Semantics Enable Standardized User Interfaces for Diagnosis in Modular Production Systems

Andreas Bunte[1] and **Alexander Diedrich**[2] and **Oliver Niggemann**[1,2]
[1]Ostwestfalen-Lippe University of Applied Sciences, Institute industrial IT
Email: {andreas.bunte, oliver.niggemann}@hs-owl.de
[2]Fraunhofer IOSB-INA, Fraunhofer Application Center Industrial Automation
Email: alexander.diedrich@iosb-ina.fraunhofer.de

Abstract

Products have become more and more customized which leads to the need for flexible production systems. To enable this flexibility for diagnosis, modular diagnosis systems are used to meet the demand. Several different diagnosing systems can be implemented in such a system, where the operator has to use all of them. Because there is no semantic information available, the operators have to know every system in detail. They need knowledge about the machine and the underlying algorithms to interpret the results, they are not able to use their known concepts; we call this the *conceptual gap*. The aim of the paper is to create a knowledge base which contains knowledge about the production system and the implemented algorithms to interpret the results automatically and thus close the conceptual gap. This enables the communication of the system's status through natural language, so that an inexperienced user can make use of it. Therefore, a natural language layer is connected to the knowledge base which enables user's question such as *"Are there any anomalies in the system?"* to be answered. 92 % of the 200 test questions are processed correctly by the natural language layer.

1 Introduction

The field of industrial automation is currently being stirred up by trends such as Industrial Internet [Evans and Annunziata, 2012] and Germany's Industrie 4.0 [Kagermann *et al.*, 2013]. These trends are mainly driven by two developments: an increased level of data availability (Internet of Things, IoT) and smart usage of these data (Smart Automation). The latter aspect is an active field of research: diagnosis algorithms abstract the data into higher-level information such as maintenance schedules, error causes, or energy assessments (Self-Diagnosis) [Pinto *et al.*, 2015] while optimization algorithms use this information to improve performance and energy efficiency (Self-Optimization) [Schmitt *et al.*, 2013] and configuration algorithms speed up the plant commissioning or plant adaption (Self-Configuration) [Scheifele *et al.*, 2014].

In all these fields, a large variety of algorithms exists: E.g. diagnosis is done by means of principal component analysis (PCA) [Eickmeyer *et al.*, 2015], automata learning [Maier and Niggemann, 2015], regression [Purwins *et al.*, 2014], model based diagnosis [Reiter, 1987; Stern *et al.*, 2013] or by means of optimization algorithms [Windmann *et al.*, 2015]. All these techniques are applied to a single module, not to a whole production plant. Thus several different modular diagnosis systems have to be monitored by the operator. Each technique uses different information on different abstraction levels (e.g. root causes versus symptoms versus mean values) and employs different wording (e.g. error versus root cause versus failure). Operators have to know the results of the algorithms to be able to use them, because there are no understandable semantics for the operator. We call this gap between the variety of algorithms and the operator the *conceptual gap*, which is illustrated in figure 1.

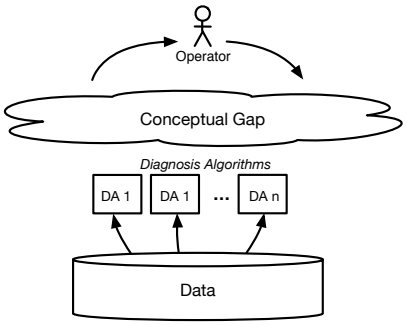

Figure 1: The conceptual gap between algorithms and operators.

Usually, the conceptual gap is solved by proprietary solutions which directly map inputs and results to the graphical user interface (GUI). It is suitable for existing plants, but because of a rising demand in flexibility, new approaches are needed which enable an efficient communication between operators and flexible production plants. To provide this flexible communication, natural language can be used as a human machine interface (HMI). Operators ask questions such as *"Why is the energy consumption too high?"* and the support system determines an answer, which is based on a common wording, thus creating new research questions (RQ):

RQ 1: Can a common wording and a common pool of concepts be established for diagnosis in Industrial Internet and Industrie 4.0?

RQ 2: How can the heterogenous wording of users' input be mapped to the homogenous wording of machines which is mentioned in RQ 1?

RQ 3: Is it possible to use a natural language interface as a HMI for diagnosis, due to the accuracy required?

To overcome the conceptual gap a solution approach with four main solution components (SC) is introduced (see figure 2):

SC 1: Knowledge and wording (RQ 1) is captured in the form of an ontology model. This model comprises *(i)* an agreement on wording and common concepts, *(ii)* a meta-model capturing general knowledge such as causalities, structural relations, and system types and *(iii)* an instance model for each specific plant.

SC 2: A reasoning engine uses these models to store events such as anomalies and to determine answers based on knowledge such as the plant's structure or measured symptoms.

SC 3: A natural language layer maps between the common wording of machines and the natural language of operators (addresses RQ 2).

SC 4: A mapping layer maps between the reasoning engine and the variety of algorithms. The semantic model contains only names (strings) and can only process these. If, for example, a value should be requested, the string has to be converted into a method call.

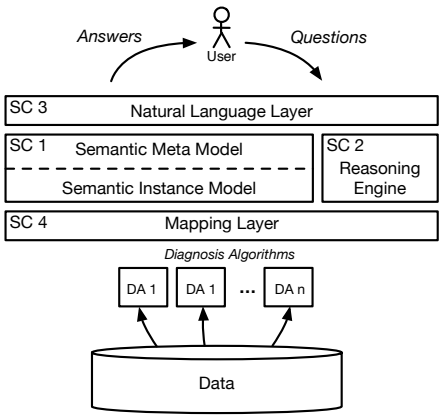

Figure 2: The solution approach with four main solution components.

The contribution of this paper is a semantic model which enables the diagnosis results to be mapped automatically to common concepts which enable the usage of different algorithms through a single user interface. The common concepts define a common wording and enable an easy communication based on natural language. A prototype is implemented with a natural language interface to demonstrate the results.

The paper is structured as follows: First, a review on related work is given in section 2. Requirements are defined based on an analyzed question in section 3. In section 4 the solution is described in detail. The results are presented in section 5. Finally, the work is summarized and discussed in section 6.

2 Related Work

Ontologies can be used to close the conceptual gap, because they include knowledge of production systems. There are several ontologies in the domain of production systems; their usage is versatile, but to the best of the authors' knowledge there is no work on a semantic layer for diagnosis in production systems to standardize the communication of different diagnosis systems.

Simon et al. [2015] introduced a framework to create models for diagnosis based on natural language. Documents such as requirements analysis, architectural design or FMEA, are analyzed and knowledge is condensed to create a model. Additional knowledge of the cyc ontology, a common sense ontology from the Cycorp. Inc., can be extracted to create the model.

DiGiuseppe and Jones [2012] developed a semantic fault diagnosis which should support developers finding the root cause by providing a detailed fault description. These descriptions are automatically generated based on, amongst others, source code, class names, method names and comments. The provided output is mostly a buzzword or some words, not a sentence. So it is only suitable for the developer and cannot be transferred into the industrial context.

Alm et al. [2015] introduce an ontology to provide context based annotations to users. Annotations are created manually in a specific context and they are used for assembly tasks. The cognitive architecture State Operation and Result (Soar) is used to identify a situation related context to provide annotations.

Ontologies are more widely used in the domain of production systems. The work of Uddin et al. [2011] deals with flexible manufacturing systems. The aim is to optimize the work flow through finding an optimal job scheduling of multiple flexible production cells. Knowledge from design tools about products is updated in the ontology. It is accessed from the shop floor by reasoning and querying to achieve an optimal job scheduling - the ontology acts as a kind of interface between the engineering department and the shop floor. Web Ontology Language (OWL) is used and extended with Semantic Web Rule Language (SWRL) and queried with SPARQL.

In the project ARUM, the authors [Harcuba and Vrba, 2015] created ontologies to optimize production processes, especially on ramp-up phases, similar to Uddin et al. [2011]. Three ontologies were created, a *core ontology* which defines generic terms, a *scene ontology* for long and medium term production planning, and an *event ontology* which reacts according to new situations.

Ontologies are used for the design and performance evaluation of production systems in [Terkaj and Urgo, 2014]. They are used during the design phase and have a shared data repository based on a common virtual factory data model which is connected to different software tools, needed for the planning and evaluation process.

The *Ontology for the description of Modular Production Systems* (OntoMoPS) is introduced in [Tsinarakis and Tsinaraki, 2013]. As the name suggests, this ontology is a semantic based description of modular production systems enabling advanced search capabilities within a digital library content. The OntoMoPS focuses on the most important features and details from the authors' perception. Buffers and production strategies are described in detail, but other aspects such as diagnosis have not been taken into account.

In [Blondé *et al.*, 2011], an ontology is introduced in the domain of biology which has a modeling structure similar to the one in this work. The knowledge is modeled on the class

layer and all instances should participate in it. The authors used OWL Full to be able to represent relations between classes and instances.

Many other ontologies can be found in the literature, e.g. eleven ontologies for sensors are known, such as *OntoSensor* or the *Semantic Sensor Network* (SSN). A detailed review on that is given by Schlenoff et al. [2013]. All these ontologies only cover a small specific part of a domain, thus they are not suitable to close the conceptual gap.

3 Requirements Analysis

The aim of the model is to establish a common wording and to interpret diagnosis results and map them to a common wording. It would be possible to use the model for diagnosis, but until now it has been too abstract and the formalism is not suitable for this task, even if the system is modeled in detail.

Typical use-cases have been identified to establish a common wording for diagnosis of production systems. Requirements of typical operators' questions on an abstract level were collected and analyzed. Some typical questions are listed below.

- How much throughput is achieved?
- How much power is consumed by module 5?
- Why is the power consumption too high?
- Is the conveyor of module 1 active?
- What is the actual state of the system?
- Are there any anomalies?
- Is the timing of the system correct?
- Why did the anomaly occur?
- Is module 1 alright?

The first finding is that operators are often faced with similar questions, which can vary in appearance through rephrasing. All identified questions regarding diagnosis can be classified into three categories: value requests, anomaly detection and root cause analysis questions. In the following we describe each of these categories in detail and derive requirements which must be fulfilled by SC 1 to determine answers to the questions.

3.1 Value Requests and Root Cause Analysis

Value requests are supplementary questions which enable the actual conditions to be checked in order to detect anomalies or identify the root-cause manually. Because systems are running fine most of the time, these requests can also be used to check whether production goals are achieved or not. So, not only should sensor values be accessible, but aggregated values, such as key performance indicators (KPI), e.g. the energy consumption of one product or the efficiency of the manufacturing system, should also be accessible.

Semantics enable definitions of aggregated values, e.g. the energy consumption of a subpart is defined as the addition of the consumption of all devices in the subpart. Such definitions are independent of the manufacturing system, so they only have to be defined once and can be applied to all systems. Furthermore, devices should be addressed by names through semantics, they do not need to be called by the names that are used in the network (also known as signal names). This enables human readable names of devices to be presented instead of signal names.

Diagnosis has the same requirements as value requests. The diagnosis is performed in underlying algorithms and has to be presented in an understandable way. So, for diagnosis, the mapping must be from the signal name to a meaningful name, whereas for value request it is the other way round. Technically it makes no difference if the relation is modeled once, the other direction can be inferred. To enable both functionalities, four requirements were derived which had to be fulfilled by SC 1.

Requirement 1. *Structural knowledge must be available to request subparts of a system and to know relations between modules.*

Requirement 2. *The signals must be mapped to meaningful names to refer to devices by known terms and not by the signal name.*

Requirement 3. *KPIs must be defined and how they are calculated must be described, e.g. the overall equipment efficency (OEE) is calculated as multiplication of the availability, performance and quality.*

Requirement 4. *General knowledge about production systems, devices and products, e.g. which products can be produced.*

3.2 Anomaly detection

One important issue for operators is the detection of anomalies to prevent machine damage and faulty products. Anomaly detection algorithms are used for this. SC 1 contains a semantic description of diagnosis algorithms which enables an interpretation of the results by the machine. Results such as points in a multidimensional space, strings, boolean values, float values, or signal names are mapped to concepts which are semantically defined. Users can communicate in terms such as *anomaly*, *root cause* or *error*, without any knowledge about the underlying algorithms.

An additional advantage is that multiple algorithms in different modules can be used and the operator will not notice anything about this, because the interface is the same for all modules and algorithms. Therefore, two additional requirements are derived form SC 1.

Requirement 5. *The semantic model must define different kinds of anomalies and their characteristics.*

Requirement 6. *Anomaly detection algorithms must be described regarding their characteristics, limitations, and the interpretation of their results.*

4 Solution

All requirements are defined in section 3, the solution components are developed in this section, with respect to the requirements. The overall solution idea is presented in figure 2. The focus of this work is SC 1 and SC 3. SC 2 is described but without much detail and SC 4 is not covered by this paper, because it is only a technical solution to communicate between the semantic model and algorithms.

4.1 Solution Component 1: Semantic model

The semantic model can be divided into a semantic meta model (SMM) and a semantic instance model (SIM) (see figure 2). The SMM includes general knowledge which holds true for all plants, e.g. wording, common concepts, algorithms, relations between these concepts, and available relations themselves; details are below. This is one of the main issues of closing the conceptual gap, because concepts are defined therein which enable machines and operators to

map their understanding to these concepts. The SIM contains machine and process specific knowledge which holds true for one specific plant and can only be reused for identical plants.

The web ontology language (OWL) is used for the semantic model (SC 1), which uses OWL DL (description logic) profile, a decidable subset of the maximum expressivity OWL full [W3C, 2004]. OWL defines classes (also known as concepts) which are organized hierarchically in the SMM. Relations between the classes provide a description of concepts. Real world objects are modeled as instances, also called individuals, and are located in the SIM.

Semantic Meta Model

The SMM defines a set of relations enabling concepts to be represented as well as enabling a common wording. Relations are ordered hierarchically and are used to characterize concepts in the SMM and to present knowledge about individuals in the SIM. Three top relations are identified: causalities, possessing and mathematical relations.

Causality is a commonly known concept which is modeled as a relation between classes or individuals. It can be further divided into unidirectional and bidirectional causalities. The relation *isProportional* is a bidirectional causality, which enables inferences such as if *Power isProportional Energy*, it can be inferred that *Energy isProportional Power*. The unidirectional causalities are split into two relations which are defined as *inverseOf*, such as *isRequiring* and *isRequiredBy*. Again this enables inferences which ease the reasoning and increase the human readability.

Possessive relations have the prefix *has*, such as *hasPart* or *hasUnit*. It is mainly used to model structure knowledge such as a *Module5 hasPart Conveyor*. Like the causalities, these relations also mark dependencies which are required and do not contain any uncertainties.

Mathematical relations are used to present calculations of values. The relations are processed by the reasoning engine (RE), because the used modeling formalism does not provide any mathematical operations directly.

Now, a set of relations is defined which can be used to model the required knowledge determined in section 3. Requirement 3 postulate the need of defining KPIs, to enable operators to request them. Because these definitions are always the same and do not contain machine specific knowledge, they are defined in the SMM. By using mathematical relations, KPIs can be defined. The RE can interpret the relations and calculate the value, so requirement 3 is fulfilled.

Different kinds of anomalies are defined to fulfill requirement 5. As an example, the continuous anomaly (CA) shown is listing 1.

Listing 1: Definition of the concept *continuous anomaly* (CA) in DL.

1. $CA \sqsubseteq Symptom$
2. $CA \sqsubseteq \exists isProvokedBy.Continuous$
3. $CA \sqsubseteq \exists isProvokedBy.ProcessParameter$
4. $CA \sqsubseteq \forall isRated.Bad$
5. $CA \sqsubseteq \exists has.Cause$

CAs are modeled as a subclass of *Symptom*. Furthermore, the CAs are modeled in such a way that they can only be provoked by continuous process parameters, so that every anomaly is bad, and so that every anomaly has some cause. The model differentiates between continuous, logical and timing anomalies which are all modeled in a similar way.

Regarding requirement 6, there is a need to describe anomaly detection algorithms. Clustering based anomaly detection will be further described as an example in DL, see listing 2.

Listing 2: Definition of the concept *clustering* in DL.

1. $Clustering \sqsubseteq Algorithm$
2. $Clustering \sqsubseteq \forall hasInput.Continuous$
3. $Clustering \sqsubseteq \exists \neg hasInput.Counter$
4. $Clustering \sqsubseteq \exists isSensing.Symptom$
5. $Clustering \sqsubseteq \forall hasResult.ContinuousAnomaly$

Clustering is placed in the hierarchical structure as a subclass of algorithm. There are not many limitations, any continuous value can be used as input, apart from a counter since counters have a negative impact on the results. In the fourth line it has been described that some symptoms can be sensed. Symptoms are indicators for anomalies or sub-optimal states. The clustering based anomaly detection algorithm can only detect continuous anomalies. Other anomaly detection algorithms such as ANODA [Maier, 2015] and PCA are described in the semantic model. There are many anomaly detection algorithms available, which could be modeled similarly. Thus, results of algorithms can be mapped to concepts and processed automatically; requirement 6 is fulfilled.

Some general knowledge is needed to define concepts and relations between concepts, as mentioned in requirement 4. Therefore, common known facts are modeled as facts about conveyors. A conveyor is an actuator which has some speed, *isRequiring* electric power, *isProvoking* some action and *isPerforming* some transportation, see listing 3. It can be extended e.g. with knowledge about wear, behavior of the torque, or capacity, depending on the required model depth. Sensors, actuators, modules, units, products, and plant statuses are modeled in a similar way to the conveyor.

Listing 3: Definition of the concept *conveyor* in DL.

1. $Conveyor \sqsubseteq Actuator$
2. $Conveyor \sqsubseteq \exists hasProperty.Speed$
3. $Conveyor \sqsubseteq \exists isRequirering.ElectricPower$
4. $Conveyor \sqsubseteq \exists isProvoking.Action$
5. $Conveyor \sqsubseteq \exists isPerforming.Transportation$

To fulfill requirement 4, knowledge about the product has to be modeled to enable questions about it. The used demonstrator of the SmartFactoryOWL[1] (SF) produces popcorn, which is an easy process containing corn and heat. The model in the ontology is shown in listing 4.

Listing 4: Definition of the product *popcorn*.

1. $Popcorn \sqsubseteq Product$
2. $Popcorn \sqsubseteq \exists ismadeOf.Corn$
3. $Popcorn \sqsubseteq \exists ismadeOf.Heat$
4. $Popcorn \sqsubseteq \exists isPerformedBy.AirHeater$

Four statements do not seem to be many to define a product, but more knowledge is covered in the linked concepts. The linked concepts contain knowledge and enable additional conditions to be checked, such as facts about an air heater.

[1] An Initiative of the Fraunhofer-Society and the OWL University of Applied Sciences, see www.smartfactoryowl.de

Semantic Instance Model

The SMM is defined above and provides a framework with general knowledge. This framework can be used to integrate machine specific knowledge, the SIM. As suggested by the name, modeled classes are instantiated and represent a real world object, such as an actuator or sensor. Requirements 1 and 2 have to be covered by the SIM.

Structural knowledge is important, because it is like a fingerprint of the machine and has a strong influence on diagnosis systems. Therefore, devices (modeled as individuals) are mapped to the plant's structure e.g. a module or subsystem. Additionally, devices are mapped to the hierarchical class structure to represent the type of device. Therefore, individuals have two types (relation: *hasIndividual*). Individuals which represent devices are linked to a type of device and to the module in which the device is mounted. Figure 3 depicts exemplary relations between devices in the SF. The permanent arrows are *hasSubClass* relations, the dashed arrows are *hasIndividual* relations.

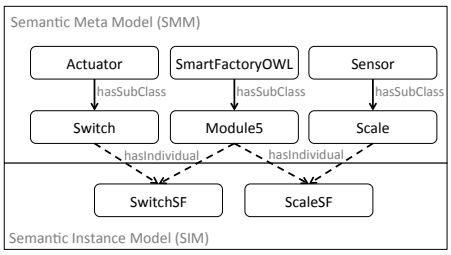

Figure 3: Individuals have two classes to express the structural knowledge as well as the type of device.

To complete requirement 1, relations between modules have to be modeled. The path of products is used for this in order to get from the input to the output. Relations such as *isMadeOf* are used to infer the process chain of the product.

Finally, devices should not be presented to the operator by their signal names, so devices and signal names have to be mapped. A device can be called by the name of the individual which can be extended with data properties to handle synonyms. Such a modeling enable devices to be referred to by meaningful names, which increases the intelligibility of a determined diagnosis significantly. This fulfills requirement 2.

4.2 Solution Component 2: Reasoning Engine

The reasoning engine (RE) is strongly related with the semantic model, because it is the active part which accesses and uses its knowledge. Every event, triggered by the machine or the operator, is received by the RE which chooses an action that has to be performed.

At first the Pellet reasoner [Sirin *et al.*, 2007] is used to check whether the semantic model is consistent or not. It is consistent if no facts lead to a contradiction. Additionally, the reasoner identifies implicit modeled relations and makes them explicit. This will not increase the expressivity, but it reduces the modeling effort.

To reason about facts from the knowledge base, SPARQL queries are generated based on templates. Rules are used to combine the templates, depending on the triggered action such as a detected anomaly or an incoming semantic frame. An example SPARQL query is presented in listing 5, which has requested all symptoms of module 5 since the first of April.

Listing 5: Example query in SPARQL to determine components for air heating.

```
1. SELECT ?anomaly
2. WHERE{
3.   ?anomalyType rdfs:subClassOf :Symptom.
4.   ?anomaly a ?anomalyType.
5.   ?anomaly a :Module5.
6.   ?anomaly :OccurredLastTime ?date.
7.   FILTER(?date >
         "2016-04-01T00:00:00"^^xsd:dateTime)}
```

4.3 Solution Component 3: Natural Language Layer

The natural language layer provides an interface between operators and the semantic layer. Natural language has a couple of advantages in comparison to other technologies. There are no menus which users have to browse through, users are familiar with their language and there are no role dependent interfaces. The main challenge is to extract the versatile information from the language and provide it formalized to the RE.

The main support system is implemented in Java, running on a PC. A user interface is implemented on a mobile device which is connected to the PC. The mobile device contains speech recognition, which can recognize a humans voice and translate speech into text. Additionally, the operator can type in the text via a keyboard. Typing in the text is more robust, while the used speech recognition framework from Android is only 60 % accurate in the context of industrially diagnosing questions. The evaluation was done by two non-native speakers. Words which are never, or only rarely, used in the common communication, such as module 5, PLC or saw, are not frequently recognized as correct.

Input texts are sent from the mobile device to the computer and formalized by the Stanford CoreNLP. The formal representation is done by a semantic frame which has six slots; *TypeOfQuestion*, *TypeOfOutput*, *IntentKey*, *EntityType*, *EntityID* and *Time*. These six slots can express questions and commands in the field of diagnosis and are processed by SC 2.

Three slots of the semantic frame are processed by one classifier for each slot. To learn the classifiers, a corpus of 200 correct classified input texts is used. The type of question is classified according to the intention: value requests, anomaly detection or root cause questions. The second classifier determines the type of output: list, boolean, value or reason. The third one determines what the question is referring to e.g. energy, power or throughput.

The two entity slots are determined with part-of-speech tagging, dependency analysis or regular expressions, depending on which method provides the best results. By using regular expressions, the spelling is used by the sentence as provided in the semantic frame, which makes the method less robust. If there is no match between the provided entity and the semantic model, the levenshtein distance is used to identify the most similar name in the model. The SUTime framework [Chang and Manning, 2012] uses regular expressions to determine the slot *Time*.

Users' questions can be formalized to enable further processing. A common use case is to check the power con-

sumption of the production plant. Users can type in questions such as *"How much power is consumed?"*. The natural language layer computes a semantic frame with *TypeOfQuestion: query*, *TypeOfOutput: value*, *IntentKey: power*, and *Time: PRESENT_REF*. *EntityType* and *EntityID* are empty, because it is not specified in the question, so it refers to the whole system.

The RE processes the incoming semantic frame. Because of the content *query* in the slot *TypeOfQuestion*, it is known that the method for value requests has to be used. To use this method the *SignalName* of the power meter must be determined. A query is formulated based on templates to request the used semantic model. The signal name is provided to the mapping layer which executes the request. The reasoning engine receives the result, determines the unit from the semantic model and provides the generated answer to the user; in this example: "The actual power consumption is 77.3 W"

5 Evaluation

The proposed system is evaluated by applying it to a demonstrator of the SF which is described in detail below. 30 questions were typed into the system manually and it was checked whether the expected results were achieved. Some exemplary questions are described in detail below. Additionally, the natural language interface is evaluated empirically with a corpus of 200 question using cross-validation.

5.1 SmartFactoryOWL

The SF is a building which contains several demonstrators with modern technologies needed for Industrie 4.0, such as versatile production, usage of smart services or user centered automation systems. To evaluate the diagnosis system, a modular and versatile production system was chosen. It is a small-size production system with typical industry processes and devices. Corn in the input silo is sorted with the help of image processing (module 0), the corn is transported into a silo (module 1+2), from the silo to a scale (module 3) and then bottles are filled with a defined portion (module 6). Bottles can be stored or emptied and transported through a blow pipe (module 7) into a heater which makes popcorn from it (module 4). The popcorn is packed at module 5, where a defined portion is determined by a scale.

Every module has 10 to 60 signals, so that the whole system has over 230 signals. For diagnosis, every module has its own diagnosing system. The storage, popcorn and packing module (modules 4, 5 and 7) are used for the evaluation.

5.2 Value Request

Values can be requested, as it is shown in subsection 4.3. It is also possible to request all devices of a module or all devices of a type. This questions can be helpful for diagnosis, but it is not the main point of this paper.

However, it is also possible to request aggregated values. Users can ask *"What is the actual OEE?"*. The semantic frame contains *TypeOfQuestion: query, TypeOfOutput: value, IntentKey: OEE* and *Time: PRESENT REF*. Again, the Entity slots are empty.

As can be seen in the frame, the OEE is requested. Mathematical relations are used to model this KPI. The RE solves the relations and derives the following formula:

$$OEE = Quality \cdot Performace \cdot Availability$$

with:

$$Quality = \frac{NoGoodQualityProducts}{TotalNoProducts}$$

$$Performace = \frac{TotalNoProducts \cdot CycleTimePerProduct}{ProductionTime}$$

$$Availability = \frac{ProductionTime}{PlannedProductionTime}$$

The single values of the formulae are available in the SIM, so the instances have to be requested and the calculation can be performed. An answer is determined and provided to the operator such as *"The actual OEE is 18,5 %."*.

All questions of value request were processed correctly. The system is capable to use names, signal names and synonyms.

5.3 Anomaly Detection and Root Cause Analysis

Anomaly detection is one main issue for operators in industries. They are faced with questions e.g. *"Are there any anomalies in the system?"* or even more specific *"Is the ScaleSF running fine?"*. The questions are formalized by the SC 1 and provided to the RE which determines the answer in the semantic model.

If an algorithm detects an anomaly, it will be stored in the SIM as shown in figure 4. The anomaly *C2* contains all the information which could be determined automatically. It is identified by the algorithm ANODAM5 (ANODA of module 5) and the model *Model_M5_02* that the signal *I_xPot* is wrong. Additionally, the start and finish time of the anomaly is saved. If it is a general question regarding the occurrence of anomalies, all available algorithms are checked whether there is an anomaly or not.

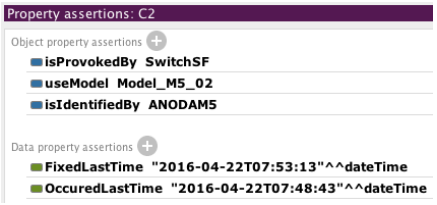

Figure 4: A detected anomaly is represented as individual.

The diagnosis provided can be processed to present the user with more detailed information. The signal name can be transformed into a meaningful name, in this case *Pot is there*. The user can request additional information e.g. about the type of sensor, some values, types of values or, if available, additional information such as a repair instruction.

For more specific question such as *"Is the ScaleSF running fine?"*, the RE has to go into more detail. Therefore, figure 5 depicts the semantic model, it contains both the SMM and the SIM. Boxes with purple diamonds are individuals (SIM), boxes with orange circles are concepts (SMM). Relations are presented as arrows, the type of relation is shown next to it. For a question regarding *ScaleSF* the RE checks whether the signal *I_rfillableCupWeightCell* is covered in the model. In this example, the signal is covered in Model_M5_02 which is used by the algorithm. The anomaly C3 also uses the model, so it is necessary to check whether the scale is the root cause or not.

The results of the anomaly detection part are good. All anomalies at the present are provided correctly, just one question with a past anomaly was not processed correctly.

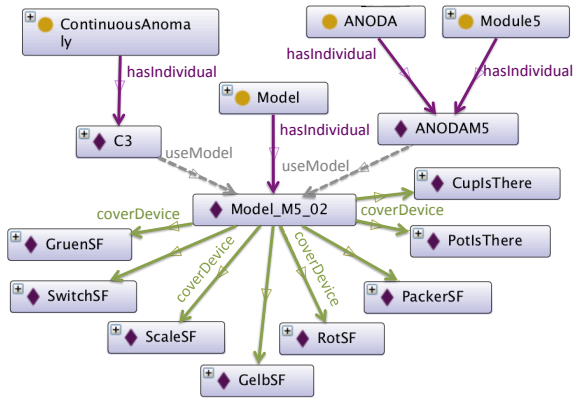

Figure 5: Relations between anomalies, signals, models and algorithms.

The root cause questions are more challenging. No answer was completely wrong, but 4 of 10 answers were not as expected, e.g. a possible cause was missing.

5.4 Empirical Evaluation

To enable communication with natural language, the system must achieve a high level of accuracy to be accepted by users, since incorrect processing lead to frustration fastly. Therefore, a statistical evaluation is made by rating the semantic frame processed by the natural language layer and compare it with the expected frame. Cross-validation is employed with a corpus of 200 questions. 90 % of the data is used for training and 10 % as test data. Initially, an accuracy of 94 % was achieved.

The used Stanford CoreNLP classifier provides a probability for every question regarding how probable it is that they belong to a specific class. If the highest probability of a class membership is lower than 60 %, the classification is suggested as wrong, since all faulty classifications were below this threshold. So these questions are suggested as wrongly classified and the user will be advised to rephrase the question. This leads to a decreased accuracy, but questions that cannot be processed correctly can be detect. Thus, a significant number of faulty semantic frames is prevented by rephrasing the question. The number of faulty answers is reduced which increase the usability of the system. Results with improvement described above are shown in table 1.

	Pred. ture	Pred. false
Correct frames	92%	2%
Incorrect frames	3%	3%

Table 1: Results of the NLP using cross validation

The columns of table 1 indicate the suggestion as to whether the classification was correct or not. Rows mark whether it was really classified correctly or not. The 92 % of the upper left cell is the accuracy which comprises all correct sentences that are correctly classified. The lower right cell is made up of 3 % of the questions, which are wrongly classified and detected as wrongly classified, so that the user gets a message to rephrase the question. 2 % of the questions are classified correctly but with an uncertainty so the user should rephrase the question (upper right cell). Only 3 % of the questions are processed incorrectly, lower left cell, which is not noticed by the natural language layer. Most of the questions are faulty because of the identified entities, the classification of the questions was correct.

Therefore, overall, only 3 % of the questions are not formalized correctly and will be not detected as faulty. Nevertheless, answers for most of the faulty questions cannot be determined, because the entity could not be found in the semantic model. For some questions e.g. *occurred* or *is* is determined as *typeOfEntity* which has to be improved. But the achieved F-measure of 97,4 % is a good result.

6 Conclusion

This paper addresses the conceptual gap which is bridged by a knowledge base that acts as an interface between algorithms and users. Through the semantic layer different diagnosis algorithms of several modules can be combined and provide a single user interface with a common wording, which leads to an easier usage for operators. Additionally, natural language can be used for communication which provides some advantages such as a single GUI which is role-independent. A prototype with the four solution components has been developed, modeled and implemented on a part of the SF. The following findings have been made incorporated into the research questions mentioned in the introduction.

- **RQ 1 - Wording:** A semantic model is developed which defines a reasonable wording based on a hierarchical structure in the domain of diagnosis in production systems.
- **RQ 2 - Mapping** The semantic frame is suitable to provide operators questions in a formalized way to the semantic layer. The RE can use the frame and map the words to known concepts. This is mainly achieved by using a classifier for the language processing.
- **RQ 3 - HMI:** The natural language processing provides an interface which simplifies the usage of production plants. The natural language layer works to an accuracy of 92 % and only 3 % of the questions are wrongly processed; 5 % of the questions have to be rephrased. Such results make it possible to use it as HMI for non-safety critical diagnosis applications. But speech recognition systems have to be adapted to the industrial question; their default accuracy of about 60 % is bad.

Further work will be on the extensions of the knowledge base to use multiple algorithms for the diagnosis of one module. Therefore, the algorithms have to be modeled in the SMM. The main challenge is the dealing with contradictory and hierarchical diagnosis. To do so, relations between different kind of results are necessary. This can obviously lead to new insights and increase the quality of diagnosis. Furthermore, it can be extended to configure algorithms automatically to reduce the manual effort.

Acknowledgment

The work was supported by the German Federal Ministry of Education and Research (BMBF) under the project "Semantics4Automation" (funding code: 03FH020I3).

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

References

[Alm *et al.*, 2015] Rebekka Alm, Mario Aehnelt, and Bodo Urban. Processing manufacturing knowledge with ontology-based annotations and cognitive architectures. In Stefanie N. Lindstaedt, Tobias Ley, and Harald Sack, editors, *I-KNOW*, pages 25:1–25:6. ACM, 2015.

[Blondé *et al.*, 2011] Ward Blondé, Vladimir Mironov, Aravind Venkatesan, Erick Antezana, Bernard De Baets, and Martin Kuiper. Reasoning with bio-ontologies: using relational closure rules to enable practical querying. *Bioinformatics*, 27(11):1562–1568, 2011.

[Chang and Manning, 2012] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pages 3735–3740, 2012.

[DiGiuseppe and Jones, 2012] Nicholas DiGiuseppe and James A. Jones. Semantic fault diagnosis: automatic natural-language fault descriptions. In Will Tracz, Martin P. Robillard, and Tevfik Bultan, editors, *SIGSOFT FSE*, page 23. ACM, 2012.

[Eickmeyer *et al.*, 2015] Jens Eickmeyer, Peng Li, Omid Givehchi, Florian Pethig, and Oliver Niggemann. Data driven modeling for system-level condition monitoring on wind power plants. In *26th International Workshop on Principles of Diagnosis (DX 2015)*, 2015.

[Evans and Annunziata, 2012] Peter C. Evans and Marco Annunziata. Industrial internet: Pushing the boundaries of minds and machines. Technical report, GE, 2012.

[Harcuba and Vrba, 2015] Ondřej Harcuba and Pavel Vrba. Ontologies for flexible production systems. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pages 1–8, 2015.

[Kagermann *et al.*, 2013] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig, editors. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry*. Secretariat of the Platform Industrie 4.0, Frankfurt/Main, 2013.

[Maier and Niggemann, 2015] Alexander Maier and Oliver Niggemann. On the learning of timing behavior for anomaly detection in cyber-physical production systems. In *International Workshop on the Principles of Diagnosis (DX)*, Paris, France, August 2015.

[Maier, 2015] Alexander Maier. *Identification of Timed Behavior Models for Diagnosis in Production Systems*. PhD thesis, University of Paderborn, 2015.

[Pinto *et al.*, 2015] Rui Pinto, João Reis, Vitor Sousa, Ricardo Silva, and Gil Gonşalves. Self-diagnosis and automatic configuration of smart components in advanced manufacturing systems. In *International Conference on Intelligent Systems and Applications (INTELLI)*, pages 164 – 169, 2015.

[Purwins *et al.*, 2014] H. Purwins, B. Barak, A. Nagi, R. Engel, U. Hockele, A. Kyek, S. Cherla, B. Lenz, G. Pfeifer, and K. Weinzierl. Regression methods for virtual metrology of layer thickness in chemical vapor deposition. *Mechatronics, IEEE/ASME Transactions on*, 19(1):1–8, Feb 2014.

[Reiter, 1987] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Scheifele *et al.*, 2014] Stefan Scheifele, Jens Friedrich, Armin Lechler, and Alexander Verl. Flexible, self-configuring control system for a modular production system. *Procedia Technology*, 15:398 – 405, 2014. 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering.

[Schlenoff *et al.*, 2013] Craig Schlenoff, Tsai Hong, Connie Liu, Roger D. Eastman, and Sebti Foufou. A literature review of sensor ontologies for manufacturing applications. In *ROSE*, pages 96–101. IEEE, 2013.

[Schmitt *et al.*, 2013] Robert Schmitt, Stephan Losse, and Eike Permin. *Achieving resource- and energy-efficient system optima for production chains using cognitive self-optimization; 1. [Aufl.]*. Universitätsverl. d. TU, Berlin, 2013.

[Simón *et al.*, 2015] Jorge Santos Simón, Clemens Mühlbacher, and Gerald Steinbauer. Automatic model generation to diagnose autonomous systems. In Yannick Pencolé, Louise Travé-Massuyés, and Philippe Dague, editors, *DX@Safeprocess*, volume 1507 of *CEUR Workshop Proceedings*, pages 153–158. CEUR-WS.org, 2015.

[Sirin *et al.*, 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Web Semant.*, 5(2):51–53, June 2007.

[Stern *et al.*, 2013] Roni Stern, Meir Kalech, Alexander Feldman, Shelly Rogov, and Tom Zamir. Finding all diagnoses is redundant. *The 24th International Workshop on Principles of Diagnosis*, pages 216–221, 2013.

[Terkaj and Urgo, 2014] Walter Terkaj and Marcello Urgo. Ontology-based modeling of production systems for design and performance evaluation. In *12th IEEE International Conference on Industrial Informatics, INDIN 2014, Porto Alegre, RS, Brazil, July 27-30, 2014*, pages 748–753, 2014.

[Tsinarakis and Tsinaraki, 2013] George J. Tsinarakis and Chrisa J. Tsinaraki. Ontomops: A modular production system description ontology. In *21st Mediterranean Conference on Control & Automation (MED)*. IEEE, June 2013.

[Uddin *et al.*, 2011] M. Kamal Uddin, A Dvoryanchikova, A. Lobov, and J.L.Martinez Lastra. An ontology-based semantic foundation for flexible manufacturing systems. In *37th Annual Conference on IEEE Industrial Electronics Society (IECON 2011)*, pages 340 – 345. IEEE, 2011.

[W3C, 2004] W3C. Owl web ontology language reference, February 2004.

[Windmann *et al.*, 2015] Stefan Windmann, Oliver Niggemann, and Heiko Stichweh. Energy efficiency optimization by automatic coordination of motor speeds in conveying systems. In *IEEE International Conference on Industrial Technology (ICIT 2015)*, 2015.