SemDia: Semantic Rule-Based Equipment Diagnostics Tool

Gulnar Mehdi
Siemens Corporate Technology
Technical University of Munich
gulnar.mehdi@siemens.com

Evgeny Kharlamov
University of Oxford
evgeny.kharlamov@cs.ox.ac.uk

Ognjen Savković
Free University of Bozen-Bolzano
Ognjen.Savkovic@unibz.it

Guohui Xiao
Free University of Bozen-Bolzano
xiao@inf.unibz.it

Elem Güzel Kalaycı
Free University of Bozen-Bolzano
ElGuezelKalayci@unibz.it

Sebastian Brandt
Siemens Corporate Technology
sebastian.brandt@siemens.com

Ian Horrocks
University of Oxford
ian.horrocks@cs.ox.ac.uk

Mikhail Roshchin
Siemens Corporate Technology
mikhail.roshchin@siemens.com

Thomas Runkler
Siemens Corporate Technology
thomas.runkler@siemens.com

ABSTRACT

Rule-based diagnostics of power generating equipment is an important task in industry. In this demo we present how semantic technologies can enhance diagnostics. In particular, we present our semantic rule language sigRL that is inspired by the real diagnostic languages in Siemens. SigRL allows to write compact yet powerful diagnostic programs by relying on a high level data independent vocabulary, diagnostic ontologies, and queries over these ontologies. We present our diagnostic system SemDia. The attendees will be able to write diagnostic programs in SemDia using sigRL over 50 Siemens turbines. We also present how such programs can be automatically verified for redundancy and inconsistency. Moreover, the attendees will see the provenance service that SemDia provides to trace the origin of diagnostic results.

CCS CONCEPTS

• **Information systems → Enterprise information systems**;

KEYWORDS

Diagnostic Systems, Rules, Ontologies, Turbines

1 INTRODUCTION

Diagnostic systems play an important role in industry since they help to maximise equipment's up-time and minimise its maintenance and operating costs [18]. In the energy sector companies like Siemens often rely on *rule-based* diagnostics to analyse power generating equipment by, e.g., testing newly deployed electricity generating gas turbines [13], or checking vibration instrumentation [15], performance degradation [16], and faults in operating turbines. For this purpose diagnostic engineers create and use complex diagnostic rule-sets to detect equipment abnormalities.

An important class of rules that are commonly used in Siemens are *signal processing rules* (SPRs) that allow to (1) filter, aggregate, combine, and compare *signals*[1] coming from sensors installed in equipment and (2) fire notification messages when a certain pattern in signals is detected. *Authoring* and *maintaining* SPR based rule-sets is a challenging problem. We now discuss the challenges in mode details and then present our solution to address them.

Challenges with Authoring SPRs. The main challenge for authoring is that SPRs in most modern industrial diagnostic systems including the ones used in Siemens are highly *data dependent* in the sense that specific characteristic of individual sensors and pieces of equipment are explicitly encoded in SPRs. As the result for a typical turbine diagnostic task engineers have to write from dozens to hundreds of SPRs that involve hundreds of sensor ids, component codes, sensor and threshold values as well as equipment configuration and design data. E.g., a typical Siemens gas turbine has about 2,000 sensors and a diagnostic task to verify that the purging[2] is over in the main flame component of a given turbine requires around 300 SPRs, most of which are similar in structure but different equipment specific data values. Thus, there is a need in industry, and in particular in Siemens for a higher level diagnostic rule language that allows to express *what* the diagnostic task should do rather than *how* it should do it for specific equipment. Such language should be high level, data independent, while powerful enough to express in a concise way most of typical diagnostic tasks in Siemens.

Challenges with Management of SPRs. Development of a diagnostic rule-set is typically a collaborative and open-ended process by a group of diagnostic engineers. Thus, the engineers may introduce rules that either *repeat* what other rules already express or *contradict* them, i.e., by stating that purging is 'over' while the other rules say that is it 'in progress'. The former problem of *redundancy* in diagnostic rule-sets affects the performance of diagnostics and the latter of *inconsistency* among rules makes diagnostic results counter-intuitive and unreliable. Moreover, the larger the rule-set gets, the harder it becomes to trace the *provenance* of the messages it fires which again affects the reliability of diagnostic results. Thus, there is a need for semi-automatic rule management support that includes detection of redundancy and inconsistency in rule sets, as well as computation of provenance for diagnostic results.

Our Solution. We rely on *semantic technologies* to address the the above mentioned challenges. In particular we rely on *ontologies* [1]

[1] Signals are are time stamped sequences of measurement values.
[2] Purging is the process of flushing out liquid fuel nozzles or other parts which may contain undesirable residues.

CIKM'17 , Singapore, Singapore

DOI: 10.1145/3132847.3133191

to define a novel SPR language and on *reasoning* [3] over ontologies to foster execution and maintenance of diagnostic tasks. In short, an ontology is a formal conceptualisation of the domain of interest that consists of a *vocabulary*, i.e., names of classes, attributes and binary relations, and *axioms* over the terms from the vocabulary that, e.g., assign attributes of classes, define relationships between classes, compose classes, class hierarchies, etc. Since ontologies are specified using a formal logical language such as the W3C standardised ontology web language OWL 2, one can query ontologies and check their properties using reasoning that typically corresponds to logical entailment and implemented in many efficient state-of-the-art reasoning systems such as HermiT [17]. We refer the reader to [1] for more details on ontologies and reasoning and to [5–7, 9–12] to our previous studies of the semantic diagnostic problem.

In order to address the authoring challenge we propose:

- an SPR language *sigRL* that treats signals as first class citizens and allows to process signals (filter, aggregate, combine, and compare signals) in a high level, declarative, and data independent fashion;
- semantic diagnostic programs that combine *sigRL* rules with diagnostic background knowledge captured using ontologies and allow to express complex diagnostic tasks in an abstract fashion by exploiting both ontological vocabulary and queries over ontologies to identify relevant information (such as sensor ids and threshold values) about the equipment that should undergo the diagnostics.

In order to address the management challenge, we developed efficient algorithms to execute diagnostic programs in bottom-up fashion, verify redundancy and inconsistency in diagnostic programs, and to compute provenance that explains the reasons for diagnostic results. Moreover, we implemented our ideas in a system *SemDia* for diagnostics of power generating equipment that (1) allows to author diagnostic programs; (2) offers a tool-kit for management of diagnostic programs, that includes detection of redundancy and inconsistency, as well as provenance computation.

Note that we designed *sigRL* in such a way that, on the one hand, it captures the main signal processing features required by Siemens turbine diagnostic engineers and, on the other hand allows for efficient execution and management of diagnostic programs.

Demo Overview. Demo attendees will be able to learn how to do diagnostics of Siemens turbines with *sigRL* diagnostic programs. To this end we prepared a deployment of our *SemDia* system on data from 50 Siemens power generating turbines, a diagnostic ontology, and a catalogue of 15 diagnostic tasks. The attendees will be able to load preconfigured diagnostic programs, deploy and execute them, author their own diagnostic programs, and try out our provenance computation and program verification services. See Section 3 for details on demo scenarios.

Due to space limit we put some formal definitions related to our *sigRL* language in the appendix.

2 OUR DIAGNOSTIC SOLUTION

We first introduce our diagnostic language *sigRL* and then describe our system *SemDia*.

2.1 *sigRL* Diagnostic Language

In our setting, a *signal* is a first-class citizen. A signal s is a pair (o_s, f_s) where o_s is *sensor id* and *signal function* f_s defined on $\mathbb{R}$ to $\mathbb{R} \cup \{\perp\}$, where $\perp$ denotes the absence of a value. We assume that we are given a finite set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of *basic* signals that are readings obtained from a single sensor (e.g., in a turbine) for different time points. We now define signal expressions that filter and manipulate basic signals and create new more complex signals. Intuitively, in our language we group signals in ontological concepts and signal expression are defined on the level of concepts. Then, a *signal processing expression* is recursively defined as follows:

$$\begin{array}{rcl} C & \leftarrow & \alpha \circ C_1 \quad | \quad agg\, C_1 \quad | \\ & & C_1 : \mathsf{filterValue}(\odot, \alpha) \quad | \quad C_1 : \mathsf{filterTime}(\odot, \alpha) \quad | \\ & & C_1 : \mathit{filterAlign}\, C_2 \quad | \quad C_1 : \mathsf{trend}(\mathit{direction}). \end{array}$$

where C, C_1, C_2 are concepts, $\circ \in \{+, -, \times, /\}$, $\alpha \in \mathbb{R}$, *agg* is one of min, max, avg, sum, $\odot \in \{<, >, \leq, \geq\}$, *filterAlign* $\in$ {within, after[t], before[t]} where t is a period and *direction* is either {up, down}. Intuitively, if $C = \alpha \circ C_1$ then C contains one signal s' for each signal s in C_1 with function defined with $f_{s'} = \alpha \circ f_s$, or if $C_1 : \mathsf{filterValue}(\odot, \alpha)$ then C contains one signal one signal s' for each signal s in C_1 with $f_{s'}(t) = \alpha \odot f_s(t)$ if $f_s(t) \odot \alpha$ at time point t; otherwise $f_{s'}(t) = \perp$. The formal meaning of signal processing expressions is defined in [9].

A *diagnostic program* is a tuple $\Pi = (\mathcal{S}, \mathcal{O}, \mathcal{M})$ where $\mathcal{S}$ is a set of basic signals, $\mathcal{O}$ is an ontology[3], $\mathcal{M}$ is a set of signal expressions. Each diagnostic program comes with a set of *message rules* that are defined on top of Boolean combinations of expressions:

$$r = msg(m) \leftarrow D, \quad where$$
$$D := \quad C \quad | \quad not\, C \quad | \quad D_1\ and\ D_2$$

We now illustrate our diagnostic programs on the following purging diagnostic task:

> *Verify that the purging is over in the main flame component of the turbine T1.*

Intuitively this task requires to check in the turbine T1 that: *(i)* the main flame was on for at least 10s and then stopped, *(ii)* 15s after this, the purging of rotors in the starting-component of T1 started, *(iii)* 20s after this, the purging stopped. The fact that the purging of a rotor started or ended can be detected by analysing its speed, i.e., by comparing the average speed of its speed sensors with purging thresholds that are specific for individual rotors. The purging diagnostic program can then consist of an ontology with one axiom:

$$\mathsf{SubClassOf}(\mathsf{RotorSensor\ SpeedSensor}).$$

two signal processing expressions and one message rule:

$$\begin{array}{l} \mathsf{PurgingStart} = avg\ \mathsf{rotorStart} : \mathsf{value}(>, \mathit{purgingSpeed}), \\ \mathsf{PurgingStop} = avg\ \mathsf{rotorStart} : \mathsf{value}(<, \mathit{nonPurgingSpeed}), \\ \mathsf{msg}(\text{“Purging over”}) = \mathsf{FlameSensor} : \mathsf{duration}(>, 10s) : \\ \qquad after[15s]\ \mathsf{PurgingStart} : after[20s]\ \mathsf{PurgingStop} \end{array}$$

2.2 *SemDia* Diagnostic System

The main functionality of our semantic rule-based diagnostics system *SemDia* is to author and maintain *sigRL* diagnostic programs, to deploy them in turbines, to execute the programs, and to visualise the results of the execution. We now give details of our system by following its architecture in Figure 1 where the solid arrows indicate data flow and dashed—access to ontologies and mappings. There are three essential layers in the architecture: application,

[3] In order to guarantee efficiency of diagnostics with *sigRL*, we consider a tractable ontology language OWL 2 QL [3] that is standardised by W3C.

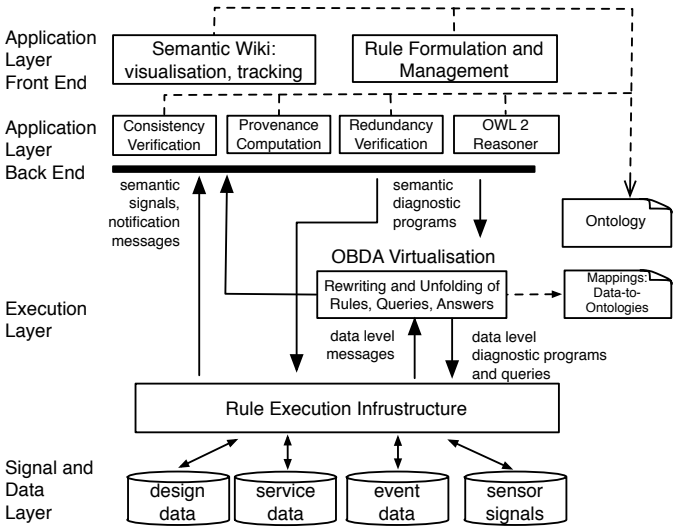

Figure 1: Architecture of *SemDia*.

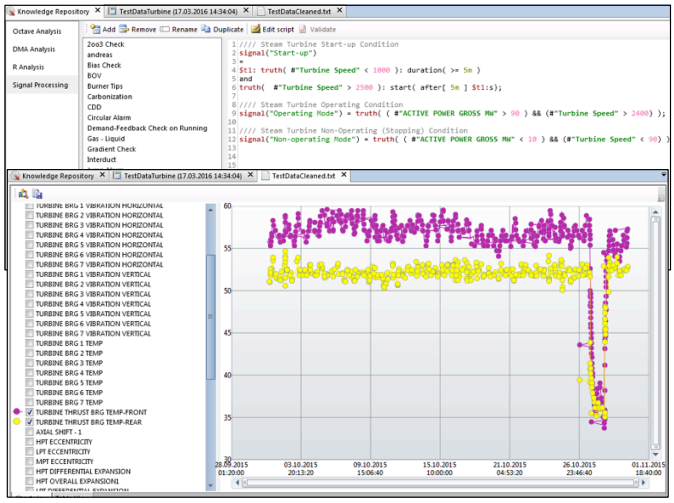

Figure 2: Screenshots: SPR editor (top), Diagnostic visualisation monitor (bottom)

rule execution, and signal and data layers. Our system is mostly implemented in Java. We now discuss the system layer by layer.

Application Layer. On this layer, the system offers two front ends and several back end components. The first front end component allows engineers to author, store, load, and maintain diagnostic programs by formulating sets of SPRs as well as message rules in *sigRL* and sensor retrieving queries. Such formulation is guided by the domain ontology stored in the system. In Figure 2 (left) one can observe a screenshot of the diagnostic program editor which is embedded in the Siemens analytical tool-kit. Another front end component is the semantic Wiki that allows among other features to visualize signals and messages (triggered by programs), and to track deployment of programs in equipment. In Figure 2 (right) one can see visualisation of signals from two components of one turbine. The back end of the application layer consist of components that rely on HermiT [17] ontology reasoning and support consistency and redundancy verification as well as provenance computation. Diagnostic programs formulated in the application layer are converted into XML-based specifications and sent to the rule execution layer that returns back messages and signals. We rely on REST API to communicate between the application layer and the execution layer of our system and OWL API to deal with ontologies.

Execution Layer. On this layer we support semantic signals that are either *native*, that is, represented in terms of the diagnostic ontology as RDF triple, or *virtual*, that is obtained through the Ontology Based Data Access (OBDA) [14] component of *SemDia*. This component allows to present signals stored in relational databases as if they were native semantic. This requires to connect the relational signals to an ontology via declarative mappings. For the OBDA layer we rely on the Ontop system [2] and its extension developed within the Optique project [4, 8] that takes care of transforming diagnostic programs written in *sigRL* into either SPRs written in the Siemens data-driven rule language or SQL. This transformation has two steps: rewriting of programs and queries with the help of ontologies (at this step both programs and queries are enriched with the implicit information from the ontology), and then unfolding them with the help of mappings. Moreover, the execution layer takes care of planning and executing rules and queries received either from the rule management or OBDA component. If the received rules are in the Siemens SPR language then the rule executor instantiates them with concrete sensors extracted with queries and passes them to the Drools Fusion (drools.jboss.org/drools-fusion.html) the engine used by Siemens. If the received rules are in SQL then it plans the execution order and executes them together with the other queries.

Signal and Data Layer. On this layer we store all the relevant data: turbine design specifications, historical information about services that were performed over the turbines, previously detected events, and the raw sensor signals. Currently *SemDia* support PostgresQL, Teradata, as well as Sparksee.

3 DEMONSTRATION SCENARIOS

Demo Setup. For the demonstration purpose we prepared the following ingredients:

- *diagnostic tasks*: 15 tasks (see Table 1) were defined during brainstorming sessions with Siemens diagnostic engineers and R&D personnel from Siemens Corporate Technology;
- *anonymised Siemens data*: from 50 power generating gas turbines gathered for 2 years that contains sensor signals, equipment specifications and configurations and maintenance data; all the data was anonymised for the demo purpose;
- *Siemens diagnostic ontology*: the ontology was inspired by the Siemens Technical System Ontology (TSO), Semantic Sensor Network Ontology (SSN), and the international standards IEC 81346 and ISO/TS 16952-10; the main module of our ontology in partially depicted in Figure 3 where in grey we present SSN and with white TSO terms. This module has 48 classes and 32 object and data properties. The other three modules are respectively about equipment, sensing devices, and diagnostic rules. They provide detailed information about the machines, their deployment profiles, sensor configurations, component hierarchies, functional profiles and logical bindings to the analytical rule definitions.
- *SemDia deployment in Siemens*: over both materialised and ontology mediated Siemens data; *SemDia* is deployed on Teradata for signals and MS SQL for other information; for rule processing *SemDia* uses Drools Fusion; for the OBDA setting we developed 376 R2RML mappings.

Task Nr	Complexity	Diagnostic Tasks
T1	Low	Variable guided vanes analyses
T2	Low	Multiple start attempts
T3	Low	Lube oil system analyses
T4	Medium	Monitoring turbine states
T5	Medium	Interduct thermocouple analyses
T6	Medium	Igniter failure detection
T7	High	Bearing carbonisation
T8	High	Combustion chamber dynamics
T9	High	Gearbox Unit Shutdown
T10	High	Surge detection

T11	Does the turbine T100 reach purging and ignition speed for 30 sec?
T12	Is the purging over in the main flame component of the turbine T1.
T13	Does the turbine T100 reach purging and ignition speed for 30 sec?
T14	Does the turbine T100 go from ignition to stand still within 1min and then stand still for 30 sec?
T15	Is the turbine T100 ready to start?

Table 1: Demo diagnostic tasks for Siemens gas turbines.

During the demo we can either remotely connect to our deployment at Siemens or to run demo on a laptop. We now explain demo scenarios in details.

Scenario 1: Preconfigured Diagnostics. We plan to present the demo starting from this scenario where the attendees will get acquainted with *sigRL* and the main functionalities of *SemDia*. For this purpose we prepared 10 tasks of different complexity, where complexity is de ned using the number of ontological terms and lines of code. These tasks can be found in Table 1, they are indicated as T1–T10. The attendees will start with the tasks of *low* complexity by loading the corresponding diagnostic programs in the *SemDia* rule editor, learning the basic signal processing operations of *sigRL*, and then deploying the programs over the turbines. Then, the attendees will study the diagnostic results on the visualisation monitors. From the visualisation monitor the attendees will be able to request the provenience for specific answers. After the tasks with *low* complexity the attendees will proceed to the tasks with *medium* and *high* complexity. We have intentionally introduced inconsistencies and redundancy in the tasks so that the attendees will be able to detect them using the tooling support of *SemDia*.

Scenario 2: User Defined Diagnostics. In this scenario we assume that the demo attendees have already learned the basics of *sigRL* and now they are ready to author diagnostic programs in our editor depicted in Figure 2. For this purpose we prepared five tasks T11–T15 that can be found in Table 1. Note that for simplicity we phrased the tasks T11–T15 in such a way that the wording follows the actual grammer of *sigRL*. Moreover, most of semantic terms relevant for these tasks are explicitly mentioned in the tasks. Observe that the first four tasks are independant from each other, while the last task, T15, combines the other four. This will show the attendees the compositionality of our language *sigRL*. For illustration consider the following diagnostic program that corresponds to the task T13:

$$\text{Ignition} = \mathit{avg}\ \text{RotorSensor} : \text{value}(<, \mathit{ignitionSpeed}).$$
$$\text{PurgeAndIgnition} = \text{PurgingStart} : \text{duration}(>, 30s) : \mathit{after}[2m]\ \text{Ignition} : \text{duration}(>, 30s).$$
$$\text{msg}(\text{“Purging and Ignition”}) = \text{PurgeAndIgnition}.$$

And now the diagnostic program corresponding to T15:

$$\text{msg}(\text{“Ready to Start”}) = \text{RampChange} : \mathit{after}[5m]\ \text{PurgingOver} : \mathit{after}[11m]\ \text{PurgingAndIgnition} : \mathit{after}[15s]\ \text{IgnitionToStand}.$$

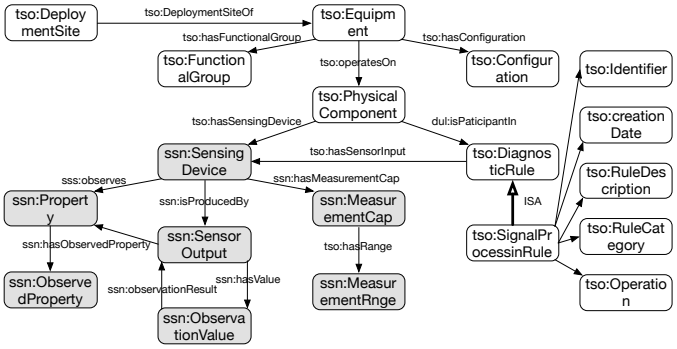

Figure 3: A fragment of the Siemens ontology that we developed to support turbine diagnostic SPRs.

While the attendees formulate the five diagnostic tasks as diagnostic programs to check they will be asked to verify them for redundancy and consistency. After they finish formulation and verification, they can run the programs and check provenance.

Acknowledgements. This research is supported by the EPSRC projects MaSI^3, DBOnto, ED^3; and by the FUB project QUEST.

REFERENCES

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8, 2017.

[3] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR*, 39(3), 2007.

[4] I. Horrocks, M. Giese, E. Kharlamov, and A. Waaler. Using semantic technology to tame the data variety challenge. *IEEE Internet Computing*, 20(6):62–66, 2016.

[5] E. Kharlamov, S. Brandt, M. Giese, E. Jiménez-Ruiz, Y. Kotidis, S. Lamparter, T. Mailis, C. Neuenstadt, Ö L. Özçep, C. Pinkel, A. Soylu, C. Svingos, D. Zheleznyakov, I. Horrocks, Y. E. Ioannidis, R. Möller, and A. Waaler. Enabling semantic access to static and streaming distributed data with optique: demo. In *DEBS*, pages 350–353, 2016.

[6] E. Kharlamov, S. Brandt, E. Jiménez-Ruiz, Y. Kotidis, S. Lamparter, T. Mailis, C. Neuenstadt, Özgür L. Özçep, C. Pinkel, C. Svingos, D. Zheleznyakov, I. Horrocks, Y. E. Ioannidis, and R. Möller. Ontology-Based Integration of Streaming and Static Relational Data with Optique. In *SIGMOD*, 2016.

[7] E. Kharlamov, D. Hovland, E. Jiménez-Ruiz, D. Lanti, H. Lie, C. Pinkel, M. Rezk, M. G. Skjæveland, E. Thorstensen, G. Xiao, D. Zheleznyakov, and I. Horrocks. Ontology based access to exploration data at statoil. In *ISWC*, 2015.

[8] E. Kharlamov, E. Jiménez-Ruiz, D. Zheleznyakov, D. Bilidas, M. Giese, P. Haase, I. Horrocks, H. Kllapi, M. Koubarakis, Ö. Özçep, M. Rodriguez-Muro, R. Rosati, M. Schmidt, R. Schlatte, A. Soylu, and A. Waaler. Optique: Towards OBDA systems for industry. In *ESWC Satellite Events*, pages 125–140, 2013.

[9] E. Kharlamov, O. Savkovic, G. Xiao, R. Penaloza, G. Mehdi, I. Horrocks, and M. Roshchin. Semantic rules for machine diagnostics: Execution and management. In *CIKM*, 2017.

[10] E. Kharlamov, N. Solomakhina, Ö. Özçep, D. Zheleznyakov, T. Hubauer, S. Lamparter, M. Roshchin, A. Soylu, and S. Watson. How Semantic Technologies Can Enhance Data Access at Siemens Energy. In *ISWC*, pages 601–619, 2014.

[11] G. Mehdi, E. Kharlamov, O. Savkovic, G. Xiao, E. Kalayci, S. Brandt, I. Horrocks, M. Roshchin, and T. Runkler. Semantic rules for Siemens turbines. In *ISWC Posters & Demos*, 2017.

[12] G. Mehdi, E. Kharlamov, O. Savkovic, G. Xiao, E. Kalayci, S. Brandt, I. Horrocks, M. Roshchin, and T. Runkler. Semantic rule-based equipment diagnostics. In *ISWC*, 2017.

[13] John S. Mitchell. *An introduction to machinery analysis and monitoring*. Pennwell Books, 1993.

[14] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[15] Robert Bond Randall. *Vibration-based condition monitoring: industrial, aerospace and automotive applications*. Wiley, 2011.

[16] B. K. N. Rao. *Handbook of condition monitoring*. Elsevier, 1996.

[17] R. Shearer, B. Motik, and I. Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In *OWLED*, 2008.

[18] G. Vachtsevanos, F. L. Lewis, M. Roemer, A. Hess, and B. Wu. *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. Wiley, 2006.