is possible. [185]. Thesauri are in Contrary to taxonomies not limited to hierarchical structures, but can also use other relations between the terms [186]. However, there is only a defined set of relations that can be used: hierarchy relations, Equivalence relations and association relations [187]. The concepts and relations of Thesauri can be defined in different languages, so that multilingual applications can easily be implemented [188]. Typical areas of application are documentation centers, libraries or archives [187]. Topic maps remove the limitations of relations, ie any relations can be defined that express the relationships between the terms [189]. This opens up a multitude of new possibilities, e.g. B. Navigation functionalities in document structures, filter functions or role definitions for access management [190]. In contrast to the other formalisms, ontologies are based on logic, so that integrity and inference rules are typically defined for them [30]. Also, for the terms in ontologies Attributes are defined, which makes a more precise description possible [30].

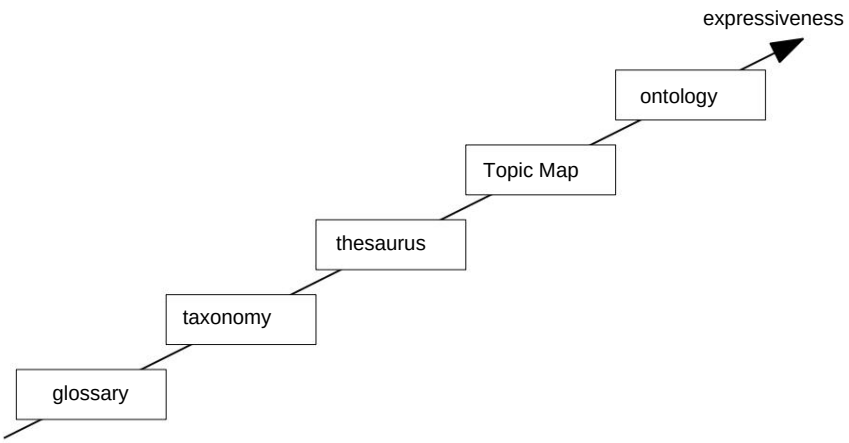

Figure 5.3: Expressiveness of different knowledge modeling formalisms, based on to [185]

Of the formalisms shown in Figure 5.3, only the ontology has sufficient expressiveness to represent the necessary knowledge for the use cases discussed here. Expressiveness is required because the knowledge to be presented is heterogeneous. On the other hand, the formalisms listed above are more for the representation of something uniform appropriate knowledge, such as B. to categorize or link different languages, or to represent always the same attributes [78, 185]. However, in addition to the ontology there there are some other formalisms with a similar expressiveness, such as *semantic networks, frames* and *production rules.* These formalisms are presented below, where only the respective idea is in focus and reference is made to the relevant literature for details. Since ontologies have already been introduced in Chapter 4.2, at this point renounced.

Semantic Networks

In semantic networks, knowledge is stored in the nodes and the attributes of the relation
presented [191], see Figure 5.4. Relations can have attributes and a cardinality
assigned, which specifies them in more detail [78]. knots can a
represent an object or property. However, it doesn't explicitly switch between objects
and properties are distinguished, ie it is only from the naming of the node and
thus only recognizable for humans [78]. The networks follow the principle of object identity,
ie each object is only mapped once, so that the information about it
object are available centrally in one place [192]. Due to the possibility of graphic
Representation, semantic networks are easily interpretable for humans. A disadvantage
about semantic networks is that it cannot be checked whether they are consistent [193].
In addition, the representation of complex issues is very time-consuming, since many
nodes and relations are required; The search process is correspondingly complex [194].

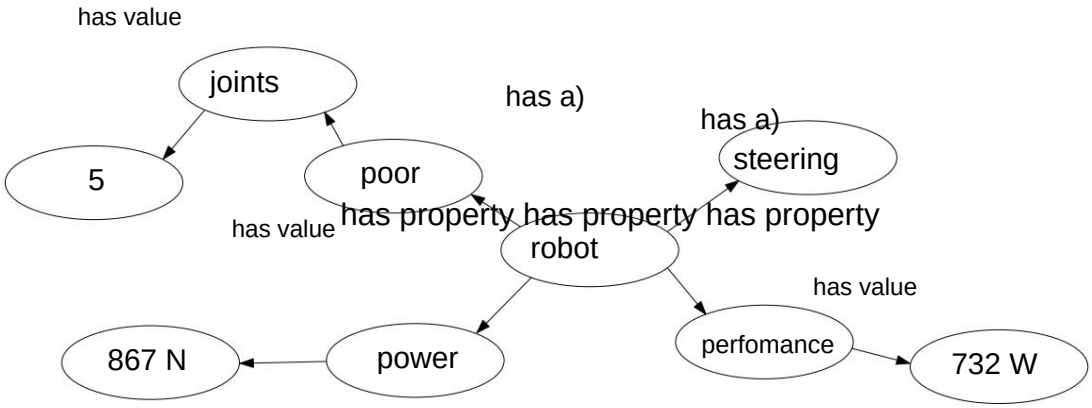

Figure 5.4: Representation of a semantic network, representation based on [78]

Semantic networks are not uniformly defined in the literature. There are some extensions
that increase the expressiveness, e.g. B. a so-called fact network that through the
Introduction of terms arises [192]. Ontologies can also be used as extensions
be understood by semantic networks [185]. In this work, the formalisms
however, be considered in their original form, which is why the ones presented above
Definitions from [78] and [185] are followed.

frames

Marvin Minsky introduced the concept of frames in 1974 [195]. Frames are built similarly
to semantic networks; however, the nodes are referred to as frames and it
relations between these can be defined [194]. Contrasted with knots are
Frames constructed according to a specific schema, where a schema provides knowledge
about an individual, an object, an event or an action through certain properties
so -called *slots,* represents [196]. A specific person is realized as a frame, for example.
The underlying scheme contains e.g. B. the slots *name, first name* and *date of birth.* Like the
In semantic networks, the frames are often named using unique identifiers
(ID's) [78]. “Metaframes” can also be defined, ie terms that define a specific scheme and
from which specific objects can be derived [78]. It

however, there is no underlying logic, so no consistency check is possible and Conclusions can only be drawn using heuristic methods [195]. Therefore Frames are often used with formalisms for the conclusion, e.g. B. rules, to hybrids systems expanded. However, this makes modeling more difficult and more similar a programming language as a language for knowledge modeling [197].

rule-based systems

A rule consists of a premise and a conclusion [41]. applies the conclusion whenever the premise is true. It is also said that the *rule fires* [41]. The knowledge base of these systems consists of objects and rules, that depict the abstract knowledge. With an application, the concrete knowledge is added, ie case-specific values that are assigned to the objects or variables [41]. for efficient procedures are in place for processing. So e.g. B. complex rules in split into several simpler rules to ensure efficient processing [41].
Many expert systems have been implemented by rule-based systems, such as e.g. B. the Credit Clearing House (CCH) for granting loans [198]. The advantage of the formalism is that people are familiar with rules and express many facts about them to let. In addition, a description of the facts is sufficient, it does not have to be an in-depth one understanding of the entire domain. A disadvantage is the large number of rules required, which makes it difficult to create and maintain the knowledge base. That's how it is the CCH from over 800 rules [198].

Comparison of modeling formalisms

In this section, the four formalisms presented are compared with each other, see Table 5.1. The requirements defined in Chapter 3 serve as the basis for the comparison. An X means the requirement is met, an (X) means it can be fulfilled with justifiable additional effort. If no entry has been made, this requirement cannot be achieved with reasonable effort.

Semantic networks meet many requirements, with some requirements, e.g. B. Hierarchy, versioning or multilingual terms, can only be fulfilled by modeling can and cannot be directly supported by the formalism. Frames also meet many requirements and also provide for time references as the only formalism. However they are not based on logic, so that, as with semantic networks, no consistency check is possible and conclusions can only be drawn heuristically. Rule-based systems are not suitable for this task, many requirements are not fulfilled or can only be realized with great effort. Ontologies fulfill the Most requirements, only time references are not provided, but can also be through appropriate modelling. Therefore, the ontologies are considered more appropriate modeling formalism chosen and used in this work.

Table 5.1: Comparison of modeling formalisms according to the requirements of Chapter 3.

requirement	semantic nets	Frames	rule-based systems	ontologies
Unique terms	X	X		X
hierarchy	(x)	X		X
Numeric Values	X	X	X	X
time references	(x)	X		(x)
Meta & instance model		X		X
Algorithm Description	X	X	X	X
data access	X	X	X	X
model access	X	X	X	X
overall model	X	X		X
expandability	X	X		X
versioning	X	X		X
Multilingual	X	X		X
reusability	X	X		X
consistency check			X	X
Conclusions	(x)	(X)X		X

5.1.2 Choosing an Appropriate Representation

There are many ontology languages that can be used to represent ontologies. Therefore, an appropriate ontology language is selected in this section.
Since not all languages can be presented and evaluated, Figure 5.5 a chronological overview of the development of different ontology languages is given, the picture does not claim to be complete. Many languages can be excluded from the outset due to dependencies between the ontologies. CycleL was the first ontology language and was presented in 1990 [199]. It was the language for the Cyc project, which wanted to formally map world knowledge [200]. First based CycL on a frame language, but the need arose to be able to represent more complex relationships, which is why it was then redesigned with higher-level logic gave [200]. In the years that followed, the *Knowledge Interchange Format (KIF)* was based on of first-order predicate logic, which can exchange knowledge between heterogeneous information systems [199]. Ontolingua is based on KIF and was the first Language that could exchange ontologies between systems. In addition, can Ontolingua generate machine-readable documents in the form of hypertext [201]. Operational Conceptual Modeling Language (OCML) is an extension of Ontolingua is more functional [202]. The Frame Logic (FLogic) integrates a frame-based approach with predicate logic [202]. Unlike previous frame-based languages, it has a proprietary syntax that is not Lisp-like [199]. LOOM is a description logic-based language for creating knowledge bases, e.g. B. in expert systems [202].
LOOM was a new development and enjoyed great popularity [203].

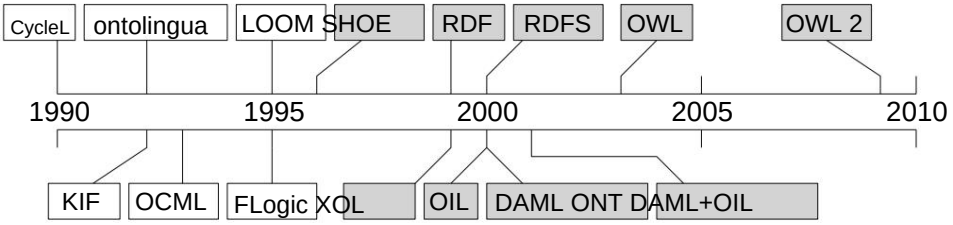

Figure 5.5: Temporal overview of the development of ontology languages based on
to [199]. Web-based ontology languages have a gray background.

From 1996, the web-based ontology languages became more popular. SHOE (SimpleHTML
Ontology Extension) was the first of these ontology languages and was developed as an extension of
HTML developed [199]. It was the goal, machine-readable knowledge in hypertext markup
Language (HTML) documents [202]. The other featured languages
based on XML. XML-Based Ontology Exchange Language (XOL) was developed as an
exchange format and not for creating ontologies [202]. ontology
Interchange Language (OIL) is a language for representation and exchange
of ontologies, so that no separate exchange format is necessary [202].
The language defines a formal semantics and enables efficient inferences,
similar to those in the description logic [204]. DARPA Agent Markup Language Ontology
(DAML-ONT) was developed almost parallel to OIL in the USA, but there
there are many similarities between the languages [205]. For this reason, a joint working
group was set up, which includes the two languages DAML-ONT and OIL
DAML+OIL summarized [206]. The result was used as a proposal for the basis of
OWL submitted to the W3C [206]. OWL is a further development of DAML+OIL
and thus very similar to this language [204]. As a further development of OWL
developed the already presented OWL 2, which has a greater expressiveness and
can reason more efficiently. It can be seen that many ontology languages are linked to one
another. Due to the dependencies of the languages among each other,
The languages OCML, LOOM and OWL are considered in more detail below.

All three considered ontology languages could be used to model the semantic model. To
select the ontology language that is most suitable,
Five characteristics are defined: *Application Programming Interface (API), Import of
Ontologies, Expressiveness, Editor* and *Community.* The characteristics are not
requirements, but fulfilling them simplifies the application. Via an API, the
Formalism were more easily integrated into a software, which represents a significant work
simplification. If the language can import another ontology,
it supports the creation of a new model. A high level of expressiveness expands the
modeling options, which simplifies the creation of the model. A
Editor for creating an ontology also supports modeling. an active one
On the one hand, community ensures that the ontology language is up to date
remains, both in the tools and in the further development of the language itself
others, with wider use, more skilled workers can be found and the
Community supports any questions.

A comparison of the different ontology languages is shown in Table 5.2. Included
an X represents that the characteristic is completely fulfilled, an (X) that the characteristic is partially

is fulfilled and if the field is empty, the characteristic is not fulfilled. For OCML
no API could be found. An API is for LOOM for the programming languages
Lisp and Java available, but it is unknown which Java versions are supported
become [207]. APIs for OWL are available in all common programming languages. A
Import of other ontologies is possible with all three languages. The power of expression is
Sufficient for all ontologies, with LOOM and OCML having a slightly greater expressiveness than OWL
[208, 209]. No editor could be found for OCML that
known WebOnto Editor is no longer available. LOOM ontologies can be created in On toSaurus [208]. The
editor is available online, however that was the last one
Update in 2010. Accordingly, the effort is somewhat greater to update the editor
getting systems up and running. With Protégé, a well-known and up-to-date ontology editor is available for
OWL. An active community only exists at OWL, this
was determined based on search queries. For this purpose, certain words were searched for
in the Stackoverflow forum in the same way as with an Internet search engine. At stackoverflow
there were no hits for the search *terms OCML Ontology* and one hit for *LOOM Ontology*
and found 3,097 hits at *OWL Ontology* . Even when searching with search engines,
with and without a time limit on the results of one year, there was always OWL
significantly more hits. Therefore it can be concluded that only OWL has an active
community has.

Table 5.2: Comparison of different ontology languages.

	OCML	LOOM	OWL
API		(X)	X
import	X	X	X
Expressiveness	X	X	(x)
editor		(X)	X
community			X

From the table it can be seen that OWL fulfills most of the characteristics. only at
expressiveness, OWL is inferior to the other two languages. There's one more
greater expressiveness requires more time for conclusions and OWL already
offers sufficient expressiveness, OWL is very good for use in the
domain appropriate.

However, OWL offers different profiles, which were presented in section 4.2.2 and
from which to choose the one that is most suitable. The differences lie in the expressiveness. With
increasing expressiveness, the time increases
needed to calculate the conclusions, up to undecidability. In the
The domain considered here must be able to be decided so that queries and
Tasks can be carried out safely and reliably. On the other hand must
Diverse and complex knowledge is presented to solve the use cases,
a high level of expressiveness is therefore required. If you choose too little expressiveness, the terms and
relations cannot be defined exactly, which means that
ambiguities or auxiliary constructions that are difficult to understand. Therefore
the modeling constructs are presented that are most important for the use cases

ten in order to then be able to select the best profile. The following are the
different modeling constructs are presented and explained which ones are relevant for this work
are important.

To create the semantic model, a hierarchy (modeling *element Sub ClassOf)* is elementary
and is required in all use cases. through the hierarchy
the model is structured and can be used much more easily because it
There are hyponyms and hyperonyms. The modeling of equivalent terms defined
necessary and sufficient conditions, so that these limit the possibilities of reasoning
extend. This is required for the description of the calculation rules in application 1 and for
the assignment of the anomalies in application 3, among other things. the
Property of the disjointness of concepts expresses that there is between the concepts
there are no divided individuals. In addition to understanding, this helps in particular to
maintain consistency, which is why the construct is used in all applications
finds. Increase the disjointness extensions *(DisjointUnion* and *DisjointClasses) .*
the expressiveness and are therefore not necessary.

Enumerations only play a subordinate role in this work, in favor of one
better extensibility hierarchies of terms can be used. The intersection and union sets are
commonly used in all use cases, e.g. B. to define the
KPIs. If these modeling constructs were dispensed with, not all the connections between
the use cases could be expressed in a comprehensible manner. The definition of the
complement, on the other hand, is less important for the use cases considered. Although it can
Complements can be used to draw conclusions, but the relationships are too complex for
the applications to exclude certain properties
allows for categorization.

The existential and universal quantifiers are required in all applications. The cardinality is
required in use cases 1 and 2. The exact cardinality is am
most important, since terms in the domain usually have an exact definition. minimal
and maximum cardinality are usually less suitable for this. The term description
based on a value *(HasValue)* is useful for use cases 2 and 3, however
the connections can also be expressed differently, so that this construct as
is considered optional.

In summary, it can be stated for the description of terms that the modeling *constructs*
SubClassOf, EquivalentClass, IntersectionOf, UnionOf, AllValuesFrom, SomeValuesFrom
and *Cardinality* are required for the considered use cases,
to create a semantic model. *DisjointWith, Enumeration, MinCardinality* and
MaxCardinality could be used since alternative modeling is possible,
however, its use is optional.

The requirements of the use cases for the modeling formalism are compared with RDFS
and the OWL profiles in Table 5.3. The top five rows of the table
describe conceptual axioms that can be used to define concepts. the
middle four rows are applied to concepts or individuals. The bottom six lines are operators
applied to relations to form concepts
describe. For a better overview, the modeling constructs identified as necessary are shown
in italics. An X means that the construct in the profile does not

Table 5.3: Comparison of expressiveness in term descriptions of RDFS and OWL profiles. X means that the construct can be used, ÿ means that at there are restrictions on the use of this construct. The modeling constructs most relevant to the use cases are shown in italics.

	RDFS	QL	EL	DL	example
SubClassOf	XXXX				sensor v device
Equivalent Class		XXX			device ÿ sensor or ... or actuator
DisjointWith		XXX			actuator ÿ sensor
DisjointUnion				X	device(sensor, ... , actuator)
Disjoint Classes		XXX			{sensor, ... , actuator}
enumeration				X	binary ÿ{true,false}
IntersectionOf		ÿ	XX		interval = hasStart u hasEnd
UnionOf				X	time object ÿ time t interval
ComplementOf		'		X	discrete = ¬ continuous
AllValuesFrom				X	pump = ÿ input.liquid
SomeValuesFrom		ÿ	XX		modules = ÿ haspart.device
MaxCardinality				X	input = product max 3
MinCardinality				X	combined power = power min 2
Cardinality				X	hatStartTime = 1 xsd:dateTime
HasValue			XX		PLC = security level hasValue "S1"

restricts can be used, ÿ means that with this construct restrictions present.

In the OWL 2 profile RL, constructs may only be used in certain contexts will. As a result, the selected representation would not be meaningful, which is why a representation is omitted. Due to the insufficient expressiveness, the suitability of the RL profile can be excluded. There are only three constructs in the QL profile restricted in a certain context, so that the table is sufficiently meaningful Has. In the case of the DL profile, global restrictions must be fulfilled in order to comply with the formalism to keep determinable. For example, certain pairs of properties may not be shared occur in relations, e.g. B. are a cardinality together with a transitivity in not allowed in a relation. Since these restrictions are very diverse and the ones for them Use cases identified constructs do not include their representation omitted in the table. Details on the restrictions can be found in [112].

When evaluating Table 5.3, the different levels of expression are clear recognize. RDFS only supports a term hierarchy (subClassOf), but no further descriptions. The OWL Profiles QL and EL are related to the definition of terms relatively similar, but have limitations in that they cannot represent union sets, all quantifiers (AllValuesFrom), and cardinalities. OWL DL supported

on the other hand, all modeling constructs. The following considers the possibility of how Relations and attributes can be described.

The same applies to the descriptions of relations as to the descriptions of terms. A hierarchy helps to structure the relations and should be present.
To describe the meaning of a relation and to draw conclusions are
the definition and target area are important. Therefore, these cannot be dispensed with.
Equivalent relations are not required for the use cases. Typically
they are used when merging different ontologies. Disjoint relations occur in the use cases, but only in small numbers. Furthermore
their relevance can be reduced by the modeling, so that the construct does not
must necessarily be available. On the other hand, inverse relations occur more frequently before, e.g. B. *ispartOf* is the inverse of *haspart.* This significantly reduces the modeling effort and increases the expressiveness, since many new
relations can be inferred. This is especially for the topological
Information required in use case 2. Chained relations *(property chains)*
are not required for the applications.

When it comes to the properties of the relations, the functional and the transitive properties stand out because they are required in all application cases. A functional relation can only be defined once for each individual and is e.g. for *has start time, has status ,* or *has* used unit. The transitivity becomes for relations like
e.g. B. *is part of* or *supplies product after* used. The remaining property values are not required for the use cases.

Table 5.4 compares RDFS and the OWL profiles with regard to the expressiveness of relations. The relations can be divided into relations (object properties), attributes (data properties) and comment relations (annotation properties).
Comment relations tie a comment to a term, relation, or
Individual. Since comments are possible in all profiles, the options for specifying this relation are not considered further. An attribute assigns a term
assigns a value to a data type or to an individual. However, the attributes can, themselves at greatest expressiveness, specified only by the first five axioms from Table 5.4
are defined as functional. Only the relations can be changed through all in
axioms shown in Table 5.4. Therefore, the following relate
remarks on the relations.

The upper seven axioms describe a relation by linking it to terms, e.g. B. definition or target area, or by other relations, z. B. Equivalent or
disjoint. The lower seven axioms describe a relation by a property
which may or may not exist, ie it is expressed via a logical value.
RDFS again stands out due to the very limited expressiveness. Only one can
Relation hierarchy as well as a definition and a target area are defined. In which
In addition, profile QL can specify equivalent, inverse and disjoint properties
become, just like symmetric, asymmetric, reflexive and irreflexive relations.
The EL profile is somewhat more restricted compared to the QL profile, since only the properties of transitivity and reflexivity can be expressed. However, you can
property chains can also be used. OWL DL can represent all constructs.

Table 5.4: Comparison of expressiveness in relation descriptions of RDFS and OWL profiles. X means the construct can be used. The modeling constructs most relevant to the use cases are shown in italics.

	RDFS	QL	EL	DL	example
SubProperty	XXXX				mathOperation add v
domain	XXXX				domain(haspart): module
Range	XXXX				Range(has part): device
equivalent		XXX			haspart ÿ consistsof
disjointed		X			X Disjoint(hasStart, hasEnd)
InverseOf		X			X hasPart = isPartOfÿ1
property chain			XX		Prod.Counter = counts ÿ partOf
Functional					X has status
inverse functional					X uses Resource
transitives			XX		is part of
Symmetric		X			X is connected to
asymmetric		X			X caused
Reflexive		XXX			equals (is equal to itself)
Irreflexives		X			X caused

The comparison has shown that RDFS has far too little expressive power for the use cases processed here. The QL profile can also do many things
do not depict necessary constructs, both on the conceptual and on the relational level.
No advantage is achieved by optimizing for larger ABoxes either. The semantic model of production plants has neither particularly large TBoxes nor a large one
ABox, so that a low memory requirement or the guaranteed maximum required memory is of no benefit. The QL profile limits the expressiveness and thus the
Modeling relatively strong, so it is not suitable. The EL profile has a
similar expressiveness as the QL profile, ie here too some relevant modeling constructs are missing. In addition, the use case of a large TBox is in this work
not given, which does not exclude the application, but also not for the use
of the profile speaks. DL supports all modeling constructs, so that many relationships can be expressed simply, i.e. using the appropriate construct. In addition
the DL profile is decidable, so all conclusions are drawn in finite time
can become. It is true that this increases the complexity of drawing conclusions compared to the other profiles, but as shown in [210] it is also possible for very large ones
DL ontologies (>100,000 axioms) to calculate conclusions in a reasonable time.
The DL profile is therefore suitable because it provides many modeling constructs
and modeling can be carried out with corresponding precision.

OWL Full has not been taken into account in the analysis so far, but it also fulfills all of them
requirements of the modeling and does not have the above-mentioned global restriction

functions like OWL DL. However, the restrictions on the considered
Use cases no restrictions. One reason against OWL Full is that the profile
is undecidable. This makes OWL Full unsuitable for industrial applications because
where reliable availability of information is expected. OWL DL has enough
expressiveness and is decidable, so it is used in this work.

To answer FF 1, four formalisms were used in this section based on the in
Requirements defined in Chapter 3 are compared with each other. It has been shown that by
The ontologies are best suited to the formalisms examined and meet all requirements.
Three languages were used to select a suitable ontology language
compared with each other and OWL identified as a suitable language. Within OWL
There are different profiles, of which the DL profile best meets the requirements and
is therefore used in this work to create the semantic model.

5.2 Model Creation

In this section, the semantic models for the three use cases are presented using the most
important modeling decisions. The methodology used
for model creation was introduced in chapter 4.4. In addition, results were sought
Publications that were created as part of this doctorate [18], [19] and [20], are used. FF 2 is
answered with the presentation of the models. In chapter 5.2.4
examines whether the combination of the models created is an approach to creating a
cross-application model. This answers FF 3.

5.2.1 Semantic Model - User Interaction

In the operation of a production plant, characteristic values, so-called KPIs, are often used to
to check the achievement of the mostly business goals. The supervision
of the KPIs is done by the user, who often still does this manually in modular systems
must calculate. If KPIs are already calculated automatically, the calculation must
typically adjusted with each process change. The use case presented here aims to ensure
that KPIs are described uniformly and generically, so that
these can be determined automatically, even if the topology of a modular production system
changes. This requirement for the production plant is z. B. described in [182]. Specifically,
the terms *total power consumption, total energy consumption,*
Total production of the day, throughput per hour and *OEE* using the semantic
model are determined. In the publication [20], approaches for modeling KPIs were already
presented as part of this dissertation, which are expanded and expanded below
be generalized. Before modeling the above terms are presented
the modeling of units of measurement and time is discussed. The units of measure
and times are basic terms that are necessary for the definition of many terms in different
use cases.

Modeling question: How can units of measurement be represented?

Units of measurement (in short: units) are necessary for the use of values, since these determine how the values are interpreted. Units allow scaling Values and an adjustment to industries or regions, e.g. B. to the Anglo-American measurement system. This work is not about tracing the units shown back to SI units, but a practical representation is required that presents the units in an understandable way and a simple conversion between different units allows.

The well-known QUDT ontology is used to represent units. This was from the NASA is developing and is supposed to standardize the representation of units in the *Semantic Web* . This is achieved through a great depth of modeling that tries to cover many subject areas to unite. It is also possible to convert units via an appropriately formulated query [176]. However, the exploitation of this ontology has proven to be quite difficult proven. This is because different ontologies have to be imported, e.g. T. not load correctly or cause a version conflict. This can possibly be attributed to the fact that the second version of the ontology is currently being rolled out will [168]. In addition to the QUDT, there are many other ontologies that represent units. An evaluation of different ontologies for representing units has been published in found inconsistencies in all examined ontologies [211].

The use of the ontologies described above is not optimal for the use cases described here, since the objectives are different. Your use would have to Consequence that only a small part of z. T. complex ontologies would be used and yet not all relevant aspects are covered. In particular refer the existing ontologies to physical units that can be traced back to SI units. However, in the field of production facilities, there are many auxiliary units. Values that are actually unitless, such as B. *percent* or *piece* get Assigned auxiliary units to make them interpretable and scalable. Therefore will without using the existing ontologies and modeling its own system of units with the aspects relevant to the domain.

An additional hierarchy level is introduced in the meta model between the general term *unit* of measurement and the concrete units, which categorizes the units represents, e.g. B. Temperature unit. All units of the same category can be converted into each other if this is possible with a linear transformation. To exactly one unit is defined as the reference unit, ie a transformation is always carried out via this unit, similar to a rule of three. A value is first entered into the Reference unit converted by applying formula 5.1. *W ertR* is the value in reference unit, *valuec* is the value in output unit, *Oc1* and *Oc2* are offset values and *Kc* is the coefficient for conversion to the reference unit. If the If the reference unit is not to be the final unit, this can be calculated using formula 5.2 will. This procedure is similar to that in the QUDT [176]. Deviating here However, two offset values are defined in order to reduce the restrictions on the conversion ren.

$$valueR = (valuec + Oc1) * Kc + Oc2 \quad (5.1)$$

$$valuec = \frac{ValueR \ddot{y} Oc2}{Kc} \ddot{y} Oc1 \quad (5.2)$$

The modeling of the units as a UML class diagram is shown in Figure 5.6. It consists of a three-level hierarchy. At the top level, the defined term is unit of *measurement.* On the second level, the units are divided into different
Sorted by categories, only three categories are shown here. Any category
has a reference entity that is defined by an attribute and used for conversion
will. At the third level, each unit is defined by a term using the spoken name as an identifier, e.g. B. *degreesCelsius.* To make it easier to understand, the name is written out in full and given an attribute *has unit symbol,*
which represents the unit symbol. The unit symbol is used, among other things, in the display for uses the user, e.g. B. °C for the temperature. As described above, everyone can
other units a conversion *coefficient Kc* and up to two conversion offsets
Oc1, *Oc2* which are also defined as attributes. Normally one would expect that the units would be defined as an individual and not as a concept. The advantage
when defined by a term is that the units are uniquely referenceable
and can therefore also be used in other definitions of terms. Through the so-called
Punning can be created individuals with the same name. This allows them
Units at both concept and individual level with the same identifier
be referenced so that unrestricted use at the individual level remains possible. Another advantage of this modeling is the clear separation of meta and
Plant model, since the units are defined as concepts, ie in the meta model.

To represent units, the semantic model is given a hierarchical structure. A value, e.g. B. for a temperature, is determined in an individual by the
Attribute *has value* represented. The connection to the unit is made through the relation *has unity.* With this modeling, values can be represented, to which units
are assigned. Furthermore, a conversion between units is possible, and the
Units can be used at both concept and individual level. In formations that are not required in industrial practice, such. B. Repatriation
on SI units, has been omitted. In the model, 31 terms were used as an example
created for this modelling. Because the amount of units is very large and through
the conversion can also integrate new units, it is sufficient if
the required units are shown. In addition, a relation and the five
Attributes *hasunit_sign, default_unit , conversion_offset1,* conversion_offset2
and *conversion coefficient* defined.

Modeling question: How can times be represented?

The representation of times is important in the area of production plants, since all events have a time reference, e.g. B. the energy consumption or the throughput. Without

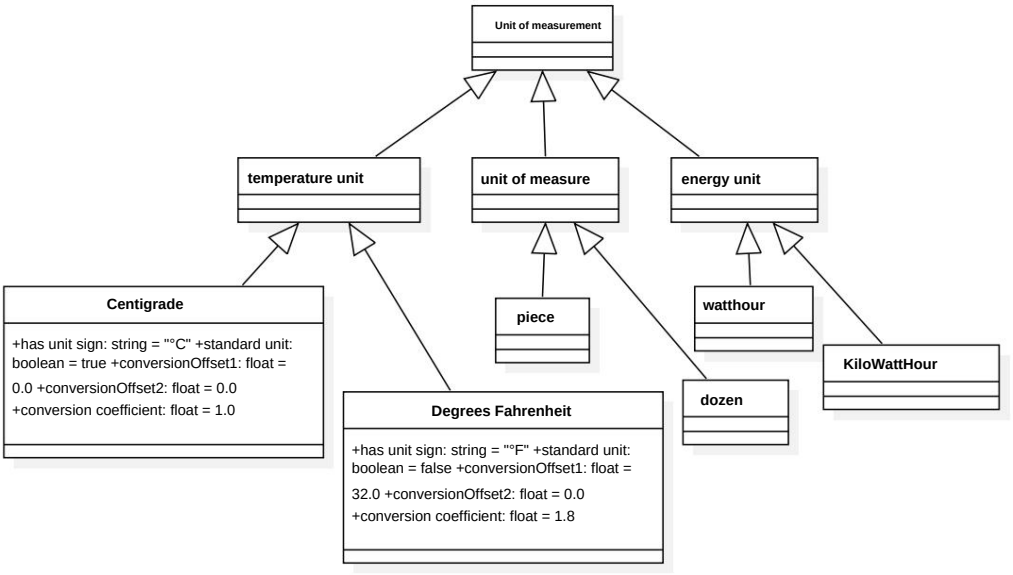

Figure 5.6: Excerpt from the unit hierarchy, with the two temperature units are shown in full.

the values are mostly useless for a time reference. The time reference can be specified by a point in time, e.g. B. at a measured value, or by a time interval, such. B. in energy consumption. Since no explicit concept for the use of times is provided in the chosen formalism, this must be covered by the modelling.

In the area of production plants, points in time and time intervals must be shown. First, the modeling of points in time is considered. For this purpose, the *W3C Time Ontology* (German: Zeitontologie) is used, which represents temporal relationships in OWL and is a W3C Recommendation, ie a standard of the W3C [169]. Since the time ontology is intended to be applicable to many domains and use cases, it is kept very general. In this work, the models relevant to the domain are selected and integrated using the IRIs. As described in chapter 4.4.2, the ontology is not imported.

In this work, a simple, clear and practical indication of the time should be used. This means that not all time-related terms have to be represented, but only those that are commonly used. So e.g. For example, the specification of a calendar system can be dispensed with without restricting applicability in the domain. The date, the time and the time zone are required so that a clear time specification is possible. The *xsd:dateTime* format, which can be defined as an attribute in an ontology, is ideal for this mapping . This format represents a date and time separated by a *T* according to the ISO 8601 standard , e.g. e.g. 2019-07-06T16:45:12. Optionally, the deviation from Coordinated Universal Time (UTC), ie the time zone, can be specified. Here, too, the ISO 8601 standard is followed, which specifies the notation by specifying signs, hours and minutes, e.g. e.g. 2019-07-06T16:45:12+01:00. If this specification is omitted, the time zone of the processing system is assumed. In many cases a correct result is achieved with this. However, since this can lead to misinterpretations, it is recommended that

use full disclosure. In the examples of this work, however, due to
mostly dispensed with due to the limited display possibilities.

The modeling of time intervals is discussed below. It will be one
Subdivision into fixed and variable time intervals. Fixed intervals, e.g. Legs
Hour, represent a running time window, ie a fixed observation period of always one hour.
This will e.g. B. required when calculating the throughput.
Variable time intervals correspond to the intervals already considered, ie they become
started at a time and the length of the interval will increase as time progresses
greater. Therefore, the time of the variable intervals is only modeled with a start time, e.g.
B. to determine the production of the current day. If an end time
is added, it becomes a fixed time interval. The distinction of the two
Interval types are not described in the time ontology, but represent a compatible extension.

Figure 5.7 shows three modeling variants for fixed time intervals. A
The time interval is represented by two points in time, see [170]. Hereinafter
three modeling variants are explained:

a) The time can be modeled by the two attributes *hasStartTime* and *hasEndTime*
will. A time attribute is assigned to an existing term, ie for
an interval has an attribute *start time* and *has end time.* The presentation is clear,
compact and the duration can be easily calculated. The downside, however, is that
the definition of time-related terms is not possible correctly. To e.g. B. the
Defining throughput at the conceptual level requires time as a referenceable element
be shown, but attributes are not. This also ensures compatibility
of the approach is not given and the expandability is severely restricted.

b) An alternative is to create a term *FixedTimeInterval.* The term re presents exactly a
fixed time interval with a defined start and end by means of attributes
endpoint. Each concrete time interval is thus represented by an individual. The
advantage is that the intervals are also explicit in the other definitions
let use.

c) Another possible representation is the definition of an interval by two points in time.
In contrast to variant b), the specification of the point in time
moved from attribute to relation. This allows times to be more detailed
be specified (extensibility). However, there is no explicit distinction between points
in time and time intervals, which makes it difficult to use. In addition, more individuals
must be created overall.

Table 5.5 shows an evaluation of the three modeling variants. The six
The evaluation criteria used were introduced in chapter 4.4.2. When evaluating
an X means the requirement is met, an (X) means the requirement
is not completely fulfilled or is better fulfilled by another variant and a free field
means that the requirement is not met or is met to a limited extent. All variations
provide a clear representation of the time and are easy to understand. Most compact
is variant **a),** while **c)** is the least compact and entails the greatest modeling effort. The
compatibility with the time ontology is in variant **c)**
fully given, while for variant **b)** the attribute (in XSD Date-Time-Stamp)

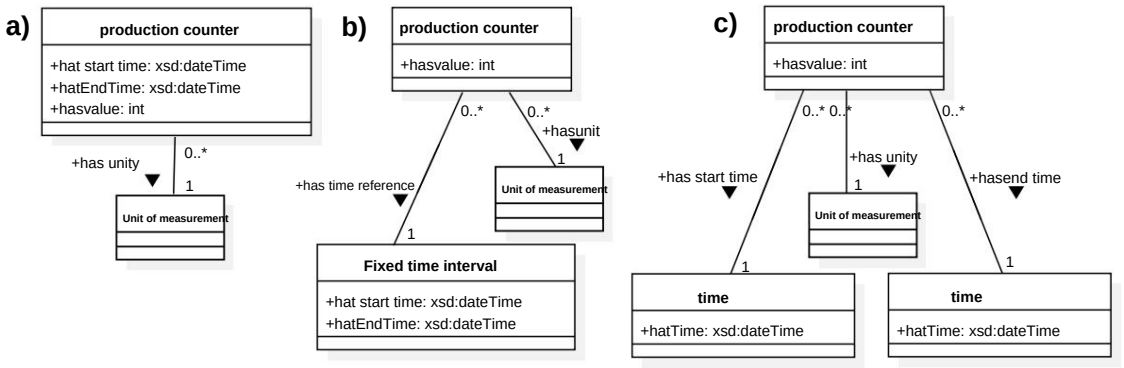

Figure 5.7: Comparison of three modeling approaches of the meta model at the moment.

must be further subdivided in order to be able to distinguish *start time* and *end time .* the However, compatibility is retained. Variant **a)** is not compatible and does not allow other terms to refer to the time. In addition, this variant is not expandable, while variants **b)** and **c)** can be expanded. With the restrictions described, the first variant is not appropriate for the central concept of the time, the other variants are judged as appropriate. In particular by the explicit distinction between point in time and time interval is the modeling of the variant **b)** more suitable and is therefore used in this work.

Table 5.5: Comparison of options for modeling time.

	a)	**b)**	**c)**
Uniqueness	XXX		
intelligibility	XXX		
Compactness	X	(X)	
compatibility		(X)	X
expandability		X	X
appropriateness		X	

If a fixed time interval is described, so that there is a start and an end time, the modeling presented is suitable. If, on the other hand, a continuous interval is described, e.g. B. for the throughput of the last hour, the start and end times would have to be constantly updated. For this purpose, modeling is available durations as used in time ontology. This defines attributes for *second, minute, hour, day, week, month* and *year* [169], meaning points in time or Time intervals can be described. For a description of points in time, this is cumbersome, but well suited for modeling a period of time. In order for the modeling to fit into the overall concept, the time specifications are given in *seconds, minutes, hours, Day, week, month* and *year* are defined as sub-terms of units of measurement and not as attribute as is the case in time ontology. This modeling consequently follows the Use of values through units of measure, is compact, extensible and understandable.

In summary, Figure 5.8 shows all three possible representations of the time intervals in this work in object diagrams using a production counter as an example.

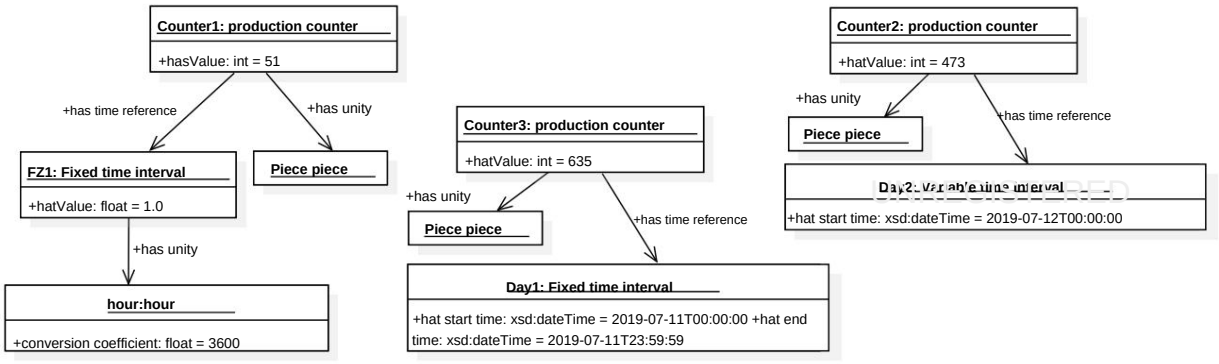

Figure 5.8: Subdividing the time intervals results in the three options shown for modeling the time, shown as a system model using production counters as an example.

In the object diagram, the notation of the header is: *individual name: concept name.* Attributes (data properties) are shown below, in the notation *name: data type = value.* Relations are shown as links, e.g. B. *has unit* or *has time reference.* The punning is easy to recognize in the units *piece: piece* and *hour: hour ,* the individuals and the terms have the same names. A fixed time interval is shown on the left using a value and the unit *hour .* A fixed time interval is also shown in the middle, but since it has already been completed, two attributes define the start and end time. A variable time interval is shown on the right, which only has a starting time and is therefore constantly increasing.

The concepts presented here can clearly represent points in time and time intervals and can be assigned to other terms. For this purpose, the terms time object are created in the term *hierarchy* as a generic term for point in *time* and *time interval* with the sub-terms fixed time *interval* and *variable* time interval . The relation *has time reference* is defined and the two attributes *has start time* and *has end time.* In this way, the time can be represented by a few newly inserted elements, since the units already defined are used.

Modeling question: How can a production counter be implemented?

This modeling question relates to how a meter can be described in the meta model. The production counter was used as an example. However, the concept can also be transferred to other meters, e.g. B. on a quality sensor that counts the good products. The production counter is a characteristic value of the production plant that is important for many areas of the company, e.g. B. Operators, maintenance personnel, purchasing, sales and the management level. One challenge is the modeling of the boundary condition and its integration into the model, e.g. B. the signal type or the position of the detecting sensor in the module. The topology is described in the model, but it is not sufficient to decide whether the sensor can count. Some sensors are installed to monitor the process or to detect error states. However, only sensors that are used to control the production plant and the product flow can usually be used for counting

to capture. This distinction is not made in the topology, so there is one
expansion required.

Counters can only be modeled using a relation, since
a counter is to be linked to a device. A realization by attributes is
unsuitable, as it is not possible to link the attributes with time, but for
a counter would be necessary. Therefore, only one solution variant is presented. Included
the term *production* counter is defined in the semantic model and linked to a sensor that
counts via the relation . The value of the sensor is used for counting.
In the case of binary sensors, the rising edges are counted, e.g. B. in an inductive proximity
sensor. With a continuous sensor, e.g. B. a scale, a
decreasing difference is measured, ie whenever the scale gets lighter, the amount is
counted as output. In order to show the different units and counting methods,
a distinction is made between a continuous and a discrete production counter. This makes
the information easier to use and more robust.

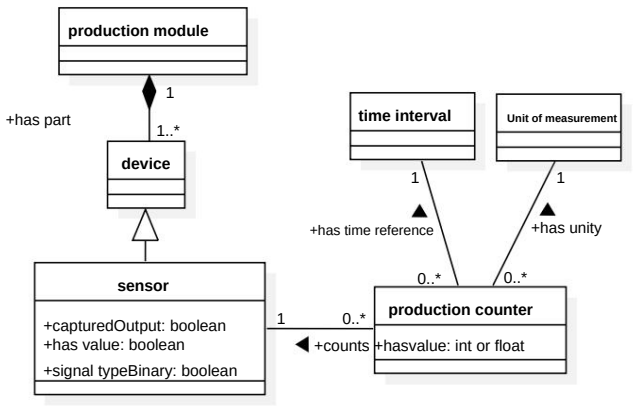

Figure 5.9: The semantic model for a production counter

Figure 5.9 shows the modeling of a production counter, the generic term of
continuous and discrete production counters. A production module consists of devices, the
generic term for sensors. Sensors can be described by the attributes has *value, signal type
binary* and *recorded output ,* among other things . About that *has value*
Attribute represents the current signal value of the sensor. Sensors can
Return value of data type boolean, integer or float. A binary signal type becomes explicit
represented by an attribute *signaltypeBinary* . The attribute *recordsOutput also* explicitly
specifies whether the sensor detects the output, ie the product stream, of the module
detected. This specification supplements the missing information so that a decision can be
made as to whether a sensor is suitable for counting. The production counter is
connected to the sensor via the relation *counts* . Each production counter is exactly one
assigned to a sensor; however, one sensor can be used by multiple production counters
become, e.g. B. for different time intervals. Furthermore, the production counter
an appropriate time interval and unit are assigned to make the value unique
can be interpreted. An example of a concrete production counter is shown in Figure 5.10. In
the example, an inductive sensor counts the filled bottles
in a filling module. The counter determines the daily production of the system.

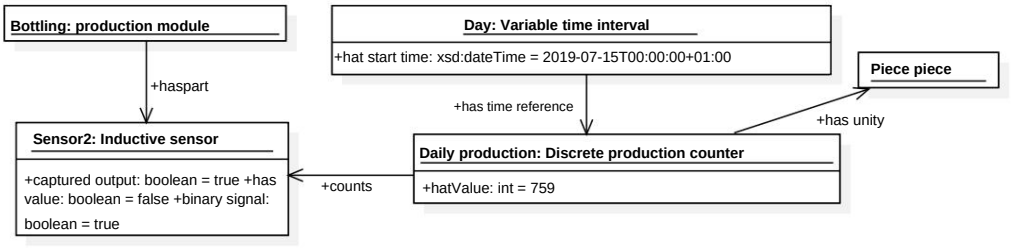

Figure 5.10: The specific system model for a production meter, as an example implementation of the semantic model from Figure 5.9.

To implement counters, the relation *counts* was added to the semantic model, as well as the term *production* counter with the distinctions *discrete production counter* and *continuous production counter.* boundary conditions, e.g. B. whether a sensor records the output, were represented by attributes, since they are easy to use and new boundary conditions can be easily added. If the necessary information is available, the attributes can be set automatically, e.g. B. by SWRL rules.

Modeling question: How can calculation rules be represented?

The operator of a production plant is often interested in KPIs and not just simple values such as e.g. B. Sensor values displayed by the system. Calculation rules must be stored in the semantic model so that KPIs can also be determined automatically after a change in the system topology. The challenge is that the formalism only provides two-digit relations, which are not sufficient for direct modeling of calculation rules. While there are many ontologies dealing with mathematics, e.g. B. OntoMathPro [212], MaMo [213] and OpenMath [214], but these mostly serve to define a general vocabulary and to structure the field of mathematics. The modeling of calculation rules is not intended in these ontologies and is therefore newly developed in this work.

The four basic arithmetic operations are required for the considered applications. As described above, the relations are two-digit, ie they can only combine one concept with production *another ,* not with several. Should e.g. B. Throughput = Fixed Time Interval total are calculated, the operands must be assigned to the throughput with a relation. The relation must represent the position of the operand in the equation so that the calculation rule can be determined from it. Three modeling options are presented below, two of which are shown in Figure 5.11. The third variant is implemented using SWRL rules and cannot be represented in UML.

a) In this variant, four relations *add, subtract, multiply* and *divide* were created. If the commutative law applies, i.e. the order of the operands does not matter, any number of operands can be processed, e.g. B. an addition

Inventory = +5 -3 +7. In the plant model, an individual is represented by the relations with the numbers (or in the model, they are the assigned values of two individuals) connected. If the commutative property does not hold, more complex ones have to be found Calculations can be broken down into sub-steps. When dividing, no explicit distinction is made between dividend and divisor. The breakdown of the division into dividends and divisor would be more mathematically exact but less expressive since the sign of the calculation could not be expressed. The dividend is through a relation that represents the sign (typically +, or *add)* and the Divisor is represented by the relation *divide ,* e.g. B. Throughput = +92 /2.

b) Instead of connecting the relations directly with the concept, so-called n-place relations are used, which are explained in more detail in [215]. Included are not real n-place relations, but represent a concept an anchor point for an n-ary relation. In addition, an incoming relation refers to the concept, from which any number of relations can then proceed. This allows an n:m relationship to be created using two-place relations. When anchor point, the generic term *calculation* rule is used, concrete calculation rules get their own term, such as e.g. B. *BV throughput,* where *BV* stands for calculation rule. By the relation *hatCalculation* or by the separate modeling of the calculation rule, a differentiation from the rest of the modeling is achieved, which improves clarity and comprehensibility. However, this does not achieve greater expressiveness, because only two relations can be used if the operation is not commutative is. In contrast to variant **a) ,** an additional relation *(has calculation)* and an additional term for each calculation rule are used for the modelling needed.

c) Another conceivable approach is modeling using SWRL rules. This one can However, they are neither represented in a UML class diagram nor in an object diagram. With this modelling, the calculation rules are carried out Rules defined and basic arithmetic represented by SWRL built-ins. Of the The advantage is that a compatible reasoner can calculate the values itself, while the other two variants have the calculation done by an external service is carried out. The disadvantage, however, is that there can be no reference from the terms to the calculation rule, so they are not in the definitions can be used. Also, the rules apply to all individuals applied to which the rule applies. Therefore, an assignment, e.g. B. through Attributes are modeled explicitly, making the modeling more complex and less clear.

The results for evaluating a suitable modeling of calculation rules are shown in Table 5.6. It can be seen that SWRL rules (variant **c))** for the modeling are not suitable as there are many limitations. The expressiveness of the two remaining approaches is the same, as is the compatibility. It is weigh whether the advantages of the n-digit relations (variant b)), the comprehensibility due to the additional relation , *calculation* and the better extensibility has the disadvantage the additional modeling (compactness). That is in the ones considered here Use cases are given, since the calculation rules are a central part of the

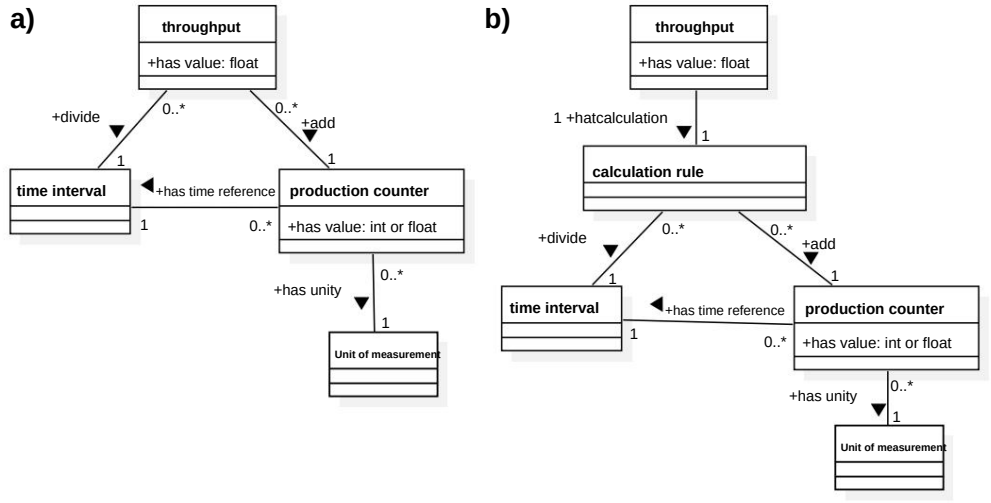

Figure 5.11: Possibilities of modeling calculation rules.

semantic model and both comprehensibility and extensibility are important
are. Therefore, for the presentation of mathematical calculations in this work
n-place relations (variant **b))** used.

Table 5.6: Comparison of three ways of modeling mathematical calculations.

	a)	b)	c)
Uniqueness	XXX		
Intelligibility	(X)	X	(X)
compactness		X	(X) (X)
Compatibility	XX		
Expandability	(X)	X	(X)
adequacy	XX		

Six were added to the semantic model for the description of calculation rules
Added relations. On the one hand, there is the relation *mathematical operations* with
add, subtract, multiply and divide the sub-relations of the four basic *arithmetic operations.*
The parent relation is only included to better structure the relations
been. On the other hand, the relation *hasCalculation was* inserted, which refers to an
individual around the n-place relation. In addition, the generic term *calculation rule* was
added with the sub-terms for specific calculation rules that
can thus be clearly represented in the semantic model. However, not all boundary conditions
for a calculation are shown explicitly. The time intervals or
Points in time must be identical for all calculations, as this is usually the only valid value
is calculated. This boundary condition can hardly be represented explicitly and must therefore be
be ensured by the calculation algorithm. This increases the generalizability
of the approach but not affected.

Modeling question: How can total consumption be determined?

The overall consumption for the operator is particularly high in modular production systems interesting because these values give an overview and not for monitoring the system the values for each module must be observed. Through the Open World Assumption (OWA) it is not possible to model an *all .* In the following, the challenge is tackled, a generic calculation rule for *all* values of a specific to model the type, e.g. B. to be able to calculate a total consumption. The solution is presented using the example of power consumption.

It is known from the module and device description which power measuring devices are available and the topology is shown via the *relation .* A power *meter measures* the relation to the devices or modules recorded by the meter will. The modeling in this section must use the term *total power consumption* Define so that the total power consumption can be calculated from the system description can be. The OWA cannot ensure that all measuring devices are known. For this reason, the term *combined power consumption* is first defined, in which at least two power values are added. For explicit representation, which Modules are taken into account in the combined power consumption, a relation *valueIncludes* introduced. This can be used to model which modules in a value were taken into account. To represent the total power consumption, an attribute *IncludesAll modules* defined. That's *true* if the combined power consumption is all known modules. This makes it possible to reduce the total power consumption to define, which corresponds to the understanding of the operator. The *total power consumption* is therefore defined as a sub-concept of the combined power consumption with the addition that that the attribute *includesAllModules = true .*

The attribute *includesAllModules* can be set automatically, theoretically e.g. B. be realized by a rule in SWRL. A rule checks whether all services of a production plant are recorded, on which the attribute for the total power consumption can be set. The rule could look like this: *production plant(?pa)* ÿ *Power Consumption(?pc)* ÿ *valueIncludes(?pc, ?*vc) ÿ *swrlb:subtract(?pa, ?*vc, 0) ÿ *includesAllModules(?pc,* true). The rule uses a SWRL built-in that counts the number of known modules with the number of modules considered. If the difference is zero, it is the total power and the attribute is set. That The problem, however, is that many reasoners do not support the SWRL built-ins. Specifically, no reasoner could be found[1], that supports this function. As long as no compatible reasoner is available, this check must be carried out together with the calculation in the corresponding service. This has the disadvantage that this rule is not explicitly represented in the semantic model, but implicitly in one Service.

The system topology is expanded in this section by the relation *measures .* A Total consumption can be defined by introducing combined consumptions in combination with the attribute *includes all modules .* Although auxiliary constructs are necessary for the modeling, the necessary extensions are manageable.

[1] As of March 2020.

The attribute *includesAllModules* and the relation *valueIncludes was* inserted. In addition, the terms to be defined must be described, in this example the *combined* power consumption and *total power consumption.*

Modeling question: How can overall equipment effectiveness be mapped?

The OEE is a KPI that covers the entire production process, which means that problems with the process or the system can be quickly identified [216]. Therefore, the GAE is of outstanding importance in the industry. The OAE is made up of various other KPIs and contains almost all the modeling constructs presented here, which is why the determination is quite complex. Because many of the previous constructs are used, the modeling follows from them, so no different variants are presented.

In the work by Hildebrand et al. [166] an ontology was presented that describes KPIs according to the ISO 22400-2 standard, including the OEE. However, the ontology does not describe its calculation rule, so that an algorithm requires an implicit calculation rule, which is to be represented in this work by the semantic model. As a result, the existing ontology does not meet the requirements, but the IRIs are adopted and the calculation rule is added to them. The explicit modeling of the calculation rule is particularly important for the OEE, since this e.g. T. subject to company-specific adjustments [216].

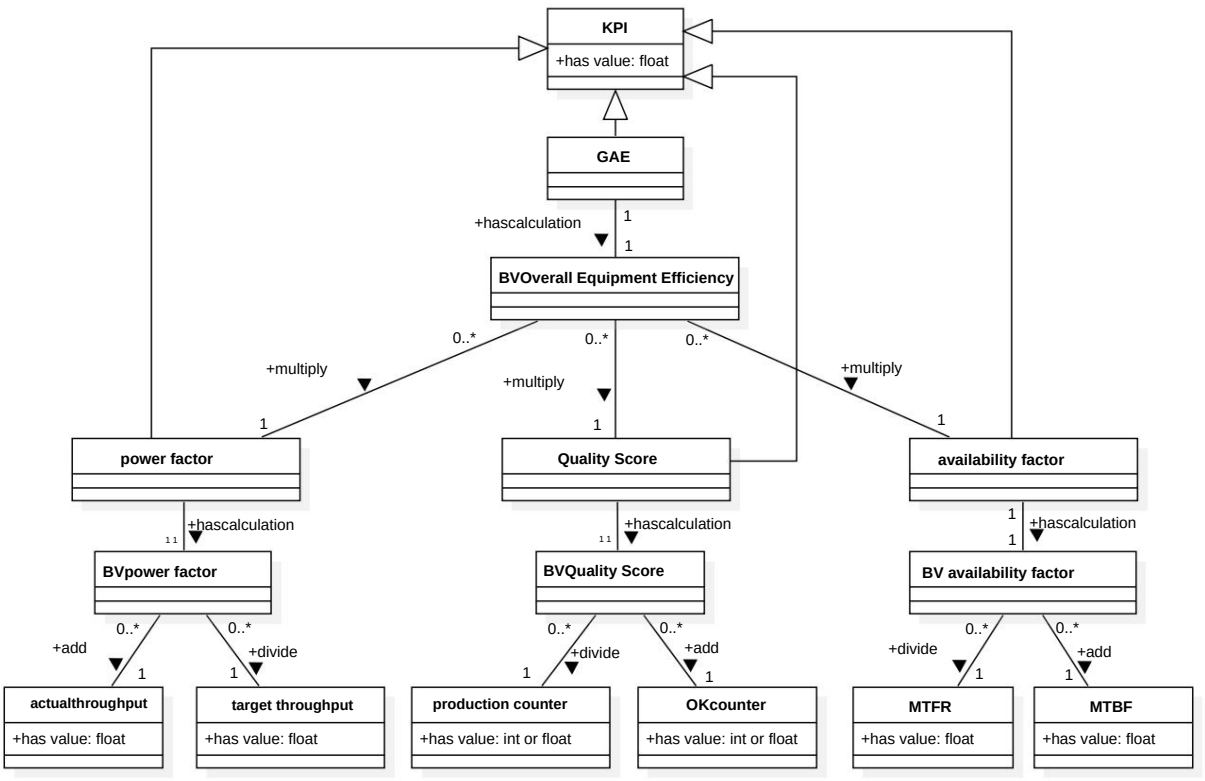

Figure 5.12: Modeling section of the semantic model for describing the overall system efficiency (OEE).

The modeling for the calculation rule is shown in Figure 5.12. It is
good to see that for the definition of the OEE cascaded calculation rules
were used. This is because OEE is the product of *Performance Factor, Quality Factor ,* and *Availability* Factor, and each of these factors divided by itself
two values are determined. The *target throughput, actual throughput, MTBF* (mean time between failures) and *MTFR* (mean time from failure to restoration) are specified,
or determined by further calculation rules in the model. The *production counter*
and the *OK counter* are linked to a sensor, so that these values can be directly
of the production plant can be determined. The *OK counter that* counts how many products the have passed quality control is shown in the model like the production counter.
Due to the cascading of several KPIs, the order must be taken into account when calculating, so that the KPIs are calculated from the process data first, ie the
Calculation in Figure 5.12 is from bottom to top. However, that won't go through
influences the modelling, but this implicit knowledge must be used in the realization
of the runtime environment (Chapter 6) must be observed.

In this section it was shown that even complex KPIs can be described with the presented modeling elements. The correct order of calculation is included
from the runtime environment. There is also a small amount of tacit knowledge here. On the one hand, as presented above, the time, on the other hand, must
the *production* counter and the *OK counter output* the same data type. if e.g. B. the Production counter outputs a volume flow of a liquid from a bottling plant as a continuous signal and the OK counter counts the bottles that the quality control
have passed, the result is an incorrect value. Even if this constellation is an exception, it must be caught by the calculation algorithm.

summary

In this section, the essential modeling decisions for the semantic model of use case 1 were presented and justified. The resulting model
is able to present relevant information about events, parameters and the production plant so that various services can use the information. It
it was also stated that not all knowledge can be explicitly presented,
but when using it, boundary conditions must be taken into account. A total of 85 terms, 18 relations and 20 attributes were required for the modelling
Extensions are possible at any time and certainly necessary, e.g. B. to integrate others Sensors, actuators, units or KPIs.

5.2.2 Semantic Model - Orchestration

If production systems are modular, the modules must be orchestrated before the production process. A corresponding plan can e.g. B. from an integrated
planning system [84] or on a separate level in the architecture,
as in [182] the processor orchestration level. Regardless of how the plan comes about, the following question must be answered: *Which modules can work together to manufacture a specific product?* The answer has many facets, including the

Compatibility of the control, the coverage of the entire manufacturing process and the Compatibility of the modules themselves. Regarding the first two points, some work has already been presented in Chapter 4, which deals with the complex topics of the description of processes and the orchestration of skills to control the systems. During the work, it is always assumed that the modules are mechanically compatible are, which is why this information was not modeled and taken into account. For the the small demonstration systems considered there, the assumption is also correct, in general however, this is not the case. No work could be found in the literature that busy with this topic. This gap is closed in this work by Compatibility of the modules with regard to the mechanics and the operating modes is dealt with.

This use case has some differences to the other two use cases considered. On the one hand, no overall model is required for this application, it is sufficient if the individual sub-models are available. On the other hand this use case does not take place in the operating phase of the plant, but during it configuration phase. Another difference from the first use case is that the industry has a greater influence on the interfaces. Therefore z. T. between between the manufacturing and the process industry.

The challenge is to uniformly record and formalize the parameters of the modules and products for different industries. Due to the large variety of different modules, it is not possible to create a model that includes all possible properties taken into account. Rather, the focus is on the basic structuring and the most important parameters, so that it can be used for many modules without much effort and can be expanded for modules with special requirements. In this application, there is a special need for an expandability of the semantic model. The compatibility check is done with the information of the model by checking the Properties of the two modules to be tested and the product compared with each other will. The product must be taken into account, given characteristics such as the size or the material can lead to the product not being processed in a module can be.

As in use case 1, the semantic model is created from scratch here. The model therefore only contains the terms, relations and attributes relevant to the application. The model created in use case 1 is used like an external model, ie if a suitable term is already defined, it is used. This ensures full compatibility and the models can be combined if necessary. With this procedure, FF 3 can be answered, by identifying the similarity of the models.

In the following, the necessary properties of the modules are determined and presented and categorized. After that, in the respective sections, a suitable modeling determined for the properties. If the properties are already from use case 1 are known, such as B. the throughput, the modeling is not considered further.

Concept question: How can the module descriptions be categorized?

This section introduces basic module descriptions, categorized and identifies the product description required for the compatibility check. There-

Table 5.7 lists and divides the module descriptions into different categories. Under *plant design properties*

descriptions are understood that are determined by the construction of the plant
and cannot be changed, e.g. B. maximum dimensions of the product or
processable materials. On the other hand, the *product* properties can change during the production process. i.e. that these properties relate to the product,
such as B. the dimensions or the weight. Obtain performance-related properties
relate to the production plant, but can usually be found within an area through the
Control program can be adjusted, e.g. B. the throughput. That can help achieve the
compatibility or for optimization.

Table 5.7: Classification of the module descriptions for manufacturing technology.

plant construction	product	performance related
Dimensions (min/max) orientation	throughput	
- length	Dimensions batch size	
- Broad	- length	process time
- Height	- Broad	
Weight	- Height	
material	Weight	
transfer type	material	
mechanical interface		

In order for two modules to be evaluated as compatible, the input module description
match the output module description and the product. The input module description
contains all parameters described above. The output module description, on the other hand, only contains the performance-related properties and a few system design properties. The product properties are represented by the product. In addition, the two can
Modules still contain service-specific interface descriptions that are required to offer a specific service. An example of this is in the use case
3 given. In Figure 5.13 are the descriptions of the modules and the product
shown, with the assignment of the properties described there. Give the colors
indicates where in the semantic model the properties are defined. The ones shown in blue
Properties are defined at the module level and are therefore dependent on the actual process
definitely. The properties shown in yellow are in or through the first or last
device of the module and can be changed if necessary by adapting this device. For example, an adapter can be mounted on a pipeline and such
change the *mechanical interface* . Features shown in red are product specific and features shown in green are optional service specific
Properties that services can add. The service-specific properties are only described, but not for evaluating compatibility
used because these services are not necessary for operation.

Each property is compared to its counterpart of the same name in the other descriptions. Only in the orientation of the product there is no property with the same name

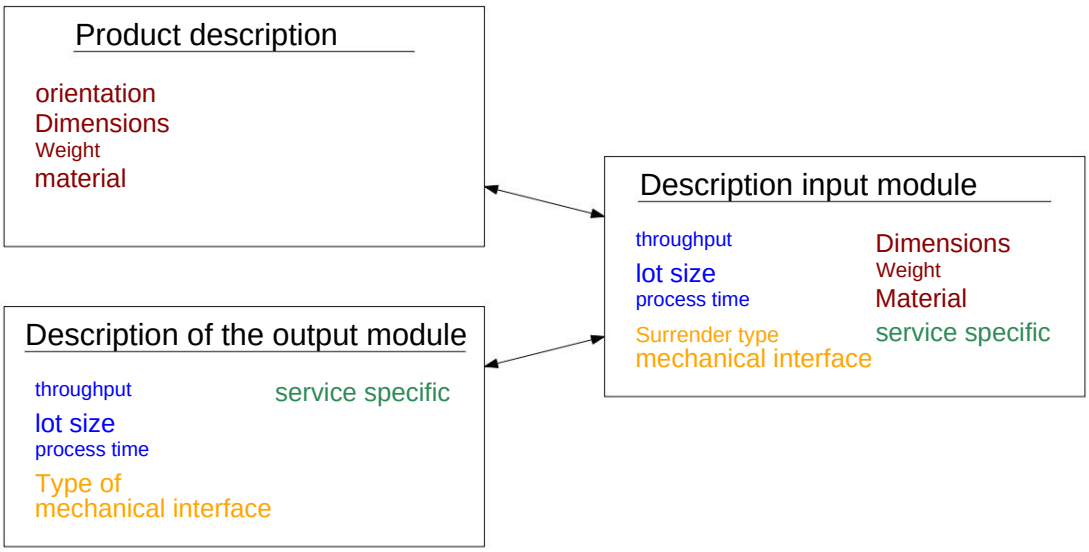

Figure 5.13: Representation and localization of descriptions used for the compatibility check be used.

in the description of the input module, as this is used to transform the dimensions is used. Is the product e.g. B. rotated by 90 degrees, about length and change Width, so the orientation and dimensions of the product in the dimension of the input module description flow. The values of the situational property of the product cannot be determined from the semantic model, but are determined by the Process planning determined and made available. Depending on the expiration of the concrete process planning, further information can be added to the semantic model, e.g. B. the manufacturing process of a module according to the DIN 8580 standard.

This explains the concept for this use case, so that the semantic model can be created below. It is implicitly assumed that a Module with a valid input also delivers a valid output. As in the previous use case, the idea is divided into different modeling questions.

Modeling Question: How can dimensions be modeled in general?

The dimensions of the product or the production plant are part of the compatibility test and must therefore be generally defined in the semantic model. the The challenge is to identify a model that describes the dimensions in general, ie for products, production modules or devices, and that is compatible the previous modeling is. For products, the dimensions can be given exactly be, while the modules have an allowable range in which the dimensions of the products are allowed to move. The modeling of an exact dimension of the product should be created according to the same scheme as the modeling of a value range. The following are three different alternatives for modeling an allowable presented, which are shown in Figure 5.14. For better clarity, only two dimensions are modeled, the transfer to the third dimension is trivial.

a) There can be a term *dimensions* with the sub-terms *length, width* and *height* to be created. The specification of which value is displayed is made by the Relations *hasMinimumValue* and *hasMaximumValue.* The benefit is high Flexibility, ease of use of knowledge, extensibility of terms and the ability to define additional attributes if required.

b) The modeling elements can also be interchanged, so that a term has the value specified, e.g. B. *MaxDimension,* and the relations represents by what value it acts. The modeling is just as clear and compact as variant **a).** The disadvantage, however, is that *length, width* and *height* are not defined as terms, so they cannot be used in the rest of the model.

c) Instead of modeling using two-place relations, an n-place relation can be used. Under the relation *has physical limits* that can Dimensions and weight are summarized. The modeling to the Dimensions themselves can be done by both variants presented above, where variant **a)** was used here. However, querying knowledge in this variant is more difficult, as is understanding, since a maximum dimension not necessarily below the relation *has physical limits* is suspected.

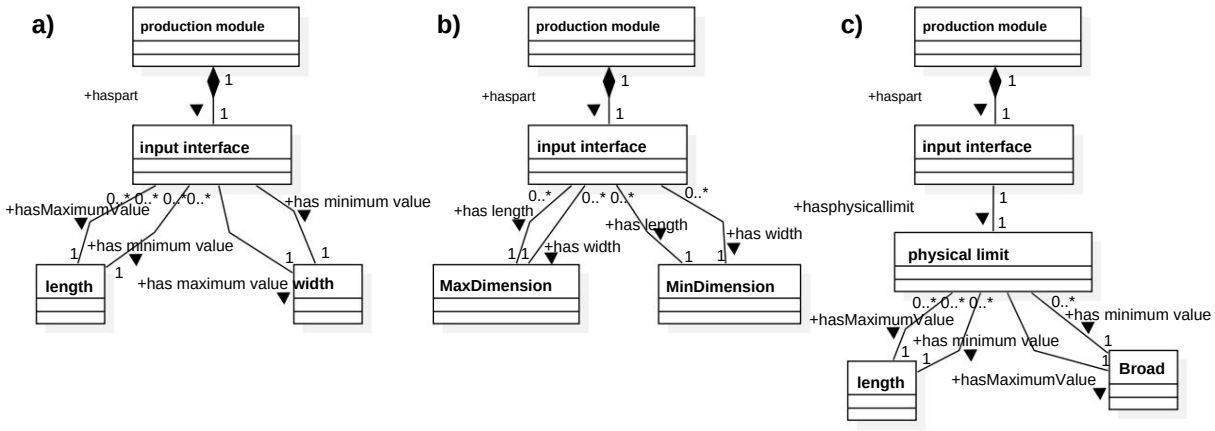

Figure 5.14: Three modeling options to represent the maximum and minimum dimensions of the interface of a production module.

Table 5.8 compares the three models. It is recognizable, that in principle all variants are suitable. Variant **c)** is a little less understandable and not compact, so this variant is ruled out. When it comes to relations Modeling (variant **b))** is the use of dimensions in other definitions possibly more difficult because *length, width* and *height* are not defined as terms. In variant **a)** the terms *length, width* and *height* are defined and relations represent whether there is a minimum or maximum value. Because the minimum and maximum value depend on the respective circumstances, the use of a relation is obvious and can be better integrated into the modelling. Therefore, variant **a)** used. For the dimension of the product, the relation *hasvalue* and is used the weight is also modeled in the same way.

For this use case, the term *input interface* is created in the meta model. This describes the interfaces of the modules and is therefore not specific to them

Table 5.8: Comparison of the four ways to model dimensions.

	a)	b)	c)
Uniqueness	XXX		
Intelligibility	XX	(X)	
Compactness	XX	(X)	
Compatibility	X	(X)	X
Extensibility	XXX		
adequacy	XXX		

Dimensions. To represent the dimensions in the metamodel, the terms *Length dimension, width dimension* and *height dimension are defined* as sub-terms of *physical size* and *length .* The relation *hasMaximumValue* and *hasMinimumValue* specify the relation *hasvalue.* The value itself is associated with a unit, as described in Chapter 5.2.1, ie via the attribute *hasValue* and the relation *hasUnit.* Due to the general validity of the relations, they can also be used to Specification of other properties, such as e.g. B. the weight can be used.

Modeling question: How can the spatial orientation of the product be represented will?

The spatial orientation of products is relevant because some actions of a production module can only be carried out in one direction. Therefore, a change the orientation of the products may be necessary, but this also means changes in the dimensions result. The semantic model can describe the orientation, but do not perform any geometric transformation.

The six degrees of freedom of a product are the spatial movements in three directions and the rotation around these three directions. Under the spatial orientation the rotation of the products is understood. The position in space is neglected since the products are guided through the production plant and these degrees of freedom with it do not need to be described. However, the orientation of the product is crucial in the manufacturing industry, since a hole z. B. typically from above (z-direction, cf. Figure 5.15) is drilled. If the hole is to be drilled in another side, either another module must be used or the orientation of the product must be changed.

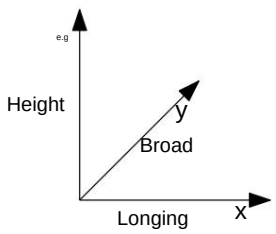

Figure 5.15: Coordinate system for orientation of the product.

The orientation is given by the angle to the original coordinate system. Accordingly, there are three angles that indicate orientation. Over a
Rotation matrix can change the original coordinate system to the rotated system be mapped. As with the modeling of the dimensions need three parameters to be discribed. In contrast to the dimensions, however, is the orientation more dynamic and represents a peripheral area of the semantic model, to which no lead references. Therefore, attributes can be used to model the orientation. To do this, an attribute *hasOrientation* is defined, which contains the subattributes *hasorientationX, hasorientationY* and *hasorientationZ* has. The three Attributes each the angle of displacement between the product coordinate system and the right-hand base coordinate system. The angle is specified in the data type *float* specified. If a product is rotationally symmetrical about an axis, then the data type string uses and receives the value *symmetric,* see Figure 5.16. There it must be a uniform designation, the term cannot be in different languages are specified. For this special case, however, this is an acceptable limitation.

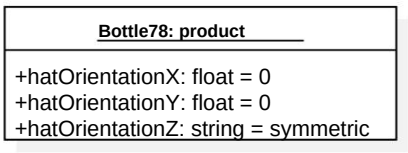

Figure 5.16: Modeling of the orientation in the system model using the example of a rotations symmetrical product, such as B. a bottle.

The spatial orientation is thus represented by three attributes that indicate the angle to the Define initial coordinate system. Only if the object is symmetrical about an axis is, the attribute is assigned a string. By modeling is no reference to orientation possible; however, no use case was found where this would be necessary.

Modeling question: How can the operating modes and the associated performance-related properties are modeled?

The performance-related properties are essentially dependent on the operating mode of the The production plant is determined, while the possible operating modes are determined by the industry. the The challenge in this section is the structuring and assignment of the performance parameters to an operating mode so that compatibility can be checked.

In the manufacturing industry and the process industry, performance can be specified by throughput. For the manufacturing industry, this means that the same time intervals a product is manufactured. In the process industry, throughput is continuous, i.e. a throughput of 3,600 l/h corresponds to a continuous flow of 1 l/s. If, on the other hand, production is carried out using the batch process, the parameters *lot size* and *process time* are relevant. The batch size is the amount of product that is sold at once can be processed, e.g. B. 600L. The process time is the time it takes to process a batches is required, e.g. B. 10 minutes. The throughput is also 3,600 l/h.

These examples show that the term throughput, which is relevant for the user, is abstract and is defined differently depending on the operating mode. Therefore, in
semantic model between the operating modes are distinguished and the respective
The corresponding performance values can be assigned to the operating mode. In this work, the modes of operation are defined as *continuous, discrete* and *batch .* The performance-related Interface parameters are shown in Table 5.9 and sorted by operating mode.

Table 5.9: Relevant performance parameters classified by operating mode.

continuous batch	discreet
throughput	process time throughput
	batch size

The operating modes are a closed set that could be modeled with an enumeration. However, the enumeration is a listing of individuals, so the
No further general meaning could then be assigned to operating modes.
Therefore, they are modeled as terms in the term hierarchy. The performance parameters are also modeled as concepts, with the throughput already being part of the model
of use case 1 is available. About a relation *has performance* parameters
the operating modes as described in Table 5.9. This is in Figure 5.17
shown.

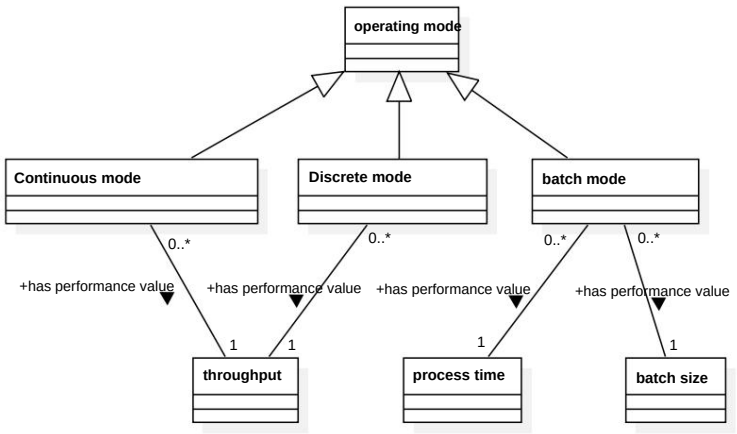

Figure 5.17: Modeling of operating modes as a hierarchy of terms.

If there is not just an allowable power value but an allowable power range, then this can be determined by the specific relation *has minimum power value*
and *has maximum power value* to be modeled. These are subrelations of *hasMinValue* and *hasMaxValue* specified in the metamodel. Since the performance values cannot be inferred from the term that they are
When it comes to performance values, this must be expressed by a specific relation.
The general relations *hasMinimalValue* and *hasMaximumValue* are not sufficient for this out of. The operating modes themselves can be *linked* to a module using the can operating mode relation and thus represent which operating modes a module supports.

In this section, eight terms and three relations were added to the semantic model added, only one term could be taken over from use case 1. In order to

the operating modes themselves and the necessary performance parameters are defined so that a compatibility check of the operating modes and the performance-related properties is possible.

Modeling question: How can the type of transfer be represented?

In modular systems, the transfer of products between the modules is an important criterion, because the modules can only be coupled if a transfer is possible.
But the type of transfer results from the combination of two modules, so it cannot be modeled within one module. The challenge is to define the types of transfer and to represent the necessary parameters in the model.

This section focuses specifically on the manufacturing industry, where there are different ways that a product can be passed between modules. In contrast, the direct connection, in which hoses or pipes are firmly connected in which liquid products and bulk goods are transported, is often encountered in the process industry. With solid products, ie in the manufacturing industry, there are many more possibilities. Three common methods are presented below.

The simplest and a frequently used method is that the products are transferred directly via a conveyor unit, ie the products run from one conveyor belt to the other. Alternatively, a multi-axis system or a robot can place a product on or off a conveyor (semi-direct transfer).

The third option considered is indirect transfer, in which a robot or multi-axis system is connected to a robot or a multi-axis system via a magazine (intermediate store) or to a (possibly autonomous) transport vehicle. The semi-direct variant, in which a robot places the product on a conveyor belt, is presented in detail. Modeling for the other two variants can also be derived from the elements used.

Table 5.10: Parameters required to describe the three transfer types.

direct	semidirect	indirect
Transfer height	Transfer height (min/max)	Transfer height (min/max)
Throughput/speed	radius of action	radius of action
connection type		

The type of transfer is one of the system design properties, but only results from the combination of two modules. has e.g. For example, if one module has a conveyor belt as the last element and another a multi-axis system as the input device, only semi-direct transfer is possible. Each module must therefore provide the parameters for all possible transfer types. In the case of direct transfer, the height and speed of the conveyor belts are relevant, since the product can be damaged if there are large differential speeds. The transfer height is compatible when the conveyor belts are at the same height. In the case of semi-direct transfer, the spatial component must be compatible, ie transfer height and radius of action. The radius of action indicates how far a component can operate beyond the limit of the module. Here, a robot for the

Feeding of parts has a positive radius of action, while a conveyor belt typically has a radius of action less than or equal to zero, ie up to the maximum of the module limit enough. An overview of the necessary parameters can be found in Table 5.10.

It makes sense to carry out the modeling using attributes or relations. This Both variants are presented below and then compared with each other.

a) When displaying with attributes, an attribute is defined for each parameter and given a corresponding value. It's compact, but it can not easily expanded. Since the interfaces in the model are not are of central importance, the more difficult extensibility is not so important. However, the speeds in particular are already in the device description in the model. According to the principle of the *single point of truth ,* all information in the only exist once in the semantic model, otherwise inconsistencies will arise can. It would also need the connection types if they are also defined by attributes are to be displayed are modeled as a string. Modeling as a string admits of no further description, whereby the subdivision, e.g. B. in symmetrical and asymmetrical connection types, is not possible.

b) Existing terms can be used in the representation using relations, so that a *single point of truth* can be realised. When modeling For the handover height, the term *height* can be used and specified by the relation *hasheightaboveground ,* and the radius of action is also specified by defines the term *length .* For the connection types, a hierarchy of terms from compatible connections can be defined. Here, between symmetrical and asymmetric connection types. In this way it can be expressed that with the symmetrical connection types both connectors must be the same; in the case of the asymmetrical ones, on the other hand, there must be a socket and a plug, to establish a connection.

Table 5.11: Comparison of different modeling options for transfer parameters.

	away)	
uniqueness		X
Intelligibility XX		
compactness	x (x)	
Compatibility (X)X		
Extensibility (X) X		
Adequacy (X) X		

Table 5.11 compares the two variants. Variant **a)** is not unambiguous , since some information is modeled several times. On the other hand, the comprehensibility is given for both modeling variants. The compactness of variant **a)** goes with a limited compatibility with existing terms and less good extensibility. Therefore, for this domain variant **b)** considered appropriate and used in this work. The modeling of workpiece carriers was not explicitly mentioned here. There is only one restriction,

that only a direct transfer between two modules is possible. The modeling presented here is therefore compatible with it.

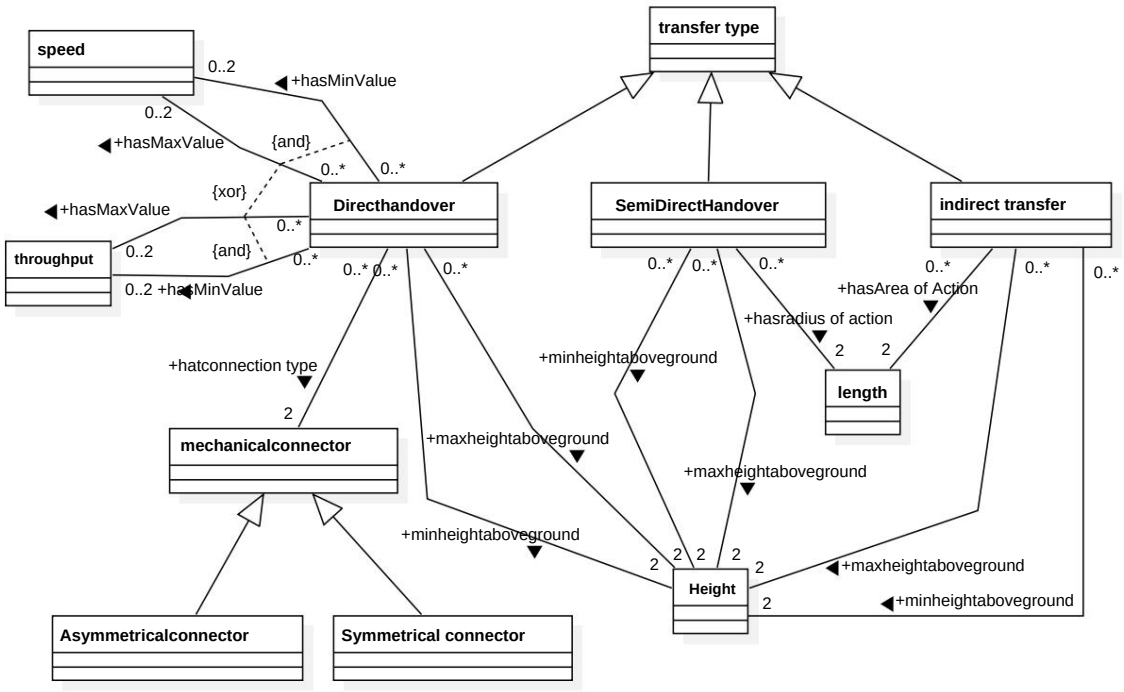

Figure 5.18: Modeling of the transfer type.

Figure 5.18 shows the meta model for modeling the transfer type. The dependencies from Table 5.10 are described there, under the premise that the existing information is not described twice. The term throughput is needed in the process industry when a product is handed over, while in the manufacturing industry the speed, e.g. B. the conveyor belt is relevant. Therefore only one of the two terms is used. Many of the terms used here already exist, such as *length, height, throughput ,* and *speed,* just like the relation *hasMinValue* and *hasMaxValue.* The relations *maxHeightAboveGround* and *minHeightAboveGround* are defined as specializations of *hasMaximumValue* and *hasMinimalValue .* This defines a valid range of values from which to select a value for operation. It is also possible to specify a fixed value instead of a range, which is done using the *hasValue relation.* This was omitted in the illustration in Figure 5.18 for the sake of clarity.

The model shown in Figure 5.18 defines which properties of the modules must be the same so that a certain type of transfer is possible. Since the information is available in the modules, the presentation only comes into play when two modules are considered. The implementation within a module is shown in Figure 5.19 as an example object diagram. If a type of transfer is specified, an input and an output interface is always necessary. In this example, an input interface has been defined. The conveyor belt is defined as an interface, ie it receives the products and thus defines the necessary transfer parameters.

The module supports the transfer types *direct* and *semidirect,* so that all necessary parameters are defined. In this case, therefore, a radius of action, the height

of the conveyor belt, the speed and the connection type are defined. There are two possible connection types: *positioning* for direct transfer and a *stopper* for semi-direct transfer. The positioning is a special form, as it is not a mechanical coupling, but the coupling can be achieved by a corresponding positioning of the conveyor belt or the module. The speed is represented by the already modeled speed value of the module. The height and the radius of action also use terms that have already been defined, which are given the desired meaning via the relations *has radius* of action and *height* above the ground.

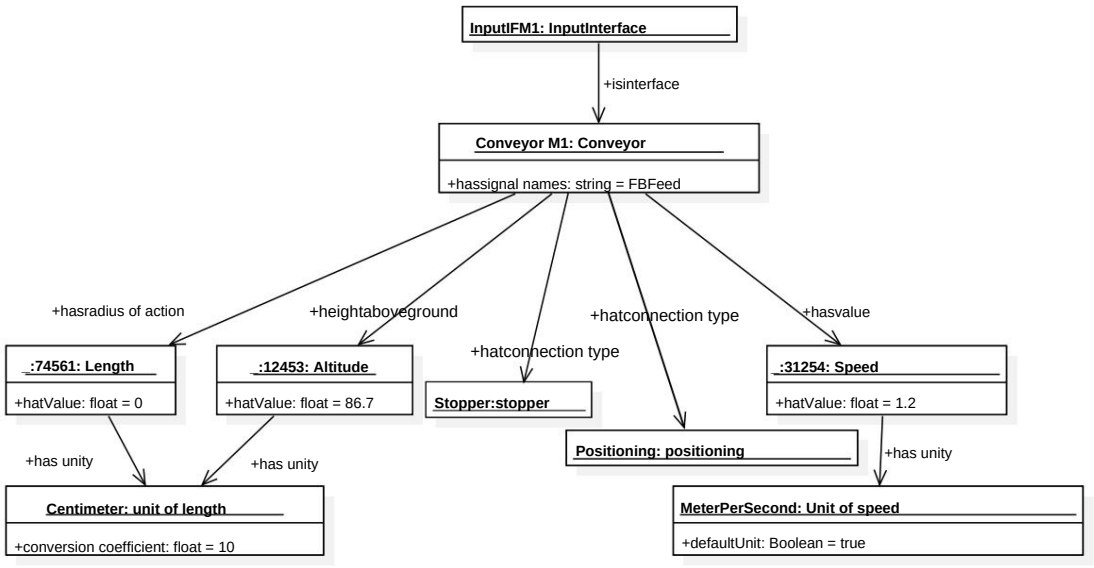

Figure 5.19: Modeling of transfer parameters of an interface using the example of a för the bands.

In this section, many terms that had already been defined could be reused, which was also required to prevent duplication. In particular, the newly defined relations could largely be realized by specifying existing relations. In this way, the transfer parameters could be displayed efficiently.

Modeling question: How is the product description modeled?

The product must be taken into account when assessing the compatibility of two modules. A widely used product classification system, eCl@ss, was introduced in Chapter 4. eCl@ss defines characteristics for each product that are necessary for their procurement. General knowledge about product categories is not available in eCl@ss. However, only general and no specific product knowledge is required for the compatibility check, which is not shown in the classic systems such as eCl@ss. Therefore eCl@ss is not suitable for this work and a separate modeling of the product is presented. Since different product types have different properties, an overview of the different product groups is given here first.

Table 5.12: Allocation of product parameters to different product categories.

bulk liquid product	solid product	
density	density	length
grain size	viscosity	Broad
material	material	Height
		orientation
		Weight
		material

With regard to the products, a distinction is only made between bulk goods, liquid and solid products. A more accurate classification into more detailed product categories, e.g. B. packaging material or detergent, is not relevant for the applications considered here and can be better covered by standards such as eCl@ss. The different
Product categories are described by various product parameters, see Table 5.12. This is an incomplete list of relevant parameters.
In addition, e.g. B. chemical properties may be of interest to check whether
oil-resistant seals are required in a plant in the process industry. An extension must therefore be guaranteed.

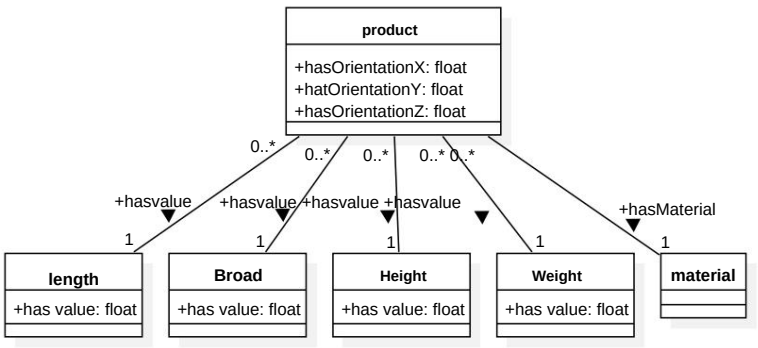

Figure 5.20: Modeling a product using a fixed product as an example.

In Figure 5.20 a model of a fixed product is shown as a class diagram.
It shows all the relevant parameters of the product. The dimensions and the weight
are connected to the product via the relation *hasvalue* . In contrast, has
the material has no value to refer to, so the relation *hasMaterial is* used for it. Such a model is used for each product that is controlled by a module in
another changes, created. The information required for this is supplied by process planning. From the modeled information it is easy to see that
the parameters for the compatibility check were selected and not the product
to describe holistically. To describe the product, other properties would be of interest, e.g. B. the volume or a specification of the shape. Because the model
can be expanded, application-specific properties can be added.

All terms, relations and attributes for modeling the fixed product were already defined. The model was given the three terms for liquid products and bulk goods
added *density, particle size* and *viscosity ;* new relations were not necessary.

summary

The special feature of the orchestration use case is that it takes place during the configuration phase of the production plant. With this, e.g. T. also marginal areas of the model shown, which are otherwise not required. Nevertheless, the modeling was continued in a similar way in order to increase extensibility and comprehensibility guarantee. This makes the model a bit more complex, but it can be used with the methods known from application 1. A total of 49 terms were used for this model, 15 relations and five attributes defined.

5.2.3 Semantic Model - Diagnosis

In this use case, a semantic model for the diagnosis is to be created. A hybrid consistency-based diagnostic method is used, with under *hybrid* is understood as the ability to use discrete and continuous signals. To do this, the method uses logical expressions and learned behavior models. Next the algorithms, the semantic model must represent cause-effect relationships in such a way that they are also suitable for modular systems. In addition, the parameters must described by the algorithms used and the diagnostic results can be interpreted. The basic diagnostic procedure was presented in [18] and is discussed in of this work extended by the aspect of modularization. The description of algorithms in a semantic model has already been presented in [19] and will be discussed in this one work generalized. First, a brief introduction to the consistency-based given diagnosis.

Consistency-based diagnostics

Consistency-based diagnosis is a specialization of model-based diagnosis [217, 218]. Model-based diagnosis was developed and presented in the late 1980s [219, 220] and can be divided into consistency-based and abductive diagnosis subdivide [217, 218]. In the literature, this linguistic distinction is often not used undertaken [221, 222]. However, the term model-based diagnosis is also used in other areas, if neither the classic consistency-based diagnosis is meant, nor the classic abductive diagnosis [223, 224, 225]. In order to prevent misunderstandings of the terms, the term *consistency* -based is used in this work *diagnostics* used.

Consistency-based diagnosis draws conclusions in the direction of causality, ie from causes to effects [218]. This has the advantage that only so-called weak-fault models (WFMs) are required, which only represent the causalities of the correct behavior of the system. With abductive diagnosis and also with heuristic diagnostic methods, conclusions are drawn in the opposite direction to the causalities, ie from effect to cause. So-called Strong-Fault Models (SFMs) are used for abductive diagnosis, ie models that describe all faults in the system. Therefore the creation of SFMs much more complex.

Three elements are required for the consistency-based diagnosis: the system description *(SD),* the components *(COMPS)* and the observations *(OBS).* The system description represents the target behavior, ie the behavior of the system when there is no error present. The observations represent the actual behavior of the system and the components contain a list of all devices in the system. First all components assumed to work, which is expressed by the term ¬ab(ci) , with *ci* ÿ *COMP* S. The *ab* stands for abnormal, so the expression is negated to denote the to represent the correct function of the components. If there is no error, the Union of the three elements *SD* ÿ *COMPS* ÿ *OBS* consistent, ie all components in *COMPS behave* correctly. If the union is inconsistent , it indicates an error in a component. For diagnosis, assumptions about correctly functioning components *COMPS* are withdrawn until consistency is restored. In formal terms, a *COMPSÿ* is created with *COMPSÿ* = {¬ab(ci) | *ci* ÿ *COMPS* \ ÿ} ÿ {ab(ci) | *ci* ÿ ÿ ÿ *COMPS}.* If the erroneous components are contained in ÿ, a consistency of *SD ÿCOMPSÿÿOBS* is reached and the components ÿ can be assumed to be faulty [219].

Residual-based diagnosis

The residue-based diagnostic algorithm (RDA) used here is a consistency-based diagnostic algorithm adapted for production systems work was developed and presented with a first evaluation in [18]. One The conventional consistency-based diagnosis needs to be expanded because the method cannot use continuous signals for diagnosis. In addition, the Runtime for calculating diagnostics exponentially with the number of components to. The RDA overcomes these limitations, making the approach suitable for production plants.

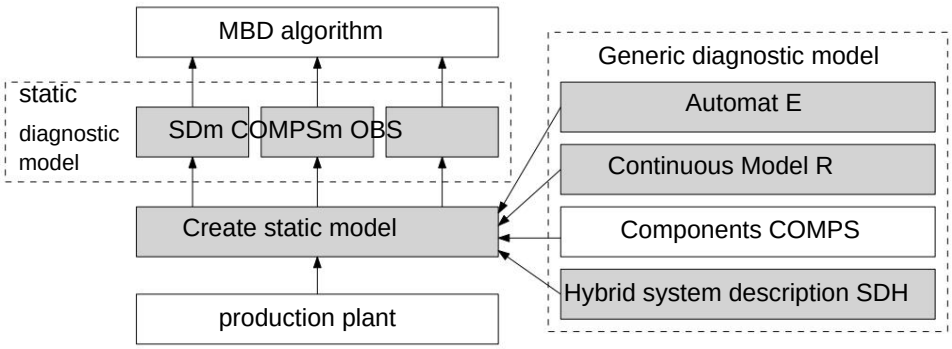

Figure 5.21: Structure of the RDA approach in the operational phase.

The relationship between the various elements in the operational phase is in Figure 5.21 detailed. The white elements are no different from the conventional consistency-based diagnosis, while the gray elements for the RDA were created or modified. Arrows represent the data flow. The algorithm processes the observations from the production plant and uses the generic Diagnostic model to get a static diagnostic model for the current operating state

produce. The generic diagnosis model consists of a finite automaton E, a continuous diagnosis model R, the components *COMPS* and the hybrid system description *SDH .* Operating states (hereinafter referred to as modes *m* ÿ *M)* are defined by the automaton. The static diagnostic model consists of *SDm, COMPSm* and *OBS* and can be processed by a conventional consistency-based diagnostic algorithm. The indexed *m* in *COMPSm* and *SDm* means that these parts are only valid for one mode *m ,* i.e. *SDH* = S *COMPS* = S *SDm* and *COMPSm.* The same convention is used for the continuous model *R* = S *Rm .* The fact that this diagnosis model can be processed by high-performance diagnostic algorithms has the advantage that existing algorithms can be used.

To make the term production plant more specific, the following definition is introduced for this application. In addition, the observability is defined because it is also of interest for the diagnosis.

Definition 19 *A **production plant** is defined as a set of devices A = {ai | i* ÿ *n;i, n* ÿ N}. *Every ai* ÿ *A has parameters defined by a set of variables V = {vj | j* ÿ m; *j, m* ÿ N}, *with V = C* ÿ *D, C* ÿ *D =* ÿ . *Where C represents the set of continuous variables and D represents the set of discrete variables.*

Definition 20 *A variable v* ÿ *V is **observable** if its values can be determined at any point in time.*

There are two main reasons why variables are unobservable. On the one hand, it may be due to the fact that it is not technically possible to determine the value, e.g. B. when distributing several liquids in a reactor. On the other hand, it may be due to the fact that no corresponding sensor was installed, e.g. B. to reduce costs or complexity.

After this brief overview of the RDA, the algorithm is presented in more detail below and its mapping into the semantic model is described.

Modeling question: How to integrate machine learning algorithms?

So far, machine learning algorithms and the processing of logical expressions have hardly been linked, but only one of the two is used depending on the task. By describing the machine learning process, this gap is to be closed and a mapping between the two processes is to be implemented. This work is limited to the description of existing algorithms, no new machine learning algorithms are developed or adapted.

To diagnose production plants, a continuous variable model must be created. A challenge here is that the continuous variables are highly dependent on the mode of the production plant, ie when the mode changes, there can be abrupt changes in the continuous signals. Therefore, different modes of the plant are defined, and each mode becomes continuous

Model created for each variable. This requires many models, but the models are simpler and have less complex shapes, which increases accuracy. Two machine learning algorithms are therefore required for diagnosis in production plants.
The first algorithm divides the behavior of the production plant into different modes based on the discrete variables. The representation takes place via finite automata, which are defined as follows.

Definition 21 *A **finite automaton** for a production plant is a 3-tuple E = (M, ÿ,* T) *where:*

- *M is a set of modes. Each mode m* ÿ *M is a tuple m = (id,* u), *where id is a unique identifier for the mode and **u*** is *a vector containing the truth values of all discrete variables **u** =* (d0, ..., *dn)* ^{T}, d0, ...dn ÿ *D.*
- ÿ *is the alphabet, ie the set of events, where an event is an alternation of at least one truth value of a discrete variable.*
- *T* ÿ *M* × ÿ × *M describes a set of transitions, e.g. B. for a transition hm, b,* m0 i *are m,* m0 ÿ *M source and target mode and b* ÿ ÿ *is the triggering event. [226]*

In each mode of the automaton, a model of all continuous variables *c* ÿ *C* is created by the second machine learning algorithm. This means that deviations in the continuous signals can later be determined and used to diagnose the system. A continuous model is defined as follows.

Definition 22 *The **continuous model** of a production plant A is defined as a set of functions R = {ec | c* ÿ *C, ec :* R + ÿ *R, }. Each function ec returns the expected continuous variable c* ÿ *C at time t* ÿ R + .

With the continuous model, a quantitative residual *ÿ* can be calculated for each continuous variable from the difference between the measured value and the expected value. The expected value is the value from the continuous model.
Thus *ÿc = c(t)* ÿ ec(t) at a certain point in time *t* ÿ R + for a continuous variable *c* ÿ C, where ec(t) is the expected value of *c* at point in time *t .*

Definition 23 *A **qualitative residual** can be for any continuous variable c* ÿ *C at any time t* ÿ R + *can be defined by one of the following predicates:*

- 0 • s *(c) if and only if |ÿc|* ÿ *ÿ*
- *s ÿ(c) if and only if |ÿc| > ÿ and ÿc <* 0
- *s +(c) if and only if |ÿc| > ÿ and ÿc >* 0

ÿ ÿ R + *is a threshold that can be chosen freely.*

Since machine learning methods require a training phase, the time sequence of the RDA is briefly shown (see Figure 5.22) before the mapping into the semantic model is explained. In the training phase, modes are first determined using a machine learning algorithm and represented by a finite state machine. for each continuous signal is learned a model by a machine learning algorithm in each mode. In the operating phase, the data from the production plant the modes and the residuals are calculated. The diagnosis for a specific point in time is then carried out together with the components and the system description. the Management of the models is taken over by the machine learning algorithm itself and not dealt with explicitly here. Also, it is assumed that each algorithm recognizes when the model is of sufficient quality and then leaves the learning phase, as e.g. B. the algorithm Online Timed Automaton Learning Algorithm (OTALA) is the case [226].

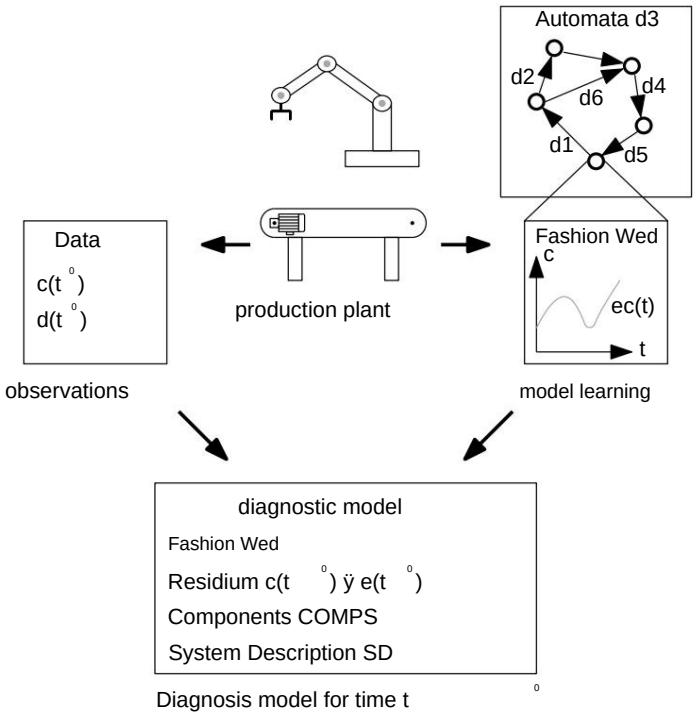

Figure 5.22: The RDA requires a training period before it begins operation.

In order to integrate the machine learning processes into the semantic model, the most important properties are described so that the input data and results required by the algorithm are known. With this it is possible to find a suitable algorithm for a specific task and to define suitable input data. the The machine learning algorithms used in this application deliver an anomaly as a result, i.e. a discrete event. The mapping of continuous signals in a semantic model was presented in [22] and will not be considered further here. The discrete results considered here can be generated with the help of metadata via a Mapping layer can be transferred to the semantic model. The mapping layer will presented in more detail in Chapter 6.

The machine learning algorithm *Hybrid OTALA (HyOTALA)* , which consists of the discrete learning algorithm OTALA [226] and the hybrid part, is suitable for the RDA

of the Hybrid Bottom Up Timing Learning Algorithm (HyBUTLA) [227].
It is therefore an existing algorithm. The modes are learned according to definition 21, ie each variable assignment is interpreted as a mode.
No algorithm is specified for learning the continuous models, but various algorithms can be used in HyOTALA. These only have to be able to display a signal curve over time. Even if two different algorithms are typically required for the training phase and the operating phase, no distinction is made between the two algorithms. Instead, the assumption is made that each machine learning method has implemented the algorithms for both phases. This means that there is no risk of errors occurring when models are exchanged, e.g. B. due to an incorrect implementation or a missing parameter of an exchange format.

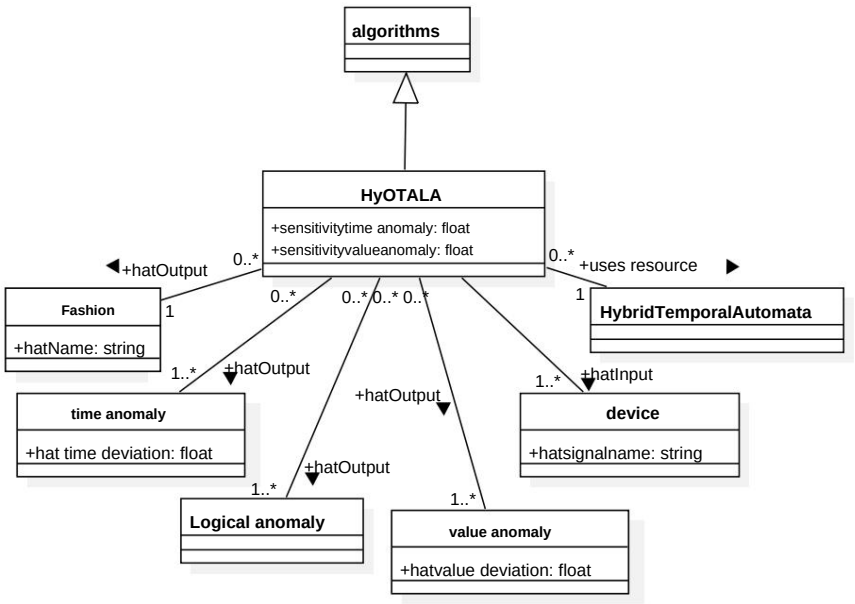

Figure 5.23: Modeling of the HyOTALA algorithm for learning hybrid time-related dead machines.

Figure 5.23 shows the modeling of the HyOTALA algorithm used. As a model, he uses a hybrid time machine that the algorithm learns during the training phase. The hybrid time automaton is an extension of the finite automaton that is not used in this work, so the model is interpreted as a finite automaton. In addition to the model, the values of the devices are required as input, which is modeled by the relation *hatInput .* Since it is a hybrid algorithm, no distinction is made between continuous and discrete signals for the input. The device is identified for the algorithm via the signal name, which means that the values can be assigned directly on the fieldbus. As a result (relation: *hatOutput)* the algorithm can deliver a time anomaly, a logical anomaly, a value anomaly and a state identified by its name. In the case of a logical anomaly, a wrong signal occurs, ie a change between two modes is observed, which was not present during training. On the other hand, with a time anomaly, the right signal occurs at the wrong time. The value anomalies are triggered by continuous signals when the distance between the

expected value and the real value is too large. This corresponds to the value anomaly the residuals described above. The sensitivity for the time and the value anomaly can be specified via two parameters, whereby the time anomalies are not included in this work are taken into account and were only modeled for the sake of completeness. When a anomaly occurs, available data is saved, such as time, size of the deviation and the deviating signal.

Many new terms and relations had to be added to describe the algorithms. In addition to the terms *algorithm* and *HyOTALA ,* the three anomaly types, model types and *mode* were introduced. A total of eight terms were defined, whereby the model types and the algorithms can be extended at will. There were added the three relations *hatOutput, hatInput* and *usesResource .* In addition, the attributes *sensitivity-time anomaly, sensitivity-value anomaly, erroneous signal, added hasnames, hastimevariance* and *hasvaluevariance .* The modeling of The timing of the anomalies is as described in Section 5.2.1.

Modeling question: How can the system description be mapped?

So far there is no uniform formalism to make the system description uniform to present [228]. The formalism of the system description must therefore match the existing algorithm. However, there must be a uniform description for modular production systems so that all modules are compatible with the algorithm. Therefore In this section, a modeling of the system description for the semantic model developed. Since the modeling is intended to represent connections that are expressed through relations, there is only one meaningful modeling that is presented here. First, the faulty assumption and, based on this, the hybrid system description are defined.

Definition 24 *The **erroneous assumption*** ab(a) *is a property (unary predicate) of a device a ÿ A, which expresses that the device a is assumed to be erroneous will.*

Definition 25 *A **hybrid system description** SDH describes the cause-effect relationships of a hybrid production facility and consists of a number of It consists of theorems of first-order logic, including the false-assurance predicates and qualitative residuals, but contains no quantifiers.*

The qualitative residuals *s* ÿ and *s* + are typically used with a negation in the hybrid system description is used because the model represents the error-free system that both residuals only occur in the event of an error. Would the residuals without negation used, it would be an SFM, which is not considered in this work.

The challenge in integrating the system description or the cause-effect relationships contained therein into the semantic model is that several (usually more than two) elements must be connected to each other. Also must establish a clear relationship between cause and effect.

Each cause-effect relationship can consist of unary predicates, variables, constants, modes, logical conjunctions, negations and an implication. Any disjunctions that may occur are not explicitly modeled, but rather expressed through separate cause-effect relationships.

When modeling the cause-effect relationship, a distinction must be made between cause and effect. This is realized by the relations *has cause* and *has effect ,* which have the concept of cause and effect *relationship* as source area, cf.
Figure 5.24. The implication is between cause and effect, each of which is linked to other elements below. Since only conjunctions occur within the cause or effect, the order of the elements is not relevant, which simplifies the modelling. The effect is linked to devices, with the relations specifying the type of link, e.g. whether it is a value, a residual, or a non-abnormal device. Concretely, the effect can consist of the predicates ¬ab *(·), signal(·), s* ÿ, *s* + and *s* or their negation.

This results in ten relations: *abnormal, notabnormal,* residualNeg, residualOk, residualPos , *notResiduumNeg , notResiduumOk , notResiduumPos, notSignal* and *Signal.* With these ten relations, the possibility of hierarchical ordering of relations was used, so that the modeling through the upper *relation system description relation* is clearer. What has been described here for the effect also applies to the cause, although the relation *hatMode* can also be used instead of the relations *signal* and *notsignal .* This simplifies the system description and makes it easier to understand.

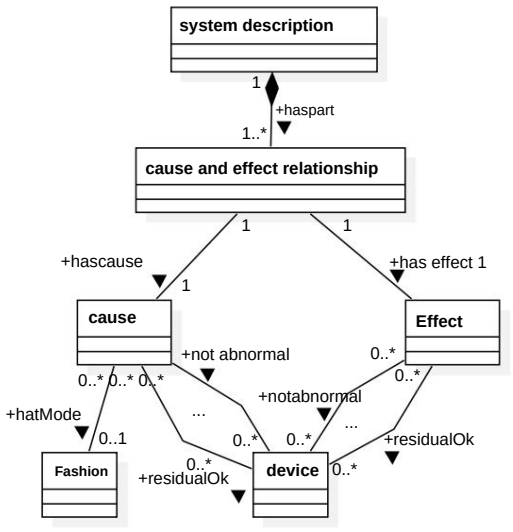

Figure 5.24: Semantic model of the system description for the hybrid model-based diagnosis. Only two relationships between cause or effect and device are shown as examples. The three dots indicate that eight further relations are defined there in the same way.

The creation of a hybrid system description is shown below using a small example. Figure 5.25 shows two modules of the versatile SmartFactoryOWL production facility. The first module is a warehouse where corn is stored. The corn is fed into the metering system via a conveyor belt and an aspirator

module promoted. A sensor is installed in the bearing to check the filling level, which detects a level that is too low. In addition, a sensor checks whether there is a There is blockage in the aspirator. The dosing module is equipped with a scale, so that the incoming corn is weighed.

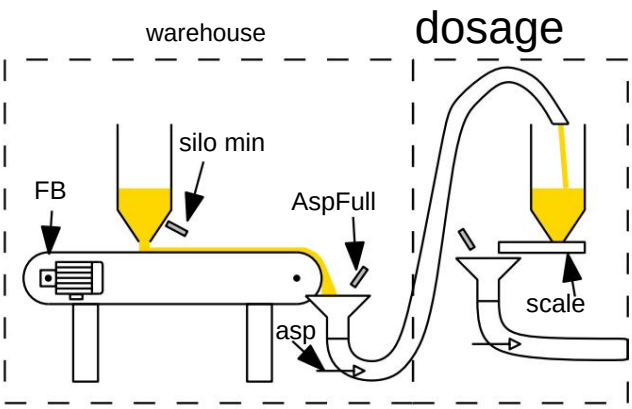

Figure 5.25: Schematic representation of two modules of a Smart FactoryOWL demonstrator, with identification of the sensors and actuators.

A hybrid system description is to be created for this system. A cause-and-effect relationship can be established for the sensor on the aspirator *(AspVoll)* of the first module determine, because if the sensor detects corn, the aspirator *(Asp)* is clogged. This relationship is shown in Formula 5.3. The value of the *AspFull* sensor must be negated because the sensor has the truth value *wrong* in normal behavior and the system description represents normal behavior. The *signal(·)* keyword indicates that it is the value of the device, in this case the logical value of the binary sensors.

$$\neg ab(Asp) \text{ ÿ } \neg ab(AspV\ oll) \text{ ÿ } Signal(Asp) \text{ ÿ } \neg AspV\ oll \tag{5.3}$$

A second dependency of the system influences the sensor signal from the scales in the dosing module. The signal from the scales is correct if all of the subcontracting devices are correct function. In addition, a situation is described via the sensors and actuators in which the context applies, which could also be expressed via a fashion. Of the The relationship described is expressed in Formula 5.4.

$$\neg ab(SiloM\ in) \text{ ÿ } \neg ab(F\ B) \text{ ÿ } \neg ab(Asp) \text{ ÿ } \neg ab(AspV\ oll) \text{ ÿ } \neg ab(\ Scale) \text{ ÿ } Signal(Asp) \text{ ÿ } Signal(SiloM\ in) \text{ ÿ } Signal(F\ B) \text{ ÿ } \neg Signal(AspV\ oll) \text{ ÿ } s^{0}(balance) \tag{5.4}$$

In the event of an error, the above statement can still be specified, since the Scale is a continuous signal. The relationship $\neg s$ *applies* ^{0} ÿ p ÿ ÿ s +, 0 or s ÿ $\neg s$ ÿ ÿ¬s +. Since the system description always describes the error-free system, a negation is used in the effect when specifying an error case. the further specification is not mandatory, but has the advantage that the diagnoses

more precise and errors can be localized more quickly. In the example, it is a negative deviation of the signal from scale *s* ÿ(scale), which can be traced back to a reduced inflow. This can be triggered by many devices. The relationships are shown in Formula 5.5. All devices that are assumed to be ¬ab(·) in the system description can cause the error.

¬*ab(SiloM in)* ÿ ¬*ab(F* B) ÿ ¬*ab(Asp)* ÿ ¬*ab(AspV oll)* ÿ ¬*ab(W aage)* ÿ Signal(Asp)ÿ Signal(SiloM *in)* ÿ *Signal(F* B) ÿ ¬*Signal(AspV oll)* ÿ ¬*s* ÿ(Balance) (5.5)

An upward deviation in flow can only be caused by the conveyor (filling level or speed) or the scale. Therefore, Formula 5.6 contains the expression ¬*ab(F* B). Since the aspirator only transports the available amount of corn, this cannot be the reason for the error. The two values Signal(SiloM *in)* and ¬*Signal(AspV oll)* limit the mode somewhat and indicate when the cause-effect relationship is valid. The silo must therefore not be empty and the aspirator must not be blocked. Optionally, the aspirator could also be included, ie it would have to be activated. That would not change the correctness of the cause-and-effect relationship and is the freedom of the modeler.

¬*ab(F* B) ÿ ¬*ab(Scale)* ÿ Signal(SiloM *in)* ÿ *Signal(F* B)ÿ ¬*Signal(AspV oll)* ÿ ¬*s* +(Scale) (5.6)

The advantages of using modes have already been explained and should be illustrated using the example of Formula 5.6. If the signals of the system description are replaced by the appropriate mode, only the signal values of the cause-effect relationships have to be compared with those of the modes. If the signals in the mode are valid, they can be replaced by the mode. Since not all discrete signals are often described in the system description, it is possible that a cause-effect relationship is also valid in several modes. Is mode m1 e.g. B. defined as Signal(SiloM *in)* ÿ *Signal(F* B) ÿ ¬*Signal(AspV oll) ,* then Equation 5.6 can be simplified to Equation 5.7.

¬*ab(F* B) ÿ ¬*ab(scale)* ÿ m1 ÿ ¬*s* +(scale) (5.7)

The implementation of the modeling in a plant model is shown in Figure 5.26 using Formula 5.3. By separating the relations *has cause* and *has effect there* is a clear subdivision. This is followed by relations to the devices or sub-terms of the devices, in this example *aspirator* and *inductive sensor.* The modeling can be viewed as a cascaded n-ary relation, where the cause-and-effect relationship is the first n- *ary* relation and cause and effect are each the second. As is usual with n-place relations, the corresponding individuals can also be executed as anonymous individuals. In Figure 5.26, however, regular individuals are used.

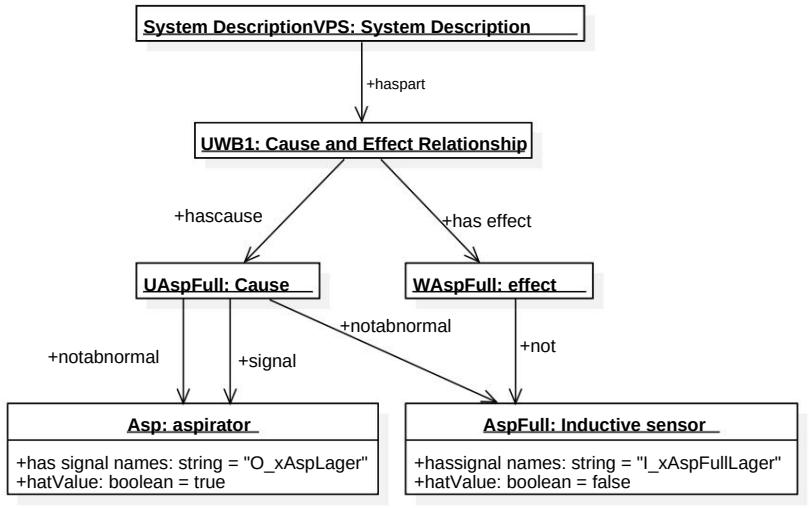

Figure 5.26: SD modeling for hybrid model-based diagnosis, an application of the meta model from Figure 5.24.

The cause-effect relationship shown is clear, so from the model again the formula 5.3 can be deduced.

Only the term *device* could be reused in this section, the other nine terms were newly introduced. In addition, 16 new relations had to be introduced, ten of them to describe cause and effect. This was however, it is possible to clearly represent the system description for the RDA. Furthermore enables the modeling to depict the system description of the conventional consistency-based diagnosis.

Modeling question: How can the RDA be integrated into the semantic model will?

So far, machine learning algorithms and the hybrid system description of the RDA defined and modeled in the semantic model. This section is intended to provide an overview about the algorithm and its mapping into the semantic model.
For this purpose, the algorithm is first presented in pseudocode for the learning and operating phase. Then the missing terms are defined and an overview is given given about the semantic model that describes the RDA.

Algorithm 1 shows the course of the learning phase. In line 1 becomes a finite state machine Learned what discrete variable names and discrete variable historical data are used for. It is important that the historical data error-free operation of the system, otherwise the model quality cannot be guaranteed can be. Not all discrete variables need be used to learn the automaton; a selection of variables relevant to the modes can be set by the operator to be hit. In line 2, all modes are determined from the finite state machine and the variable *M* is initialized with it. It is iterated over all modes (line 3) and the continuous models for the modes and the overall system are initialized

(Line 4+5). For each continuous variable, a model is learned from historical data (rows 6+7) and added to the continuous model for the current mode *Rm* (row 8). In line 9, the continuous models of the modes are combined into an overall model. The finite state machine and the continuous model are returned (line 10) and the learning phase is ended.

Algorithm 1: Learning Modes and Continuous Models

Input: variables *V* historical data *V* ÿ
Output: finite automata E, continuous model *R*

1 *E* ÿ *learnAutomaton(D,* Dÿ)
2 modes *M* ÿ *getAllModes(E)* **3**
for **all** *modes m* ÿ *M* **do**
4 *R* ÿ ÿ
5 *Rm* ÿ ÿ
6 **for all** *continuous variables c* ÿ *C* **do**
7 *c me* ÿ *learnExpectedValues(m, c,* Cÿ)
8th *Rm* ÿ *Rm* ÿ *{e mc}*
9 *R* ÿ *R* ÿ *Rm*
10 return *E, R*

The course of the operating phase is shown in Algorithm 2. In line 1, the current mode is determined using the measured values and the finite state machine. An initialization of *SDm, COMPSm* and *OBS* follows in line 2-4. It is then determined for each cause-effect relationship of the hybrid system description *SDH* whether it is valid in the current mode (lines 5+6). If the cause-effect relationship is valid, it is added to the mode's system description (line 7) and the included devices are added to the mode's components (lines 8+9). To determine the observations, a distinction must be made between continuous and discrete signals (lines 10+11). For the continuous signals, the residual is determined (line 12), compared with the threshold value and the resulting symbol is added to the observations (line 13). The discrete signals can thus be added to the observations (lines 14-18). Then the current mode is added to the observations (line 19). A diagnosis algorithm, as known from consistency-based diagnosis, is called and determines the diagnoses that are defined as the output of the algorithm (lines 20+21).

The structure of the RDA offers some advantages, making it suitable for production facilities. In addition to the division into modes already explained above, the reduction of the overall model to a smaller static diagnosis model is decisive. To do this, the system description is reduced in such a way that it only contains cause-effect relationships that are valid in the current mode. Building on this, the *COMPS* components are reduced to the components that appear in the system description. Since the calculation effort increases exponentially with the number of components [229], this is an effective method to reduce it. Although the static model has to be recalculated for each point in time, the running time for this is linear to the overall model.

In a modular system, a linear total running time could be achieved under certain conditions [18].

Algorithm 2: Flow of the RDA operation phase

Input: variable V , measurement V^t, finite automaton E, continuous model R, hybrid system description *SDH ,* components *COMPS*

Output: Diagnoses *ÿ*

```
1  Mode m ÿ determineMode(E, V^t)
2  SDm ÿ ÿ
3  COMPSm ÿ ÿ
4  OBS ÿ ÿ
   // Compute SDm and COMPSm
5  for all ÿ ÿ SDH do
6      if ÿ holds true in m then
7          SDm ÿ SDm ÿ ÿ
8          if ÿ contains a ÿ COMPS then
9              COMPSm ÿ COMPSm ÿ a
10         end
11 final
12 end
   // Compute OBS
13 for all v ÿ V do
14     if v ÿ C then
15         ÿv ÿ v(t) ÿ ev(t)
16         OBS ÿ OBS ÿ determineResidual(ÿv, ÿ)
17     else
18         if v(t) then
19             OBS ÿ OBS ÿ v(t)
20         else
21             OBS ÿ OBS ÿ ¬v(t)
22         end
23 end
24 end
25 OBS ÿ OBS ÿ getID(m)
26 ÿ ÿ computeDiagnoses(SDm, COMPSm, OBS)
27 return ÿ
```

Since the diagnostic procedure is based on determining consistency, and the semantic Model, which is mapped in an ontology, can be checked for consistency, the idea of carrying out the diagnosis in the semantic model makes sense. This is Although possible, the modeling is much more complex than if only the System description is stored in the semantic model. In addition, the modeling required for this does not correspond to the general conventions of modeling. the System description would have to be modeled at concept level, as well as the observations and components. By convention, however, these parts are closed by individuals represent. In addition, only the consistency check would be carried out in the semantic model, but not the retraction of the assumptions of components, that would have to be in each case by an external service. Therefore it makes more sense to

To carry out the diagnosis process with a specialized service outside of the semantic model and only to represent the input data for the diagnosis in the model. The integration of the elements into the semantic model that are required by the RDA is presented in the following section.

In order to fully describe the algorithm, the definition of the *component* and the *observation is still missing.* These are defined as follows.

Definition 26 *The **components** COMPS are the set of all devices COMPS = A.*

Definition 27 *The **observations** OBS are a finite set of predicate logic statements that represent the discrete variables and qualitative residuals of the production plant at time t* ÿ R + .

The mapping of the RDA into the semantic model is shown in Figure 5.27. The system description, the components and the observations are connected to the RDA via the relation *hatInput .* The system description is linked to the cause-and-effect relationships by the has - part *relation .* To model the *COMPS* components , the two relations *abnormal* and *non-* abnormal are used and connected to the devices, ie actuators and sensors. Although the *COMPS* components can be determined and changed by the diagnostic algorithm, the assumption can be explicitly stated via the relations, e.g. B. when it is known that a component is defective. The *OBS* observations are composed of the values of the devices and the residuals. For this, the relation *has* observation is introduced.

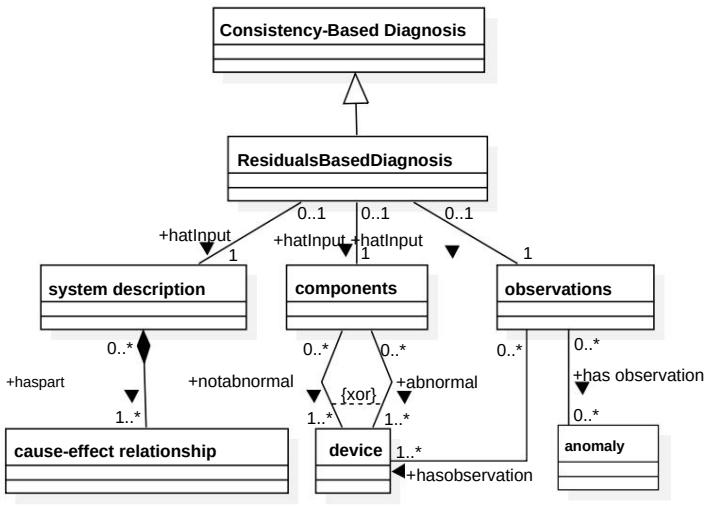

Figure 5.27: Overview of the modeling of the RDA.

For mapping in the semantic model, only the three terms *residual-based diagnosis, components* and *observations* had to be added to the model. Apart from that, only one relation had to be redefined. The algorithm presented is thus mapped in the semantic model, but the system description is not yet modular. However, this is an essential requirement so that the concept can also be used in modular production systems. Therefore, the modularization of the system description is dealt with in the next section.

Modeling question: How can the system description be modularized?

In modular systems, each module has its own system description. Depending on the topology, these must be combined to form an overall system description. In particular, interfaces for the cause-effect relationships between the modular system descriptions must be defined and integrated into the modeling concept. The difference to use case 2 is that use case 2 only refers to the mechanical interfaces of the modules. However, these cannot be transferred to this application, since this modularization is integrated into the diagnostics concept
got to.

The modularization of the system description cannot simply be achieved by dividing the cause-effect relationships of an overall system into different module descriptions. This is due to the fact that in modular systems the dependencies also change when the topology changes and dependencies exist across the module boundaries. In addition, the cross-module dependencies are particularly valuable for diagnostics, since they identify errors that cannot be detected by a single diagnostic system within a module. Also, chained faults across multiple modules are very difficult for the operator to diagnose. Since the modes also change with a changed system topology, the system description should be done in the classic way via the signals.

It is assumed that there can be dependencies at each coupling point of the modules. The reason for this is that deviations in a plant can propagate along the material flow [86]. The cause-effect relationships of each module must therefore be described at the coupling points so that the models can be connected there. For this purpose, three identifiers are introduced, which are based on the three residuals. *x* ÿ is the identifier for a decreasing product stream, e.g. B. by an empty silo, a defective conveyor belt or a blocked aspirator. *x* + denotes an increasing product stream, e.g. B. by a defective control valve. *x* normal operation as used in classic model-based diagnostics.

0 denotes the

These generic identifiers are used as placeholders in the cause and effect relationships of the individual modules that have dependencies on the input and output interface. A cause-effect relationship of an output interface has the form *Ursa cheM1ÿ x* and represents that the *cause* has influence *x* on the following module, where = { +, ÿ } applies. Accordingly, the identifier in the relationship of an input interface *CauseM2* ÿ *Effect* is used for the cause-effect templates can be processed by deleting the identifier and assembling the rest of the expression, resulting in the cause-effect *relationship CauseM1* ÿ *CauseM2* ÿ *Effect .* In this way, cross-module relationships can be represented in modular systems.

Cause-effect relationships that are not cross-module are not affected by the modularization.

In the following, the procedure is presented using the example of the versatile production plant described above. In the example, Formula 5.5 must be modularized because it is cross-module. Formula 5.8 results for the part of the storage module and formula 5.9 for the part of the dosing module. To set up the formula 5.8

all components of the module that can cause a reduced flow are combined. That this is a reduced flow,
becomes by ÿ *x* ÿ shown and is to be read as *caused reduced flow* . Of the
Arrow indicates that this is the premise. Formula 5.9 shows the module-specific addition to the second module in the above formula. In doing so
only added the *scale* component as a possible cause of the error and the same
Component is added assuming correct function as the cause.

¬*ab(SiloM in)* ÿ ¬*ab(F* B) ÿ ¬*ab(Asp)* ÿ ¬*ab(AspV oll)* ÿ Signal(Asp)ÿ
Signal(SiloM *in)* ÿ *Signal(F* B) ÿ ¬*Signal(AspV oll)* ÿ *x* ÿ (5.8)

X ÿ ÿ ¬*ab(scale)* ÿ ¬*s* ÿ(scale) (5.9)

Joining the two formulas for a given configuration is done by
replacement of ÿ *x* ÿ by formula 5.9 in formula 5.8 and the remove a syntactic
of the identifier in formula 5.9. Through this process can be from the different
create an overall system description based on modular system descriptions, regardless of the topology.

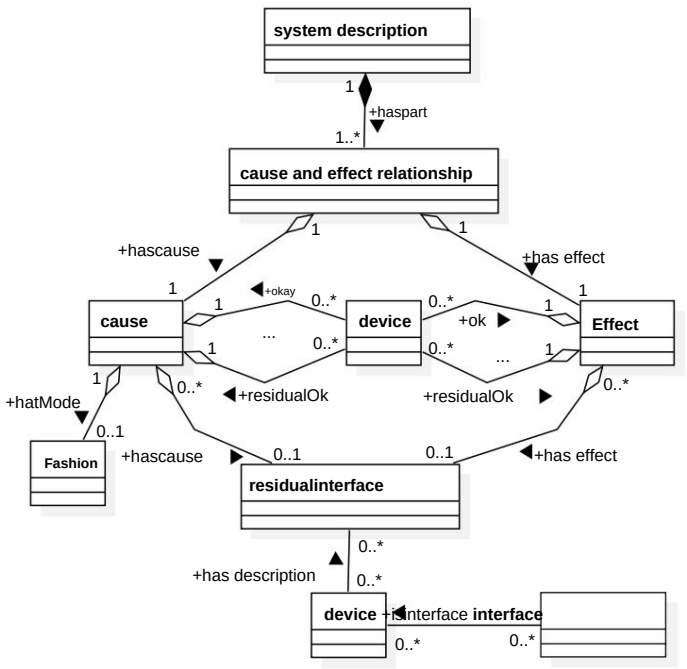

Figure 5.28: Semantic model of the system description with extension for creation of modular cause-effect relationships.

After the concept of the modular system description was introduced, the mapping into the semantic model is described below. The interface for the

System descriptions have been created by the identifiers. For modeling be + and *x* , the
a term: *ResiduumInterfaceNeg, ResiduumInterface* identifiers xÿ are first assigned. *x* each assigned
and *effect* . To do this, the identifiers are connected with the relation *hascause* with the term *cause* or with the relation *haseffect* with the term *effect,* see Figure 5.28. Since a module can have multiple interfaces, the identifier is assigned directly to a device that is the interface. This way there can be no ambiguity.

The two relations *has cause* and *has* effect seem a bit unusual at first glance. However, they show that there is an extension, ie further causes or effects, so that the terminology is appropriate and the relation is clear.

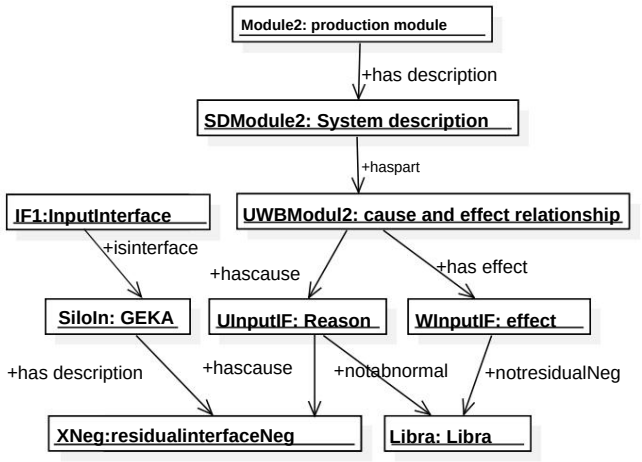

Figure 5.29: Modeling of the cross-module formula 5.9.

The modeling is presented below using a specific example. The modeling for Formula 5.9 is shown in Figure 5.29. The system description is assigned to module 2 (dosage), via the relation *has description.* This is followed by the structure already presented with the cause-effect relationship. In this formula, the cause is linked to the identifier *x* ÿ.

In the example considered so far, bulk material was processed, ie it is a process industry plant. Therefore, a small example from the manufacturing industry is presented below. The difference from the previous example is that there are more binary signals in the manufacturing industry. In the example, two conveyors are juxtaposed and a product is transferred from one module to the other, as shown in Figure 5.30. The previous and subsequent process steps are not relevant for the following considerations. The conveyor belt of the first module has a separator so that the product is transferred as required. The second module has a light barrier at the entrance so that an incoming product is detected here.
The overall system description is shown in Formula 5.10.

¬*ab(F* B1) ÿ ¬*ab(SP1)* ÿ ¬*ab(F* B2) ÿ ¬*ab(LI1)* ÿ *signal(F* B1)ÿ
signal(F B2) ÿ *signal(SP1)* ÿ *signal(LI1)* (5.10)

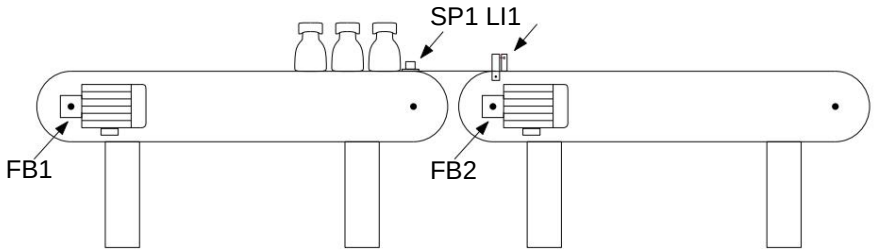

Figure 5.30: Two modules transferring a product (bottles) via a conveyor belt.

As in the previous example, Formula 5.10 can be divided among the modules. There the system under consideration contains only binary signals, the residuals are not required, 0 so only the identifier *x* is used. Typically, production systems in the manufacturing industry also contain continuous signals, such as e.g. B. power consumption, Pressure of the compressed air or speed of a drive, so that the example sketched represents a special case. The modularization of the system description results in the formulas 5.11 and 5.12, which by a syntactic replacement again to formula 5.10 can be combined.

$$\neg ab(F\,\mathrm{B1}) \ddot{y} \neg ab(SP1) \ddot{y} \ddot{y} Signal(F\,\mathrm{B1}) \ddot{y} Signal(SP1) \ddot{y} x^{0} \quad (5.11)$$

$$^{0}x \ddot{y} \neg ab(F\,\mathrm{B2}) \ddot{y} \neg ab(LI1) \ddot{y} signal(F\,\mathrm{B2}) \ddot{y} signal(LI1) \quad (5.12)$$

On the one hand, the example makes it clear that the methodology can also be used in the manufacturing industry. On the other hand, it can be seen that in the manufacturing industry or if only binary signals are present and thus the conventional consistency-based diagnosis can also be used, only the identifier *x* ^{0} is used. Through this model creation is simplified in contrast to hybrid modelling. the Mapping by the semantic model is possible in both variants.

Across all industries, it can be said that modularization with relative little effort is possible. Only the three terms were added to the semantic model Added ResidualInterfaceNeg , *ResidualInterfaceOk* and *ResidualInterfacePos* . Creating a modular formula isn't much different than that Creation of a classic system description.

Modeling question: How can results be presented and interpreted?

In this work, machine learning algorithms are described, but the exact allocation of the results to terms not yet presented in detail. Furthermore the results of the diagnosis must also be represented by terms. In both cases, the results should be linked to the model so that modeling can be excluded by attributes. First, two variants for the modeling of an anomaly presented in Figure 5.31:

a) In this variant, the hierarchy of the anomalies is formulated at concept level and each anomaly is presented as an individual of the corresponding concept. This allows each event to be accurately described and additional information to be easily added.

b) Alternatively, the anomaly type can be described using a single individual. A relation is then used to identify a device as the trigger or cause of this type of anomaly. However, this type of modeling is less specific; so no direct connection can be made between a trigger and the cause. It is only known that a corresponding device is the cause or
trigger of such an anomaly. This means that not all information can be mapped and no expansion is possible, so that additional information, e.g. B. the duration of the anomaly, can not be modeled.

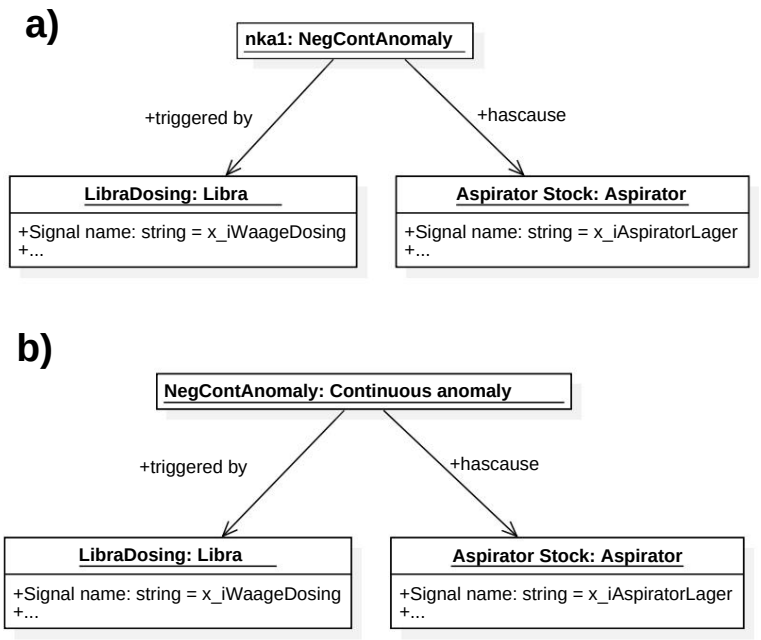

Figure 5.31: Two modeling options for the results of the diagnosis as an object diagram, since variant **b)** cannot be sensibly represented in a class diagram.

A comparison of the two approaches is shown in Table 5.13. It can be seen that variant **a)** is more suitable, especially since it can be easily expanded.
It also allows more information to be presented, which is why the anomalies are clearly described. These advantages come at the expense of compactness, although variant **a)** requires only one more individual per anomaly than variant **b).**

When modeling the results of the RDA, the causes of the error must be shown. However, it often happens that there are several possible causes of failure that have to be represented alternatively, which makes the modeling more complex. For this purpose, three modeling variants are presented, which are shown in Figure 5.32 as an object diagram. The object diagram was used because this is the only way anonymous individuals can be represented and their use makes sense in this modeling. at

Table 5.13: Comparison of different modeling options for diagnostic results.

	away)	
uniqueness	x (x)	
Intelligibility XX		
compactness	(X)X	
Compatibility XX		
Extensibility X		
Adequacy X (X)		

In previous modeling, it has proven useful to use n-place relations, such as B. in the calculation rule, so that a clear structure is achieved. This is also valid for the modeling of possible error causes, so that these were used in all three variants.

a) In this variant, two relations about anonymous individuals are used to represent the causes. If the relation used *has* multiple possible causes from an individual, then there are multiple possible causes, so this should be interpreted as *or .* If, on the other hand, the relation *hasCause* is used, there are several causes, so that the relation is to be understood as *and .* It is therefore the simplest solution in terms of modeling effort, but it is at the expense of comprehensibility.

b) Since the interpretation of the relations in variant **a)** must be known and is therefore not readily understandable, two new relations were introduced in variant **b) .** The explicit representation of *and* and *or* makes the modeling more understandable. In addition, the term *PossibleErrorCause* was added so that it is better structured and not only anonymous individuals are used.
As a result, every anomaly can be explicitly assigned to possible causes, which increases the comprehensibility, but also the scope of the model.

c) The term *possible* error cause was also created in this variant. However, this is used in such a way that only devices are linked via the relation *has cause* and a possible combination of defective devices is mapped exactly. To do this, only the relation *hatPossibleCause has* to be added to the modeling.

An evaluation of the modeling is shown in Table 5.14. Variant **a)** is not suitable with the two relations, as is easy to see. Variants **b)** and **c)** are comprehensible . However, variant **b)** requires more modeling elements and the expandability is not that great there, since the individual conjunctions of possible causes of error cannot be expanded with additional information, such as e.g. B. a probability of their occurrence. Therefore, variant **c)** is the most suitable modeling and is used in this work.

In this section, models for the results of machine learning algorithms for anomaly detection and diagnostic algorithms were developed. The result

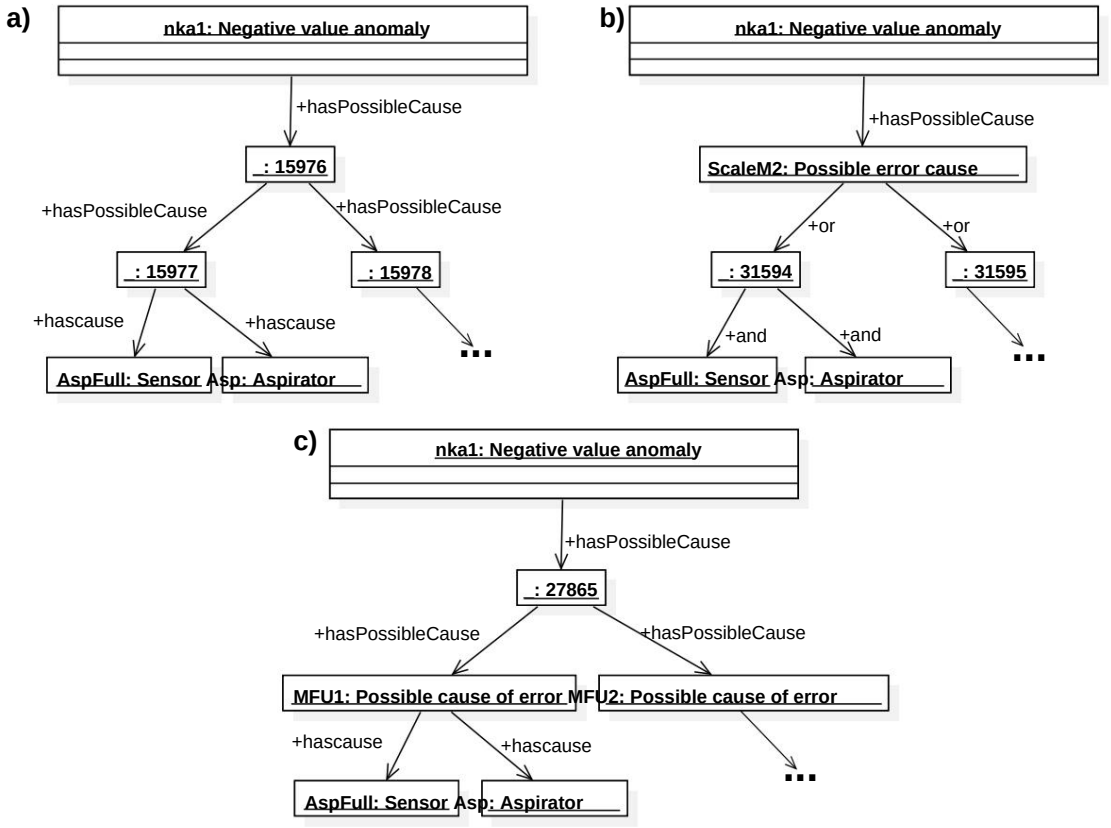

Figure 5.32: Modeling options for possible error causes.

Se of the machine learning algorithms can be represented relatively easily, the anomaly types could be used by the algorithm description, so that only the Relation *triggered* by had to be added. In the more complex modeling of the Possible error causes also only had to be defined as a *possible* error cause and has a relationPossibleCause . This completes the presentation of the results possible with little effort.

Table 5.14: Comparison of different modeling options for possible error causes.

	a)	b)	c)
Uniqueness	XXX		
intelligibility			XX
Compactness	X	(X)	X
compatibility			XX
expandability			(X)X
appropriateness			XX

summary

In this section, a semantic model for diagnosing hybrid production systems was presented. To this end, the RDA diagnostic procedure was first introduced in order to derive the information to be modeled from it. After modeling the system description of the entire production plant, the concept and
adapted the modeling for modular production plants. For this purpose, the cause and effect relationships that apply between the modules were stored in a generic form so that they can be combined according to the topology. The cause and effect relationships are present in the semantic model in the relations
and not explicitly modeled in a term. The semantic model contains in addition to the Description of sub-symbolic algorithms the system descriptions and a description of the results so that they can be interpreted. Total were for
the modeling uses 38 terms, 24 relations and 14 attributes. Of that could
ten terms, three relations and three attributes from the previous use cases
be taken over.

5.2.4 Semantic Model - Overview

In the previous three subchapters, the essential modeling decisions of the semantic models for three use cases were presented. With the imagination
of the models, FF 2 was answered. This section provides an overview of the
Connection of the models of the different use cases are given. The three models are combined to form an overall model. This makes FF 3
Answered: how similar the semantic models are and whether they can be reused across applications. This is important because of the approach presented
can only be used meaningfully if the number of required terms converges.

Figure 5.33 shows the overall semantic model. It is about
However, this is not a complete depiction of the model; only the terms from the first hierarchical level were used. Selectively, these were given sub-concepts
added what was indicated by the relation of the generalization. Furthermore, for reasons of clarity, not all of the relationships between the terms have been included
shown. With the 39 terms, the picture nevertheless gives a good overview of this
overall model. The details such as B. lower hierarchical levels and other relations are
can be found in the individual use cases of the previous sections.

It can be clearly seen that the overall model is *one* model, ie there are many
Relations between the terms of different use cases. That is also possible
recognize by the number of common terms. The second use case uses 16 terms from the previous use case, and use case 3 does
ten terms. That's over 25% of the terms to be defined, which is a high proportion because the use cases are very different. It can be assumed that the
Terms for more similar use cases correspond much better. about that
to check, four additional typical use cases are considered below

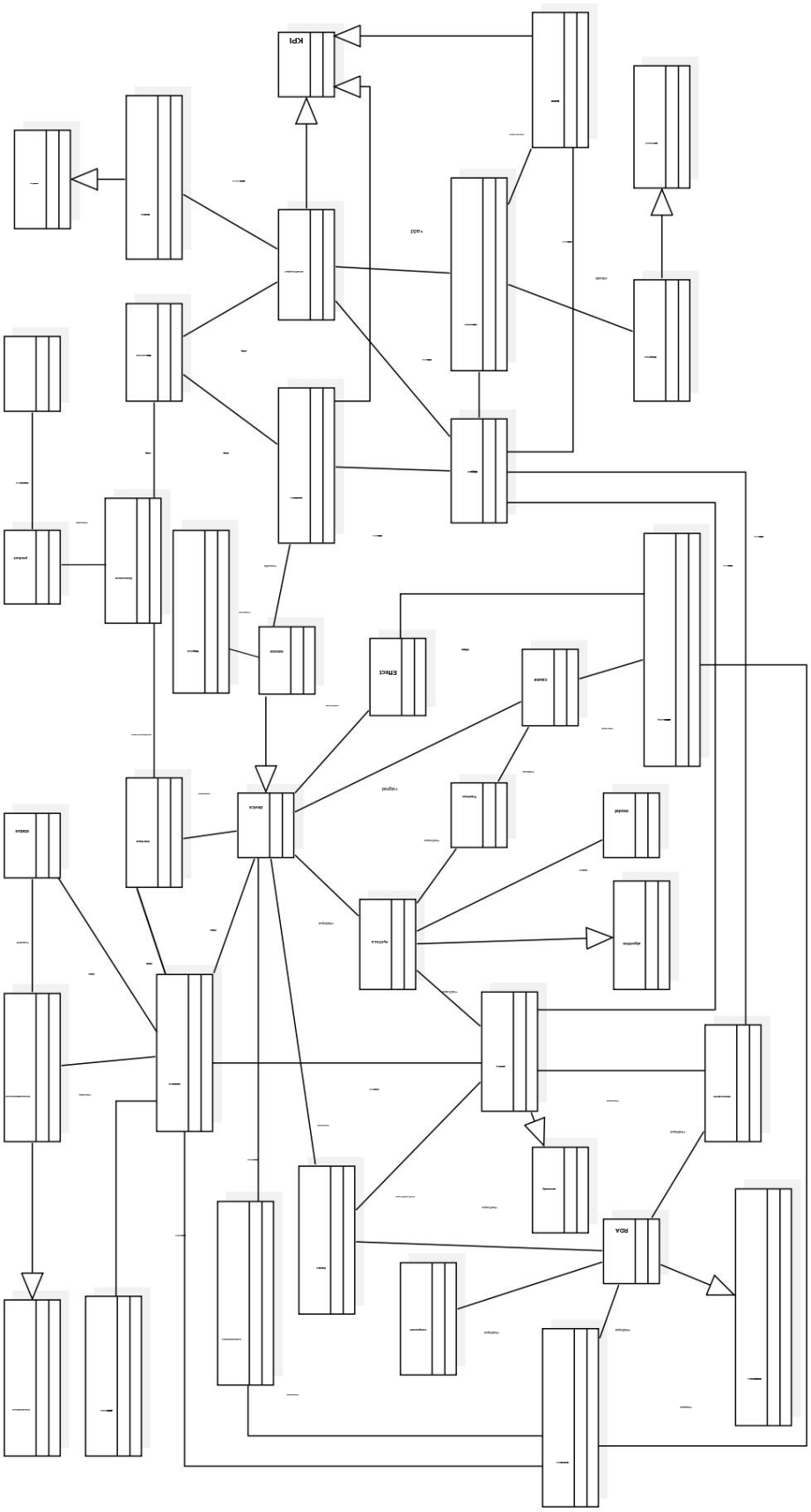

Figure 5.33: Overview of the overall model of all use cases. This is a section of the model. All terms of the first hierarchical level are shown and individual sub-terms where it is helpful. Only some, not all, dependencies between the terms are shown. Multiplicity and reading direction are not specified.

can be found in [3], among others. The use cases considered are predictive maintenance, optimization, digital twin and order negotiations and are therefore typical Use cases of I4.0.

In the use case of predictive maintenance, data-driven algorithms used to identify changes in the production system and use this to plan maintenance as required. This requires information in the semantic model Algorithms, production modules, topologies as well as planned and performed maintenance are presented. The previous model already provides the information very well from, only little things have to be added, e.g. B. a representation for performed maintenance or special algorithms for predictive maintenance.

Data-driven algorithms are also used in the optimization that have to find better settings for the production facility. This is the quality or the throughput is increased or the resource consumption is reduced. The semantic model must describe information about algorithms, models, module parameters, boundary conditions and target values such as energy, time or quality parameters. Also much of this information is already available in the model or can be used with the existing elements are modeled, e.g. B. the boundary conditions. Added only the target values and the quality parameters have to be defined.

The digital twin should collect all information about the production plant and to simulate their behavior. The description already shows that the use case cannot be clearly defined. Typically, a digital twin contains Information about models, production modules, devices, module parameters, maintenance, production times and quantities. When listing, all terms from the known from previous use cases. If the digital twin has other information should provide, some extensions may be necessary.

Order negotiation is a use case that aims at the automated formation of value chains [13]. This means that the production modules should work together negotiate which modules will be involved in the production process. For this he will process step is described and the modules make offers for this, e.g. B. to which Price, in which quality or processing time the completion is possible. have to in the model information about the capabilities of the modules, the orders, the costs, the qualities and capacities. These aspects have only been considered to a very limited extent in the semantic model so that the integration of this Use case is only possible with many extensions in the semantic model.

Of the four use cases considered, three can be implemented without much effort existing model can be covered, ie the overall model only has to be expanded by a few terms for these applications. The fourth use case, on the other hand, can little use of existing definitions. This is because the other use cases considered were technical and related to the production facility or theirs service had to do. In contrast, the last use case deals more with topics of Production planning and business administration and thus has an orientation that cannot be served by the existing model.

In addition to the use case orientation, it can be observed that some terms be used across applications. This includes in particular descriptions

genes and characteristics of the production plant or the product and general terms such as B. Energy, time or units. The application-related terms, on the other hand, are more specific and often only relevant for certain applications, e.g. B. Cause and Effect Relationship. There is no sharp boundary between general and the application-specific terms. So it is not possible from the created overall model to take out a set of terms to use as base terms for use different use cases. Based on the knowledge gained here instead, the scope of the model is limited so that the number of Terms converged faster. The restriction does not lead to major expansions, as would be the case with the use case of contract negotiations. Because to On the one hand, extensions are time-consuming and, on the other hand, they are carried out manually, from which a new semantic model develops, which no longer works with everyone services is compatible. So that this approach can be applied consistently and the models remain compatible, there should be a model that contains all terms at the upper hierarchical levels. These levels are important for compatibility. In the lower levels, terms such as B. a new sensor added and used if existing modeling elements are used. In order to be able to create an almost complete model, as many technical use cases as possible should be included be modeled.

To answer FF 3, it can be summarized that many use cases can be implemented with the created semantic model. However is not a differentiation of the terms into generally applicable and application-specific possible. Instead, the perspective of the use case must be considered, which was technical in the cases considered here. The knowledge gained in this work suggests that for such an overall model from a technical perspective only a few hundred terms are necessary. If other, non-technical aspects are to be taken into account, the model described here is not suitable or it would have to be greatly expanded, which would increase the complexity.

6 Evaluation

To implement the use cases, an overall model must be created during the configuration phase, which is used in a so-called runtime environment during the operating phase. In subchapter 6.1 the configuration and operation phases are explained
and the runtime environment is presented. For this purpose, two publications can be used that were created as part of this dissertation. In [21] could be shown
that OWL ontologies can be mapped in OPC UA. This will not be here
further considered, but assumed as given. The mapping of signals to terms at runtime is considered in [22], so that the knowledge gained from this can be used in the
the mapping level presented here. The test system used is described in Chapter 6.2 presented and the implementation explained. For implementation, the plant model
based on the semantic metamodel presented in Chapter 5, which will not be discussed in detail here. In subchapters 6.3, 6.4 and 6.5 the
three use cases evaluated. In addition to the correctness of the results,
the computing time is considered to ensure that the necessary performance is given is.

6.1 Creation of the runtime environment

The aim of the work is to use the semantic model to achieve a uniform description of the production system and thus to be able to solve use cases generically. This means,
it should be able to be transferred to other production systems without manual effort. So that the semantic model can be used to solve the use cases,
a runtime environment is required that allows access to the model and the necessary Periphery defined. These runtime environment requirements in the operational phase are summarized in Table 6.1. The presentation of the examples developed here Runtime environment is divided into configuration phase and operation phase.

6.1.1 Configuration phase

The aim of the configuration phase is to generate an overall model of the production plant from the partial models of the individual modules. This includes dependency
to be inserted between the partial models, because by combining modules
are no longer independent. Many process parameters can change, such as
e.g. B. the total power consumption or the OEE. The configuration process takes place in the

Table 6.1: Runtime environment requirements

No.	Request for data	description
1	collection	The data collection must be easy on others Systems to be adaptable.
2	Incorporation of algorithms	Process data must be able to be processed in algorithms so that further information can be obtained.
3	Configuration algorithms	The algorithms must be configurable and it must be possible to easily add new algorithms.
4	data availability	The process data and the results of the algorithms should be in the semantic model available in the appropriate individuals without having to add them manually.
5	Standardized model access	From the outside, the model should have a standard dissed interface must be available.
6	multiservice	The model should not be limited to one service, but any number of them services can be executed simultaneously.
7	distributed system	Services should not only be executed centrally can be, but also on distributed systems so that remote access is possible.

integration level and is shown in Figure 6.1. The process is divided into three components in this work, but depending on the granularity this can be even further be subdivided, such as e.g. in [183].

- **Discovery:** The discovery component has the task of recognizing all modules located in the network so that all partial models of the modules are available at the integration level.

- **Topology:** The topology component establishes topological dependencies between the modules so that these dependencies are represented in the overall model be able. This information can e.g. B. be extracted from planning or engineering data or the fact that the modules recognize their neighboring modules, so that the topology can be deduced.

- **Merge:** The merge component combines the partial models into one model. Through adding relations between the modules it becomes an overall model representing the production plant.

The merge component that creates the overall model is independent of the CPPS and the technologies used. Therefore, the component is presented in detail. the

the other two components, on the other hand, are implementation-dependent. There are several Implementation options and these may depend on the circumstances of each facility can be selected. Therefore, the components cannot be explained in detail, but only to the extent necessary for the evaluation.

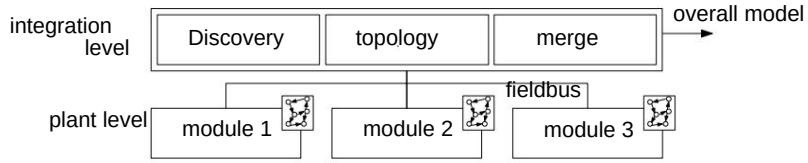

Figure 6.1: Logical structure of the solution approach during the configuration phase.

Discovery component

In a flexible production environment, new production modules are constantly being added and others disappear, e.g. B. due to maintenance. So now all ready for operation Modules can be discovered and communicated with, a discovery server is used. The server must be known to all modules or through a corresponding method to be discoverable. A new module registers with the discovery server and stores relevant information, e.g. B. your own address or security requirements. If information from a module is now required, the module can be sent to the server be queried and a connection can be established with the returned details.

The OPC UA standard used in the industry offers the possibility of a discovery server. A distinction is made between a local and a global discovery server [95]. The basic functionality is similar for both servers. each new module, or the integrated OPC UA server, reports to the discovery server at. A list of all active modules is created and maintained in the server, including the Details necessary to communicate with the module. A client, e.g. B. the Merge component, can query the discovery server which servers are in the network available. This allows the client to connect to any production module and access the necessary information. The information is defined in an information model so that more complex data can also be displayed. That's the way it is possible, with few restrictions, ontologies in the OPC UA information model [21] so that the partial models can be transferred directly via OPC UA. Alternatively, a corresponding link with direct access to the semantic model submitted. In any case, the discovery component implemented in OPC UA ensure that all available submodels can be accessed.

topology component

In order to create a correct overall model, the topology must be known, ie how the modules are connected together. Depending on the processes in the company, there are many ways of determining or defining the topology

can. This can e.g. B. through engineering data, production planning data, localization tion of the production modules or sensors on the modules, with each possibility being subject to certain limitations. For the approach used here is not what matters is how the data was obtained, only that it is available. for a flexible manufacturing would typically use the production planning system topology provide. The information becomes available to the merge component provided, which creates an overall model from the topology and the partial models. For the The topology is entered manually for evaluation, as there is no corresponding planning system and no sensors on the modules.

merge component

The merge component covers the two core aspects of the configuration phase: *(i)* merging the sub-models and *(ii)* inserting relationships between the models. In the context of OWL and the Semantic Web, merging becomes semantic and syntactic similarities of terms and relations are searched for and accepted as equal if the quality is sufficient [121]. However, this method is not suitable for production systems because of the high demands on reliability in the industrial environment and correctness of the model. Therefore, in this work, partial models used, which were created according to a uniform meta model and are therefore simpler can be put together.

Each object (concepts, relations and individuals) in the semantic model gets one unique ID. By using the IDs, each object can be clearly assigned, so there can be no ambiguity. This does not make it expandable limited, any number of other terms and relations can be defined and used. Furthermore, not all terms and relations need to be taken from the standard are used, but only exactly those that are actually required. When merging multiple models, the same IDs are grouped together so that each The term only exists once in the overall model. Unless a term only in a model is available, it is transferred to the overall model with all relations. in the In contrast, the individuals must appear separately from each other in the overall model, since each individual represents a certain aspect of a module, which is also after the merging is retained. To do this, the individuals must be given unique IDs. The use of a prefix is recommended for this. This prefix must be used when creating of the sub-models are defined and can e.g. B. the manufacturer and the serial number include. This enables correct and unambiguous merging of the partial models.

However, it can also happen that mapping between different models is necessary, e.g. B. because different versions of the semantic model were used. This will occur as the terms of a semantic model change over time to change. For this purpose, a mapping rule between the semantic models is developed, which defines how terms can be mapped onto one another, ie the terms are defined as equivalent even if the IDs are different. Because the creation such a mapping is only necessary once, it should be created manually in order to to ensure the necessary quality.

After the overall model has been created, relationships between the modules must be shown. This concerns information about the product flow and the horizon of measuring devices, both of which result from the topology. This means that the relations are *measured* and *suppliesProductTo are* added. provides the necessary information the topology component ready. The merge component can be viewed as an initialization service that is always executed first after a configuration and that topology information into the overall model. In addition, use case-dependent relations can be inserted, such as B. in the system description of the diagnostic use case. However, these are not inserted directly by the merge component, but rather the corresponding services add the information at the first call independently. In this way, each service can process the necessary information and insert necessary relations.

6.1.2 Operating Phase

This section explains how the semantic model can be used to solve the use cases. This requires more solution components than just that of the semantic model. The logical structure for the solution approach in operation is shown in Figure 6.2, which was created according to the requirements from Table 6.1.
The CPPS, at the lowest level, is not represented in a modular way, as the modules are represented by the Merging the models in the configuration phase can be viewed as a single entity. The CPPS signals are transmitted to the mapping layer and can also be sub-symbolic algorithms are processed in order to obtain new additional information to generate. The parameterization of the algorithms is carried out by the services, ie if a service needs additional information, it configures an appropriate algorithm. All information collected through the CPPS is managed by the Transferring the mapping layer to the semantic model. That's all the information summarized about the production plant in the semantic model and are in one uniform description. The reasoning component checks the consistency and can To draw conclusions. Since the semantic model and the reasoning component are closely connected to each other and the reasoning component needs direct access to the model, the two components were drawn in a box. In order to use the information from the semantic model, queries from the services to the semantic model. The processing of the request and therefore the model is typically accessed via the reasoning component [230].
The actual use case is solved by services at the application level. The services make and process requests via a defined interface the information in e.g. T. complex algorithms and provide them if necessary for the operator ready. The advantage of the uniform description in the semantic model is that the Services are generic, so they can be used for many systems with the same use case. First the interfaces and then the individual ones Components presented in detail.

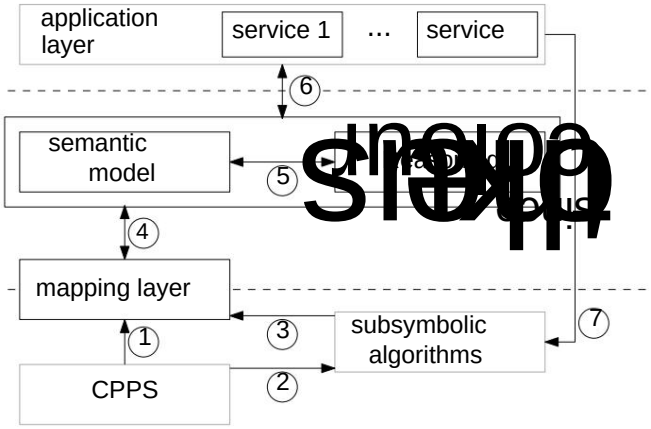

Figure 6.2: Logical structure of the solution during operation. The core elements of the approach are framed in black and arranged according to Figure 1.2 in the integration layer.

interfaces

Interfaces 1-7 of the runtime environment from Figure 6.2 are presented below:

1. The CPPS transmits current process data, e.g. B. sensor values, actuator values or process parameters to the mapping layer. OPC is suitable for accessing the data UA, since this platform-independent service can also provide metadata, which is made possible by a simple mapping.

2. The interface between the CPPS and the sub-symbolic algorithms is similar of interface 1. The same data is transmitted, so that OPC UA can be used. Depending on the dynamics of the production process, it can be necessary to use *OPC UA over TSN ,* a real-time extension of OPC UA, which enables shorter sampling times [146].

3. The results of the algorithms are defined strings of characters that can be assigned a specific meaning. The character strings are transferred to the mapping layer, which adds the result to the semantic via interface 4 model one.

4. The mapping level transfers the process data and the results of the algorithms into the semantic model. The query language SPARQL is used for this. Of the Data flow is bi-directional, so the mapping layer also makes requests to the model can provide and the assignment is as efficient as possible.

5. About the interface 5 is knowledge between the semantic model and the Reasoning component replaced. Typically, the two components executed in a framework (Jena Framework or OWL API) so that the reasoner has direct access to the model, e.g. B. in the pellet reasoner [231].

6. Services can query information from the semantic model and write new information to the model, e.g. B. possible diagnoses for an occurred Anomaly. SPARQL is also used for this.

7. Access from the application layer to the sub-symbolic algorithms
provided for the configuration. Accordingly, each algorithm has a configuration routine
that is called by a service and necessary parameters
provides.

mapping layer

The mapping layer is an active component, ie signals from the CPPS and the algorithms are retrieved and written into the semantic model. thereby become
the data is provided in a symbolically interpretable form and can be used to solve the use cases. The mapping level provides the connection
between the z. T. proprietary, control signals and the uniformly defined
Conceptualized in the semantic model. In general, due to the heterogeneity of
Technologies to communicate with the CPPS implementing the mapping layer only
done proprietary. Only if standardized technologies are used, such as
e.g. B. OPC UA, a generic solution is possible, in which a few settings may be necessary (e.g. IP address) must be adjusted. Due to the standardized interfaces to the semantic model or the query component, there is no adjustment effort for the
Model access required.

The signal names of the devices are used for the mapping. The signal names are
for each sensor of a module in the submodels and are thus also in
taken over the overall model, so that the values can be assigned to the individuals in the model is possible. The recorded process data contain metadata, ie in particular signal names that can be compared with the semantic model. This is a
automatic mapping of the process data into the semantic model possible. But
in addition to the process data, the results of the sub-symbolic algorithms must also be indicated be transferred to the mapping layer. In addition, the results are given an algorithm-specific name, comparable to a signal name, so that they also have a
Individual, which represents an algorithm, can be assigned [22]. Since the
Algorithms can be used on many production systems, the mapping is also transferable, so that it only has to be created once for each algorithm. Accordingly
New algorithms can also be added for which only a mapping needs to be created, provided they have already been described in the semantic model.

semantic model

The semantic model can be divided into a meta model and a plant model. The meta model is the generic part of the model, the terms and relations of a
Domain defined as presented in the previous chapter. The meta model
can also be understood as abstract knowledge, because it does not represent originals, ie no real world objects. The meta model is therefore not plant-specific.
In contrast, the system model describes the system-specific part, such as e.g. B. the components of the system, signal names and characteristic values of the components. For this they will
Concepts instantiated (instances are called individuals in OWL), get concrete
assigned values and relations. Not all individuals represent a concrete object of

real world, e.g. B. configurations or the total energy consumption are also called individuals modeled.

The type and amount of information required in the plant model depends on the use case to be implemented. Therefore, the models for the use cases were always considered separately. Although it is desirable that the production plant is described as completely as possible in the plant model, since then any implement use cases. On the other hand, there is resource consumption and Modeling effort, especially as long as no method for automatic model creation has been established and the information has to be modeled manually. the Needs-based modeling also has the advantage that the complete meta model is not required, but that it can be reduced to the terms used. This allows the consistency of the models to be checked faster and the time to Calculating conclusions decreases.

The semantic model is passive, i.e. it makes knowledge available on request, but does not perform any independent actions. It is a pure knowledge base with external reasoning functionality. The model is updated from the outside by the mapping layer, the reasoning component or an application layer service writes information into the model. The interface required for this is provided by the SPARQL realized because it is a technology-open interface.

Reasoning

The reasoning component has two tasks: Checking the consistency of the semantic model and deriving conclusions. The consistency check is important because only one consistent model can be used. Inconsistencies can lead to incorrect statements, which is why they must be avoided at all costs. They arise through logical contradictions, e.g. B. if an instance has been explicitly assigned to a term and this has a relation which, according to the definition of the term, must not exist. They are typically to modeling errors or incorrect use of a relation attributed and should not occur in operation. Therefore, in the context of this Work no automatic way to resolve inconsistencies provided.

When deriving conclusions, implicitly contained logical connections are determined and explicitly presented. That is, through the logical description of a object (concept, individual or relation) the existence of another property can be inferred, which is not represented in the model, e.g. B. the Conceptual affiliation of an individual or relation between individuals. This will the knowledge base is complemented and only then do the implicit connections become apparent usable. For example, from a transitive relation *haspart* from *production plant* to *module 1* and from *production plant* to *conveyor belt 1 ,* it can be concluded that a product from the *production plant* also comes to *conveyor belt 1* and thus the relation *haspart* is inserted.

application layer

In the application layer, services are executed that solve application-specific problems, such as B. Human-machine interaction, orchestration or diagnostics. Through a uniform model and standardized access to the model
these services can be used without adjustments for different production plants
will. In addition to the typical in-house services, this also enables the use of external services, which can be rented, for example, according to the software-as-a-service paradigm (SaaS). The services do not have to be available on a central system
SPARQL also enables access via the Internet, e.g. B. with Apache Jena Fuseki [232]. So every user can choose the right service, and it can be a
There will be competition for high-performance, resource-saving or cost-effective services.

There is one service that is not individualized but is an integral part of the application layer. This *update service* monitors the semantic model for consistency and executes calculation rules after the model has been updated. There
not all services necessarily identify an inconsistency, this service takes care of that Monitoring and notifies the user if necessary. The calculation of values becomes like this executed only once, resulting in consistent performance. Furthermore
this reduces the complexity of the services, especially since the calculation
of nested parameters is complex. This always stands for the other services
a consistent and up-to-date model is available.

Because services can transfer data into the model, they can interact with each other additional services can be realized. So there can be an application for service teams, which, in the event of an error, are shown the diagnostics determined by the system and then enter which was the faulty component. This shows how easy it is to extend the approach. In addition, e.g. B. with little effort life cycle information
Services are collected and processed by service teams. In addition, it is also possible Exchange information with other systems. For this purpose, services can implement interaction protocols as designed by the Industry 4.0 platform [13].
However, these use cases are not considered in detail in this work.

Subsymbolic Algorithms

Sub-symbolic algorithms are able to extract additional information from process data to be generated via the production plant and without any manual effort. With these additional information, new and more complex services can be implemented would not be possible without a subsymbolic algorithm. About the description in Using the semantic model, services can identify and configure available algorithms. By choosing the algorithm, setting the parameters and choosing
of the inputs, services can generate new information using machine learning processes. So e.g. For example, in use case 3, the *HyOTALA* algorithm is used.
The algorithm detects deviations from the learned model and reports an anomaly that specifies the deviation. In order for the data to be integrated into the semantic model, each algorithm has an identifier, ie a name and a
uniform format in which the data is made available. With that they can

Results are displayed and used independently of the algorithms, eliminating the need for algorithm-specific user interfaces.

summary

In this section, a concept for a runtime environment was presented, with which the adjustment effort for services in modular production systems should be minimized. The requirements from Table 6.1 were implemented. The data collection is easily adaptable, so that little manual effort is required. Algorithms can configured and integrated so that the results are available in the semantic model. The process data is also integrated into the semantic model. Several services can access the semantic model via a standardized interface access. All requirements have thus been successfully implemented. The generic ones Interfaces keep the runtime environment open so that it can be expanded and can be individualized.

6.2 Implementation in the demonstrator

The runtime environment presented in the previous section was implemented for the evaluation. In this section, the test system used for the evaluation is presented, as well as some implementation details.

6.2.1 Versatile production system of the SmartFactoryOWL

The VPS, which is in the SmartFactoryOWL, becomes an evaluation of the approach used. In the production plant, corn is processed into popcorn. The ability to change is achieved, among other things, by a modular structure of the system, in which each module has its own controller. The controls use a skill-based approach, which combines complex functions of a module into skills that are simply called can become. This makes it possible to add individual modules with relatively little effort remove or add. For this work the four initial modules are used, ie the *delivery, storage, dosing* and *production* modules shown in Figure 6.3 see are. There is an OPC UA server on each controller for data acquisition and control available, which can be accessed via the *Profinet* fieldbus system.

The incoming corn enters the production system via the delivery module. The module consists of a silo, three proximity sensors, a conveyor belt and an aspirator. The silo serves as a buffer and is monitored by two proximity sensors monitored, which recognize when the silo is full or empty. The conveyor belt transports the corn via a hopper to the compressed air operated aspirator that pulls the corn promoted to the next module. The connection between the modules is realized via hoses that are connected with a GEKA coupling. At the funnel above A proximity sensor is attached to the aspirator, which detects a blockage in the hose

Figure 6.3: The four modules of the VPS from the SmartFactoryOWL are used for evaluation of the approach used.

detected. A total of three sensors, two actuators and a controller are installed in the module.

The storage module has a similar structure. The corn arriving via a hose is filled into a silo, where a sensor also warns of an overflow and another warns when the silo is empty. The silo is emptied via a vibrating channel, which is driven by a pneumatic muscle. The removal also takes place via an aspirator, which is checked for blockage by a sensor.

The module also contains three sensors, two actuators and a controller.

The corn arriving via a hose is also fed into a silo of the dosing module. The silo is attached to a load cell so the weight of the corn can be accurately determined. As a result, the fill level of the silo is known and no additional sensors are required on this silo. The silo is emptied by a screw conveyor, which enables precise dosing. As in the other modules, the corn is fed into an aspirator via a hopper and transported to the next module. The dosing module contains two sensors, an actuator and a controller.

In the production module, the incoming corn is fed directly into the reactor. The production process is a batch process, so the process time and lot size are relevant parameters for productivity. So only a certain amount of corn can be filled into the reactor. The reaction of the corn popping up is caused by a hot air gun which is switched on for 100 seconds (process time). Due to the lower weight-to-volume ratio, popcorn is carried out of the reactor with the hot air and then down a chute past a switch into a cup

Libra headed. There, a certain amount of popcorn is weighed and the popcorn is filled into the packaging by turning the cup, with a sensor checking that the packaging is in the correct position. When the scale's target weight is reached, the diverter switches so the popcorn goes into a large bin. The reactor must be opened at regular intervals to remove unpopped corn kernels. A sensor monitors whether the reactor is open. In addition, the diverter has a sensor for checking the two positions, the scale itself is a sensor, and the rotation of the cup is also monitored by two sensors.

With the hot air gun, the air flow and the heating element are controlled separately so that they can be viewed as two actuators. A stepping motor switches the points and the scale's measuring cup is rotated by an actuator powered by compressed air. In total, this module has four actuators, seven sensors and one controller.

Most of the modules can be interchanged because the mechanical interfaces are compatible and the control programs are correspondingly flexible. Only the delivery module cannot have an upstream module since it has no input interface, and there cannot be a post-production module since a packaged product is not a valid input for any module. Taking into account that the production module should be present in the configuration and under the boundary conditions described above, there are eight valid configurations.

6.2.2 Implementation/Libraries used

The implementation was made in Java. The Java Development Kit (JDK) was used in version 12, the Java Runtime Environment (JRE) in version 8u211. Java is the standard programming language in the field of ontologies. The two well-known OWL libraries *OWL API* and the *Apache Jena* framework are written in Java, as is the ontology editor *Protégé,* which uses the OWL API. In this work, *Apache Jena* version 3.13.1 was used to manage the ontology. Jena is free, provides all required functions and is well documented. It includes an interface to RDF (RDF API), a SPARQL 1.1 compliant query engine (ARQ), a triples store (TDB), SPARQL queries over the web (Fuseki), interfaces to RDFS and OWL (Ontology API), and an inference engine no (Inference API). The Triple Store (TDB) and Fuseki are not used in the implementation of this work, but could be of interest for commercial use as they increase performance and flexibility. The Inference API provides various reasoners that have different expressiveness and thus different runtimes. In this work, the Reasoner *mini* is used, which, contrary to what the name suggests, has the second greatest expressiveness, which is only slightly smaller than the Reasoner *full.* The expressiveness of the *mini* is sufficient and the clear runtime advantages speak for it. External reasoners can also be integrated, but the selection of available reasoners is limited and not necessary due to the flawless function of the integrated system. If reasoning is of greater importance, e.g. B. in large models, the use of the

OWL API makes sense because there is a larger selection of external reasoners for it.

The connection of the production plant or the individual modules is implemented via OPC UA. A Java Software Development Kit (SDK) of an OPC UA client from Prosys in version 2.2.0 is used for this. The information model can be searched to identify the signals of the plant. However, the implementation is more complex and the result is the same, which is why the simpler variant was chosen for the proof-of-concept and this is carried out manually. The more complex implementation in connection with a discovery server brings a clear advantage. However, this server was not implemented because it is not directly related to the concept presented and does not bring any advantages for the use case under consideration with four modules.

Version 1.10 of the Tweety library was used to process logical expressions of the consistency-based diagnosis in application 3. Tweety offers comprehensive functions for many logics and the documentation is good, so that the implementation effort is low. The focus of the library is on the logical aspects of artificial intelligence and knowledge representation. The consistency check is carried out by the included Sat4j solver. [233, 234]

The implementation aims to show the feasibility of the approach by implementing the three use cases. Efficiency can still be increased in many implementation details. The tools used, e.g. B. the reasoner or the framework, just as influence on the performance as the formulation of the SPARQL queries.
Due to the objective, these topics were not considered further in this work.

6.3 Evaluation Use Case Interaction

The aim of the use case interaction is to automatically determine KPIs using the semantic model. In this section, the VPS of the SmartFactoryOWL is checked to see whether the goal is achieved with the presented concept. For this purpose, a model is created for each module, which is combined into an overall model during the configuration phase. The KPIs are then determined from this overall model. This process is performed for four different configurations of the VPS. With the conventional determination of the KPIs, the calculation must be adjusted manually for each configuration. The correctness of the determined values and the required computing time are evaluated so that the user experience is not negatively influenced. For this purpose, the logical structure of the solution from Figure 6.2 was implemented and the necessary service was implemented as a prototype.

6.3.1 Test Scenarios

The three test scenarios are described below. These were chosen to cover different aspects and thus pose different challenges to the model. The following configurations are used for this:

- Use of all four modules as described in Section 6.2.1
- Use of the last two modules (dosing and production)
- Use of the first two modules (delivery and storage)
- Use of the first three modules (delivery, storage and dosing)

The first configuration with all four modules was described in section 6.2.1. This is the typical configuration and serves as a starting point. The difficulty is at not higher in this application, the computational effort only increases with the size of the model, so that the greatest computational effort is expected with this configuration will.

In the second configuration, only the last two modules are used. that can e.g. This can be useful, for example, if only a small quantity is to be produced that does not require continuous loading of the dosing module. Therefore, this represents a typical configuration in flexible production with small lot sizes. Determining the KPIs changes compared to configuration 1 due to the absence of some modules.

As a third configuration, only the first two modules delivery and storage are used. Such a configuration is conceivable when a delivered product should not be processed immediately. The peculiarity of the configuration is that in only a few sensors are installed in both modules, so that the calculability of the KPIs is impaired. There are no sensors installed that recognize the product, so that KPIs based on them, e.g. B. production counter, throughput or the OEE, not calculated can become. This application differs the most from the other test scenarios.

In the last configuration considered, the plant is without the production module operated, so that a certain amount of corn is issued here, e.g. B. to a packaging module. The product is different in this configuration, so the configuration reflects the character of the flexible use of production modules. Since the Determination of many KPIs based on the sensor values of the last module, e.g. B. the production counter, quality control or production time, many KPIs are driven by others sensors detected.

Many challenges can be presented with these four use cases, since modules with different equipment are used for determining the KPI will. This results in changes that normally require manual adjustment require. An automatic determination would offer added value in the form of time and resource savings.

6.3.2 Test Execution

The evaluation takes place exclusively in the operational phase. The correct creation of the overall model is therefore only evaluated indirectly. To get correct results, Both the modeling and the assembly of the overall model must be correct to be lost In order for the overall model to be created correctly, the topology be known to the production plant. Since the VPS has no technology to change the topology

recognized independently, this must be specified by the user. Then can
all models are put together and the necessary relations are inserted. Also the
Relations for the calculation rules of the KPIs have to be adjusted because they are
determined across modules and therefore have to be adjusted to each new configuration.
For this purpose, the calculation rules defined at the concept level are transferred to the
instance level and assigned to the corresponding individuals there. The interpretation
The merge component is responsible for a suitable assignment, since the concept level only
the form dictates, but not the specific allocation. When all assignments have been made,
the overall model is finished and it goes into the operating phase.

In the operating phase, the overall model is loaded, an OPC UA client is initialized,
connects to all modules of the production line, and the service for calculation
the KPIs is initialized. The OPC UA client provides the signal name for each signal
and the associated value. In the mapping layer, the individual with the
corresponding signal names are determined and the value updated. When all signals have
been updated, the calculation service is called to determine the KPIs. Around
Working with a consistent dataset will not update at this time
of the values. After the calculation, all KPIs are available in the model and can be displayed
to the operator or used in other ways. All queries and
Updates to the model are made through SPARQL.

For the evaluation, all data, ie the current values, which the individuals
assigned, and the calculated KPIs recorded at ten different points in time and checked
manually. The evaluation is carried out with ten repetitions,
so that the production plant is in different states and different values are calculated and
randomly correct values are recognized. The number of
Repetition was chosen arbitrarily, but is considered sufficient since
the calculation will not sporadically deliver incorrect values.

The *total power consumption, total energy consumption, throughput* and *overall* system
efficiency values are used for the evaluation . These cover a wide spectrum
on values, which contain all the challenges of the modeling presented.
If these values are determined correctly, the challenges have been successful
mastered. Since not every module of the VPS has a separate power or energy measurement,
the measured value was randomly selected in the data acquisition
divided and distributed to the existing modules so that the calculation of the values can be
tested. In addition, the VPS does not have a sensor that can determine the quality of the
end product determined. A static value was therefore used for the quality factor. However,
both limitations relate to changes in data collection
and do not influence the rest of the procedure.

The tests were carried out on a notebook with an Intel Core i5-8250U CPU, 16 GB of RAM,
64-bit Windows 10 operating system and an SSD hard drive. Execution
takes place directly from the Netbeans 10.0 programming environment. For the calculations
only uses one CPU core, only the connections to the OPC UA servers
are executed in different threads, so the data query was executed on multiple cores.

6.3.3 Results

Figure 6.4 shows an example of a KPI, the total power consumption. the
Assignment to the calculation rule with the relation *has calculation* was made when creating the individual. By calculating, the value became the unit
and assigned the time reference. With these three pieces of information, the value can be clearly interpreted and used. The values highlighted in yellow are conclusions,
determined by the reasoner, ie these were not explicitly modeled.

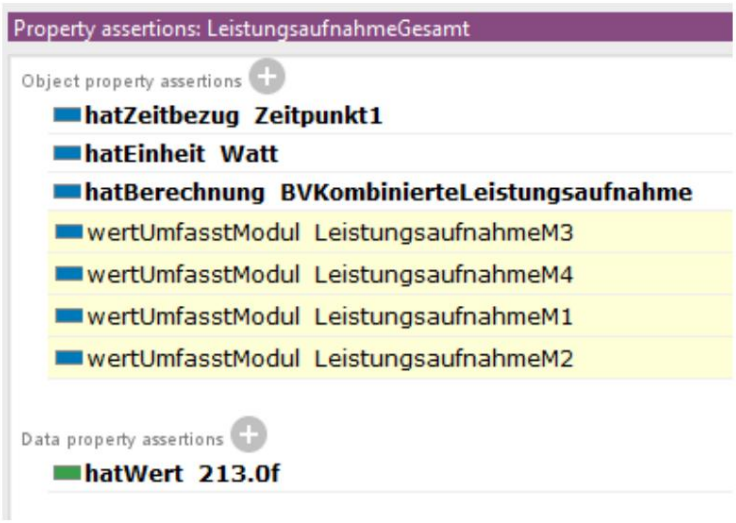

Figure 6.4: Section of the total power consumption from the overall model as a screenshot from Protégé.

The results of the test series with four different configurations and four KPIs, each with ten iterations, are presented below. In addition, the time required to calculate all values is specified for each configuration. The results
are shown in Table 6.2. An X means that all values were determined correctly,
while - means that the values are not determined due to the configuration
could. An incorrect value was not determined in any configuration, only in configuration 3 was it not possible to determine the throughput and the OEE. the reason for this
are missing sensors.

Table 6.2: Overview of correct calculations of the various KPIs.

	total power	total energy	throughput	GAE time	
configuration 1	X	X	X	X	294ms
configuration 2	X	X	X	X	173ms
configuration 3	X	X	-	-	100ms
configuration 4	X	X	X	X	199ms

The times given for the calculation in Table 6.2 are the means of 100
calculations. As expected, the size of the model has a direct impact
on the calculation time. In particular, the long calculation time for configuration 1

surprised because the model is not significantly larger than configuration 4. Since only three values have to be calculated in configuration 3, the calculation is as expected done faster. Times are sufficient for user interaction as a
Most of the values are integrated over hours and therefore do not change suddenly. However, there are also a number of measures that can be implemented to increase the efficiency of the implementation. All calculation rules, unit conversions and units are queried for each calculation or the units
are reset. Since the values within a configuration typically do not change change, these could be stored in an internal memory. This could, for example 2/3 of the SPARQL queries are saved. Furthermore, there are increases in efficiency possible through the use of other tools or the extension to multithreading.

6.3.4 Discussion

The evaluation has shown that the concept presented in this work and the modeling are suitable for the presentation of KPIs. In all configurations examined, all KPIs could be determined correctly. Only in one configuration could
Two KPIs cannot be determined due to missing sensors, but this is also only possible manually with a great deal of background knowledge. Due to the lack of ability of the VPS to determine the topology itself, this had to be entered manually. All further steps were carried out automatically. The calculation rules defined at concept level were transferred to instance level. The assignment to individual
Instances are performed by the merge component through a predefined procedure. Even in the prototypical implementation on a normal notebook it was
the calculation time of the KPIs sufficiently fast. Overall, the approach is flexible and reduces the manual effort to determine the KPIs significantly.

6.4 Evaluation Use Case Orchestration

In this section it is checked whether the modeling presented in chapter 5.2.2 for the assessment of the compatibility of production modules is appropriate. For this they will Compare descriptions of modules and products to decide if they are mechanically compatible. For the evaluation, the VPS of the SmartFactoryOWL
used.

6.4.1 Test Scenarios

With the four modules of the VPS, there are 16 possible paired connections between the modules that can be evaluated. However, four of them correspond to the compound of a module with itself. These four possibilities have practically no relevance, however, the results should be just as correct. Therefore, for reasons of representation, they are still considered. It should also be noted that the order of the modules is crucial. So e.g. B. deliver the *dosing* module to *production ,* the *production*

but not for *dosing* because the *doser* cannot handle popcorn and there is no proper connection for the two modules. Since the VPS has a sequential production flow, there is no branching of the product flow in any scenario.

All properties described in Chapter 5.2.2 are used for the compatibility check. Only the service-specific descriptions are not considered, since these are application-specific. No additional services are required for normal operation, but an example is given in use case 3.

6.4.2 Test Execution

The evaluation of this use case is carried out offline, ie without a connection to the VPS, since no current values from the system are required. Only the semantic models of all modules are required. The product description would typically be provided by a planning system. However, only two different products are processed in the VPS, corn and popcorn, which is why the product descriptions were created manually. The product description was added to the module description so that each module also provides a product description. This is possible because each module in the VPS always provides the same product.

However, this is only a different provision of the data, the rest of the process is not affected by this.

Compatibility is checked by a service, as described in Chapter 5.2.2. Two modules are checked for compatibility with the associated product based on their description. The service only needs access to the two semantic models, which is done via SPARQL. The properties of the two semantic models are determined separately and then checked.

It is therefore not necessary to combine the models. Properties checked in this scenario include mode of *operation, throughput, compound, material, density* and *grain size.* The service then outputs the result of the check.

6.4.3 Results

The results of the semantic model are presented in Table 6.3. The first module is shown in each row, ie the module for which the output description is used for the compatibility check. The columns each represent the second module, i.e. the module that receives the product, so that the input description is used here for the compatibility check. All combinations are shown in the table, including combinations of the modules with themselves. B. two modules of type *delivery* are not compatible, which is indicated by a -. An X means that this module combination is compatible.

The results were shown to the person responsible for the plant, who confirmed the correctness of the determined results. Thus, the compatibility could be correctly determined from the modeling created. To determine the duration of the compatibility check

Table 6.3: Overview of the compatibility determined by the semantic model between the modules.

	input module			
	Delivery	**Storage**	**Dosing**	**Production**
delivery	-	X	X	X
storage	-	X	X	X
dosage	-	X	X	X
production	-	-	-	-

100 tests were carried out in different configurations. An average of 71 ms per calculation was determined, using the same notebook as in
Use case 1 was used and the application was not multithreaded. the
Differences between different configurations were small. As in the others
Due to the prototypical implementation, this value can only be used as a guide in certain use cases. However, in the configuration phase the time constraints are not
very strict, which means that the result is available quickly enough.

6.4.4 Discussion

In this section it could be shown that the modeling created in this work for the orchestration of production modules of the VPS works correctly. To
all possible combinations of the modules were evaluated by the created service. The results were checked by an expert from the VPS. The average
Checking time for two modules was relatively short at 71 ms, so this is not
entails limitations in use. When the models are created, there is no
manual effort necessary; the check is carried out automatically. So here it is
The concept presented and the modeling for the orchestration of modular production systems are well suited.

6.5 Evaluation Use Case Diagnosis

In this section, the modeling presented in Chapter 5.2.3 is evaluated. For better structuring, three error classes are defined, into which every possible error can be classified
can be categorized by production systems. In addition to the error classes, different configurations of the VPS are presented, since different errors can occur in each configuration. However, mainly cross-module error causes are considered, since these are more complex and the modeled relationships in the

need the full semantic model. This allows the suitability of the diagnostic algorithm and the semantic model to be evaluated.

6.5.1 Test Scenarios

In order to evaluate the diagnosis use case, three generally valid ones are first identified Error classes are presented, into which each error can be classified. Because of the multitude of errors only a few are presented in detail. If possible, the errors are included Cross-module cause-effect relationships are described, as these are of the greatest relevance in the present work.

Error class 1: Discrete error without time delay
In this error class, an event immediately triggers the erroneous behavior. Applied to production systems, this means that the impact of the event is within of a cycle of the PLC must be recognizable so that in the next observation the effect becomes apparent. This error class is the typical area of application the classic consistency-based diagnosis. In real production plants there is However, there are only a few errors of this class, since the reaction of the mechanics is frequent is slower than the cycle time of the PLC.

Error class 2: Discrete error with a time delay
In this error class, the effect of an error in a discrete signal is not immediately visible. A so-called dead time between the triggering event and the effect is often found in production plants. Because the consistency-based diagnosis does not offer any suitable description options for this dead time, the handling of this error class is a challenge. The handling can't go through the diagnostic algorithm, but must be manually integrated into the system will.

Error class 3: Continuous error
This error class contains continuous signals in the cause-effect relationships and is only made possible by the RDA. Typically, production plants have an appreciable number of continuous signals, the affect the process and cause errors. A distinction regarding the time delay is not necessary in this error class, since the learned continuous models show the development over time. Thus, any time delays do not pose an additional challenge.

In order to evaluate the flexibility of the approach, the tests are carried out with different module configurations. Four module configurations are presented below that are suitable for evaluating the use case.

- **Configuration 1:** In this configuration, all modules are in order *delivery, storage, dosing* and *production .* This represents the standard configuration in which the VPS is often operated.
- **Configuration 2:** The *storage* module is removed from configuration 1, so that the *Delivery* module is directly connected to the *Dosing* module . This configuration is possible, for example, with a continuous delivery of corn.
- **Configuration 3:** Instead of removing the *storage* module, you can also use the module *Dosing* can be removed from configuration 1. So that's not a very accurate one

Dosing possible, but the process runs faster and there are operating costs saved for the module.

- **Configuration 4:** As a minimum configuration, the modules *Delivery* and *production* available. In this configuration, the complexity is low and, due to the small number of modules, so are the operating costs.

All possible errors for each configuration of the VPS are assigned to an error class in Table 6.4. It can be seen that the error class 1 only to a small extent occurs. Error class 2 is the most common in the VPS considered here, but error class 3 also occurs more frequently. This distribution is relatively typical, where the division between error classes 2 and 3 depends on the components used is. There are also systems in which error class 3 predominates.

Table 6.4: It is the number of possible errors for each of the four configurations used shown, with each error being assigned to an error class.

	Error class 1	**Error class 2**	**Error class 3**
configuration 1	3	13	8th
configuration 2	3	10	8th
configuration 3	3	12	4
configuration 4	3	5	4

The number of possible errors in the VPS is too large to list them all. Therefore, in One error from each error class is presented below to give you a concrete idea to give about the mistakes. Class 1 errors only occur within a module on, while error class 2 and 3 also include cross-module errors.

Example of error class 1: A switch is installed in the production module that popcorn to the scale or into a large container. The two end positions are equipped with sensors so that the correct switching of the points can be monitored. The switch is operated via compressed air and the adjustment does not require any large forces, giving it a fast response time. is in normal operation one of the two limit switches is always active. Typically, only one comes in the event of an error limited number of components to consider, namely only those that are immediate affect the faulty component. In this specific case, the cause can be a faulty actuator, a faulty sensor or jammed popcorn, which the switch is blocked.

Example of error class 2: In this error class, the proximity sensor for level monitoring of the store can report that the store is empty. Despite the delivery of new corn, the value of the sensor does not change even after some time. On the one hand, the challenge lies in the chronological sequence of the cause-effect relationships the effect can only be observed with a fairly long delay (more than 5 seconds). can. In addition, the cause-effect relationships are cross-module, ie the system description is divided into several semantic models and must be checked before the diagnosis be put together. As is typical for this error class, many components can be the cause of the error. Specifically, in the *delivery* module , the conveyor belt,

the level sensor or the aspirator may be defective. In the case of a defective aspirator
the monitoring sensor detects corn. Detected by the arrangement of the sensor
However, this does not cut the corn in the first production cycle because of the amount of corn
not sufficient for this. In the *storage* module , the sensor can be used to monitor the filling level
be defective or the conveyor belt unintentionally conveys out of the warehouse, e.g. B. due to a
defect in the control. In the latter case, the sensor on the aspirator would
Time to detect corn if its function is given. It can be seen that the
Cause-effect relationships are more complex.

Example of error class 3: An error of this class occurs in the VPS, e.g. B. on when in
the scales of the production module do not receive enough or no popcorn, although corn is in
the reactor should be filled. The production process cannot be monitored using the sensor data, ie
the cause-effect relationships cannot be divided into sub-steps
be disassembled. There are many possible causes for this error in the *dosing* or *production modules.*
So could the screw conveyor and the scales of the dosing module
be defective, the aspirator and possibly the sensor for monitoring the aspirator. in the module
production , the heater, fan or popcorn scale could be defective. In contrast to error class 2,
however, the diagnostic process is only triggered when it actually occurs
there is a deviation, so that the error is reacted to appropriately.

Error classes were introduced in this section and it was shown that all error classes occur in the considered VPS. In addition, various configurations of the VPS were presented and possible errors were assigned to an error class. In
all configurations, all error classes are represented as well as cross-module ones
Cause-effect relationships exist, so the VPS used is well suited for the evaluation.

6.5.2 Test Execution

In preparation for the test, system descriptions were created for each of the four modules. In order to simplify the modelling, these were created using software in
integrated the semantic model, so that only the logical expressions had to be created manually, but not the modeling of the semantic model. So could
modeling errors are excluded as far as possible. Furthermore, the merge component has been expanded in such a way that the application-specific interfaces of the system description can also be automatically combined. Besides the expressions
the classic system description can also contain residuals and residual interfaces
are processed. The interfaces for the evaluation were exclusively included
the residuals x^{+} and $x^{ÿ}$ modeled, which contain more information than the residual x^{0}. If both variants are used mixed, only the simpler Va 0 can be inferred. Furthermore, a function riante x was implemented that
the *COMPS* components are determined from the system description and the semantic model. As a result, the overall system description can be completely and automatically omitted
create the classic modular system descriptions without manual intervention
are necessary.

The evaluation was conducted offline. For this purpose, data from the VPS via OPC UA recorded. The HyOTALA algorithm was trained with this data learns an automaton from the binary values and creates a model of the continuous values in each mode. Self-Organizing Maps (SOMs)) used to represent the continuous values [235, 236]. The SOMs had a size of 20x20 neurons with a toroidal topology, ie the Corners of the map are connected. Over 100 epochs were learned and an error threshold *(ÿ)* of 1 was set because good results were obtained with these values. After training the models, a dataset containing an anomaly was used for evaluation. The results of the algorithms were integrated into the data set. This contains in particular the information as to whether and which anomalies are present, as well as the causative signal.

The sequence described here can also be applied online by using the data of the production plant can be used directly for evaluation in HyOTALA. For reasons the reproducibility was omitted. As the part of data provision is located in the mapping layer, this has no influence on the sequence in the semantic model, the service or on the results.

The observations obtained from the file are read in by the mapping layer and written into the ontology. This includes the values from sensors and the Residuals determined by the SOMs. The hybrid diagnostic algorithm was realized as a service and receives all input data from the semantic model. For this purpose, the information about SPARQL is queried in the model and then to the diagnostics used. The RDA is always called when new data has been stored in the semantic model. It is checked whether there is an inconsistency. If this is the case, appropriate diagnoses are determined, issued to the user and integrated into the semantic model.

The suitability of the modeling of the semantic model should be checked. To it is examined whether the correct error causes are determined, since this depends on the correct interpretation of the system description, the components and the observations can be closed. To do this, however, it is also necessary to explain the algorithm in more detail considered so that the correctness of the results can be assessed. As with the other use cases, time is also considered, although it is expected that the values are higher due to the prototypical and non-optimized implementation. The basic order of magnitude can nevertheless be estimated.

6.5.3 Results

For the evaluation, the possible ones determined by the diagnostic algorithm are used first Causes of errors are considered, which should be correct and complete. For this purpose, the identified causes of error were assessed by an expert. Table 6.5 shows the correctly determined error causes for all configurations and all error classes. This Results are explained in more detail below.

As expected, the correct error causes were identified in all configurations for error class 1 errors. In error class 2, all error causes were also identified

Table 6.5: Number of error causes correctly determined for each of the four confi gurations and the assignment to the error classes.

	Error class 1	Error class 2	Error class 3
configuration 1	3	*13*	8th
configuration 2	3	*10*	8th
configuration 3	3	*12*	4
configuration 4	3	*5*	4

determined correctly in all configurations. Because the time lag between the cause and the effect cannot be shown, the diagnoses are based on the principle identified at the wrong times. To use the method, these premature diagnoses must be intercepted and only displayed to the user in the event of an error. the Results are therefore correct, but still need further processing, which is why they are in Table 6.5 are shown in italics. This limitation does not exist in error class 3, so that here, too, all causes of errors in all configurations were correctly and completely identified. The temporal behavior is intercepted by the continuous model, so that the exact time of the error is known.

In all three error classes, no changes between the different configurations could be determined; the causes were correctly determined in all configurations. It could thus be shown that the presented modeling approach works in modular production plants and that the diagnosis can be carried out without manual effort in the

reconfiguration is possible.

The results of the RDA are mapped in the semantic model as described in Chapter 5.2.3. This includes the triggering component, the time of the error and the determined possible causes of error. This is shown as an example in Figure 6.5, as a screenshot from Protégé. For documentation purposes, the relation *hasPossibleCause* to Cause can be replaced by the relation *hasCause* and an end time can be added will. This means that an additional service can be used to quickly create evaluations, e.g. B. which device causes how much downtime.

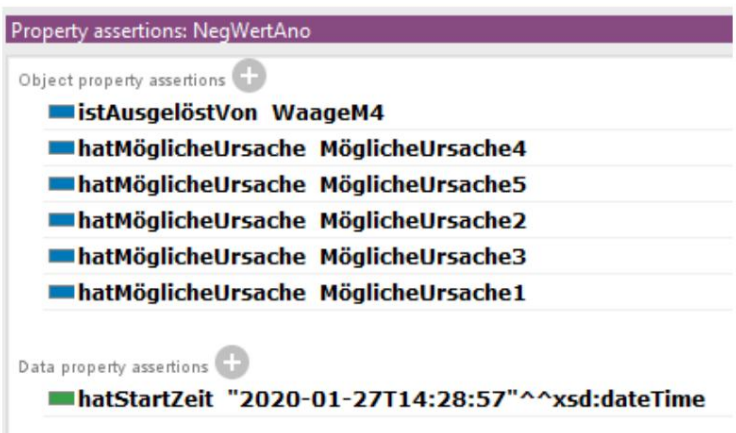

Figure 6.5: Representation of the identified error as a screenshot from Protégé.

The time that the RDA needs is particularly dependent on whether an inconsistency
present. If there is no error, the system description check takes about 105 ms on average. Reading in the system description, the components and the
observations takes about 78 ms in total. Results will be in this bug free case
not integrated into the semantic model, so no time is required for this.

If there is an inconsistency, the time increases exponentially with the number of components. This also results with the standard algorithm used without optimization
for the small system times from several minutes to almost an hour. Through
the division of the system description into modes described in chapter 5.2.3, the
time required can be significantly reduced. Since the assignment of the modes is not general,
but can only be made for a specific system, the modes will follow
added to continuous model learning. With this one-time calculation,
which takes place automatically, the times for determining possible causes of errors can be reduced to just a few seconds. The most complex calculation thus lasted in the
presented example only 3.2 s. The duration is not the overall system size
decisive, but only how many components are considered in the current fashion
Need to become. Overall, this results in an enormous time advantage, which significantly increases the applicability for complex systems. There is, however, the restriction
that the use of modes is only applicable to continuous variable errors
is. When a binary signal changes its value, the mode also changes.
This means that an incorrect system description would be used. The HyOTALA algorithm, which detects anomalies in binary signals, can help here
which allows the correct system description to be selected. Through this
expansion, the time required for the calculation is also not decisive in the case of consistency, since a calculation is only made if there is an error.

6.5.4 Discussion

In this section it was examined whether a hybrid modular consistency-based diagnosis can be realized with the semantic model developed in this work.
For better structuring, three error classes were defined and four different ones
Configurations of the VPS used for evaluation are presented. In error class 2 in particular, the limitation of the consistency-based approach became clear. Times cannot be represented, which can lead to difficulties with cause-effect relationships in which the effect is not immediately related to the cause

entry. In error class 3, the time is implicitly present because the learned models
of the continuous signals they represent. Overall, very good results were achieved, as all error causes determined by the algorithm were correct. The duration of
Calculation of possible error causes can be very small due to the approach used
be kept, which also makes it possible to use the approach in large systems
is. The semantic model has mapped all relationships correctly and understandably.
The modular system descriptions could be aggregated without manual effort when the module arrangement was changed. It could thus be shown that the
The approach presented here is very well suited for diagnostics in modular systems.

7 Critical Discussion

In this chapter, the solution approach of this work is critically examined in order to show the potential and limitations of the approach. The need with a semantic Solving several use cases in a model has already been identified in several publications [13, 16, 17]. But only in this work could a concrete implementation to be shown. For use in a real production environment, however, further criteria must be taken into account, such as e.g. B. the effort involved in creating the model, conflicting goals or limitations. First, the possibilities of the approach are highlighted. Then the challenges regarding the model creation, the necessary implicit knowledge, compatibility and the possibility of error. Finally, a conclusion is drawn.

In this work it was shown that with OWL DL there is an ontology language that all meets the requirements of a semantic model in the industrial environment defined in Chapter 3 (FF 1). This ontology language can be used to describe modular Create production plants as shown in Chapter 5.2 for three use cases (FF 2). This makes it possible to provide relevant information via a uniform interface, so that services for different use cases can be implemented based on this can be defined, see Chapter 6. Although no parts of the model could be defi ned as universally valid, it was shown that the number of terms for technical use cases converges. This makes it possible to create a semantic model that can be used for many technical applications. Thus, a reusability given (FF 3). The semantic model makes it possible for services to be automatically adapted to the production plant and the functions implemented with it without manual effort can be used. This is an important step towards the efficient production of small batch sizes with modular production systems. Also can Services created by third parties and similar to applications on the smartphone be used. This results in a new business field for innovative services for companies. Up to now there was no access to the data of the production plant, so that the services always had to be adapted to the specific plant. In the semantic model, terms and relations can be defined multilingually, so that the language for can be easily adapted to the user and services are also offered internationally can become. Consistency is checked by a reasoning component, which can also draw conclusions about implicit relationships. Together results This enables easy handling of higher-level functions in modular production systems that can be used without manual adjustments.

In order for the above advantages to be realised, models of all production modules must be available. So far, however, the system model has been created still manual, which creates additional effort. An automatic creation or

Integration of information from other sources, e.g. B. from AML, would accelerate the model creation. This was due to the large scope in this feasibility study
not considered. Furthermore, only one production plant may be mapped in each semantic model, as this simplifies the creation, use and management in the runtime environment and increases the efficiency for inquiries and conclusions. Since several semantic models are operated in parallel on one server

can be, this circumstance does not represent a significant restriction
there are conflicting goals when creating the semantic model. So e.g. B. between compactness and expandability or comprehensibility. In this work
the criterion of extensibility was generally given higher priority, since there are niche applications in production that are not fully covered by the model, and thus
extensions are needed. However, for expandability is not just the number
of the terms is decisive, but also the degree of description of the individual terms and
relations. Are terms and relations defined very precisely, ie they contain many
Restrictions, this limits the extensibility of the model. Especially relations
can then no longer be used for similar relationships, and new relationships must be defined. Also increases with more detailed description
usually also the complexity of the model, which increases the time for the consistency check and
for identifying conclusions increases. The goal is a good compromise between the uniqueness of the terms and relations and the extensibility. This is for
the three considered use cases succeeded.

In this work, the three use cases were implemented with the help of the semantic model. However, there are some limitations when using the model
the implementation of services must be taken into account. The model represents a lot of knowledge explicitly, but implicit knowledge is still necessary to use the model,
which is not shown in the model. This knowledge cannot be represented explicitly in the model, or not with justifiable effort. However, most of these points
self-evident and are therefore intuitively implemented correctly. So e.g. B. Not explicitly
shown that the time intervals when calculating values must be the same, like
unit conversion works or that with complex calculations like
of the OEE first the independent KPIs have to be calculated. Because this knowledge is not
is explicitly specified, it must be given to the creator of a service outside of the semantic model are transmitted.

The approach used in this work provides a uniform interface for services.
It was shown that the necessary technologies are available and that such a model can be created. So that an application can be implemented across manufacturers
standardization is necessary. Furthermore, it was found during the implementation that the results of different SPARQL engines easily
deliver different results. The different behavior of the engines must
are taken into account when creating services so that they do not lead to problems.
In addition, the runtime environment must ensure that the data of the production plant is available consistently in the semantic model and e.g. B. no obsolete values in the
model remain. When evaluating on a real system, the response times were
of the approach for the typical higher-level tasks, such as B. User Interaction
or diagnosis, always sufficiently quickly; However, the implementation does not meet real-time requirements.

Services accessing the model can also cause inconsistencies and thus render the model unusable, e.g. B. by the incorrect use of relations.
This could not be taken into account for the feasibility study carried out here, but for productive use concepts should be implemented that prevent these scenarios. It could e.g. For example, access restrictions for the services, backups that are restored in the event of an inconsistency, or systematic testing and certification of the services in advance. In addition, it would be possible to introduce provenance information that allows for traceability and shows the origin of changed information. Depending on the application, an appropriate approach can be selected and implemented.

In this work, a semantic model for the three use cases *interaction, orchestration* and *diagnosis* was created. It was shown that the use of the models enables information processing to be automated. After a change in the system topology, services access the information via a standardized interface and adapt themselves so that no manual effort is required to configure them. This reduces conversion times, effort and errors on the one hand, and additional functions are implemented on the other, e.g. B. a cross-module diagnostics that supports the operator and further increases the availability of the system. As part of the evaluation, it was shown on a SmartFactoryOWL production system that the approach presented here can already be applied to a real production system. As shown in this chapter, there are still a few challenges to be solved for productive use, which, however, are mainly implementation-dependent and have therefore not been considered in detail in this work. All of the challenges considered are solvable, so that the approach can be used productively, whereby the standardization of the model is seen as a prerequisite for this.

8 Summary and Outlook

This chapter summarizes the work and gives an outlook on possible future work. In the outlook, on the one hand, from a technological point of view
questions that are still open were raised, and on the other hand the possibility of integrating the work into activities of the Industry 4.0 platform was considered.

8.1 Summary

Shorter product life cycles and individualized products require adaptable production facilities. These are implemented by modularizing the production systems so that each module has its own controller and the modules can be flexibly combined. The various production modules are typically from different manufacturers and are also implemented with proprietary interfaces. The challenge
is to superordinate functions, such as e.g. B. Human-machine interaction or diagnostics, independently of the topology and without any manual effort. The implementation
this can be done with semantic technologies since they are cross-module terms
define and map the proprietary signals to these terms [8]. In this way, they enable uniform access to the data, through the services to the information
use and adapt independently. In the context of this work,
presented three use cases (interaction, orchestration and diagnostics) for which the
The approach was implemented as a prototype and evaluated.

First, 15 requirements for the formalism were defined and with formalisms
compared, which are potentially suitable for creating a semantic model. the
Requirements form the basis for answering FF 1: *Which modeling formalism for semantic models meets the requirements for industrial use cases?* Of the four formalisms examined (semantic networks, frames, rule-based systems and ontologies), only the ontologies met all the requirements.
Since several ontology languages had to be selected, five characteristics were used to evaluate the applicability of the language. It turned out
found that the ontology language OWL is best suited. Because OWL multiple profiles with different expressiveness, the necessary expressiveness became
determined and compared with the different profiles. The data to be mapped in
The use cases considered are heterogeneous, so that a relatively high level of expressiveness is required and the profile DL was identified as suitable. This could be shown that the formalism OWL DL meets all requirements for industrial applications, which means FF 1 is answered.

The semantic models can now be created with the selected ontology language. This results in FF 2: *How can a semantic model for the use cases of interaction, orchestration and diagnosis look like?* To answer the FF several publications can be accessed that were created as part of this work [18, 19, 20]. When creating the semantic models, the Use case structured and alternative illustrations in the model were sought for all parts. The best option was selected based on six criteria, so that conflicts of interest that arise are resolved as comprehensibly as possible. Whether the created Models for which use cases are suitable were checked in the evaluation.

For the evaluation, the approach was implemented as a prototype and based on test scenarios for the three use cases. In the presented runtime environment *(i)* conclusions are drawn in the model, *(ii)* signals are mapped, *(iii)* sub-symbolic algorithms are used and *(iv)* the execution of services is implemented. The runtime environment was developed as part of this work, the important aspect of the mapping has already been discussed in two publications [21, 22]. The evaluation was carried out on a modular production system in the SmartFactoryOWL. The desired results could be achieved in all application cases, so that all higher-level Services were running correctly. This enabled the suitability of the created semantic Models for solving the use cases are shown. Despite the prototypical implementation, which was not runtime-optimized, low runtimes were determined. In the Running on a notebook, depending on the application, mean runtimes of 70 to 300 ms measured, so that the requirements for non-time-critical applications be fulfilled. Only in the diagnostic use case was one of the RDA detected inconsistency, a maximum runtime of 3.2 s was observed, which is a good value for displaying the causes of errors. The longer calculation time is due to a high effort in determining the cause of the error, so that only a small part due to the use of the semantic model. For real-time applications however, the approach is not appropriate. The successful evaluation showed that the models created are suitable for solving the three use cases and to answer FF 2.

FF 3 aims to merge the models: *How similar are the semantic models of the use cases under consideration and can a cross-use case model be created from them?* For this purpose, the models of the three use cases were used compared and similarities identified. Although the use cases varied widely, over 25% of the terms were required in more than one use case. To check whether the number of terms converges, the necessary Identified terms from other use cases. Typical engineering use cases had good agreement with the existing model, so few some terms had to be added. For non-technical use cases, the Significantly less agreement, so that integration of such use cases is not an option. If you limit yourself to typical technical applications, the model converges at a few hundred terms. FF 3 can thus be answered in this way that there is not *one* model for all applications or that it cannot be implemented sensibly. But for technical use cases, a model can be created which can be used for many applications.

The approach presented here for the semantic description of production plants for Increasing the degree of automation was successfully implemented. By describing the modules, it is possible for higher-level functions to become independent can be configured and used without manual effort after the production system has been modified. This statement is supported by the good results of the evaluation. Uniform access to information makes it possible to generate generic To create services that implement higher-level functions. The use of such services is much more attractive due to the reduced effort. Services take care of you Added value and can increase the availability and efficiency of equipment and costs save, e.g. B. by a diagnostic functionality. In addition, they offer companies a new business field, since services are offered through uniform data access that do not have to be specially adapted to a system. Overall is the The approach presented here is promising and is being supported by the increasing networking of Production plants are becoming even more important as part of I4.0.

8.2 Outlook

Some insights were gained in this work, which can be used in further work. In this work, the model was still created manually. For a broader application of the approach, the creation of the model should be simplified. Existing data sources such as B. Engineering data used will. Such an approach was pursued in [142] and would have to be adapted to this work will. This requires in particular a suitable mapping between the engineering data and the semantic model are created. Using a suitable tool with an integrated A plant expert can then add missing information via the user interface without having to know the underlying formalism. Because the creation of a Such tools are complex and dependent on the specific modelling, standardization is a prerequisite for companies to have investment security. for standardization requires the support of many actors, e.g. B. with eCl@ss or oneM2M. The model presented here can serve as a basis for standardization and supplemented by additional aspects and other use cases.

When the system is put into productive use, a consistent model must always be used can be restored if an inconsistency occurs. The bigger and more diverse the models and services become, the greater the need for such an approach. One approach is to certify all components involved, thereby reducing the risk of an inconsistency is reduced, but the effort to create services is increased. the Introduction of traceability of changes in the semantic model provides a represents a different approach that is similar to the provenance data in [237]. Here can better on unforeseen events can be reacted to, but the computational effort increases using the semantic model. The easiest option would be regular However, creating backups of the model and restoring them if necessary This can lead to a small amount of data loss. So there are approaches where Advantages and disadvantages have to be weighed before they are converted into an overall concept will.

In this work, technical use cases were implemented in which a convergence of the model size could be observed. When integrating non-technical use cases, the semantic model becomes significantly larger. Therefore, a delimitation of the areas is proposed in this work. It remains to be clarified which areas are sensibly represented in different models and how the different models can be linked. For this purpose, suitable non-technical use cases must first be analyzed in order to be able to determine the corresponding requirements.

With regard to the activities of the Industry 4.0 platform, there are several overlaps with this work. So e.g. For example, many platform publications mention ontologies [13, 16] or even specific technologies such as RDF, OWL and SPARQL [17, 238] about the administration shell – a communication and management unit of an object. In terms of the technologies used, the work carried out here has an intersection with the subject areas relating to the Administration Shell. However, the activities of the Industry 4.0 platform have so far been quite general overall; the first part of a specification for the administration shell was only published in November 2019 [238]. This specifies the exchange of information between two partners in a value chain. However, this area was not addressed in the present work. Nevertheless, due to the overlaps in the use cases, it can be assumed that specifications from the administration shell will follow for the topics dealt with here. The knowledge gained in this work, modeling patterns or even parts of the model can be used for the specification activities of the Industry 4.0 platform.

List of abbreviations

ABox Assertional Knowledge.

AML Automation Markup Language.

API Application Programming Interface.

CAEX Computer Aided Engineering Exchange.

CCH Credit Clearing House.

COLLADA Collaborative Design Activity.

CPPS Cyber-Physical Production System.

CPS cyber-physical system.

DAML-ONT DARPA Agent Markup Language Ontology.

DL Description Logic (German: Description Logic).

ER approach Entity relationship approach.

EU European Union.

FF research question.

FLogic Frame Logic.

OEE Overall Equipment Effectiveness.

HBG hybrid bond graphs.

HMI Human Machine Interface.

HTML Hypertext Markup Language.

HyBUTLA Hybrid Bottom Up Timing Learning Algorithm.

HyOTALA HybridOTALA.

I4.0 Industry 4.0.

ID identifiers.

IRI Internationalized Resource Identifier.

JDK Java Development Kit.

JRE Java Runtime Environment.

KEI KPI item information.

KIF Knowledge Interchange Format.

KPI Key Performance Indicators.

MOF Meta Object Facility.

NeOn Lifecycle Support for Networked Ontologies.

OCML Operational Conceptual Modeling Language.

ODP Ontology Design Pattern.

OIL Ontology Interchange Language.

OMG Object Management Group.

OPC UA OPC Unified Architecture.

OTALA Online Timed Automaton Learning Algorithm.

OWA Open World Assumption.

OWL Web Ontology Language.

PSL Process Specification Language.

QUDT Quantities, Units, Dimensions and Types.

RDA residual-based diagnostic algorithm.

RDF Resource Description Framework.

RDFS RDF schema.

SaaS Software as a Service.

SDK software development kit.

SFM Strong Fault Model.

SKOS Simple Knowledge Organization System.

SMT Satisfaction Modulo Theory.

SOM Self Organizing Map.

SPARQL SPARQL Protocol And RDF Query Language.

PLC programmable logic controller.

SWRL Semantic Web Rule Language.

TBox Terminological Knowledge.

TCG Temporal Causal Graph.

TSN Time-sensitive Networking.

UML Unified Modeling Language.

UTC Coordinated Universal Time.

VPS Versatile Production System.

W3C World Wide Web Consortium.

WFM weak fault model.

XML Extensible Markup Language.

XOL XML-Based Ontology Exchange Language.

bibliography

[1] H. Arndt, *Influence of Megatrends on Logistics.* Wiesbaden: Gabler, 2008, pp. 8-26 Available: https://doi.org/10.1007/978-3-8349-9743-2_2

[2] B Beihoff, C Oster, S Friedenthal, C Paredis, D Kemp, H Stoewer, D Nichols, and J Wade, *A World in Motion – Systems Engineering Vision 2025.* INCOSE, 01 2014 .

[3] European Factories of the Future Research Association (EFFRA), *Factories of the Future: Multi-annual Roadmap for the Contractual PPP Under Horizon 2020,* ser. EDC collection. Publications Office of the European Union, 2013.

[4] A Barnard Feeney, S Frechette, and V Srinivasan, *Cyber-Physical Systems Engineering for Manufacturing.* Cham: Springer International Publishing, 2017, pp. 81–110. Available: https://doi.org/10.1007/978-3-319-42559-7_4

[5] J. Haupert, X. Klinge, and A. Blocher, *CPS-Based Manufacturing with Semantic Object Memories and Service Orchestration for Industry 4.0 Applications.* Cham: Springer International Publishing, 2017, pp. 203–229. Available : https://doi.org/10.1007/978-3-319-42559-7_8

[6] R. Anderl, K. Bauer, T. Bauernhansl, S. Berlik, M. Broy, B. Diegner, J. Diemer, A. Fay, J. Gausemeier, D. Goericke, J. Grotepass, C. Hilger , G. Hornung, J. Jasperneite, J. Kalhoff, U. Kubach, G. Lanza, U. Löwen, G. Menges, J.-P. Meyer-Kahlen, JS Michels, W. Nebel, G. Reinhart, C. Ripperda, H. Rödig, F. Schmidt, T. Stiedl, M. ten Hompel, and C. Zeidler, "Research Agenda Industry 4.0 - Update of Research Needs, " Federal Ministry for Economic Affairs and Energy (BMWi), Berlin, Tech. Rep., 2016. Available: https://www.plattform-i40. de/I40/Redaktion/DE/Downloads/Publikation/forschungagenda-i40.html

[7] H. Kagermann, N. Gaus, K. Euler, J. Hauck, J. Beyerer, W. Wahlster, and H. Brackemann, “Fachforum Autonome Systeme : Chances and Risks for Economy, Science and Society : Final Report - Long Version ,” Specialist forum autonomous systems in the high-tech forum, Berlin, Tech. Republic, 2017. Available: http://publications.rwth-aachen.de/record/729403

[8] Bunte A, Wunderlich P, Moriz N, Li P, Mankowski A, Rogalla A, and Nig Gemann O, “Why Symbolic AI is a Key Technology for Self-Adaptation in the Context of CPPS ,” in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2019, pp. 1701–1704.

[9] IEEE, *Standard Glossary of Software Engineering Terminology, 610.12-1990,* IEEE Standard, Dec 1990.

[10] D. Schilberg, M. Hoffmann, S. Schmitz, and T. Meisen, *Interoperability in Smart Automation of Cyber-Physical Systems.* Cham: Springer International Publishing, 2017, pp. 261–286. Available: https://doi.org/10.1007/978-3-319-42559-7_10

[11] *OPC Unified Architecture Specification; Part 1: Overview and Concepts,* OPC Foundation, Standard, Rev. 1.02, Jul. 2012.

[12] A. Bunte, A. Brozmann, and O. Niggemann, "Semantic self-description as a first step towards intelligent industrial automation (Semantics4Automation): final scientific report on the BMBF research project," Ostwestfalen-Lippe University of Applied Sciences, Lemgo, Tech. Republic, 2018.

[13] J. Vialkowitsch, O. Schell, A. Willner, F. Vollmar, T. Schulz, F. Pethig, J. Neidig, T. Usländer, J. Reich, D. Nehls, M. Lieske, C. Diedrich , A. Belyaev, J. Bock, and TH Deppe, "I4.0 Language: Vocabulary, Message Structure, and Semantic Interaction Protocols of the I4.0 Language: Discussion Paper,"
Federal Ministry for Economic Affairs and Energy (BMWi), Berlin, Tech. Rep., 2018, as of April 2018. Available: https://www.plattform-i40.de/I40/Redaktion/DE/ Downloads/Publikation/hm-2018-sprache.html

[14] T. Bauernhansl, J. Krüger, G. Reinhart, and G. Schuh, "WGP-Standpunkt Industrie 4.0," Scientific Society for Production Engineering WGP e. V., Darmstadt, position paper, Jun 2016. Available: https://wgp.de/wp-content/ uploads/WGP-Standpunkt_Industrie_4-0.pdf

[15] A. Fay, C. Diedrich, M. Thron, A. Scholz, P. Puntel Schmidt, J. Ladiges, and T. Holm, "How does Industry 4.0 become important?" *atp edition - automation technical practice,* vol. 57, p. 30, 07 2015.

[16] M. Abramovici, P. Gebus, and S. Philipp, "Engineering smarter products and services platform industry 4.0 study," acatech - German Academy of Science and Engineering Office, Munich, Tech. Rep., 2018, as of March 2018. Available: https://www.plattform-i40.de/PI40/ Redaktion/DE/Downloads/ Publikation/hm-2018-fb-smart.html

[17] P. Adolphs, S. Auer, H. Bedenbender, M. Billmann, M. Hankel, R. Heidel, M. Hoffmeister, H. Huhle, M. Jochem, M. Kiele-Dunsche, G. Koschnick, H Koziolek, L. Linke, R. Pichler, F. Schewe, K. Schneider, and B. Waser, "Structure of the administration shell - further development of the reference model for the Industry 4.0 component," Federal Ministry for Economic Affairs and Energy (BMWi), Berlin , tech Rep., 2016. Available: https:// www.plattform-i40.de/PI40/Redaktion/DE/ Downloads/Publikation/struktur-der-verwaltungsshell.html

[18] A. Bunte, B. Stein, and O. Niggemann, "Model-Based Diagnosis for Cyber-Physical Production Systems Based on Machine Learning and Residual-Based Diagnosis Models," in *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19),* Ha waii, USA, Jan 2019.

[19] A. Bunte, A. Diedrich, and O. Niggemann, "Semantics Enable Standardized User Interfaces for Diagnosis in Modular Production Systems," in *International Workshop on the Principles of Diagnosis (DX),* Denver, CO, USA, Oct 2016 .

[20] A. Bunte, A. Diedrich, and O. Niggemann, "Integrating Semantics for Diagnosis of Manufacturing Systems," in *21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Berlin, Sep 2016.

[21] A. Bunte, O. Niagemann, and B. Stein, "Integrating OWL Ontologies for Smart Services into AutomationML and OPC UA," in *23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2018, pp. 1383–1390.

[22] Bunte A, Li P, and Niggemann O, "Mapping Data Sets to Concepts Using Machine Learning and a Knowledge Based Approach," in *International Conference on Agents and Artificial Intelligence (ICAART).* Madeira, Portugal: SCITEPRESS, Jan 2018.

[23] Bunte A, Fischbach A, Strohschein J, Bartz-Beielstein T, Faeskorn-Woyke H, and Oliver N, "Evaluation of Cognitive Architectures for Cyber-Physical Production Systems," in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2019

[24] A. Diedrich, A. Bunte, A. Maier, and O. Niggemann, "Cognitive architecture for concept learning in technical systems," in *Machine Learning for Cyber Physical Systems,* O. Niggemann and J. Beyerer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 75–85.

[25] O. Niggemann, G. Biswas, JS Kinnebrew, H. Khorasgani, S. Volgmann, and A. Bunte, *Data analysis in the smart factory.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 471-490. Available : https://doi.org/10.1007/978-3-662-53248-5_73

[26] O. Niggemann, G. Biswas, JS Kinnebrew, H. Khorasgani, S. Volgmann, and A. Bunte, “Data-Driven Monitoring of Cyber-Physical Systems Leveraging on Big Data and the Internet-of-Things for Diagnosis and Control,” in *International Workshop on the Principles of Diagnosis (DX),* Paris, France, Aug 2015.

[27] A. Bunte, P. Li, and O. Niggemann, "Learned Abstraction: Knowledge Based Concept Learning for Cyber Physical Systems," in *3rd Conference on Machine Learning for Cyber Physical Systems and Industry 4.0 (ML4CPS),* Jun 2017 .

[28] O. Niggemann, S. Windmann, S. Volgmann, A. Bunte, and B. Stein, "Using Learned Models for the Root Cause Analysis of Cyber-Physical Production Systems," in *International Workshop on the Principles of Diagnosis (DX).* Graz, Austria, Sep 2014.

[29] A. Liew, "DIKIW: Data, Information, Knowledge, Intelligence, Wisdom and their Interrelationships," *Business Management Dynamics,* vol. 2, no. 10, pp. 49 – 62, 2013. Available: http://bmdynamics.com/issue_pdf/bmd110349-%2049-62.pdf

[30] P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure, *Semantic Web: Fundamentals.* approx lin: Springer, 2008.

[31] A. Hoppe, R. Seising, A. Nürnberger, and C. Wenzel, “Wisdom - the blurry top of human cognition in the DIKW-model?” in *EUSFLAT Conf.* Atlantis Press, 2011, pp. 584–591.

[32] T Groff and T Jones, *Introduction to Knowledge Management: KM in Business.* Butterworth-Heinemann, 2003.

[33] BM Lake, "Towards more human-like concept learning in machines: Compositionality, causality, and learning-to-learn," Dissertation, Massachusetts Institute of Technologies (MIT), 2014.

[34] *DIN 2342:2011-08, Terminology terms,* German Institute for Standardization eV, Standard, August 2011.

[35] D. Brändle and S. Kirchmayer, “Tell me, how do you feel about the terminology?” *MDÜ - Trade journal for interpreters and translators,* 2012.

[36] S. Loebner. (2015) Semantics. An Introduction, 2nd Edition. Available: https: //user.phil.hhu.de/~loebner/semantik_2/term.htm#BasicLevel

[37] J. Pafel and I. Reich, *Introduction to Semantics: Fundamentals - Analyzes - Theo ria.* JB Metzler, 2016.

[38] H. Reinalter and P. Brenner, *Encyclopedia of Humanities: Subject Terms - Disciplines - People.* Böhlau, 2011.

[39] *DIN 2330:2013-07, Terms and designations - General principles,* German Institute for Standardization eV, Standard, Rev. DIN 2330:2013-07, July 2013.

[40] U. Kastens and H. Büning, *Modelling: Fundamentals and formal methods,* 4th ed. Carl Hanser Verlag GmbH & Company KG, 2018.

[41] C. Beierle and G. Kern-Isberner, *Methods of knowledge-based systems: fundamentals, Algorithms, Applications,* 5th ed. Wiesbaden: Springer Vieweg, 2014.

[42] SJ Russell and P Norvig, *Artificial Intelligence: A Modern Approach, Third Ed tion.* Pearson, 2014.

[43] W. Hesse and HC Mayr, “Modelling in software engineering: A Inventory,” *Informatics Spectrum,* vol. 31, no. 5, pp. 377–393, Oct 2008. Available: https://doi.org/10.1007/s00287-008-0276-7

[44] S. Zelewski, *Ontologies for the structuring of domain knowledge: An approach attempt from a business perspective.* PIM, 1999.

[45] W. Scholze-Stubenrecht, *Duden, The German spelling: [the comprehensive Standard work based on the official rules] - .* Futbergen: RM book and media distribution, 2013.

[46] C. Wagenknecht and M. Hielscher, *Formal Languages, Abstract Automata and Compiler - textbook and workbook for basic studies and advanced training,* 2nd ed. Hei delberg: Springer-Verlag, 2014.

[47] J. Hromkoviÿ, *Theoretical Computer Science - Formal Languages, Computability, Complexity Theory, Algorithmics, Communication and Cryptography,* 5th ed. Heidelberg: Springer-Verlag, 2014.

[48] Department of application fields of automation, *language for I4.0 components - structure of messages,* VDI/VDE Society for Measurement and Automation Technology (GMA), Standard, Jan. 2019.

[49] J. Rowley, "The Wisdom Hierarchy: Representations of the DIKW Hierarchy," *J. Inf. Sci.,* vol. 33, no. 2, pp. 163–180, Apr. 2007. Available : http://dx.doi.org/10.1177/0165551506070706

[50] J. Amelingmeyer, *Knowledge management: Analysis and design of the knowledge base of companies,* ser. Strategic competence management. Deutscher Universitätsverlag, 2013.

[51] M. Frey-Luxemburger and R. Bischoff, *knowledge management - basics and practical application: An introduction to the IT-supported management of knowledge resources.* Vieweg+Teubner Verlag, 2013.

[52] T McFarland and R Parker, *Expert Systems in Education and Training.* Educational Technology Publications, 1990.

[53] R Sun, RC Mathews, and SM Lane, *Implicit and Explicit Processes in the Development of Cognitive Skills: A Theoretical Interpretation with Some Practical Implications for Science Education.* New York: Nova Science Publishers, 2007.

[54] G. Ryle, "Knowing How and Knowing That: The Presidential Address," *Proceedings of the Aristotelian Society,* vol. 46, no. n/a, pp. 1–16, 1945.

[55] T. Wagner, "Agent-supported engineering of automation systems," dissertation, University of Stuttgart, 2008.

[56] U. Schöning, *Logic for computer scientists.* Spectrum Academic Publishers, 2000.

[57] H. Stachowiak, *General Model Theory.* Vienna, New York: Springer Verlag, 1973. Available: https://archive.org/details/Stachowiak1973 General Model Theory

[58] G. Maletzke, *models.* Wiesbaden: VS publishing house for social sciences, 1998, pp. 56-80. Available: https://doi.org/10.1007/978-3-322-80363-4_5

[59] JP Van Gigch, *System design modeling and metamodeling.* Plenum Press New York, 1991.

[60] C. Atkinson and T. Kuhne, "Model-driven development: a metamodeling foundati on," *IEEE Software,* vol. 20, no. 5, pp. 36–41, Sept 2003.

[61] J. Adersberger, "Model-based Extraction, Representation and Analysis of Traceability Information," PhD thesis, University of Erlangen-Nuremberg, 2012.

[62] *OMG Unified Modeling Language (OMG UML),* Object Management Group, Standard, Rev. 2.5.1, Dec. 2017. Available: https://www.omg.org/spec/UML/2.5.1/PDF

[63] A. Laarman and I. Kurtev, "I.: Ontological Metamodeling with Explicit Instantiation," in *Conference on Software Languages Engineering, SLE 2009. LNCS.* Springers, 2009.

[64] T. Kühne, “Matters of (Meta-)Modeling,” *Software & Systems Modeling,* vol. 5, no. 4, pp. 369–385, Dec 2006. Available: https://doi.org/10.1007/s10270-006-0017-9

[65] W. Hesse, “More matters on (meta-)modelling: Remarks on Thomas Kühne's “matters”,” *Software & Systems Modeling,* vol. 5, no. 4, pp. 387–394, Dec 2006. Available: https://doi.org/10.1007/s10270-006-0033-9

[66] M. Glinz, "Metamodels," University of Zurich, Tech. Republic, 2005.

[67] *OMG Meta Object Facility (MOF) Core Specification,* Object Management Group, Standard, Rev 2.5.1, Nov 2016. Available: https://www.omg.org/spec/MOF/2.5. 1/PDF

[68] PP-S. Chen, "The Entity-relationship Model Toward a Unified View of Data," *ACM Trans. Database Syst.,* vol. 1, no. 1, pp. 9–36, Mar. 1976. Available: http://doi.acm.org/ 10.1145/320434.320440

[69] S Robinson, G Arbez, LG Birta, A Tolk, and G Wagner, “Conceptual modeling: Definition, Purpose and Benefits,” in *Winter Simulation Conference (WSC),* Dec 2015, pp. 2812–2826 .

[70] Kidawara Y, Zettsu K, Kiyoki Y, Jannaschk K, Thalheim B, Linna P, Jaakkola H, and Duzí M, “Knowledge Modeling, Management and Utilization towards Next Generation Web,” in *19th European-Japanese Conference on Information Modeling and Knowledge Bases (EJC),* Maribor, Slovenia, 2009, pp. 387-402. Available: https://doi.org/10.3233/978-1-60750-477-1-387

[71] A. Olivé, *Conceptual modeling of information systems.* Springer, 2007. Available: https://doi.org/10.1007/978-3-540-39390-0

[72] AT McCray, "Conceptualizing the world: Lessons from history," *Journal of Biomedical Informatics,* vol. 39, no. 3, pp. 267 – 273, 2006. Available: http://www.sciencedirect.com/ science/article/pii/S1532046405000808

[73] V. Tiuriumin and A. Massel, "Integration of Situation Semantic Models Based on Ontology System," in *2018 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC),* Aug 2018, pp. 1–5.

[74] M. Taheriyan, CA Knoblock, P. Szekely, and JL Ambite, "A Scalable Approach to Learn Semantic Models of Structured Sources," in *2014 IEEE International Conference on Semantic Computing,* June 2014, pp. 183–190.

[75] YT Lee, CR McLean, and Y Luo, "Information Modeling and Model Implementation," in Proceedings *of the International Simulation Conference,* 2006. Available: https://www.nist.gov/ publications/information-modeling -and-model implementation

[76] D. Kolberg, “Development of a reference architecture for the implementation of lean production methods using digital technologies,” dissertation, Technical University of Kaiserslautern, 2018. Available: http://nbn-resolving.de/urn:nbn:de: hbz :386-kluedo-53310

[77] A. Selig, *Information model for the functional typing of automation devices,* ser. ISW-IPA Research and Practice. Jost-Jetter, 2011. Available: https://www.effra.eu/sites/default/files/factories_of_the_future_2020_roadmap.pdf

[78] A. Dengel, Ed., *Semantic Technologies.* Heidelberg: Spectrum, 2012.

[79] H. Stuckenschmidt, *Ontologies: Concepts, Technologies and Applications,* ser. In format in focus. Springer Berlin Heidelberg, 2011.

[80] J. Lunze, *Artificial intelligence for engineers: Methods for solving engineering problems using rules, logical formulas and Bayesian networks,* 3rd ed. Berlin: De Gruyter Oldenbourg, 2016.

[81] S. Beißel, *Basics for problem solving.* Wiesbaden: Gabler, 2011, pp. 17-40 Available: https://doi.org/10.1007/978-3-8349-6232-4_2

[82] F. Fuchs, “Semantic modeling and reasoning for context information in infrastructure networks,” PhD thesis, Ludwig-Maximilians-University Munich, October 2008. Available: http://edoc.ub.uni-muenchen.de/9348/

[83] F. Stenger, D. Schmalz, T. Bieringer, A. Brodhagen, C. Dreiser, and A. Schwei ger, "Modular Plants: Flexible Chemical Production by Modularization and Standardization - Status Quo and Future Trends," Temporärer ProcessNet working group "Modular Plants", Tech. Republic, 12 2016.

[84] A. Rogalla and O. Niggemann, "Automated process planning for cyber-physical production systems," in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2017, pp. 1–8.

[85] Zhang P, Jeinsch T, Ding S, and Liu P, “Process Monitoring and Fault Diagnosis—Status and Applications,” *IFAC Proceedings Volumes,* vol. 44, no. 1, pp. 12 401 – 12 406, 2011, 18th IFAC World Congress. Accessible: http://www.sciencedirect.com/science/article/pii/S1474667016456107

[86] L. Christiansen, “Knowledge-based diagnostic concept by combining Plant structure and process model,” dissertation, Helmut Schmidt University, Holstenhofweg 85, 22043 Hamburg, 2015.

[87] O. Avci, “Why do errors arise in the requirements analysis? A Synthesis of Empirical Findings of the Last 15 Years,” in *Industrialization of Software Management,* 2008.

[88] A. Wilson and J. Hendler, "Linking symbolic and subsymbolic computing," *Connec tion Science,* vol. 5, no. 3-4, pp. 395–414, 1993.

[89] L. Hundt, A. Lüder, R. Drath, and B. Grimm, *Description of behavior with PLCopen XML, R.Wire ,* Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Available: https://doi.org/10.1007/978-3-642-04674-2_4

[90] *AutomationML white paper; Part 1 - Architecture and general requirements,* Car mationML consortium, Standard, Oct. 2014

[91] M. Schleipen, Ed., *Practice Manual OPC UA.* Würzburg: Vogel Business Media, 2018

[92] J. Lange, F. Iwanitz, and TJ Burke, *OPC From Data Access to Unified Architecture,* 4th edition. VDE Verlag GmbH, 2010.

[93] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture,* 1st ed. Sprin ger Publishing Company, Incorporated, 2009.

[94] OPC Foundation. (2020, Mar) OPC Foundation website. Available: https: //opcfoundation.org/

[95] *OPC Unified Architecture Specification; Part 12: Discovery,* OPC Foundation, Stan dart, Rev. 1.02, Jan. 2014.

[96] I. Consult, "eCl@ss - Benefits, Goals and Use," IW Consult, Tech. Republic, 2018.

[97] S. Kuhlins and H. Ströbel, “eCl@ss for product classification in price comparison services,” *Price Comparison Services Workshop - Concepts, Business Models and Architectures,* 2003.

[98] eCl@ss. (2019, Dec) eCl@ss search. Available: https://www.eclasscontent.com/ index.php?language=en&version=11.0

[99] eCl@ss. (2019, Dec) eCl@ss website. Available: https://www.eclass.eu/

[100] A. Bondza, C. Eck, R. Heidel, M. Reigl, and S. Wenze, “With data and semantics on the way to Industry 4.0,” eCl@ss e. V., Tech. Republic, 2018.

[101] *VDI/ VDE 3682 sheet 1, formalized process descriptions - concept and graphic representation,* VDI/VDE society measurement and automation technology, standard, May 2015.

[102] *VDI/ VDE 3682 sheet 2, formalized process descriptions - information model,* VDI/VDE Society Measurement and Automation Technology, Standard, May 2015.

[103] *DIN 8580:2003-09, manufacturing processes - terms, classification,* German Institute for Standardization eV, Standard, Sep. 2003

[104] *DIN 8580, manufacturing processes - terms, classification DRAFT,* Deutsches Insti tut für Normung eV, Standard, Nov. 2019.

[105] *Technical Specification ETSI TS 118 112; oneM2M; Base Ontology, version 2.0.0,* European Telecommunications Standards Institute (ETSI), Standard, Sep. 2016

[106] oneM2M. (2019, Dec) Current Members . Available: http://onem2m.org/ membership/current-members

[107] oneM2M Partners Type 1, “Technical Report: Industrial Domain Enablement, TR 0018-V-4.0.0,” oneM2M, Tech. Republic, Sep 2018.

[108] T. Finin. (2019, Nov) Mailing List. Available: http://lists.w3.org/Archives/Public/ www-webont-wg/2001Dec/0169.html

[109] R. Cyganiak, D. Wood, and M. Lanthaler. (2014, Feb) RDF 1.1 Concepts and abstract syntax. site. W3C. Available: https://www.w3.org/TR/ rdf11-concepts/#referents

[110] D. Brickley and R. Guha. (2014, Feb) RDF Schema 1.1. site. W3C. Available: https://www.w3.org/TR/rdf-schema/

[111] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, DL McGuinness, PF Patel-Schneider, and LA Stein. (2004, Feb) OWL Web Ontology Language: Reference. site. W3C. Available : https://www.w3.org/TR/2004/REC-owl-ref-20040210/

[112] B. Motik, PF Patel-Schneider, and B. Parsia. (2012, Dec) OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition). site. W3C. Available : https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

[113] *Ontology Definition Metamodel,* Object Management Group, Standard, Rev. 1.1, Sep. 2014. Available: https://www.omg.org/spec/ODM/About-ODM/

[114] J. Bao, EF Kendall, DL McGuinness, and PF Patel-Schneider. (2012, Dec) OWL 2 Web Ontology Language: Document Overview (Second Edition). site. W3C. Available: https://www.w3.org/TR/owl2-overview/

[115] P. Hitzler, M. Krötzsch, B. Parsia, PF Patel-Schneider, and S. Rudolph. (2012, Dec) OWL 2 Web Ontology Language: Primer (Second Edition). site. W3C. Available: https://www.w3.org/TR/2012/REC-owl2-primer-20121211

[116] B. Motik, BC Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. (2012, Dec) OWL 2 Web Ontology Language: Profiles (Second Edition). site. W3C. Available: https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

[117] Grau BC, Horrocks I, Motik B, Parsia B, Patel-Schneider P, and Sattler U, “OWL 2: The next step for OWL,” *Journal of Web Semantics,* vol. 6, no. 4, pp. 309 – 322, 2008, Semantic Web Challenge 2006/2007. Available: http://www.sciencedirect.com/science/article/pii/S1570826808000413

[118] C. Golbreich and EK Wallace. (2012, Dec) OWL 2 Web Ontology Language: New Features and Rationale (Second Edition). site. W3C. Available: https://www.w3.org/TR/2012/REC-owl2-new-features-20121211/

[119] J Zhang and Y Lv, “An approach of refining the merged ontology,” in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery,* May 2012, pp. 802–806.

[120] Miyoung Cho, Hanil Kim, and Pankoo Kim, "A New Method for Ontology Merging based on Concept using WordNet," in *8th International Conference Advanced Communication Technology,* vol. 3, Feb 2006, pp. 1573-1576.

[121] S. Amrouch and S. Mostefai, "Syntactico-semantic algorithm for automatic ontology merging," in *International Conference on Information Technology and e-Services,* March 2012, pp. 1–5.

[122] P. Shvaiko and J. Euzenat, "Ontology Matching: State of the Art and Future Challenges," *IEEE Transactions on Knowledge and Data Engineering,* vol. 25, no. 1, pp. 158-176, Jan 2013.

[123] M Fahad, N Moalla, and A Bouras, “Towards ensuring Satisfiability of Merged Ontology,” in *Proceedings of the International Conference on Computational Science (ICCS),* vol. 4, 2011, pp. 2216 – 2225. Available: http://www.sciencedirect.com/science/article/pii/S1877050911003000

[124] I. Horrocks, PF Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. (2004, May) SWRL: A Semantic Web Rule Language Combining OWL and RuleML. site. W3C. Available: http://www.w3.org/Submission/SWRL/

[125] M. Krötzsch, "Rules for OWL," University of Freiburg - Institute for Computer Science, Tech. Republic, 2010

[126] M. O'Connor. (2019, Dec) SWRL Language FAQ. Available: https://github.com/protegeproject/swrlapi/wiki/SWRLLanguageFAQ#What_are_DLSafe_SWRL_Rules

[127] C. Rui, Y. Shi, Z. Zhi-Hao, H. Luo-Kai, Z. Kai, and W. Qing, “An approach to Construct and Parse the OWL-Based Service Ontology,” in *International Symposium on Intelligent Ubiquitous Computing and Education,* May 2009, pp. 465–468.

[128] A Jounaidi and M Bahaj, “Designing and implementing XML schema inside OWL ontology,” in *International Conference on Wireless Networks and Mobile Communications (WINCOM),* Nov 2017, pp. 1–7.

[129] Z Bo, D Chong-jiang, and Z Jie, “On the formalization of aggregate relation based on ontology and its implement of OWL,” in *6th International Conference on Computer Science Education (ICCSE),* Aug 2011, pp. 1323-1326.

[130] Xu J, Wang H, and Trimbach H, "An OWL Ontology Representation for Machine Learned Functions Using Linked Data," in *IEEE International Congress on Big Data (BigData Congress),* June 2016, pp. 319-322.

[131] N. Sadovnikova, A. Matohina, O. Shabalina, D. Shirmanova, and A. Romanova, “Ontology-based urban transport system modeling for dynamic goal setting and decision making,” in *7th International Conference on Information, Intelligence, Systems Applications (IISA),* July 2016, pp. 1–5.

[132] Meridou DT, Kapsalis AP, Papadopoulou MC, Karamanis EG, Patrickakis CZ, Venieris IS, and Kaklamani DI, “An Ontology-Based Smart Production Management System,” *IT Professional,* vol. 17, no. 6, pp. 36–46, Nov 2015.

[133] IP Gunawan, BI Santoso, Y Ahmadin, GP Kuntarto, and FL Moechtar, “Dwipa ontology II: A semi-automatic ontology population process for Bali Tourism based on the ontology population methodology,” in *International Conference on Smart Cities, Automation Intelligent Computing Systems (ICON-SONICS),* Nov 2017, pp. 42–47.

[134] M. Loskyll, “Development of a methodology for dynamic context-based orchestration of semantic field device functionalities,” PhD thesis, Technical University of Kaiserslautern, Jun 2013.

[135] J. Zedlitz, “Conceptual modeling with UML and OWL – investigation of similarities and differences using model transformations,” Dissertation, Christian-Albrechts-University of Kiel, 2013. Available: https://d-nb.info/103734278X/34

[136] G. Guizzardi, "Ontological foundations for structural conceptual models," Dissertation, University of Twente, 10 2005.

[137] M. Compton, H. Neuhaus, K. Taylor, and K. nguyen Tran, “Reasoning about Sensors and Compositions,” in *International Semantic Sensor Networks Workshop, CEUR WS,* 2009, pp. 33–48.

[138] G. Engel, T. Greiner, and S. Seifert, "Ontology-Assisted Engineering of Cyber–Physical Production Systems in the Field of Process Technology," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 6, pp. 2792–2802, June 2018.

[139] T. Tudorache, "Employing Ontologies for an Improved Development Process in Collaborative Engineering," Dissertation, Technische Universität Berlin, Jan 2006.

[140] K. Kumagai, M. Fujishima, H. Yoneda, S. Chino, S. Ueda, A. Ito, T. Ono, H. Yoshida, and H. Machida, “KPI element information model (KEI Model) for ISO22400 using OPC UA, FDT, PLCopen and AutomationML,” in *56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE),* Sep. 2017, pp. 602–604

[141] M Staron, W Meding, K Niesel, and A Abran, "A Key Performance Indicator Quality Model and Its Industrial Evaluation," in *Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA),* Oct 2016, pp. 170–179.

[142] H. Walzel, M. Vathoopan, A. Zoitl, and A. Knoll, "An Approach for an Automated Adaptation of KPI Ontologies by Reusing Systems Engineering Data," in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2019, pp. 1693–1696.

[143] K. Kritikos, D. Plexousakis, and R. Woitch, "A Flexible Semantic KPI Measurement System," in *Cloud Computing and Service Science,* D. Ferguson, VM Muñoz, J. Cardoso, M. Helfert, and C. Pahl, Eds. Cham: Springer International Publishing, 2018, pp. 237–261.

[144] P. Zimmermann, E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski, and P. Zanini, “Skill-based Engineering and Control on Field-Device-Level with OPC UA,” in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2019, pp. 1101–1108.

[145] Industry 4.0 platform. (2019, Dec) Website Platform Industry 4.0 - Glossary. Available: https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/Glossar/glossary.html

[146] J. Pfrommer, A. Ebner, S. Ravikumar, and B. Karunakaran, “Open Source OPC UA PubSub Over TSN for Realtime Industrial Communication,” in *23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* volume 1 Sep 2018, pp. 1087–1090.

[147] S. Malakuti, J. Bock, M. Weser, P. Venet, P. Zimmermann, M. Wiegand, J. Grothoff, C. Wagner, and A. Bayha, “Challenges in Skill-based Engineering of Industrial Automation Systems*,” in *23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* vol. 1 Sep 2018, pp. 67–74.

[148] K. Dorofeev, S. Profanter, J. Cabral, P. Ferreira, and A. Zoitl, “Agile Operational Behavior for the Control-Level Devices in Plug Produce Production Environments,” in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2019, pp. 49–56.

[149] H. Bloch, T. Grebner, A. Fay, S. Hensel, A. Menschner, L. Urbas, M. Hoernicke, T. Knohl, J. Bernshausen, and B. Ag, "Orchestration of Services in Modular Process Plants," in *44th Annual Conference of the IEEE Industrial Electronics Society (IECON),* Oct 2018, pp. 2935-2940.

[150] Z Shi, P Zeng, and H Yu, “An ontology-based manufacturing description for flexible production,” in *2nd International Conference on Advanced Robotics and Mechatronics (ICARM),* Aug 2017, pp. 362–367.

[151] L. Kathrein, K. Meixner, D. Winkler, A. Lüder, and S. Biffl, “A Meta-Model for Representing Consistency as Extension to the Formal Process Description,” in *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Sep. 2019, pp. 1653–1656.

[152] R. Chen, Z. Zhou, Q. Liu, DT Pham, Y. Zhao, J. Yan, and Q. Wei, “Knowledge modeling of fault diagnosis for rotating machinery based on ontology,” in *13th IEEE International Conference on Industrial Informatics (INDIN),* July 2015, pp. 1050–1055

[153] LT Abele, “Resource Monitoring in Industrial Manufacturing Using Knowledge Based Technologies,” Dissertation, Technical University of Munich, 2014.

[154] A. Diedrich and O. Niggemann, “Model-based Diagnosis of Hybrid Systems using Satisfiability Modulo Theory,” in *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19),* Hawaii, USA, Jan 2019.

[155] A. Grastien, "Diagnosis of Hybrid Systems with SMT: Opportunities and Challenges," in *ECAI,* ser. Frontiers in Artificial Intelligence and Applications, vol. 263 IOS Press, 2014, pp. 405-410.

[156] S Narasimhan and G Biswas, "Model-Based Diagnosis of Hybrid Systems," *IEEE Transactions on systems, manufacturing and Cybernetics,* vol. 37, pp. 348–361, 2007.

[157] Prakash, Om and Samantaray, Arun K. and Bhattacharyya, Ranjan, “Model-Based Diagnosis of Multiple Faults in Hybrid Dynamical Systems With Dynamically Updated Parameters,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* Jun 2017.

[158] A. Feldman, GM Provan, and AJC van Gemund, "Approximate Model-Based Diagnosis Using Greedy Stochastic Search," *J. Artif. intel. Res.,* vol. 38, pp. 371–413, 2010.

[159] S. Siddiqi and J. Huang, "Sequential Diagnosis by Abstraction," *J. Artif. international Res.,* vol. 41, no. 2, pp. 329–365, May 2011. Available: http://dl.acm.org/citation.cfm?ID=2051237.2051247

[160] A. Metodi, R. Stern, M. Kalech, and M. Codish, "Compiling Model-Based Diagnosis to Boolean Satisfaction," in *AAAI Conference on Artificial Intelligence,* 2012.

[161] D. Jannach, T. Schmitz, and K. Shchekotykhin, "Parallelized Hitting Set Computation for Model-Based Diagnosis," in *AAAI Conference on Artificial Intelligence,* 2015.

[162] M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo, "METHONTOLOGY: from Ontological Art towards Ontological Engineering," in *Proceedings of the AAAI97 Spring Symposium,* Stanford, USA, March 1997, pp. 33– 40

[163] C. Hildebrandt, S. Törsleff, B. Caesar, and A. Fay, "Ontology Building for Cyber Physical Systems: A domain expert-centric approach," in *14th IEEE International Conference on Automation Science and Engineering (CASE),* Aug 2018, pp. 1079–1086.

[164] NF Noy and DL Mcguinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford University, Tech. Republic, 2001.

[165] NKY Leung, SK Lau, and N. Tsang, "A new methodology to streamline ontology integration processes," in *Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP),* May 2014, pp. 174–179 .

[166] C Hildebrandt. (2019, June) Industrial Standard Ontology Design Patterns. site. Available: https://github.com/ConstantinHildebrandt/ Industrial-Standard-Ontology-Design-Patterns

[167] S. Borgwardt, "Logic-Based Ontology Engineering - Part 1: Introduction to Ontology Engineering, OWL 2, and Description Logics," TU Dresden, Tech. Rep., 2018. Available : https://tu-dresden.de/ing/informatik/thi/lat/quellen/daten/lboe18/Part1.pdf

[168] QUDT. (2019, May) QUDT CATALOG - Quantities, Units, Dimensions and Data Types Ontologies. site. Available: http://www.qudt.org/release2/qudt-catalog. html

[169] S. Cox and C. Little. (2017, Oct) Time Ontology in OWL. site. W3C. Available: https://www.w3.org/TR/owl-time/

[170] V. Presutti, A. Gangemi, S. David, GA de Cea, M.-C. Suárez-Figueroa, E. Montiel-Ponsoda, and M. Poveda, "D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies." Open University, Tech. Rep., 2008. Available : http://neon-project.org/deliverables/WP2/NeOn_2008_D2.5.1.pdf

[171] E. Daga, A. Gangemi, V. Presutti, and A. Salvati, "Ontology Design Patterns," *ISWC 2008,* 2008. Available : http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-401/iswc2008pd_submission_84.pdf

[172] G Guizzardi, “Ontological Patterns, Anti-Patterns and Pattern Languages for Next Generation Conceptual Modeling,” in *Conceptual Modeling,* E Yu, G Dobbie, M Jarke, and S Purao, Eds. Cham: Springer International Publishing, 2014, pp. 13–27.

[173] K. Hammar, “Content Ontology Design Patterns: Qualities, Methods, and Tools - Supplementary Materials,” Dissertation, Linköping University, Department of Computer and Information Science, 2017.

[174] O. Corcho, C. Roussey, LMV Blázquez, and I. Pérez, "Pattern-based OWL Ontology Debugging Guidelines," in *International Conference on Ontology Patterns,* ser. WOP'09. Aachen, Germany: CEUR-WS.org, 2009, pp. 68–82. Available: http://dl.acm.org/citation.cfm?id=2889761.2889767

[175] TR Gruber, “Toward principles for the design of ontologies used for knowledge sharing?” *International Journal of Human-Computer Studies,* vol. 43, no. 5, pp. 907 – 928, 1995. Available : http://www.sciencedirect.com/science/article/pii/S1071581985710816

[176] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL,* 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[177] C. Maria Keet, MC Suárez-Figueroa, and M. Poveda-Villalón, "Pitfalls in Ontologies and TIPS to Prevent Them," in *Knowledge Discovery, Knowledge Engineering and Knowledge Management,* A. Fred, JL Dietz, K , Liu, and J. Filipe, Eds. Springer Berlin Heidelberg, 2015, pp. 115–131.

[178] A. Miles and S. Bechhofer. (2009, Aug) SKOS Simple Knowledge Organization System Reference. site. W3C. Available : https://www.w3.org/TR/2009/REC-skos-reference-20090818/

[179] BC Grau, B. Parsia, and E. Sirin, “Working with Multiple Ontologies on the Semantic Web,” in *The Semantic Web – ISWC 2004,* SA McIlraith, D. Plexousakis, and F. van Harmelen, Eds. Springer Berlin Heidelberg, 2004, pp. 620–634.

[180] R. Anderl, K. Bauer, T. Bauernhansl, B. Diegner, J. Diemer, A. Fay, J. Goericke, Dietmar Grotepass, C. Hilger, J. Jasperneite, J. Kalhoff, U. Kubach, U. Löwen, G. Menges, JS Michels, F. Schmidt, T. Stiedl, M. ten Hompel, and C. Zeidler, "Updating the Application Scenarios of the Industry 4.0 Platform," Federal Ministry for Economic Affairs and Energy (BMWi), Berlin, tech Rep., 2016. Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/ Publikation/fortress-application-scenarios.html

[181] J. Bernshausen, A. Haller, H. Bloch, M. Hoernicke, S. Hensel, A. Menschner, A. Stutz, M. Maurmaier, T. Holm, C. Schäfer, L. Urbas, U. Christmann , C. Fleischer-Trebes, and F. Stenger, “Plug & Produce on the Go in

the market," *atp magazine,* vol. 61, no. 1-2, pp. 56–69, 2019. Available: http://ojs.di-verlag.de/index.php/atp_edition/article/view/2400

[182] A. Klose, S. Merkelbach, A. Menschner, S. Hensel, S. Heinze, L. Urbas, C. Schäfer, S. Szmais, M. Eckert, T. Ruede, T. Scherwietes, P. Santos, F. Stenger, T. Holm, W. Welscher, N. Krink, T. Schenk, A. Stutz, M. Maurmaier, and F. Apitz, "Orchestration requirements for modular process units," Technische Universität Dresden, tech Republic, 05 2019.

[183] H. Bedenbender, A. Bentkus, U. Epple, T. Hadlich, R. Heidel, O. Hillermeier, M. Hoffmeister, H. Huhle, M. Kiele-Dunsche, H. Koziolek, S. Lohmann, M Mendes, J. Neidig, F. Palm, S. Pollmeier, B. Rauscher, F. Schewe, B. Waser, I. Weber, and M. Wollschlaeger, "Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation,"
Federal Ministry for Economic Affairs and Energy (BMWi), Berlin, Tech. Rep., 2017. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/ Publikation/Industrie-40-Plug-and-Produce.html

[184] T. Kuhn, "A Survey and Classification of Controlled Natural Languages," *Computational Linguistics,* vol. 40, no. 1, pp. 121–170, 2014. Available: https://doi.org/10.1162/COLI_a_00168

[185] T. Pellegrini and A. Blumauer, Eds., *Semantic Web: Ways to a networked knowledge society,* ser. X.media.press. Berlin: Springer Vieweg, 2006.

[186] B. Geyer-Hayden, *Knowledge Modeling in the Semantic Web.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 127-146. Available : https://doi.org/10.1007/978-3-540-72216-8_7

[187] *DIN 1463 Part 1 - Creation and further development of thesauri: Monolingual thesauri,* German Institute for Standardization eV, Standard, November 1987.

[188] *DIN 1463 Part 2 - Creation and further development of thesauri: Multilingual thesauri,* German Institute for Standardization eV, Standard, Oct. 1993

[189] F.-P. Glawar, J. Küng, T. Luckeneder, K. Steiner, AM Tjoa, R. Wagner, and W. Wöß, "Applications of topic maps in knowledge management systems," *IJEB,* vol. 2, pp. 508–518, 01 2004.

[190] *ISO/IEC 13250: Information technology - SGML applications - Topic maps,* International Organization for Standardization, Standard, May 2003.

[191] F.-J. Auernigg, *Semantic Networks.* Munich: GRIN Verlag, 2005.

[192] K. Reichenberger, *Compendium semantic networks - concepts, technology, modelling.* Springer, 2010. Available: https://doi.org/10.1007/978-3-642-04315-4

[193] G. Stumme, "Chapter 8: Semantic Networks," Uni Kassel, Tech. Rep., 2006. Available: https://www.kde.cs.uni-kassel.de/wp-content/uploads/lehre/ws2006-07/KI/ folies/4_08_Teil1_SemantischeNetze.pdf

[194] L. Dittmann, “Languages for representing knowledge - egg
ne investigative representation,” University of Essen, Tech. Republic, 2002.
Available: https://www.pim.wiwi.uni-due.de/uploads/tx_itochairt3/publications/
Languages_for_knowledge representation.pdf

[195] M. Minsky, "A Framework for Representing Knowledge," Massachusetts Institute of
Technology (MIT), Cambridge, MA, USA, Tech. Republic, 1974.

[196] W. Petersen, "Representation of concepts as frames," *The Baltic International Year book of Cognition, Logic and Communication,* vol. 2, pp. 151–170, 01 2007.

[197] PJF Lucas and LC van der Gaag, *Principles of expert systems,* ser. International
computer science series. Addison-Wesley, 1991.

[198] R Jambor, S Jow, D Meder, and P Klahr, “The Credit Clearing House
Expert System,” in *Third Conference on Innovative Applications of Artificial Intelligence,* Ser. IAAI '91. AAAI Press, 1991, pp. 255-269. Accessible:
http://www.aaai.org/Papers/IAAI/1991/IAAI91-017.pdf

[199] J. Martinez Lastra, I. Delamer, and F. Ubis, *Domain ontologies for reasoning machines in factory automation.* International Society of Automation (ISA), 2010.

[200] D. Foxvog, *Cyc.* Dordrecht: Springer Netherlands, 2010, pp. 259–278. Accessible:
https://doi.org/10.1007/978-90-481-8847-5_12

[201] T. Gruber. (2019) Ontolingua. Available: http://ksl-web.stanford.edu/kst/
ontolingua.html

[202] O. Corcho and A. Gómez-Pérez, "A Roadmap to Ontology Specification Languages,"
in *Knowledge Engineering and Knowledge Management Methods, Models, and Tools,*
R. Dieng and O. Corby, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000,
pp. 80–96.

[203] R MacGregor. (1999, Retrospective) on Loom. Ver
https://web.archive.org/web/20131025063241/http://www.isi.edu/isd/
available: LOOM/papers/macgregor/Loom_Retrospective.html

[204] S. Arroyo, R. Lara, Y. Ding, M. Stollberg, D. Fensel, and U. Innsbruck, “Semantic
Web Languages. Strengths and Weakness,” in *In International Conference in Applied computing (IADIS),* 2004, pp. 23–26.

[205] D. Mcguinness, R. Fikes, L. Stein, and J. Hendler, *DAML-ONT: An ontology language for the semantic web.* MIT Press, 01 2003, pp. 65-93.

[206] I. Horrocks, PF Patel-Schneider, and F. van Harmelen, “Reviewing the Design of
DAML+OIL: An Ontology Language for the Semantic Web,” in *Eighteenth AAAI Conference on Artificial Intelligence (AAAI).* AAAI Press, 2002, pp. 792-797.

[207] PowerLOOM. (2019, Sep) PowerLoom Documentation. Available: https://www.isi.
edu/isd/LOOM/PowerLoom/documentation/documentation.html

[208] A Sawsaa, *Ontological Engineering Approach of Developing Ontology of Information Science.* Anchor Academic Publishing, 2015.

[209] O. Corcho, *A Layered Declarative Approach to Ontology Translation with Knowledge Preservation,* ser. Frontiers in artificial intelligence and applications. IOS Press, 2005.

[210] B. Parsia, N. Matentzoglu, RS Gonçalves, B. Glimm, and A. Steigmiller, "The OWL Reasoner Evaluation (ORE) 2015 Competition Report," *Journal of Automated Reasoning,* vol. 59, no. 4, pp. 455–482, Dec 2017. Available: https://doi.org/10.1007/s10817-017-9406-8

[211] JM Keil and S. Schindler, "Comparison and evaluation of ontologies for units of measurement," *Semantic Web,* vol. 10, no. 1, pp. 33–51, 2019. Available: https://doi.org/10.3233/SW-180310

[212] Kazan Federal University. (2019, Aug) OntoMathPro - A Hub for Math LOD. site. Available: http://ontomathpro.org/

[213] COmputational Modeling in BIology NETwork (COMBINE). (2019, Aug) Mathematical Modeling Ontology (MAMO). site. Available: https://co.mbine. org/standards/mamo

[214] OpenMath Society. (2019, Aug) OpenMath. site. Available: https://www. openmath.org/

[215] N. Noy and A. Rector. (2006, Apr) Defining N-ary Relations on the Semantic Web. site. W3C. Available: https://www.w3.org/TR/swbp-n-aryRelations/

[216] C. May and A. Koch, "Overall Equipment Effectiveness (OEE) : Tool for increasing productivity," *Journal of Management Consulting : ZUb,* vol. 3, no. 6, pp. 245–250, 2008.

[217] D. Poole, “Representing diagnosis knowledge,” *Annals of Mathematics and Artificial Intelligence,* vol. 11, no. 1, pp. 33–50, Mar 1994. Available: https://doi.org/10.1007/BF01530736

[218] B. Peischl and F. Wotawa, "Model-Based Diagnosis or Reasoning from First Principles," *Intelligent Systems, IEEE,* vol. 18, pp. 32–37, 06 2003.

[219] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence,* vol. 32, no. 1, pp. 97–130, 1987.

[220] J. de Kleer and BC Williams, “Diagnosing Multiple Faults,” *Artificial Intelligence,* vol. 32, no. 1, pp. 97–130, 1987. Available : http://dx.doi.org/10.1016/0004-3702(87)90063-4

[221] RT Stern, M Kalech, S Rogov, and A Feldman, “How Many Diagnoses Do We Need?” in *Twenty-Ninth Conference on Artificial Intelligence (AAAI),* Austin, TX, USA, 2015, pp. 1618–1624. Available : http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9952

[222] J. de Kleer, A. Feldman, and I. Matei, "Correcting Design Errors in Components and Connections," in *International Workshop on the Principles of Diagnosis (DX),* 2019, Updated version of DX 2018 paper: The Duality of Design and Diagnosis. Available: http://dekleer.org/Publications/ijcai2019r.pdf

[223] B. Liu, M. Ghazel, and A. Toguyéni, "Model-Based Diagnosis of Multi-Track Level Crossing Plants," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 2, pp. 546–556, Feb 2016.

[224] A. Sidhu, A. Izadian, and S. Anwar, "Adaptive Nonlinear Model-Based Fault Diagnosis of Li-Ion Batteries," *IEEE Transactions on Industrial Electronics,* vol. 62, no. 2, pp. 1002–1011, Feb 2015.

[225] Z. Gao, C. Cecati, and SX Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques —Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches," *IEEE Transactions on Industrial Electronics,* volume 62, no. 6, pp. 3757–3767, June 2015.

[226] A. Maier, "Online passive learning of timed automata for cyber-physical production systems," in *12th IEEE International Conference on Industrial Informatics (INDIN),* July 2014, pp. 60–66.

[227] O. Niggemann, B. Stein, A. Vodenÿareviÿ, A. Maier, and H. Kleine Buning, "Learning Behavior Models for Hybrid Timed Systems," in *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12),* July 2012.

[228] B. Hofer, R. Mateescu, W. Serwe, and F. Wotawa, “Using LNT Formal Descriptions for Model-Based Diagnosis,” in *29th International Workshop on Principles of Diagnosis (DX),* 2018. Available: http: //ceur-ws.org/Vol-2289/paper2.pdf

[229] A. Metodi, R. Stern, M. Kalech, and M. Codish, "A Novel SAT-based Approach to Model Based Diagnosis," *J. Artif. international Res.,* vol. 51, no. 1, pp. 377-411, Sep. 2014 Available: http://dl.acm.org/citation.cfm?id=2750423.2750433

[230] B. DuCharme, Learning *SPARQL: Querying and Updating with SPARQL 1.1,* 2nd ed. Beijing: O'Reilly, 2013. Available: https://www.safaribooksonline.com/library/view/learning-sparql-2nd/9781449371449/

[231] E. Sirin, B. Parsia, BC Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Journal of Web Semantics,* vol. 5, no. 2, pp. 51 – 53, 2007, Software Engineering and the Semantic Web. Available: http://www.sciencedirect.com/science/article/pii/S1570826807000169

[232] The Apache Software Foundation. (2019, Sep) Apache Jena Fuseki. Available: https://jena.apache.org/documentation/fuseki2/index.html

[233] M. Thimm, "The Tweety Library Collection for Logical Aspects of Artificial Intelligence and Knowledge Representation," *Artificial Intelligence,* vol. 31, no. 1, pp. 93-97, March 2017.

[234] M. Thimm. (2019, Dec) Tweety Project. Available: http://tweetyproject.org

[235] A. von Birgelen and O. Niggemann, *IMPROVE - Innovative Modeling Approaches for Production Systems to Raise Validatable Efficiency: Intelligent Methods for the Factory of the Future.* Springer Vieweg, Aug 2018, chap. Anomaly Detection and Localization for Cyber-Physical Production Systems with Self-Organizing Maps, pp. 55–71.

[236] A. von Birgelen, D. Buratti, J. Mager, and O. Niggemann, "Self-Organizing Maps for Anomaly Localization and Predictive Maintenance in Cyber-Physical Production Systems," in *51st CIRP Conference on Manufacturing Systems,* vol. 72, 2018, pp. 480 – 485. Available : http://www.sciencedirect.com/science/article/pii/S221282711830307X

[237] P. Li and O. Niggemann, "A Data Provenance based Architecture to Enhance the Reliability of Data Analysis for Industry 4.0," in *23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* vol. 1 Sep 2018, pp. 1375–1382.

[238] Platform Industrie 4.0, "Specification: Details of the Asset Administration Shell - Part 1 - The exchange of information between partners in the value chain of industrie 4.0 (Version 2.0)", Federal Ministry for Economic Affairs and Energy (BMWi), Berlin, tech Rep., 2019. Available: https://www.plattform-i40.de/PI40/Redaktion/DE/ Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html

Declaration of honor

I hereby declare on my honor that I have completed this thesis without the unauthorized help of third parties and without using any tools other than those specified.
The data and concepts taken directly or indirectly from other sources are marked with the source. Parts of the work that have already been the subject of examination papers are also clearly marked.

Other people were not involved in the content-material preparation of the present work. In particular, I have not used the paid help of mediation or consulting services (doctoral advisors or other persons). No one has received direct or indirect monetary benefits from me for work related to the content of the submitted dissertation. The work has not yet been submitted to another examination authority in Germany or abroad in the same or a similar form. I certify that I have told the truth to the best of my knowledge and have not concealed anything.

Lemgo, April 11, 2020

Andrew Colorful