FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Combining Ontologies and Statistics for Sensor Data Quality Improvement

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

im Rahmen des Erasmus-Mundus-Studiums

Computational Logic

eingereicht von

Nina Solomakhina
Matrikelnummer 1228203

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Prof. Dr. Thomas Eiter
Mitwirkung: Dipl.-Inf. Thomas Hubauer (Siemens AG)

Wien, 31.03.2014

(Unterschrift Verfasserin) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Combining Ontologies and Statistics for Sensor Data Quality Improvement

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science (M.Sc.)

in

Computational Logic

by

Nina Solomakhina
Registration Number 1228203

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Prof. Dr. Thomas Eiter
Assistance: Dipl.-Inf. Thomas Hubauer (Siemens AG)

Vienna, 31.03.2014

(Signature of Author) (Signature of Advisor)

Acknowledgements

I would like to offer my special thanks to all those people who have made this thesis possible. First and foremost, I would like to express my very great appreciation to Professor Thomas Eiter, for his valuable feedback and useful critiques of this research work. Without his supervision and support this thesis would not have been possible. I am deeply grateful to Thomas Hubauer, for his useful guidance, constructive remarks and engagement through the learning process of this master thesis. Furthermore, I wish to thank the academic staff of the EMCL consortium's universities in Dresden, Bolzano and Vienna, they taught me a lot during my studies and I am very grateful for that. I also take this opportunity to express my sincere gratitude to the EMCL Joint Commission for making my studies possible on the EMCL Master program. My special thanks are extended to my colleagues at Siemens AG in Munich, for their cooperation and assistance in the course of this work. I would specially like to thank Dr. Mikhail Roshchin, who introduced me into the topic and supported me throughout the work.

I want to offer a special thanks to my family: my parents and my grandmother made me believe that I could achieve anything I set my mind to. I want to thank them for their faith in me and continuous support. My sincere thanks goes to my brother, who always encourages me and gives a good advice. Finally, I am very grateful to my friends, for bringing great joy into my life.

Abstract

In large industries usage of advanced technological methods and modern equipment comes with the problem of storing, interpreting and analyzing huge amount of information. Typical sources for this data include a myriad of sensors mounted at the industrial machinery, measuring qualities such as temperatures, movement and vibration, pressure, and many more. However, these sensors are complex technical devices, which means that they can fail and their readings can become unreliable, or "dirty". Low quality data makes it hard to solve the original task of assessing system and process status and controlling the system behavior. So, data quality is one of the major challenges considering a rapid growth of information, fragmentation of information systems, incorrect data formatting and other issues. The aim of this thesis is to propose a novel approach to address data quality issues in industrial datasets, in particular, measurements of sensors mounted at power generation facilities.

The most common approach to detect anomalies in data is the analysis by means of the statistical and machine learning techniques. However, analyzing data alone can not always give satisfactory results. For instance, suspicious sensor readings may not indicate at bad quality of data, but at an appliance functioning abnormality detected by this sensor. Therefore, we propose to use additional available information on the domain. The approach presented in this work brings together several well-known techniques, which come from the worlds of computational logic and statistics, improving the results of data quality assessment and improvement procedure. The application domain and the dependencies between its objects are represented as a knowledge-based model, while statistics identifies data anomalies, such as outlying or missing values, in sensor measurement data. In this work we represent domain knowledge in OWL ontology, which covers the topology of an industrial equipment and an information about measuring devices installed. Providing statistical computations with the additional information from the model allows to validate and improve the results. Thus, comparing and analyzing readings provided by sensors of the same type and mounted at the same component of an appliance helps to identify possibly damaged sensors, as well as to distinguish between data quality inconsistencies found in single sensor readings from anomalies in machinery functioning detected by other measuring devices.

Based on the proposed approach a software demonstrator has been implemented and tested, proving that the usage of the additional information provided by the semantic model improves the results of statistical analysis.

Kurzfassung

Die zunehmende Nutzung fortschrittlicher Technologien und Maschinen in der Industrie geht Hand in Hand mit der Herausforderung der Speicherung, Interpretation und Analyse immenser Informationsmengen. Diese werden unter anderem produziert von einer Vielzahl verbauter Sensoren zur Messung von Temperatur, Bewegung, Vibration, Druck, und einer Vielzahl anderer Qualitäten. Allerdings handelt es sich bei diesen Sensoren um hochentwickelte technische Systeme, die selbst eine Fehlfunktion entwickeln und in Folge dessen fehlerhafte (oder "unsaubere") Daten liefern können. Das ursprüngliche Ziel bestimmung von System- und Prozesszustand wird hierdurch zumindest erschwert.

Angesichts immer größerer und zunehmend heterogener Datenmengen wird Datenqualität so zu einer zentralen Herausforderung in der Industrie. Ziel dieser Arbeit ist die Entwicklung eines neuen Verfahrens zur Behandlung von Datenqualitätsproblemen in industriellen Datensätzen im Allgemeinen, und in Sensordaten von Turbinen im Speziellen.

Die am häufigsten verwendeten Ansätze zur Erkennung von Anomalien in Daten sind statistische Analyse und Techniken des maschinellen Lernens. Jedoch sind deren Ergebnisse oft nicht ausreichend, so können beispielsweise verdächtige Sensorablesungen nicht nur durch geringe Datenqualität, sondern auch durch eine Fehlfunktion des Geräts verursacht werden. Der hier vorgestellte Ansatz kombiniert daher bekannte Techniken aus den Bereichen der angewandten Logik und der Statistik mit dem Ziel, die Erkennung und Behebung von Datenqualitätsproblemen zu verbessern. Die Anwendungsdomäne und Abhängigkeiten zwischen den Objekten werden hierbei als deklaratives Modell dargestellt, wohingegen statistische Methoden zur Erkennung von Anomalien in Sensordaten wie etwa Oszillationen oder fehlenden Werten genutzt werden. Das deklarative Wissen umfasst hierbei Informationen über Topologie und Messeinrichtungen des betrachteten Geräts, dargestellt als OWL Ontologie. Diese zusätzlichen Informationen ermöglichen eine Validierung und Verbesserung der Ergebnisse in der statistischen Komponente, indem Messwerte von Sensoren desselben Typs, die an derselben Komponente montiert sind, miteinander verglichen werden. Fehlerhafte Sensordaten können so von tatsächlichen Anomalien des beobachteten Systems unterschieden werden.

Basierend auf dem vorgeschlagenen Ansatz wurde ein Software-Demonstrator implementiert. Unsere Evaluierung zeigt, dass durch die Nutzung der zusätzlichen Informationen aus dem semantischen Modell die Ergebnisse der statistischen Analyse signifikant verbessert werden können.

Contents

CHAPTER 1

Introduction

Motivation

Each phenomenon around us can be documented and expressed by pieces of information. There exist numerous different ways to represent real-world objects: categories, descriptions, numerical measurements, geographical coordinates, sound, graphics and a lot of other formats. All this collected data is retrieved and manipulated by people and software systems for various purposes. Day by day the amount of stored data grows drastically due to advanced technological progress: as more and more data is being collecting whereas storage becomes cheaper.

However, there exists a number of factors that might corrupt and damage the data. One example of such factors is the noise in signal processing. The presence of noise in a sound makes it difficult to hear it accurately. In a digital photograph noise affects brightness, color display, and other parameters, which result in unclear and distorted images. Similarly, a digital signal may be affected by random unwanted fluctuations or distortion.

Another example of factors affecting the data are typos. While filling in textual data, e.g., addresses for a courier company, patient data collected for a hospital, or employees personal information for a hiring company, there is a possibility of writing names or addresses wrongly by mistake. Likewise, error in calculations affect quantitative data.

Consequently, we can not always rely on the existing data and blindly trust it: data requires continuous control and quality improvement in case of a presence of factors that could cause mistakes and inaccuracies in data. The design of a convenient technique which is able to handle low quality data has become a formidable challenge.

Problem statement

Automatic processing of data for the purpose of determining operating states and identifying faults has become essential for many modern industrial systems. Typical sources for this data include hundreds of sensors mounted at the device, measuring qualities such as temperatures, movement and vibration, pressure, and many more. However, these sensors are complex techni-

cal devices, which means that they can fail and their readings can become unreliable, or “dirty”. Potential flaws in data include:

- *inliers* - values within defined range but deviating largely from previous and following values;
- *outliers* - values exceeding defined thresholds;
- various types of *missing data* - visible or hidden by noise;
- *oscillations* and other anomalous data.

Such low quality data makes it hard to solve the original task of assessing system and process status and controlling the system behavior. Overall it considerably reduces reliability of the system and in particular invalidates system analysis results. To reduce cost of misguided decisions and the workload of human operators, it is therefore vital to have effective processes for assessing and improving the quality of sensor data in an automatic or semi-supervised way.

One of the most significant branches in industry is power generation. Gas and steam turbines, power generators and other complex machinery become more complicated, high-powered and automated. Caused by technological improvements, decreasing software cost and increasing demands on information, the amount of data produced, processed and stored by companies grows continuously. Operations engineers retrieve this data for analysis such as diagnostics in case of appliance malfunctions, regular maintenance, improving product design, draw up long-term operation plans, and for other numerous purposes. Even small amounts of poor quality data may cause problems and costly consequences, and Data Quality Assessment has become one of the most crucial tasks in information collection and database management.

One of the most fundamental tasks in energy domain is a fault detection and condition monitoring of the power generation machinery. Energy companies run thousands of power plants, where each operates with several power generation facilities. Hundreds of sensors are installed on a single piece of equipment for a close control of its status and processes. Every sensor produces measurements at a rate between 1 Hz and 1000 Hz, depending on its purpose and characteristics. They continuously gather quantitative characteristics for subsequent analysis, such as diagnostics, detection of abnormal behavior, and predictive maintenance planning. Therefore, a huge amount of data is generated per one day of functioning. In the case of any malfunctions of the appliance, engineers have to conduct diagnostics based on the sensor measurements. However, the required data might be lost, corrupted or damaged. The reasons for that include:

- sensor failures and inaccuracies;
- poor connection between an appliance and a database;
- operation mistakes;
- failures of a control unit.

Thus, engineers have to deal with a large and complex dataset, that may contain errors and missing data. Taking into account that conducting an analysis and diagnostics with the wrong data may lead to a serious consequences, we need to detect and clean "dirty" data.

Experience shows that this task can not always be solved satisfactorily by looking at the data alone. However in the engineering domain we have an abundance of additional information available, including the complex topology of an appliance with its numerous components and measuring devices attached. This additional knowledge of the appliance's structure and functioning should be used for a more precise data quality assessment.

However, currently the largest portion of the known approaches to sensor data quality assessment is rooted in the application of machine learning and statistical methods. Well-known and effective methods, such as descriptive statistics and its moving window applications, exploratory analysis (the analysis of trends and seasonal effects), fitting regression models, usage of clustering algorithms and neural networks for outlier detection, and sophisticated time-series analysis methods such as the ARIMA model and the Kalman filter. However, most of these approaches neglect that in the context of industrial appliances, there is additional information on the system (e.g., structural, functional, and process information) which could be employed to improve both the accuracy detecting low quality data and the "likeness" of generated replacement data to the unknown original data. This leads us to cases where errors such as outliers are not identified appropriately, for example:

- There is a failure of an appliance occurred, e.g., emergency shutdown or a trip. Sensors monitoring the functioning of a turbine detected the change in the process. Analyzing sensor measurements, statistical methods the most probably identify measurements taken during turbine failures as bad quality data points, which have to be corrected. Similarly, in case of turbine malfunction, some sensor readings might oscillate, e.g., in case of high vibrations of an appliance. These oscillations periods are possibly identified as data quality issues as well by statistical approaches, whereas these anomalous data is necessary to have for diagnostics of an appliance. Therefore, it is important to distinguish between anomalies caused by turbine failures and anomalies caused by sensor faults or bad connection. Using statistical analysis only, we can not do that, as in this case the best method to distinguish these anomalies is to get access to duplicated sensors and sensors deployed at the same component of the turbine, in order to analyze, if they observed this anomaly as well, and the information on sensors, which we can use for the comparison, is contained in the domain as a knowledge.

- There are operational ranges for each sensor defined. For instance, consider a sensor installed in the combustor of the turbine (the component of the turbine where the fuel is burned) and the temperature slowly decreases from 700^oC to 50^oC. Statistical methods might identify that as the simple trend. However, it is obviously an anomaly (fire can not have only 50^oC temperature): either turbine had a shutdown and combustor chamber cools down or the data was corrupted. In both cases, we need domain knowledge to doubt that; without knowing the operational conditions and ranges of measuring devices we can not identify this situation as a suspicious.

Therefore, it is necessary to consider additional domain information and appliance structure to correctly identify errors in sensor measurements.

Aim and Contributions of the work

The main idea of the thesis is to elaborate an approach, that helps to detect and to eliminate data quality inconsistencies in industrial datasets, in particular, in data generated by various sensors and measuring devices during functioning of power generation facilities in the energy domain. In order to detect data quality inconsistencies in sensor readings as precise as possible, the aim of the thesis is to make use of the available domain information and to design and prepare for use a comprehensive model of the domain, which would work with the exploratory statistical methods applied to measurement data. The *underlying hypothesis* is that the usage of knowledge-based component and available domain information gives better results in detecting anomalies in sensor measurement data and increases measures of accuracy and precision.

The theoretical part of the thesis includes a study of the field of data quality. We provide basic notions of data quality and procedures of data quality assessment and improvement, and also an overview of existing frameworks and approaches designed for data cleaning and for sensor readings in particular, which are used in industrial applications. Additionally, we present a literature study with the comprehensive overview for methods available for:

- outlier detection and correction;
- oscillation detection and smoothing;
- missing data prediction.

We also provide necessary preliminaries for description logics and Web Ontology Language, as the resulting approach shall integrate statistical with knowledge-based approaches for maximum effectiveness.

The second part of the thesis is practical and focuses on implementing a prototypical system realizing the proposed methodology. First of all, we thoroughly considered the application domain and devised a holistic turbine model. This model is represented as OWL 2 ontology; it covers most of the domain knowledge and restrictions, namely, describes the following three main aspects of the use case:

- turbine structure: its platforms, units and components;
- sensors: types, characteristics, deployment in an appliance;
- diagnostics: failures of turbines and their symptoms.

Second, we devise statistical methods that we apply to the available sensor measurement data from the domain: we revisit the overview of the methods we provide in theoretical part and describe our choice for the implementation of our approach. Another aspect for the statistical method is to determine an order of operations we conduct, for example, we explain, why is it undesirable to correct outliers first and then run an oscillation detection procedure and give a corresponding example.

Then, we show, how statistical and knowledge-based interact with each other in order to analyze sensor measurements and detect anomalous data points in it.

And, finally, we provide evaluation results and analyze them, providing some evidence in support of the hypothesis.

The current work is conducted in the context of the EU-funded project "Optique" [78].

Methodological approach

As motivated before, the best effectiveness is expected to be achieved with the combination of knowledge-based methods and statistical techniques for data quality assessment and improvement in order to use knowledge provided by the domain to avoid appearance of situations outlined above, where the application of statistical approaches alone is not sufficient. We developed components operating with sensor data, one employing statistical approaches and the other making use of knowledge-based techniques, and established an interaction between them, that provides better results in improving data quality. The two components used in the process of work, are:

1. *The statistical/probabilistic component* applies univariate analysis for a measurements of a single sensor to identify possible outlier points, oscillation periods and other anomalies, and refers to the *knowledge-based part*, if there are sensors, which measurements can be used for validating found anomalies and to distinguish between data quality issues and turbine unusual behavior detected by other sensors. We use the following methods:

   - the Hampel filter [82] for detection and correction of outliers;
   - regression [83] and ARIMA models [16] for prediction of missing values;
   - analysis of the autocorrelation function and moving window techniques for oscillation and noise detection and correction.

   For application of the methods and implementation of the component the programming language R for statistical computing [3, 49] is used.

2. *The knowledge-based component* encodes domain knowledge as an OWL 2 ontology and describes three main aspects of the use case:

   - turbine structure: its platforms, units and components;
   - sensors: types, characteristics, deployment in an appliance;
   - diagnostics: failures of turbines and their symptoms.

   The domain knowledge determines sensor clusters; for each such group we use the *statistical part* for multivariate analysis to detect cases when the measurements differ and do not correlate well. Also, the knowledge base contains domain restrictions that have to be checked with the statistics as well, for example, measurement ranges for each sensor. In particular, for each individual sensor we encode the following information in the model:

- affiliation with a certain component part or mechanism in the appliance,
- measuring capabilities, value and unit,
- set (or *cluster*) of duplicating sensors, if any,
- set (or cluster) of comparable sensors (e.g., that are installed on the same component and monitor the same process),
- precision, accuracy, sensor vendor and other specifications.

The statistical and knowledge-based modules support each other in identifying false positives reported (and/or false negatives not reported), thereby improving precision and recall of the combined solution in comparison with the application of statistical analysis to data alone, as we also detail in Chapter 4.

The proposed method is suitable to be used in the context of model-based data access control. For example, so called *Ontology-Based Data Access* (OBDA) systems [56] [41], that provide end-users with domain-language access to data by automatically translating user questions into queries of the underlying data base(s), can be extended by the proposed methods, it is one of the goals for the "Optique" project.

Structure of the work

The thesis is structured as follows: in Chapter 2 we present the theoretical aspects of the topic, i.e., introduce the field of study and define necessary concepts and notation. Moreover, we perform literature studies and provide an overview of existing frameworks and methods. Next, in Chapter 3 we concentrate on our particular use case from the area of power generation. We introduce how the appliances function and how sensors monitor their functioning. Next, we specify our approach to solve data quality deficiencies in measurements, i.e. describe in detail statistical and semantic components and underlying concepts and methods. Chapter 4 presents an implementation and evaluation. In Chapter 5 we conclude the thesis and discuss open issues and the possibilities for the improvement of our approach. The Appendix contains some additional information on the use case and precise schemes of the ontologies designed in the process of this work.

CHAPTER 2

Analysis of existing approaches

2.1 Preliminaries

The Notion of Data Quality

Having data available does not guarantee the possibility to manipulate the information and to derive benefit from it: the data might happen to be incorrect or damaged. In that case the data has a *poor quality* and is not suitable for use. That is the simplest one of the multiple definitions of data quality [92] - a measure of how "bad" is the data, i.e., whether it is corrupted and represents wrong information. Broader definitions list more criteria of good quality, such as relevance, consistency and comprehensiveness of data. Therefore, *Data Quality (DQ)* is the measure of data to be valid, correct, trustworthy and suitable for its purposes. Clear, unambiguous, consistent data which can be processed immediately by demand is called *high-quality data*. However, in real life data seldom has a high quality. Reasons for that might be data entry mistakes, measurement errors, semantic heterogeneity, different data types for similar columns and much more [92].

Data quality is a complex and multidisciplinary field, intersecting with other research areas related to manipulation of data, involving various techniques, and applicable to many real-life domains, such as life sciences [36], business [29], health care [38, 90], industry [48], and others. Figure 2.1 illustrates the intersections of the DQ area with other research fields and lists problems that they have in common.

Data Quality Dimensions

Researchers in the field of DQ deal with a broad variety of types and formats of data, sources it comes from, its purposes, and errors in it. Therefore, they propose different classifications of notions and entities they operate with: data, its characteristics, and information systems which manages it. For instance, data is divided into categories based on its parameters: structure, types, level of processing, change frequency and others [11]. Information systems are also classified according to various criteria, such as distribution, heterogeneity, and autonomy [94]. In this section we concentrate on the description and categorization of DQ attributes and issues.

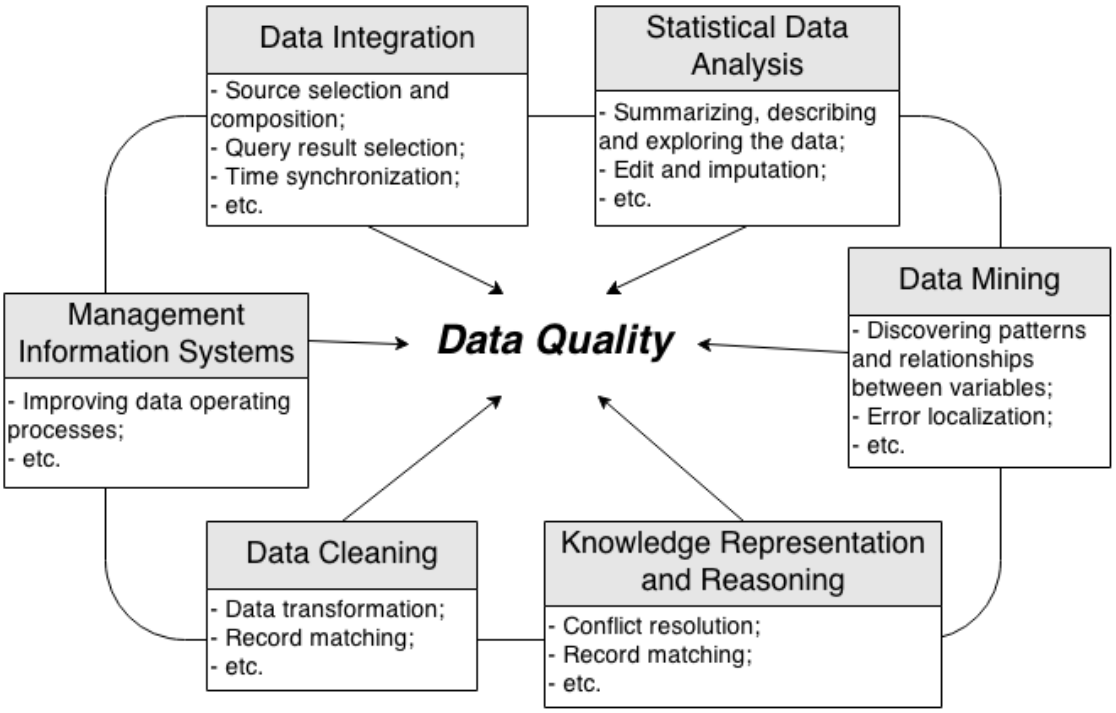


Figure 2.1: Research issues related to Data Quality [11].

In order to precisely describe data deficiencies, there exist specific attributes of data. They are called *Data Quality Dimensions* (or *Data Quality attributes*, *quality criteria*) and they are aimed to provide comprehensive overview of data and its "weak spots". Although the word "dimension" usually implies multidimensionality of the concept, term Data Quality Dimensions is taken as standard to name characteristics of data and is widely observed in the literature [11, 54, 105]. The most widely used DQ attributes are listed in Table 2.1 [54].

Depending on purpose and structure of information, the literature defines a different number of DQ dimensions [105]. In some cases insignificant dimensions are omitted and others are split up into more attributes, because in various fields of actions some particular characteristics are more important than others, and the most attention is therefore paid to them. For instance, for military and government information security is a major feature, whereas for postal services complete and free-of-errors address database is more of a priority. Nevertheless, to some extent each dimension is important for databases of any purpose.

By means of these attributes, researchers can: (i) fully describe quality issues for a particular dataset, i.e., from which side (in which "dimension") there is a lack of quality in the information, (ii) set priorities for their improvement according to the purpose of data set, and (iii) predict data analysis errors.

Further, for easier prioritizing and handling DQ issues, these dimensions can be clustered in three *hyperdimensions* [55]:

Table 2.1: List of Data Quality Dimensions.

Attribute	Definition
Accessibility	the extent to which information is available or easily and quickly retrievable
Appropriate Amount of Information	the extent to which the volume of information is appropriate for the task at hand
Believability	the extent to which information is regarded as true and credible
Completeness	the extent to which information is not missing and is of sufficient breadth and depth for the task at hand
Concise Representation	the extent to which information is compactly represented
Consistent Representation	the extent to which information is presented in the same format
Ease of Manipulation	the extent to which information is easy to manipulate and apply for different tasks
Free-of-Error	the extent to which information is correct and reliable
Interpretability	the extent to which information is in appropriate languages, symbols, and units, and the definitions are clear
Objectivity	the extent to which information is unbiased, unprejudiced, and impartial
Relevancy	the extent to which information is applicable and helpful for the task at hand
Reputation	the extent to which information is highly regarded in terms of its source and content
Security	the extent to which access to information is restricted appropriately to maintain its security
Understandability	the extent to which information is easily comprehended
Value-Added	the extent to which information is beneficial and provides advantages from its use

- **Process**: characteristics related to a maintenance of data, such as Ease of Manipulation, Value-Added, and Security.
- **Data**: characteristics of the information itself, such as Believability, Completeness, Free of Error, Objectivity, and Relevancy.
- **User**: characteristics related to usage and interaction with users, such as Appropriate Amount of Information, Accessibility, Timeliness, and Understandability.

Thus, *high-quality data* can also be defined as data that satisfies all data quality dimensions. On the other hand, if there exist any defects and insufficiencies in data affecting one or more

Table 2.2: Address database.

Row	Name	Age	Gender	City	Country
1	Holmes, Sherlock	M	London	France	
2	Sawyer, Tom	12	M		USA
3		23	Female	Munich	Germany
4	Anna Karenina	285	F	Saint Petersburg	Russia

dimensions, data has poor quality. As proposed by [75], defects and insufficiencies of data are roughly divided into three groups: (i) *syntactical*, (ii) *semantic*, and (iii) *coverage*.

Consider a small database of personal information in Table 2.2 as an illustration of types of data deficiencies. Syntactical anomalies include errors in values and in format such as lexical errors, domain format errors, irregularities, such as non-uniform usage of abbreviations and units, and other mistakes. Line 1 violates the defined format for a data, i.e., instead of a tuple with the expected format *(Name, Age, Gender, City, Country)* it contains a tuple *(Name, Gender, City, Country)*. A domain format error is shown in the first column of line 4, i.e., the domain is specified as *(Surname, Name)*, but in this case the format is *(Name Surname)* without a comma, which is an anomaly. Additionally, in the field *"Gender"* of line 3 another designation is used to indicate feminine gender, which is clearly an irregularity.

Semantic anomalies concern non-comprehensive and redundant representation of data such as integrity constraint violations, contradictions, duplicates, invalid tuples. In Table 2.2 *"Name"* is a primary key, but since it is not present in line 3, it is an integrity constraint violation. Contradiction is illustrated in tuple 1 for values *"City"* and *"Country"*. Finally, the *"Age"* value in line 4 is too high for a person's age, which is an invalid value.

Coverage anomalies indicate at the lack of the information, e.g., missing values and tuples in the data. The examples of a missing values is in a line 2 for column *"City"* and in line 3 for column *"Name"* in Table 2.2.

Each anomaly disregards one or several DQ attributes in one way or another. For instance, data with coverage anomalies might not satisfy Completeness and Appropriate Amount of Information, whereas data with syntactical anomalies may fail to comply with Believability and Free-of-Error.

The above classification can be further extended, e.g., in accordance with the information system type: single-source and multi-source problems, as proposed in [87], depending on how many sources need to be integrated. In a multi-source case, besides all above-described issues, data quality issues include heterogeneous data models and schema designs, and overlapping, contradicting and inconsistent data.

Improving Data Quality

Before employing the information, it is checked thoroughly and all the inconsistencies, errors, and inaccuracies are removed in order to increase its quality. The whole process is called *Data*

Quality Audit and consists of the following steps [11]:

1. *Data Quality Assessment (DQA)* (or data profiling [85]) - the procedure of discovering data deficiencies and estimating DQ criteria. It is performed using various techniques, such as:

   - dictionaries of words to check whether attributes belong to their domains;
   - algorithms to detect anomalies in data;
   - algorithms to detect functional dependencies and their violations etc.

2. *Data Cleaning* (or *data scrubbing*, *data cleansing*), *Transformation and Enrichment* - the subsequent step of modifying the data as to remove identified on the previous step deficiencies and errors (i.e., to improve its quality and thus to fit its quality criteria). Data Cleaning (DC) procedures use:

   - dictionaries of words;
   - libraries of predefined cleaning functions and transformation rules;
   - machine learning techniques;
   - methods for merging duplicates, etc.

   Data enrichment procedures improve data using statistical models, logical inference, additional information and other approaches.

3. *Data Analysis* - further manipulating with the clean information to extract data patterns, rules, summary and significant points. Methods involved are (i) statistical evaluation, (ii) logical study, (iii) data mining algorithms, etc.

There exist different techniques and methods for DQA and DC. The choice of an approach strongly depends on the type and purpose of the data. However, obtaining acceptable DQ often requires a lot of time and resources and at times even manual correction to ensure cleanliness of data. In general, based on results of the Data Quality Audit, there exists four possible outcomes [55]:

- discard the data - information quality is very low, no use can be found for it;
- characterize the data, but no improvement is executed - if data quality is not high, but it is not feasible or costly to conduct an improvement;
- improve data quality - information quality is not good enough, but it is possible to apply methods for its improvement;
- improvement is not needed - the best but the rarest outcome: data quality is high.

This thesis proposes an approach for sensor reading quality assessment, which confirms whether measurements are precise enough, require data cleaning or smoothing, or they are unreliable and measurements should not used for diagnostics. We concentrate on quantitative data and its semantic anomalies, namely, invalid values. First, we formulate general statistical notions for analyzing quantitative data, then we provide an overview of different techniques for detection of DQ inconsistencies.

Description Logics

The current preliminary section is based on the information provided in [18] and [10].
Knowledge Representation (KR) is the field of Artificial Intelligence (AI) that focuses on methods for providing high-level descriptions of the world and management of complex kinds of information. Methods of knowledge representations are sometimes roughly divided into two categories: (i) logic-based formalisms, and (ii) non-logic-based representations. The latter are often used for more cognitive purposes, such as network structures or rule-based representations, whereas in logic-based approaches the representation language is usually a variation of first-order logic, therefore they are more general-purpose. *Description Logics (DL)* is an example of logic-based formalism for knowledge representation. It is a family of formal knowledge representation languages, specifically designed to represent and reason on structured knowledge. The domain of interest is modeled as a composition of *objects* and is structured into:

- *concepts* corresponding to classes and denoting sets of objects,
- *roles* corresponding to binary relationships and denoting relations on objects,

The knowledge is formulated using logical axioms called *assertions*. In general, Description Logic is characterized by the following:

- *A description Language* providing means for defining concepts and roles. It consists of two finite alphabets of *atomic concepts* and *atomic roles* (i.e., two sets of names) and a set of *constructors* allowing to build complex concepts and roles. A description language is identified by the set of constructs. Concept constructors are listed in Table 2.3, role constructors are listed in Table 2.4. Different sets of constructs form different kinds of DL with various expressiveness and complexity. For instance, concepts marked with the bold type in Table 2.3 constitute the basic language $\mathcal{AL}$ of the family of $\mathcal{AL}$ languages of DL. Adding further constructs gives further more expressive languages; for example, adding full negation to $\mathcal{AL}$ result in $\mathcal{ALC}$ language, adding inverse roles gives us $\mathcal{ALI}$.

  The formal semantics of DL is given in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of: (i) a nonempty set $\Delta^{\mathcal{I}}$, called the *interpretation domain*, and (ii) an interpretation function $\cdot^{\mathcal{I}}$, which maps each atomic concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each atomic role P to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to complex concepts and roles according to their syntactic structure (see Tables 2.3, 2.4).

- *TBox* - a mechanism to specify knowledge about concepts and roles, i.e., formulate assertions. It consists of logical axioms of the following form:

  - Inclusion assertion on concepts: $C_1 \sqsubseteq C_2$, $C_1 \equiv C_2$,
  - Inclusion assertions on roles: $R_1 \sqsubseteq R_2$,
  - Property assertions on atomic roles: (transitive P), (reflexive P), (symmetric P), (functional P), (domain P C), (range P C) and others.

- *ABox* - a mechanism to specify properties of objects, i.e, assertions on individual objects (c_i denotes individuals):
  - Membership assertions for concepts: $A(c)$,
  - Membership assertions for roles: $P(c_1, c_2)$.
  - Equality and distinctness assertions: $c_1 \approx c_2$, $c_1 \not\approx c_2$.

Table 2.3: Concept constructors, bold constructs constitute the language $\mathcal{AL}$

Construct	Syntax	Example	Semantics
bottom	$\bot$		$\emptyset$
top	$\top$		$\Delta^{\mathcal{I}}$
atomic concept	A	Human	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
atomic role	P	hasChild	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
atomic negation	$\neg A$	¬Male	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
conjunction	$C \sqcap D$	Human ⊓ Male	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	Male ⊔ Female	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
(full) negation	$\neg C$	¬(Human⊓Male)	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
unqualified existential restriction	$\exists R$	∃ hasChild	$\{o \mid \exists o'.\, (o, o') \in R^{\mathcal{I}}\}$
qualified existential restriction	$\exists R.C$	∃hasChild.Male	$\{o \mid \exists o'.\, (o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	∀hasChild.Male	$\{o \mid \forall o.\, (o, o') \in R^{\mathcal{I}} \rightarrow o' \in C^{\mathcal{I}}\}$
number restrictions	$(\leq k\, R)$	($\leq$ 2 hasChild)	$\{o \mid \#\{o' \mid (o, o') \in R^{\mathcal{I}}\} \leq k\}$
	$(\geq k\, R)$	($\geq$ 2 hasChild)	$\{o \mid \#\{o' \mid (o, o') \in R^{\mathcal{I}}\} \geq k\}$
qualified number restrictions	$(\leq k\, R.C)$	($\leq$ 2 hasChild.Male)	$\{o \mid \#\{o' \mid (o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \leq k\}$
	$(\geq k\, R.C)$	($\geq$ 2 hasChild.Male)	$\{o \mid \#\{o' \mid (o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \geq k\}$

An ontology is a representation scheme that describes a formal conceptualization of a domain of interest. The specification of an ontology comprises two different levels:

- *The intensional level* specifies a set of conceptual elements and of constraints/axioms describing the conceptual structures of the domain,

Table 2.4: Role constructors

Construct	Syntax	Example	Semantics
atomic role	P	hasChild	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
role negation	$\neg P$	$\neg$hasChild	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \backslash P^{\mathcal{I}}$
inverse role	R^-	hasChild$^-$	$\{(o, o') \mid (o', o) \in R^{\mathcal{I}}\}$

- *The extensional level* specifies a set of instances of the conceptual elements described at the intensional level.

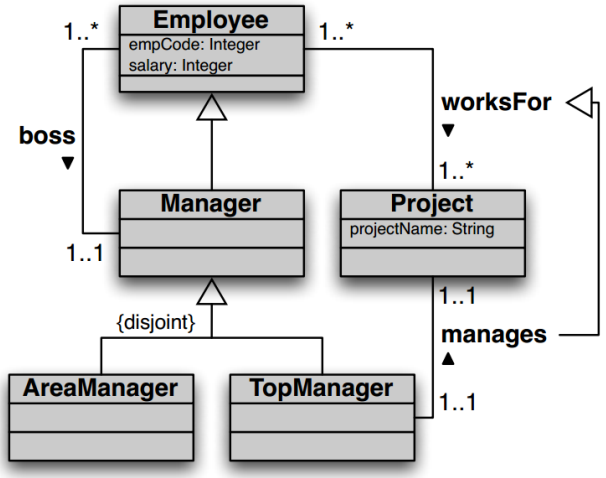
DLs provide a foundation and logical formalisms for ontology languages, as they are logics specifically designed to represent knowledge and reason upon it. The intensional level of the ontology is expressed with the TBox, whereas the extensional level with the ABox. Ontology is said to be *satisfiable* if there exists a non-empty model $\mathcal{I}$ satisfying its TBox and ABox. The W3C Standard Web Ontology Language (OWL) is defined as different versions of DL with various description languages. OWL comes in the following variants [70, 73]:

- OWL 1 Lite - description logic $\mathcal{SHIF}(D)$, where each letter denotes the following construct capabilities:
  - $\mathcal{S}$ - description logic $\mathcal{ALC}$ extended with transitive roles,
  - $\mathcal{H}$ - role hierarchy, i.e., role inclusion assertions,
  - $\mathcal{I}$ - inverse roles,
  - $\mathcal{F}$ - functionality of roles,
  - (D) - data types, which are necessary in any practical knowledge representation language.
- OWL 1 DL - description logic $\mathcal{SHOIN}(D)$, where:
  - $\mathcal{O}$ - usage of nominals, which means the possibility of using individuals in the TBox (i.e., the intensional part of the ontology),
  - $\mathcal{N}$ - unqualified number restrictions.
- OWL 2 DL is the new version of OWL. It is the description logic $\mathcal{SROIQ}(D)$, which adds the following functionality to concepts and role construction:
  - $\mathcal{Q}$ - qualified number restrictions,
  - $\mathcal{R}$ - regular role hierarchies, where role chaining might be used in the left-hand side of role inclusion assertions, with suitable acyclic conditions.

  Additionally, OWL 2 DL defines three profiles corresponding to its sub-languages: OWL 2 QL, OWL 2 EL, OWL 2 RL. Restrictions of each profile guarantee better computational properties in comparison with OWL 2 DL, and each profile targeted for a specific use.

Profile OWL 2 QL is of particular interest for us, it is based on the *DL-Lite* family of DL [73]. *DL-Lite*$_{\mathcal{A}}$, an expressive member of *DL-Lite* family, which has the following features:

- distinction between objects and values: values are connected with objects through attributes rather than roles;
- concept constructors: atomic concept, unqualified existential restriction, top, domain of an attribute, full negation to express disjointness;
- role constructors: atomic roles, inverse roles, role negation to express disjointness;
- attribute constructors: atomic attribute, negation to express disjointness;
- assertions: concept, role and hierarchy inclusions, disjointness assertions, functional and identification property assertions.



$$
\begin{aligned}
\text{Manager} &\sqsubseteq \text{Employee} \\
\text{AreaManager} &\sqsubseteq \text{Manager} \\
\text{TopManager} &\sqsubseteq \text{Manager} \\
\text{AreaManager} &\sqsubseteq \neg\text{TopManager} \\
\text{Employee} &\sqsubseteq \delta(\text{empCode}) \\
\delta(\text{empCode}) &\sqsubseteq \text{Employee} \\
\rho(\text{empCode}) &\sqsubseteq \texttt{xsd::int} \\
(funct\ \text{empCode}) & \\
(id\ \text{Employee empCode}) & \\
\exists\text{worksFor} &\sqsubseteq \text{Employee} \\
\exists\text{worksFor}^{-} &\sqsubseteq \text{Project} \\
\text{manages} &\sqsubseteq \text{worksFor} \\
&\ldots
\end{aligned}
$$

Figure 2.2: Example of an *DL-Lite*$_{\mathcal{A}}$ ontology [18].

Figure 2.2 illustrates an example of an ontology expressed in *DL-Lite*$_{\mathcal{A}}$ as an Unified Modeling Language (UML) diagram and shows a part of its TBox assertions [18].

Other members of *DL-Lite* family are sub-languages of *DL-Lite*$_{\mathcal{A}}$: *DL-Lite*$_{\mathcal{R}}$, which allows role hierarchy and does not allow functional assertions, and *DL-Lite*$_{\mathcal{F}}$, which allows functional assertions and does not allow role inclusions. In both *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$ data values are not considered, and hence attributes are dropped.

In the syntax of *DL-Lite* we can trivially simulate concept conjunction on the right hand side of inclusion assertions:

$$
\begin{aligned}
B &\sqsubseteq C_1 \\
B &\sqsubseteq C_2
\end{aligned}
\quad \Leftrightarrow \quad B \sqsubseteq C_1 \sqcap C_2
$$

Similarly, qualified existential restrictions can be simulated on the right hand side of inclusion assertions using role inclusion and introduction of a new role:

$$\begin{array}{ll} Q \sqsubseteq P & \\ A_1 \sqsubseteq \exists Q & \Leftrightarrow \quad A_1 \sqsubseteq \exists P.A_2 \\ \exists Q^- \sqsubseteq A_2 & \end{array}$$

In complexity of *DL-Lite* ontology satisfiability is PTIME in the size of TBox and AC^0 in the size of ABox, and query answering for conjunctive queries is PTIME in the size of TBox, AC^0 in the size of ABox and NP-COMPLETE in the size of query [18].

The *DL-Lite* family is a maximally expressive ontology languages with good computational properties, it has the same data complexity for query answering as relational databases and thereby it provides new foundations for the *Ontology-Based Data Access* - approach to query answering over relational databases using domain knowledge provided by an ontology. Therefore, ontologies expressed in the OWL 2 QL profile are very well suited to underlie Ontology-Based Data Management Systems.

Statistics

Statistics is a branch of mathematics applied to analysis and interpretation of data. It comprises a load of approaches for describing, modeling and predicting data, and has close connections with machine learning and data mining. It is necessary to apply simple statistical analysis in case of quantitative data in order to summarize data and to detect possible inconsistencies. For example, setting a simple threshold of human life span for values in a field *"Age"* in Table 2.2 would help us detect the anomaly in line 4. Similarly, we can doubt about good quality of data if we observe winter temperatures in Alaska and get a positive number after calculating its average. In this subsection we introduce the fundamental statistical notions used further in this work.

Descriptive Statistics

Statistical methods that summarize and describe characteristics of a dataset are called *descriptive statistics* [46]. Consider a dataset $\{x_n\} = \{x_1, x_2, \cdots, x_n\}$, where $x_i \in \mathrm{R}$. The main parameters describing $\{x_n\}$ are:

- the *mean* or *expected value* of $\{x_n\}$ is an average of all sampled values and is a basic measure of the central tendency of $\{x_n\}$:

$$\overline{x} = \frac{1}{n} \sum_{1}^{n} x_i \tag{2.1}$$

- the *standard deviation* shows how much on average values in $\{x_n\}$ deviate from their mean, i.e.,

$$\sigma = \sqrt{\frac{1}{n} \sum_{1}^{n} (x_i - \overline{x})^2} \tag{2.2}$$

Additionally, the dataset $\{x_n\}$ is characterized by its *minimum* and *maximum* points, and *quartiles* - three points in a data, which divide the dataset into four equal groups: (i) 25th percentile, which divides 25% of data from other 75%; (ii) 50th percentile that is a *median* $\tilde{x}$ of a dataset: exactly half of data have lower value then $\tilde{x}$ and other 50% have greater value; (iii) 75th percentile that is a middle between the median and the maximum value of a dataset.

There exist various estimators of these statistical parameters, which are robust to outliers and very helpful for analyzing data in the presence of outliers and noise. Typical robust methods for estimating the central tendency include:

- Median of a dataset, i.e., 50th percentile.
- *Trimmed* (or *truncated*) mean - calculation of a mean value discarding $k\%$ (usually from 5% to 25%) of data from each end. One of the examples of the truncated mean is *Interquartile Mean (IQM)* that trims 25% of data from each end.
- *Trimean* - weighted average of three quartiles of data. For Q_1 denoted as 25th percentile, Q_2 denoted as median, and Q_3 for 75th percentile:

$$T = \frac{Q_1 + 2Q_2 + Q_3}{4} \tag{2.3}$$

Methods for robust estimation of variance include:

- *Median Absolute Deviation (MAD)* - calculated as median of absolute deviations from data median in dataset. For $\{x_n\}$:

$$MAD = median\{(|x_i - \tilde{x}|)\}, \tag{2.4}$$

- *Average (or mean) Absolute Deviation* is similar to MAD - a mean of absolute deviations of data from its mean. Generally, any measure of central tendency could be chosen for calculation of absolute deviations:

$$AAD = \frac{1}{n}\sum_{1}^{n} |x_i - M(\{x_n\})|, \tag{2.5}$$

  where $M(\{x_n\})$ represents mean, trimmed mean, median, or any other measure.

- *Interquartile Range (IQR)* - difference between 75th and 25th percentiles of a dataset $\{x_n\}$, i.e.

$$IQR = Q_3 - Q_1 \tag{2.6}$$

We listed only several well-known and widely used estimators of statistical parameters, which we are going to be used and mentioned further in the thesis. Practically there exist many more different estimators classified in broad classes [43], but their usage is out of scope of this work.

Additionally, sometimes in order to detect long-term trends or changes in data, statisticians analyze and calculate basic statistical parameters of different subsets of data set, using a *moving* (or *rolling, running*) *window*. Given a dataset $\{x_n\} = \{x_1, x_2, \cdots, x_n\}$ and natural number $k < n$, we analyze each subset generated by moving a k-sized window along the full dataset, i.e., $\{x_1, x_2, \cdots, x_k\}$, $\{x_2, x_3, \cdots, x_{k+1}\}$ and so on. In total there are $n - k + 1$ such subsets of $\{x_n\}$.

For two datasets $\{x_n\} = \{x_1, x_2, \cdots, x_n\}$ and $\{y_n\} = \{y_1, y_2, \cdots, y_n\}$ we use the measures of *covariance* and *correlation* in order to analyze how they depend on each other. For $x = \{x_n\}$ and $y = \{y_n\}$ covariance is calculated as:

$$cov(x, y) = \frac{1}{n} \sum_{1}^{n} (x_i - \overline{x})(y_i - \overline{y}) \tag{2.7}$$

Covariance is zero if there are no dependence between x and y and non-zero if they have any dependence between them, the bigger absolute value of covariance shows greater interdependence. Normalized covariance is called correlation and it changes between -1 and 1, where the value 1 indicates at perfect direct linear dependence between x and y (i.e., y increases if x increases and decreases when x decreases), -1 indicates at perfect inverse relationship (*anticorrelation*). The closer the correlation value is to 1 or -1, the stronger the dependence between datasets, and similarly, correlation value closer to 0 indicates at weaker relationship between the datasets. There exist multiple formulas to measure correlation of values examined in [66], but the most familiar and widely-used way to measure correlation is the *Pearson's correlation coefficient*:

$$cor(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}, \tag{2.8}$$

where σ_x and σ_y denote standard deviations of $\{x_n\}$ and $\{y_n\}$ correspondingly. This formula is used further in this thesis.

In the multivariate case, i.e., for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_k)^T$, where each component x_i is a dataset $\{x_i\} = \{x_{i,1}, x_{i,2}, \cdots, x_{i,n}\}$, the *covariance matrix* is a $k \times k$ matrix, whose (i, j) element is a covariance value between i^{th} and j^{th} elements of $\mathbf{x}$. Similarly, *correlation matrix* is a $k \times k$ matrix, whose (i, j) element is a correlation value between x_i and x_j.

Time Series

Sensors observe temperature, pressure and any other physical quantities with a certain frequency; their measurements form a sequence of data points collected in the course of time. Such sequences are called *time series data*. There are distinguished into (i) *discrete time series*, when observations are made at fixed time intervals and form a discrete set, and (ii) *continuous* time series, when observations are recorded continuously in time [16]. Time series can be either *stationary* in case parameters such as mean and standard deviation do not change over time and do not follow any trends, and *non-stationary* otherwise.

Time series analysis is a branch of statistics that provides methods for analyzing time series data in order to derive meaningful characteristics of data. Specifically, there are several common

approaches for modeling time series data for analysis or forecasting. Models represent observations as a sum of several independent components, such as trend or season (cyclic reiterations). One of the most common time series model is the ARIMA model [16]. Generally, ARIMA models combine autoregression, namely fitting of data points to linear combination of previous data, and moving averages to estimate the next value.

All statistical parameters defined in the previous section apply and can be calculated for time series data as well. Additionally, we define *autocorrelation* for a single time series dataset $\{x_n\} = \{x_1, x_2, \cdots, x_n\}$ as a correlation of x with time-shifted version of itself:

$$acf(\tau) = \frac{1}{\sigma^2} \sum_{1}^{n} (x_i - \overline{x})(x_{i+\tau} - \overline{x}), \tag{2.9}$$

where τ is a time lag for which we shift initial time-series to calculate the correlation between them. Thus, autocorrelation is expressed as a function of time lag, and it is equal to 1 for zero time lag. Additionally, autocorrelation is the so called *even function*, i.e., $acf(\tau) = acf(-\tau)$.

2.2 Approaches to Data Cleaning in Quantitative Data

This section provides an overview of techniques to detect and clean data quality inconsistencies in quantitative data and in time series in particularly. First, we observe a related work of general-purpose data cleaning methods as well as designed specially for time series and sensor data cleaning. The rest of the section is dedicated to techniques and methods of detecting and cleaning DQ insufficiencies, in particular, we concentrate on quantitative and time series data.

Overview of Existing Data Cleaning Methods

Data cleaning is a vivid field and there have been developed a lot of various frameworks and methods for cleaning and transforming data from different fields of application. As data mining techniques becoming more widespread for growing amounts of data, more and more attention is drawn to the quality of data and to the methods of improving its quality. We made a survey on some existing general-purpose methods for data preprocessing and cleaning. This section discusses approaches to DC and provides an overview of existing general-purpose DC systems. First, we list the most remarkable frameworks designed in the field of data cleaning, and then concentrate on works on assessing and improving data quality of sensor readings.

General-purpose Data Cleaning Frameworks

In Table 2.5 we provide a brief overview of existing DQ frameworks designed for DQA and DC of data of different types and purposes. Most of the works in the field of DC, as well as DC systems presented in Table 2.5, has been done for textual and address databases.

Data Cleaning for Sensor Readings

However, the area of sensor networks and sensor data processing receives a lot of attention recently and there exists a tremendous amount of work in this area. In this section we focus on

Table 2.5: Existing data cleaning frameworks and tools

Name of the system	Year	Description, method	Features
AJAX [37]	2000	Transforms existing data into a target schema, uses declarative SQL-based language	Data transformations: mapping, view, matching, clustering, merging.
ARKTOS [103, 104]	2001	Models Extracting-Transformation-Load (ETL) process with integrated DC part to create a data warehouse uses two declarative languages, based on XML and SQL.	Semantic, coverage anomalies and domain mismatch errors.
ERACER [69]	2010	Uses probabilistic relational model, convolution and regression models for DC in Database Management Systems (DBMSs)	Semantic, syntactical, coverage errors, SQL interface
FraQL [91]	2000	Declarative language for ORDBs - extension of SQL, user-defined functions	Schema transformations, data transformations, missing values, smoothing noisy data.
IntelliClean [65]	2000	Targeted mainly for textual databases, uses knowledge-based rules and algorithm Rete	Transforming and updating, specified by conditions in rules, duplicate elimination.
MapReduce [28]	2004	A data processing model based on two operations Map and Reduce, widely used.	Duplicate elimination, syntactical anomalies, data enrichment.
NADEEF [27]	2013	An extensible and generalized DC system for DBMSs, uses user-defined quality rules.	Syntactical and semantic anomalies.
Potter's Wheel [88]	2001	Interactive data cleansing system with spreadsheet-like interface	Data transformations, error detection, executed interactively with the user specifications.
XClean [106]	2007	Predefined cleaning operators and their combinations for cleaning XML data	Syntactical anomalies, duplicate elimination.

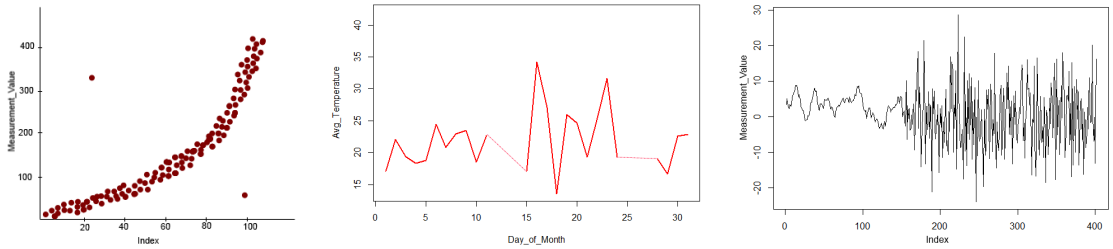
the methods and systems developed particularly for sensor data quality assessment and improvement.

One of the commonly used method is statistical inference. For instance, [34] presents a Bayes-based statistical approach for online cleaning and reduction of uncertainty of sensor measurements. In order to obtain more accurate estimates of sensor readings it combines prior knowledge of the true sensor readings, such as information about the distribution of the true measurements, and an error model - noise and inaccuracy characteristics of each sensors, which is basically a distribution of the noise that affects measurements. A similar idea for cleaning corrupted values in sensor data is presented in [62] - their probabilistic approach also consists of a clean model for true measurements, error model for noise, and corruption model for measured values corrupted by noise. Many more statistical techniques are used to model sensor measurements in order to clean the noise and to restore missing values: regression models [84], and Kalman filtering [95] just to mention a few. In [30] a dynamic Bayesian network model is introduced for analyzing sensor measurements and distinguishing true readings from sensor failures. Its application domain is atmospheric data and air temperature sensors, and the conditional Gaussian Bayesian network models the interaction between the sensor and the air temperature process.

ERACER [69] (also mentioned above in Table 2.5) is a statistical framework designed for relational databases, and cleaning sensor measurements is one of the use cases for this framework. It models the database as relational dependency network and uses convolution and regression models for data cleaning. The framework SwissQM [74] is based on a specialized virtual machine that builds a sensor network with sensor and gateway nodes and allows to query and process sensor data. Although it is designed for data acquisition in sensor networks, it performs data processing and cleaning steps. Other examples are [51] that presents integrated model for data cleaning on signal processing systems of sensor networks, and the ESP (Extensible Sensor stream Processing) framework [50] that builds a sensor data cleaning infrastructure over relational data streams and provides a high-level declarative language for users to specify cleaning operations.

Among all above-listed frameworks only ERACER uses the similar idea as presented in this thesis: using available domain information and interconnections between entities. It captures attribute dependencies with graphical models and combines graph theory and statistics to reason with joint distributions. However, these graph-based models are not expressive enough to capture all types of potential domain restrictions in a compact and concise way in our use case, therefore we turned to the usage of semantic ontologies.

Additionally, it is important to mention the ICM-Wind system for the condition monitoring of wind turbines [58], which presented in the recent work and accepted to the 29th Symposium On Applied Computing. It applies linear discriminant analysis to operational data and encodes the domain knowledge and restrictions in an expressive logical formalism: as OWL 2 ontology and SPIN rules. Although the ICM-Wind system is not designed for data quality assessment and improvement, but rather for fault diagnosis and monitoring, the overall approach is similar to the method presented in this thesis.



a) b) c)

Figure 2.3: Types of data anomalies in time series data: a) point outliers, b) missing data in monthly temperature observations, c) observational data affected by random fluctuations.

Data Cleaning Techniques

We conducted some research on existing techniques for identifying anomalies in data. The main focus of the work is quantitative and time series data and by "anomalies" in time series data we understand one of the following:

- Erroneous values: *inliers*, falling within the expected range; and *outliers*, falling outside of the expected range [102]. In the following, we write *outlier* meaning both inliers and outliers, i.e., considering erroneous data points, distant and/or different from other observations. In time series analysis such erroneous values are also called *point outlier* or *additive outlier*. Figure 2.3a shows observations with outlying values. Presence of outliers in data influences analysis, in particular, distorts the main statistical parameters and autocorrelation values.

- Missing data: absence of data values during some period. Figure 2.3b shows an example: observing average daily temperatures during two periods data is completely missing, no measurements are provided or they were erased. That affects analysis of data, its comprehensive overview, and conclusions that could be drawn from the data.

- Noisy data: data with *statistical noise* and *oscillations*, i.e., repeating random fluctuations in value around the mean over time. Figure 2.3c shows a situation when observations are affected by oscillations and noise: it is hard to estimate a real measured value due to the fluctuations and conduct a comprehensive analysis of this data.

Thus, we want to solve the problem of coverage anomalies and invalid tuples in sensor measurement data in order to increase the measures of Free-of-Error, Believability and Completeness of information.

More formally, given that $\mathcal{D}_R$ is our real dataset of available data, and $\mathcal{D}_I$ is a dataset that contains ideal error-free data, we say that *data imperfections* (or data quality deficiencies) are differences between the datasets $\mathcal{D}_R$ and $\mathcal{D}_I$ [83]. Using this definition, we can define data deficiencies for quantitative data more precisely. For $x \in \mathcal{D}_I$ - the ideal data value, $\mathcal{R}x \in \mathcal{D}_R$ - the corresponding real data value (if any), an arbitrary small positive quantity $\varepsilon > 0$, and a

scalar-valued distance function $\rho(x, y)$ defined on $\mathcal{D}_I \times \mathcal{D}_R$ we distinguish the following types of DQ deficiencies [83]:

1. *observation error*: $x \in \mathcal{D}_I$, $\mathcal{R}x \in \mathcal{D}_R$, $\rho(x, \mathcal{R}x) \leq \varepsilon$,
2. *gross error*: $x \in \mathcal{D}_I$, $\mathcal{R}x \in \mathcal{D}_R$, $\rho(x, \mathcal{R}x) \gg \varepsilon$,
3. *simple missing data*: $x \in \mathcal{D}_I$, $\mathcal{R}x \notin \mathcal{D}_R$,
4. *coded missing data*: $x \in \mathcal{D}_I$, $\mathcal{R}x = m* \in \mathcal{D}_R$, where $m*$ - a special defined symbol, e.g. 'NA', '?', 'NULL',
5. *disguised missing data*: $\mathcal{R}x = y$, y is an arbitrary value.

In this categorization, ε is a small value to distinguish between an observation error and a gross error. In practice, observation errors are almost always present in measurements, depending on accuracy and precision qualities of a measuring device. Gross error values are simply erroneous values (or outliers) defined above. Whereas imperfections of types 3 and 4 above explicitly report on missing data, type 5 implicitly shows data losses. Specifically, noise is an example of DQ deficiency of type 5.

In the Appendix we give a detailed instances of above-listed data imperfections detected in real-world data from the use case.

Outlier detection

There exists a broad variety of algorithms for identifying erroneous and out-of-range values in a dataset, from statistical to machine learning models. A choice of the method highly depends on the data character and purpose. For instance, although all methods from the overview in this chapter can be applied to quantitative data, but not all of them are suitable for time series data. For time series we need to take into account that outliers need not be extreme with respect to the overall range of data variation, but have to be considered as an outlier with respect to local values [83]. For example, observing the weather in Europe during the year, value 25^oC is the normal value and not an outlier, unless this value is not dated in December.

We summarize below well-known types of approaches for outlier detection in data [13, 60] and give specific examples of their application to time series data.

Classification algorithms identify outliers as misclassified values, i.e., objects that after a classification do not belong to any group or cluster. The most remarkable examples of methods in this group are Support Vector Machines (SVM) [53], Replicator Neural Networks [44], and Self-Organizing Maps (SOM) [76]. Classification algorithms require a training set and their complexity depends on a chosen classification, for example, nonlinear SVM generally requires $O(n^3)$ time in the worst case for training with n as number of observations.

Depth-based approaches identify outliers as points located at the border of k-dimensional data space, whereas and normal data lies in the center of data space. The main idea is to calculate layers of depth in data and distinguish outlying values as point with $depth \leq k$. There exist a number of efficient algorithms for $k = 2, 3$, but for $k \geq 4$ algorithms become inefficient [15]. Well-known examples of depth-based algorithms are ISODEPTH [89] and FDC [52]. Lower bound complexity for these algorithms is $\Omega(n^{k/2})$ for n objects in k-dimensional space.

Deviation-based approaches define outliers as points, after which removal variance of the dataset is minimized. On of the methods in this category is computing exception sets using smoothing factors [9], this method has a linear complexity $O(n)$ in case of a good choice of parameter function, such that it computes necessary parameter values in constant time. A pseudo-deviant algorithm for univariate and multivariate time series data is presented in [77], it has complexity $O(n^2k)$ for n observations and parameter k in the univariate case.

Distance-based approaches define outliers as points that are far apart from their neighborhood. Methods of this category are efficient also for large k-dimensional datasets with large values of k, e.g., $k \geq 5$. Examples in this category are methods based on k-Nearest Neighbors (kNN) algorithm (e.g., [6], RBRP algorithm in [39]), index-based algorithms [59], nested-loop algorithms (e.g., [12]). Worst case complexity for these algorithms is $O(kn^2)$, but [59] presents an algorithm with complexity linear in n and exponential to k. A moving window distance-based algorithm STORM for data streams and time series is presented in [5] with time complexity $O(\Delta \log n)$, where Δ is a cost of computing distance between two objects.

Density-based approaches identify normal data as points that have similar density to their neighbors, and the density around an outlier is lower than the density around its neighbors. Approaches in this category differ mostly in methods of estimating density. Examples od density-based approaches are Local Outlier Factor (LOF) [15] and its variants (e.g., (COF) [96]), cluster analysis algorithms (e.g., DBSCAN [35], OPTICS [7]). These algorithms have $O(n \log n)$ on average for n objects, but in the worst case, e.g., in case of high-dimensionality, the complexity is $O(n^2)$. Application of DBSCAN to time series data is presented in [57].

High-dimensional approaches are methods for finding full-dimensional outliers in high-dimensional data. They include angle-based outlier degree (ABOD) [61], grid-based subspace outlier detection [2]. Complexity for ABOD is $O(n^3)$ and for its faster modifications $O(n^2 + nk^2)$ for n objects and parameter k.

Outlier score in statistics define outliers as data points, that variate the most from other values. In a data sequence $\{x_n\}$ they are determined according to the rule $|x_i - x_0| > t\varsigma$, where $i \in \{\overline{1, n}\}$, x_0 is a reference value, ς is a measure of variation and t is a chosen threshold. The value $z_i = (x_i - x_0)/t\varsigma$ is called *outlying score*. For the multivariate case extension of the outlying score is called *Mahalanobis distance* [83]: for $\mathbf{x}$, its covariance matrix S, and vector of mean values $\overline{x} = (\overline{x_1}, \ldots, \overline{x_n})$ the Mahalanobis distance is $d(x_k, \overline{x}) = (x_k - x)^T S^{-1}(x_k - x)$.

Outlier detection rules are very frequently applied to time series data and differ in parameters choice. Examples are:

- 3σ *edit rule* with a parameter set $x_0 = \overline{x}$ - mean of $\{x_k\}$, $\varsigma = \sigma$ - standard deviation of $\{x_k\}$, $t = 3$,
- *Hampel identifier* with $x_0 = \tilde{x}$ - median, $\varsigma = S$ - MAD of $\{x_k\}$,
- *boxplot outlier rule* with $x_0 = \tilde{x}$ - median, $\varsigma = Q$ - IQR of $\{x_k\}$.

In the last two examples the parameter t can be varied, higher values of t make the filter more lenient, whereas low values of t make it more sensitive. In [83] it is shown that the 3σ rule is less effective in comparison with Hampel and boxplot outlier detection methods. A moving window usage of Hampel identifier is called *Hampel filter* [82].

Statistical tests and models fit a distribution or a model to quantitative data and indentify outliers as strongly deviating data values, i.e., points with a low probability to be generated by the distribution or a model. There exist various tests for checking the hypothesis whether data follows a certain or mixture distribution, both for univariate and multivariate data [93]. The well-known examples are Grubb's test for univariate normally distributed data, Dixon's Q test. Most of tests however are defined for univariate distributions.

The most frequently used models for prediction of values are (i) regression and least square model, (ii) ARIMA model family for time series data, and others. ARIMA modeling is the most common approach for fitting, analyzing and predicting time series.

Oscillation and noise detection

In time series data oscillations are usually characterized as periods in data with high deviation from the mean value in comparison to other observations. In this case oscillations have to be smoothed. Random uncorrelated fluctuations in data is called noise and corresponds to disguised missing data. Table 2.6 describes several well-known methods used to detect noise and oscillations in quantitative data and Table 2.7 lists several approaches to smoothing oscillating data.

Missing data

We defined three cases of missing data: simple, coded and disguised missing data. They differ only in representation of the fact that we lack a portion of data, but in all three cases we need either to omit missing data intervals or to forecast or approximate missing values. Missing data significantly complicates further analysis, as some methods, such as moving window techniques, assume that data is regularly sampled and there are no missing values [83]. Table 2.8 summarizes several known methods used for prediction.

Table 2.6: Oscillation and noise detection

Approach	Description
Absolute Error	In industrial applications analysis of absolute value is used to detect oscillations, i.e., analyzing regularity of large intervals of absolute error and magnitude of integrated absolute error [42].
Autocorrelation function	Analysis of autocorrelation coefficients plays a key role in statistics and may provide useful information for process monitoring. For instance, noise has zero autocorrelation values and autocorrelation function of oscillatory signal is also oscillatory. There exist a number of works that use autocorrelation for oscillation detection, such as [71] and [98].
Discrete Cosine Transform [67]	Isolates different sequences of data points as sum of cosine functions oscillating at different frequencies, i.e., after transform application components with different frequencies are distributed separately in the transformed domain.
Tests for *heteroscedasticity*	Used to detect subsets of data which have significantly different variance than the other values. The most frequently used is Cochran's C test [22].

Performance metrics

To validate out approach we define a number of performance metrics widely used in various domains in order to evaluate the performance of classification techniques, in our case classifying normal and anomalous data points. Thus, classified data points can be classified as [79]:

- *true positive* (TP) - an anomaly was classified correctly as an anomaly,
- *false positive* (FP, also known as *false alarm* or *type I error*) - a correct value was erroneously classified as an anomaly,
- *true negative* (TN) - a correct value was classified as correct value,
- *false negative* (FN, also known as *type II error*) - an anomaly was wrongly classified as correct value.

We define the following performance metrics based on the above-introduced notions [57,79]:

Table 2.7: Smoothing data

Approach	Description
Moving (or rolling, running) average/mean	Creates series of averages/means for different subsets of the full dataset, shifting forward a k-sized window. There exist simple methods and their modifications with using weights. Used to smooth fluctuations and also to highlight long-term trends.
Exponential moving average [17]	Variation of weighted moving average algorithm with weights that change depending on previous observations. They change in geometric progression, that is a discrete version of exponential function, therefore the method is called exponential.
Exponential smoothing [17]	Techniques with the same idea of calculating weights as for exponential moving average: double (or second-order) exponential smoothing, often referred to as Holt-Winters smoothing, and triple exponential smoothing. These methods are more effective for data with trends or seasonal changes.
Fixed-interval smoothing [33]	Application of Kalman filter to data for smoothing. Kalman filter uses system model and observation data to produce estimated values and has numerous applications in technology. Several algorithms based on Kalman filter are used for smoothing data.

1. *True Positive Rate* or *Recall* - is a measure of completeness for anomaly detection, i.e., amount of identified anomalous points among all anomalies:

$$TPRate = \frac{TP}{TP + FN} \tag{2.10}$$

2. *True Negative Rate* or *Specificity* - a measure of completeness for correct values, expresses amount of correct values identified as correct among all correct values:

$$TNRate = \frac{TN}{TN + FP} \tag{2.11}$$

3. *False Positive Rate* - rate of Type I errors:

$$FPRate = \frac{FP}{TP + FP} \tag{2.12}$$

Table 2.8: Predicting data

Approach	Description
Modeling	Usage of models based on existing values to predict missing values. Regression models [83], time-series model such as ARIMA [16] are frequently used to predict missing values in dataset.
Simple imputation methods [83]	One of the most well-known technique is mean imputation: missing values are replaced with a mean of an appropriately defined group of non-missing values. However, this method leads to wrong estimations of variance and other parameters, if imputed values are treated as real. Properties and drawbacks of the method with illustrative example are discussed in more detail in [83]. Another example is hot-deck imputation, where missing values are replaced with existing values from a dataset, different procedures determine, which values exactly are used.
Smoothing methods	Methods presented in Table 2.7, such as exponential smoothing methods or fixed-interval smoothing are widely used for prediction of values as well [16].

4. *False Negative Rate* - rate of Type II errors:

$$FNRate = \frac{FN}{TN + FN} \tag{2.13}$$

5. *Accuracy* is the amount of true results in the whole dataset:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

6. *Error Rate* is the amount of erroneous results in the whole dataset:

$$ER = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.15}$$

7. *Precision* is a measure of exactness, i.e. amount of anomalies identified correctly:

$$P = \frac{TP}{TP + FP} \tag{2.16}$$

8. *F-measure* measures the balance between precision and recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R}, \tag{2.17}$$

where the parameter β is chosen depending on which characteristic is more important for validating the algorithm. If precision and recall are equally important for algorithm validation, β is chosen to be 1 and then the F-measure becomes a harmonic mean between precision and recall:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{2.18}$$

Some of the above-introduced metrics are related to each other:

$$ER = 1 - A \tag{2.19}$$

$$P = 1 - FPRate \tag{2.20}$$

These metrics are going to be used further in Chapter 4 for evaluating the performance of our approach.

In this chapter we discussed fundamental notions and existing techniques for DQA and DC. Next, in Chapter 3 we describe the use case, specify data types of sensor readings and purposes of its usage, and devise appropriate techniques for DQ insufficiencies detection and cleaning in the use case data.

CHAPTER 3

Methodology

3.1 Application Domain

The Optique project

Several European universities and two large industrial companies launched the European project "Optique" in 2012, aimed at providing scalable end-user access to Big Data [78]. The main goals of the project are given as follows:

- to provide a semantic end-to-end connection between users and data sources;
- to enable rapid formulation of intuitive queries using vocabularies familiar for the user and captured using an ontology and declarative mappings;
- to integrate data spread across multiple heterogeneous data sources, including streams;
- to exploit massive parallelism for scalability far beyond traditional RDBMSs and thus reducing the turnaround time for information requests to minutes rather than days.

Two European companies Siemens and Statoil support the project with detailed use cases and corresponding data sets. The Statoil use case is concerned with geospatial data analysis, whereas Siemens provides a use case from the Energy domain and concerns condition monitoring of industrial gas and steam turbines. The data sets include measurements and observations from sensing devices and control panels on the turbines.

The current work is a part of the Siemens use case. Research, conducted in the course of this work is based on the use case information, and data cleaning methods apply to the use case data. Further in this chapter we describe the field of work, namely the structural design of Siemens gas turbines, the information path from sensing devices mounted at the appliance to the database, and tables and data types which are stored in the database. The Siemens use case of the Optique project therefore serves as a guiding example for the design and evaluation of the methodology devised in this thesis.

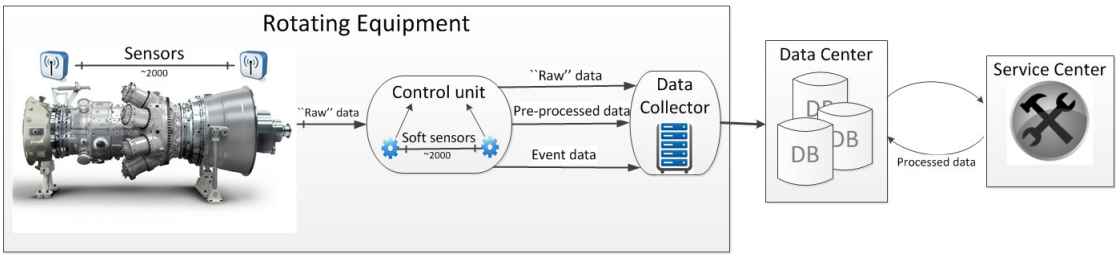


Figure 3.1: Appliance structure and data flow.

Appliance Structure and Sensor Settings

Siemens Energy Services maintain thousands of devices related to power generation, including gas and steam turbines, compressors, and generators - called *appliances*, *units* or *rotating equipment* from here on. Operational support is provided through a global network of more than 50 service centers. These service centers are in turn linked to a common database center, where the appliance data is stored in several thousand databases. In the following we describe the structure and monitoring facilities of an appliance. Next, we detail the data processing procedure and show sample database tables. Figure 3.1 presents a high-level overview of the components addressed in this Chapter and depicts schematically an appliance and its data flow.

Each gas turbine consists of three main parts [64] serving the following functions (see also Figure 3.2[1]):

1. *compressor*, that accelerates a gas and increases its pressure by decreasing gas volume;
2. *combustor*, that heats a gas with a constant pressure, using gas or liquid fuel;
3. *power turbine* and *generator*, which extract power from a hot gas flow.

Each stage of gas transformation and unit functioning is monitored by measuring devices or *sensors*, mounted at every compartment of the appliance. They conduct measurements of physical quantities such as temperature, pressure and speed of the gas at a rate between 1 Hz and 1000 Hz, depending on its purpose and characteristics. Additionally, there are sensors monitoring positions of valves and vanes in the appliance, gas detectors, detectors for level of fuel and others. Moreover, sometimes in one location in the appliance several sensors of the same type are installed to duplicate each other. Also, sensor measurements of different types and in the same location might correlate with each other. Groups of duplicated or comparable sensors are called *sensor clusters*. Overall, several hundreds of hardware sensors and measuring devices are mounted at a single appliance. Industrial computers operate on information from sensors in order to monitor unit functioning and send the information to the main database.

Sensor measurement data is stored in schemes similar to the ones shown in Table 3.1 and Table 3.2 (note that we only show the same data in both tables for the reader's convenience - normally, different tables contain information on different entities). Additional information

[1]Picture source: Siemens

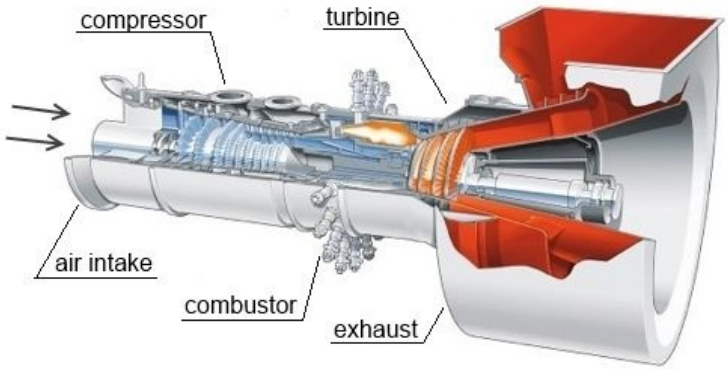


Figure 3.2: Internal structure of an industrial gas turbine (on the example of gas turbine Siemens SGT-600).

concerning sensor readings, as depicted in Table 3.1, may involve either time parameters from the local sensor source (i.e., time fixed in control unit vs. time set at sensor side) or information tag about data filtering (i.e., in a situation, when data is taken once per second, but only one measurement per minute is used for further analysis) or something else, specific for a particular sensor and appropriate analytical usage.

Table 3.1: Raw data (variant 1)

SensorID	Time	Value	Additional Information 1	...
TMP23	2010/07/23 23:11:55	44	49	...

Table 3.2: Raw data (variant 2)

Timestamp	Sensor name	Sensor value
2010/07/23 23:11:55	TMP23	44

Tables of this category contain mostly numerical data and have extremely large size. From a database operations engineers retrieve this data for an analysis, such as diagnostics in case of appliance malfunctions, regular maintenance, and for other purpose. The Siemens use case of the "Optique" project introduces three different scenarios of how engineers use the data:

1. reactive and preventive diagnostics in case of malfunction of an appliance,
2. predictive analysis for regular maintenance of an appliance,

3. product engineering and maintenance support for optimizing product design.

For that purposes, data has to have sufficient data quality. However, in reality it is not always the case. Sensor readings may have the following types of errors and anomalies, also considered in Chapter 2: (i) erroneous values: *inliers* falling within the expected range, and *outliers* falling outside of the expected range, (ii) absence of data values during some period (missing data), (iii) noisy data containing oscillations and noise, and other unexpected behavior that needs to be investigated. Typical reasons for that include sensor inaccuracies, device failures, and poor or absent connection between an appliance and the database. For a more detailed overview of the use case data and precise examples and illustrations DQ inconsistencies in sensor readings of the use case see Appendix. Thus, data cleaning is an important preprocessing step for making use of sensor readings for condition monitoring and diagnosis of turbines. The current work aims to develop a system which detects and removes DQ issues in sensor measurements affecting Correctness, Believability and Accuracy dimensions of data.

There are three possible conclusions one can make while observing listed sensor data irregularities, such as outliers, oscillations and others:

- The identified abnormality is a data quality issue caused by a minor device fault, it is correctly identified as an issue (true positive, as defined in Chapter 2) and occurs once, e.g., a single outlier or oscillation during a short-term period. In that case it is preferably to smooth it by means of appropriate DC methods.

- The abnormality is caused by a defective measuring device and repeats for a period of time. In that case it is preferable to inform the user about a problem and to exclude device measurements from analysis in order not to corrupt result of the diagnostic process.

- The observed abnormality in sensor measurements is not a DQ issue or sensing device malfunction but an actual turbine behavior, and these measurements are considered anomalous because this behavior is not typical for an appliance (e.g., unit malfunction that needs to be diagnosed). In this case we call a found issue false positive and do not apply DC methods, but report it to the user.

Hence, it is essential to distinguish these three cases and to be able to identify actual DQ issues and false positives. In this thesis we focus on the detection of missing values, outliers, oscillations, and uncorrelated sensor duplicates, and achieve this goal by means of statistical methods and semantic model of turbine structure.

Therefore, in this Chapter we propose a novel method for data cleaning that is aimed at sensor readings of power generation facilities and employs not only data cleaning methods, but also appliance structure and properties. In particular, it is crucial to take into consideration the following aspects:

- the type and location of the sensor in the appliance,

- the existence of duplicating sensors, and

- its measurement characteristics.

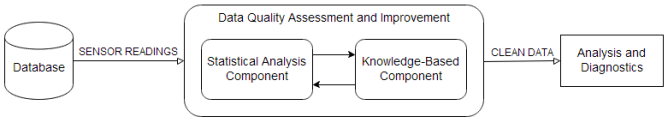


Figure 3.3: High-level interaction of semantic and statistical component

Having an appliance model available which contains information on its components and list of sensors mounted of that component is a prerequisite for that.

We propose to validate anomalies detected in sensor readings consulting duplicate and neighboring sensors in order to distinguish between anomalous appliance behavior (when the equipment faults and sensors detect it), and data quality issues caused by sensor inaccuracy, bad connection and other factors. On the other hand, duplicate sensor measurements may differ which can indicate a problem as well, such as a faulty sensor. The following examples highlight the benefit of our approach:

- Assume that the analysis of sensor measurements detected abnormal behavior such as an outlier. The proposed solution then queries the semantic model whether there are any sensor duplicates, i.e., sensors measuring the same value at the same location in the appliance. If duplicating sensors show a similar behavior, the observed behavior is hardly a data quality problem but turbine behavior. If, on the other hand, the duplicating sensors show no outlier value, it is likely that the outlier is a DQ issue and has to be smoothed.

- The model of an appliance contains information on sensor clusters with duplicating sensors. It is therefore possible to use multivariate analysis to check whether their values correlate and do not have non-simultaneous behavior or large dissimilarity in observations. In case such discrepancy is detected and one of the sensors has abnormal behavior in comparison to the other measuring devices, there is a possibility that this sensor is faulty and its measurements should not be considered as credible.

Thus, using the proposed way of combined evaluation of measurement data and appliance model helps us not only detect and clean DQ inconsistencies caused by inaccuracies and data losses, but also identify possibly faulty measuring devices. The approach consists of two main components (see also a Figure 3.3 showing a high-level basic picture of the proposed method):

- a semantic component, that describes an appliance and sensors mounted on it and provides the information on duplicating sensors, and

- a statistical component, which conducts an analysis on sensor readings in order to identify DQ problems using information provided by semantic component.

Moreover, another challenge in the current use case is the retrieval of relevant data that engineers need for diagnostics and analysis. Currently a project is running development which is aimed to develop a platform that uses query rewriting and answering in an Ontology-Based Data Access (OBDA) system [78]. To achieve that, one of the key components is an ontology

that captures in a familiar language available domain knowledge, such as appliance metadata (ID, model and other information), its structure and list of its components, mounted sensors and their characteristics (type, frequency of measurements, precision, etc), diagnostics knowledge (failures of turbine, their causes and detection with sensors). The semantic model developed during this thesis work is the ontology, purposed to be used further for the development of an OBDA system.

In order to demonstrate the effectiveness of applying our approach to our use case, we designed a use case model (see Appendix for detailed use case description) used for the semantic component, chose several simple statistical analysis methods for anomaly detection in time series data, and implemented a demonstrator for the approach. In this Chapter we detail the design of semantic and statistic components, whereas Chapter 4 contains more information on the demonstrator implementation.

3.2 Semantic Component

The main aim of the semantic component is to describe the structure of turbine, the hierarchy of its components, and its mounted measuring devices. We decided to represent this information in an ontology in order to use it further within the "Optique" project mentioned in the Introduction and detailed in the Appendix. The ontology expressed in the OWL and defines all the concepts related to an appliance and their interconnections. In the scope of the current thesis work the ontology allows us to use a turbine structure information to validate sensor measurements.

Prerequisites for the model

The key components of the "Optique" platform are an ontology to capture user conceptualizations in a familiar language and declarative mappings providing the relationship between the ontology and underlying data [41]. Since one of the project activities is application of query rewriting and answering approaches to OBDA [31], there are some limitations for expressiveness of an ontology language used. Although there exist methods for approximating ontologies, i.e., moving OWL 2 ontologies to less expressive ontology language [14,81], we do not consider them due to the risk of losing model properties and gaining bad quality of data, and thereby contradicting with goals of the work.

Therefore, one of the main requirements for an ontology is its language profile. In order to ensure efficient query answering over an ontology and a good overall performance of an OBDA system, it is beneficial to use the OWL 2 QL language profile wherever possible: this profile was designed to perform query answering using Relational Database System (RDBS) [73]. The OWL 2 QL profile is based on the *DL-Lite* family of Description Logics and allows query answering in PTIME with respect to the size of ontology and AC^0 with respect to the size of data [18] (see also Chapter 2).

Additionally, the design ontology has to be easily extended and manipulated, and cover most of the main entities and properties from the use case. Specifically, the Optique project poses the following functional requirements on the models:

1. specification of an appliance structure, i.e., its components and subcomponents,

2. constitution of main functional parts of the turbine, i.e., ability to express the functional purpose of each component,
3. information for a single measuring device installed on the appliance comprising at least:
    - sensor type, measured value (e.g., thermocouple, measures burner tip temperature);
    - location in the appliance (which part and component it is mounted at, e.g., a combustor);
    - device information (such as its model, ID, vendor, time settings etc);
    - observational characteristics (precision, detection limit, drift, measurement range, frequency etc);
    - measurement characteristics (measurement unit, data type of measurements etc);
4. ability to group sensors in clusters, such as duplicating sensors, sensors mounted at the same component, ot measuring similar qualities,
5. meta-information on the observations, such as timestamp representing the instant the observation was made, the relation between measurements and derived events, etc.,
6. diagnostics information, such as a connection between specific events observed by monitoring devices, symptoms for failure, and particular diagnosis for a turbine.

With respect to the scope of this thesis, only requirements 1-4 must be fulfilled. However, in order to provide the fullest model of the use case for further use, we decided to cover requirements 5-6 as well, and provide mechanisms for extending the ontology in the future. To address requirements to sensor structure and to simplify the construction of ontology, we decided to introduce a specific sensor ontology module into our overall model.

Available Sensor Ontologies

Recently sensor networks and sensor data receives a lot of attention, since digital sensors are everywhere around us: each electronic device is equipped with various sensing devices. Semantic web technologies are one of the good methods to overcome volume, complexity and heterogeneity of data for various sensors and systems. In general, there exist two standards which are used to construct sensor ontologies and to describe sensors, their characteristics and operation:

- SensorML: an approved Open Geospatial Consortium standard [2] [80]. It provides standard models and an XML encoding for describing process as well as the geometric, dynamic, and observational characteristics of sensors and sensor systems.
- Observations and Measurements (O&M): an international standard [1] that defines a conceptual schema for observations, and for features involved in sampling when making observations. The XML implementation of the standard [25] is widely used and approved as a standard by the Open Geospatial Consortium.

[2] `http://www.opengeospatial.org/standards/sensorml`

To identify a suitable ontology for representing sensor and measurement data, we had a look at the existing sensor ontologies [24] and their available material on the Internet (such as documentation, XML files, UML diagrams), trying to find one, which would fit all requirements after minimal changes. Most of the existing ontologies contain concepts and hierarchy according to the devices and sensors they use in particular in their field of application. Although recently the W3C Semantic Sensor Networks Incubator Group (SSN-XG)[3] started developing a general and expressive ontology for sensors [24]. It is called Semantic Sensor Network (SSN) Ontology, covers large parts of the SensorML and O&M language standards, and it can describe sensors in terms of capabilities, measurement processes, observations and deployments [23]. SSN includes and is based on the foundation ontology DUL[4] for general concept descriptions, such as qualities, regions, object categories and others. The full SSN ontology consists of 41 concepts and 39 object properties and inherits 11 DUL concepts and 14 DUL object properties. The key concepts and relations of the SSN ontology are shown on the Figure 3.4. The central part of the SSN Ontology is the Stimulus-Sensor-Observation (SSO) ontology design pattern, which links measuring devices with the observations they make. In general, the creators of SSN claim that the ontology can be seen from the four different perspectives [23]:

- A sensor perspective, focused on the measuring device and what does it sense;
- An observation perspective, focused on the sensor readings and observations;
- A system perspective, focused on systems of sensors and where they are deployed;
- A property perspective, focused on the particular property and observations made about it.

However, SSN is designed to be a general ontology for sensors, and some of the concepts and blocks included in the ontology are meaningless in our case and only added overhead that led to inconsistencies, which were hard to resolve. Therefore, we decided to build an ontology on our own, building on best practices from SSN.

Final Ontology Design

General block structure SSN follows a certain block structure, i.e., concepts and relations are split into 10 modules roughly presented in Figure 3.5a and in more details in Figure 3.4. This structure however turned out to be too complex for our use case, including too many superfluous concepts and thus redundant blocks. Therefore, we adopted the idea and created a downscaled version of the ontology with a similar but simplified block structure adjusted to our domain (see Figure 3.5b).

Splitting the ontology Focusing on the OWL 2 QL profile leads to certain limitations with respect to diagnostics capabilities required for the "Optique" platform. But in order to formulate large amounts of diagnostic knowledge, i.e. (i) event and messages indicating at a certain

[3] `http://www.w3.org/2005/Incubator/SSN`

[4] Upper ontology DUL `http://ontologydesignpatterns.org/ont/dul/DUL.owl`

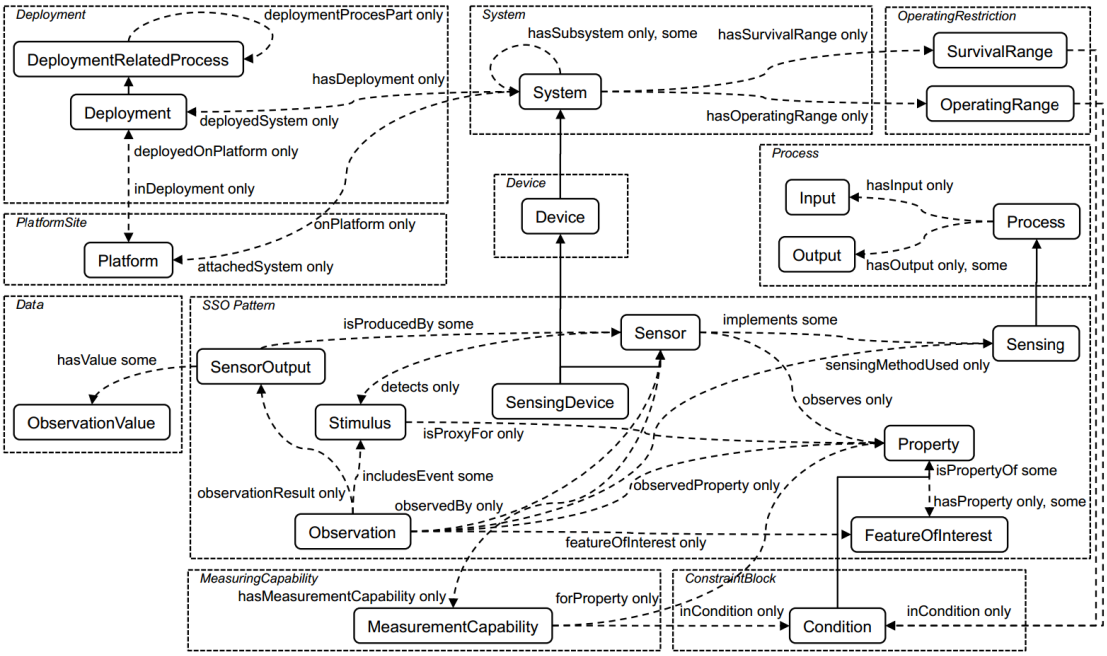


Figure 3.4: The SSN ontology.

symptom, (ii) symptoms and other signs implying a certain diagnosis, and other dependencies between observations and turbine condition, we need to use qualified existential restrictions used on both sides of inclusion assertions. The OWL 2 QL however supports existential quantification on the right hand side only: although the *DL-Lite*-family does not support them in general, but they still can be simulated on the right hand side of class expressions using role assertions in *DL-Lite*$_{\mathcal{R}}$ (see Chapter 2). Nevertheless, in our use case it is necessary to use qualified existential restriction on the left hand side in domain axioms as well. As an alternative to these restrictions, it is also possible to create several roles in OWL 2 QL and to specify their domains and ranges. However, it is not acceptable for our case, and we show that below.

As an example, consider a diagnosis that can be determined by two symptoms, so they are represented as classes connected with relation `hasSymptom` (see Figure 3.6). In the OWL 2 DL we can express this model by using an existential restriction, whereas in the OWL 2 QL we have to restrict domain and range of the corresponding relation:

In OWL 2 QL:

dom(hasSymptom) = Diagnosis

ran(hasSymptom) = Symptom1 $\sqcap$ Symptom2

in OWL 2 DL:

Diagnosis $\sqsubseteq \exists$ hasSymptom.Symptom1

Diagnosis $\sqsubseteq \exists$ hasSymptom.Symptom2

However, the usage of existential quantifier implies that there exists at least one connection

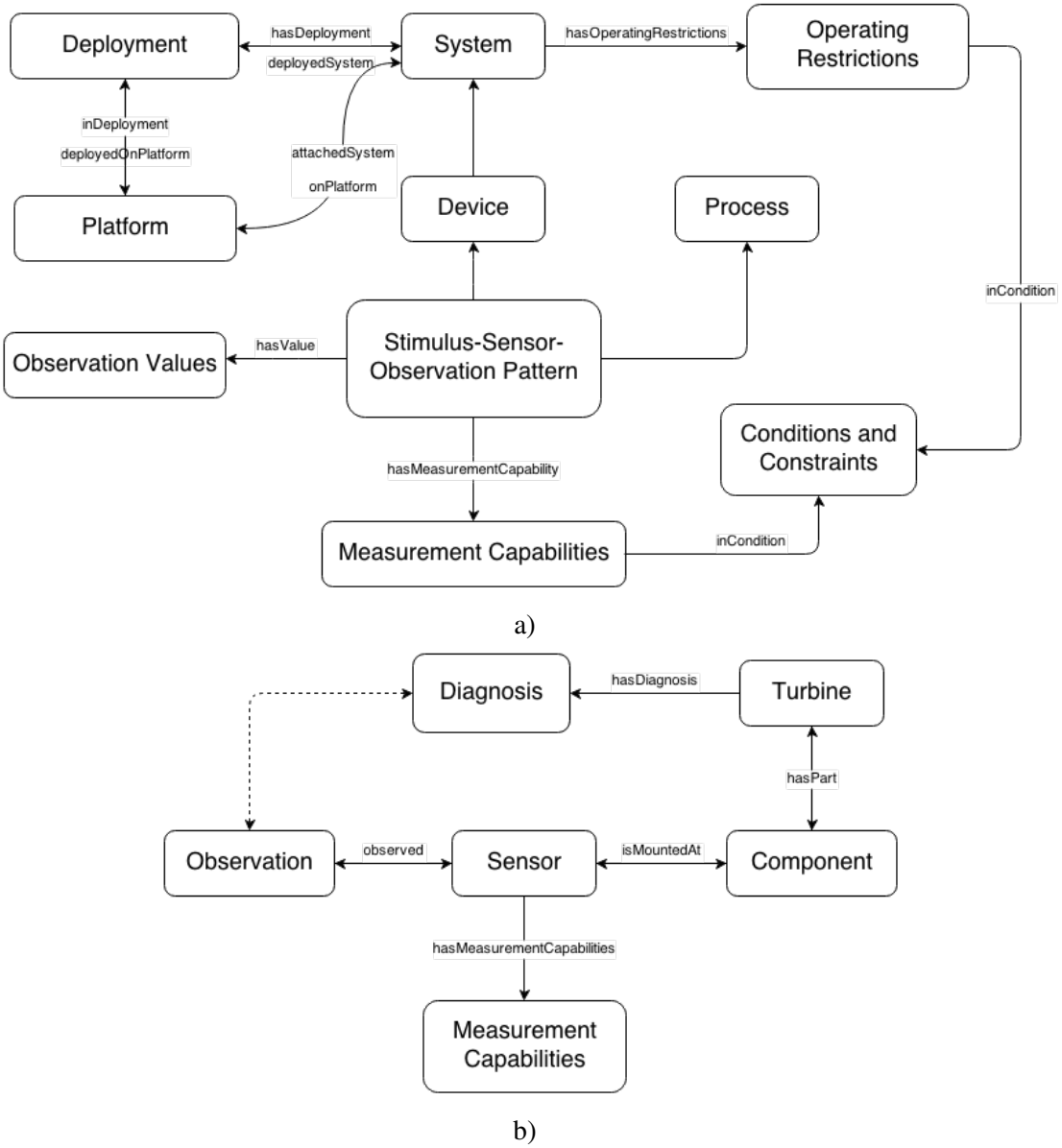


Figure 3.5: Schematic representation of a) modules in SSN ontology, b) modules in the developed ontology.

of diagnosis with the symptom, whereas restricting domain and range of the `hasSymptom` in OWL 2 QL does not enforce this restriction and `hasSymptom` relation might be empty. This difference in expressiveness plays key role in formulation of dependencies between diagnoses, symptoms, and observations. For instance, the axiom below formulates that symptoms detected specifically in the combustor of an appliance and shutdowns of turbine (denoted by messages of Category 3, for a detailed notation see Appendix) indicate the possible diagnosis Flame Failure

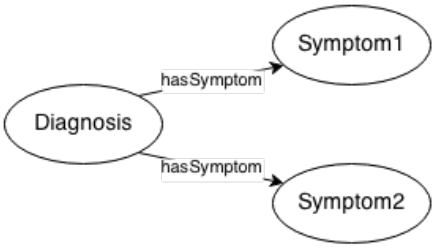


Figure 3.6: Illustration for motivation of using more expressive logic for diagnostics

of an appliance:

$$\exists \text{ hasDiagnosis.FlameFailure} \sqsubseteq \text{Turbine} \sqcap \exists \text{ hasSymptom.}(\exists \text{ isDetectedBy.}(\exists \text{ isMountedAt.Combustor})) \sqcap \exists \text{ observed.Category3}$$

where each concept contains the following objects:

Turbine	- class of turbines,
∃ hasDiagnosis.FlameFailure	- turbines with the "Flame Failure" malfunction,
∃ hasSymptom.A	- turbines which show the symptom "A" in its behavior,
∃ isDetectedBy.B	- observations detected by sensor "B",
∃ isMountedAt.Combustor	- measuring devices mounted at the "Combustor" component of the appliance,
∃ observed.Category3	- turbines, during which functioning events of "Category3" (i.e., shutdowns of the turbine) happened.

However, usage of restricted domains and ranges instead of qualified existential restrictions would not imply, that each concept above in the axiom is not empty. Thus, we want to use OBDA systems that require the OWL 2 QL on the one hand, but on the other hand we need to use more expressive logic for diagnostics purposes. The solution we took is the following: to split an ontology into several parts, each expressed in its own language profile. This way we will be able to use OBDA system with the appliance and sensor information and at the same time express all available diagnostics knowledge in a more expressive language profile. With respect to the blocks in Figure 3.5b we formed three ontology modules using specific OWL profiles:

- blocks "Diagnosis" and "Observation" form the Diagnostics ontology expressed in OWL 2 DL,
- blocks "Turbine" and "Components" form the Turbine ontology expressed in OWL 2 QL, and
- blocks "Sensor" with "Measurement Capabilities" form the Sensor ontology, also expressed in OWL 2 QL.

Although Turbine and Sensor ontologies are expressed in the same language profile, we decided to make them independent for easier manipulation in further use, since they define different and largely independent aspects of the domain.

The next step is to link the created ontologies to a consolidated Use case ontology, that covers all aspects of the application domain described above in this chapter. The problem of integrating heterogeneous (i.e., expressed in different formalisms) ontologies is well-known in modular ontology design, and there exists a number of solutions for integration and interoperability of ontologies, such as frameworks MAFPA [68], Ontology Integration System [19], and others. One of the most significant solution for integration of distributed ontologies is the *Distributed Ontology Language (DOL)*, which is currently being standardized within the Ontology Integration and Interoperability (OntoIOp) activity of ISO/TC 37/SC 3 - international standard for systems to manage terminology, knowledge and content [5] [63]. The core of the DOL is the graph of different existing ontology languages and translations between them, which provides mechanisms to relate ontologies expressed in different formalisms [72]. The DOL covers all basic ontology languages and provides a meta-level on top of them, enabling the following features for distributed and heterogeneous ontologies [72]:

- relates ontologies in different formalisms, including propositional logic, first-order logic with equality $FOL^{=}$, various OWL profiles, UML, Resource Description Framework (RDF) and others;
- allows to re-use ontology modules, even if they are formulated in different formalism;
- allows to re-use ontology tools, such as theorem provers, along translation between formalisms;
- uses IRI for unique identification, ontologies can be distributed over several sources or over the Web;

However, OWL has a mechanism that allows to import one OWL ontology into another OWL ontology [86]. Since above-described three fundamental ontologies are expressed in OWL ontology language and are not distributed across the web or different sources, this import mechanism is sufficient for our use case. We created one consolidated ontology providing a comprehensive model of the Siemens Use Case of the "Optique" project, which imports three fundamental ontologies describing different aspects of the domain. This integrated Use Case ontology connects the modules using `EquivalentClass` assertions for classes representing the same entities in different ontologies. Figure 3.7 schematically depicts the design and interconnection of the three module ontologies and the consolidated Use Case ontology.

Detailed description of ontologies Each of the three module ontologies is independent and describes one of the fundamental entities of the use case. For development of the ontologies we used the well-known free open-source ontology editor Protégé[6], version 4.3. For ontology

[5] `http://www.iso.org/iso/standards_development/technical_committees/other_bodies/iso_technical_committee.htm?commid=48136`

[6] `http://protege.stanford.edu/`

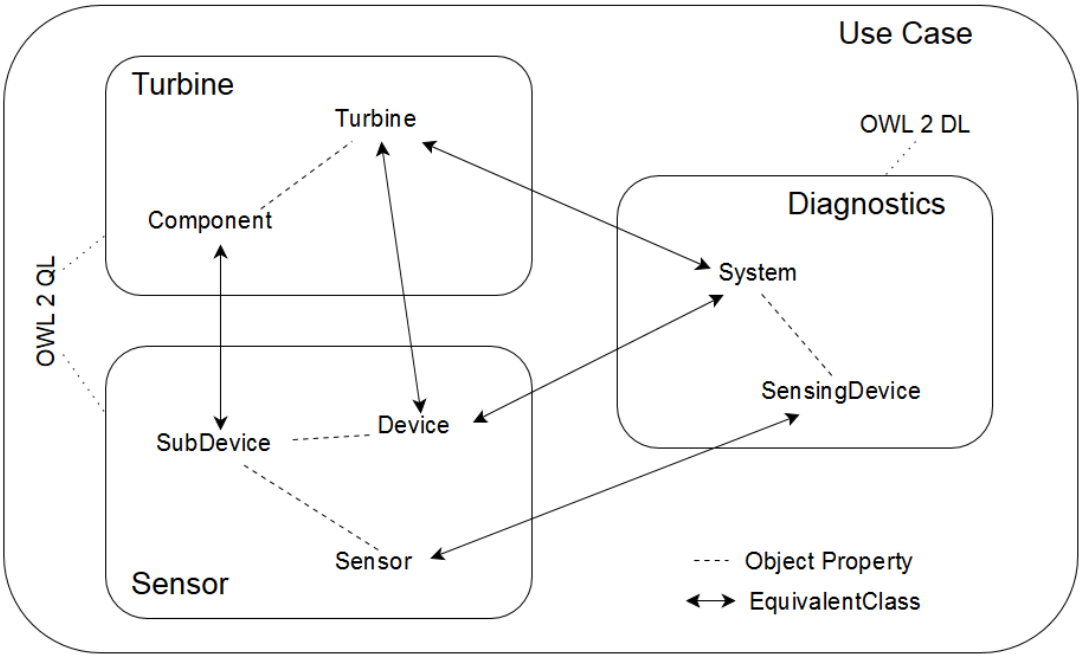


Figure 3.7: Final ontology design.

visualization we used an OntoGraf plug-in bundled with the standard installation of Protégé. Below we detail each ontology and provide an illustrating diagram.

1. **The turbine ontology** addresses requirements 1-2 formulated above in this setion, i.e., describes the internal structure of an appliance, i.e., it lists all its parts, functional units, and their hierarchy. It contains approximately 60 classes, 15 object and data properties. The central class `SystemElement` contains three subclasses: `Turbine`, `Component` and `FunctionalUnit`.

   a) Subclass `Turbine` models product families and contains turbines as individuals.

   b) Subclass `Component` describes the inner structure of a turbine, its main parts and their hierarchy, using relations such as `hasPart`, `hasDirectPart` and others.

   c) Subclass `FunctionalUnit` lists important functional blocks of an appliance, such as `GasPath`, `GasFuelSystem`, `LiquidFuelSystem`, and others. Each such system groups turbine components with such properties as `hasFunctionalSuccessor`, `hasFunctionalPredecessor`.

   The axioms in this ontology enforce the exact structure of a piece of machinery. For instance, we require that an appliance can not be component part of anything:

$$\text{Turbine} \sqsubseteq \neg\, (\exists\ \text{isPartOf})$$

But on the other hand, every turbine must have among others a Control System, a Generator, and a LubOilSystem:

$$\begin{aligned} \text{Turbine} &\sqsubseteq \exists\, \text{hasDirectPart.ControlSystem} \\ \text{Turbine} &\sqsubseteq \exists\, \text{hasDirectPart.Generator} \\ \text{Turbine} &\sqsubseteq \exists\, \text{hasDirectPart.LubOilSystem} \end{aligned}$$

Similar axioms are used for defining the structure of elements:

$$\text{LiquidFuelPump} \sqsubseteq \exists\, \text{isPartOf.LubOilSystem}$$

The full ontology scheme is depicted in Figure A.3 in the Appendix.

2. **The sensor ontology** addresses requirements 3-4, i.e., lists and categorizes types of measuring devices mounted at the turbine. It has approximately 40 classes and 20 properties. The main class is `Sensor` listing all types of measuring devices mounted on an appliance (e.g., gas detector, temperature sensor, etc) and with further branching of classes gives more detailed characteristic information on them (e.g., temperature sensors could measure: burner temperature, inlet temperature, compressor exit temperature, etc). Using location, type, detailed characteristic, measurement properties, and other information from the ontology about a sensor we acquire duplicate and comparable sensors. Grouping sensors by other features can be defined additionally using sensor clusters. Added axioms assert, for example, that each sensor is mounted at some turbine component or functional unit:

$$\text{Sensor} \sqsubseteq \exists\, \text{isMountedAt.SubDevice}$$

Sensors of different types are disjoint: one sensor cannot be mounted at different locations of the turbine and measure different values at the same time:

$$\begin{aligned} \text{CompressorExitTemperature} &\sqsubseteq \neg\, \text{ExhaustTemperature} \\ \text{CompressorExitTemperature} &\sqsubseteq \neg\, \text{GasDetector} \\ &\ldots \end{aligned}$$

Temperature sensors measure temperature values and cannot measure pressure or something else:

$$\begin{aligned} \text{Temperature} &\sqsubseteq \exists\, \text{measures.Temperature} \\ \text{Temperature} &\sqsubseteq \neg\, (\exists\, \text{measures.Pressure}) \end{aligned}$$

The full ontology is shown in Figure A.4 in the Appendix.

3. **The diagnostics ontology** addresses requirements 5-6, i.e., formalizes the connection between events generated by measuring devices and the control unit, and typical symptoms of different faults of the turbine. At the moment there are approximately 30 classes and 10 properties. The core classes are `Observation` and `Diagnosis`, connected with the relation `indicatesAt` for listing characterizing symptoms for each diagnosis. Structural requirements enforced by this ontology include: each diagnosis has to be assigned to some System Element, i.e. to a turbine or its component, and must be supported by some symptoms:

$$\begin{aligned} \text{Diagnosis} &\sqsubseteq \exists\, \text{indicatesAtDiagnosis}^{-}.\text{Symptom} \\ \text{Diagnosis} &\sqsubseteq \exists\, \text{hasDiagnosis}^{-}.\text{System} \end{aligned}$$

   The Diagnostics ontology also detailed in Figure A.5 in the Appendix.

4. **The use Case ontology** imports all three above-described ontologies and connects their entities together by introducing equivalences between corresponding classes using the `EquivalentClass` property. For instance, the Sensor ontology contains a class `SubDevice` connected to `Sensor` via the property `isMountedAt`. In the Use Case Ontology, this class is equivalent to the `Component` class of the Turbine ontology. Similarly, the `Sensor` class in the Sensor ontology is equivalent to `SensingDevice` in the Diagnostics ontology, which is connected via `hasMeasured` to `Observation`.

In the current thesis work for sensor data quality assessment and improvement, Sensor and Turbine ontology do however play the most important role. But we encode the full application domain into the ontology for two following reasons:

1. In this work we focus on validating detected data anomalies using clusters of sensors located in the same component and monitoring the same process. Additionally, we check measurement ranges for sensors, if there are any defined in the ontology. But, focusing on the location and topology of the appliance now in this work, and proving that the employing domain knowledge information gives us noticeable advantage, we get wide opportunities for using great amount of knowledge that is present int he ontology. For instance, employing knowledge in Diagnostics ontology and integrating it with the statistical analysis would allow us to upgrade the method from data quality assessment of sensor data and detecting faulty sensors to a strong condition monitoring system. But this is one of the aspects for the future work, which we discuss in Chapter 5.

2. The proposed model is suitable to be used in the context of model-based data access control. For example, OBDA systems, that provide end-users with domain-language access to data by automatically translating user questions into queries of the underlying data base(s), can be used with the designed model, and this is one of the directions of future work in the context of the “Oprique” project, described before in this chapter.

3.3 Statistical Component

In this section we present the statistical component designed to interact with the knowledge-based component introduced before. To reduce development time, we chose to rely on the statistical computing language R [7] and some of its additional packages for implementation. For details see Chapter 4.

The statistical methods are intended to identify and remove data irregularities in sensor measurements, namely (i) outliers, (ii) oscillations, (iii) missing data, and (iv) differing behavior of comparable sensors. In this section we revisit some of the approaches for treating outliers, oscillations and missing data considered in Chapter 2 and Tables 2.6, 2.8, 2.7 in Chapter 2 and choose particular methods applicable to data of the use case. We decided to use the most effective and simple methods for our approach, in order not to complicate our statistical component, but still to demonstrate the novelty and efficiency of our idea: enhancing statistical methods using comprehensive domain models.

Missing values

It is essential to check sensor measurements for simple and coded missing data (as defined in Chapter 2), i.e., when there is no measured value for a timestamp. In some cases only a value or two may be lost, whereas in the other cases there are long periods of missing data. In order to predict missing values we use methods in Table 2.8 of Chapter 2 depending on how many values are lost. For a few missing values (up to 4 in a row) we use simple imputation methods. For larger periods of missing values we avoid using imputation, because as it was mentioned in Chapter 2, it might lead to a wrong estimation of variance and other statistical parameters. Therefore, for longer periods of missing data there are two possibilities depending on the information provided by the semantic model:

1. the sensor does not have any duplicate sensors or correlation values with them are lower than a permissible tolerance: we fit an ARIMA model to sensor readings and predict missing values using the model, or

2. the sensor has one or several duplicate sensors: we estimate missing values using a regression model based on measurements of other sensors.

However, before value prediction, we also estimate a rate of missing values. If during the day too much data is lost, e.g., more than 90%, we inform the user about that and discard the data as a low quality data.

Outliers

First, we consider detection of outliers in the univariate case, i.e., for analysis of measurements of a single sensor. Sensor readings can be understood as a stochastic process, namely discrete time series data, since sensors take measurements not continuously but with a certain rate. As

[7] http://www.r-project.org/

we also mention in Chapter 2, for time series data we need to take into account that outliers need not to be extreme with respect to the overall range of data variation, but have to be considered as an outlier with respect to local values. In this case we have to estimate how different the value is with respect to its neighbor values some time ago and/or some time in the future.

The most effective and widely used approaches for outlier detection in time-series are (i) the Hampel filter [82], and (ii) outlier detection using fitted ARIMA models [21]. Both methods have their drawbacks and benefits. The Hampel filter uses a parameter k for a number of future and past observations a single value have to be compared with, and it can not be used before at least $2k$ observations are made. That happens due to its moving window approach (see Chapter 2). That is clearly a drawback, because in that case the Hampel filter does not detect outliers in the first and last k observations. Another disadvantage relates to the parameter k as well: as any other moving window approach, the Hampel filter is highly reliant on the parameter value. Finding an optimal value for a parameter is very challenging. If k is too small, the filter becomes more sensitive and may produce far more type I errors, i.e., consider correct values as anomalous, whereas if k is chosen to be high, filter becomes more forgiving and may not detect anomalous values accepting them as correct, thus producing errors of type II. However, the Hampel filter is often considered as practically effective [13, 82] and successfully localizes local outliers in time series data, which is clearly an advantage of an approach. Unfortunately, there exist no general technique of choosing the optimal value k; in Chapter 4 we provide a comparison of Hampel filter sensitivity depending on the parameter k. On the other hand, ARIMA models can effectively describe a wide variety of industrial problems [8, 13], as with enough points used in regression and averaging, it is possible to fit the model to almost any time series.

Therefore, we have employed both Hampel filter and ARIMA model for outlier detection. Additionally, we tried to combine the results of outlier detection by both algorithms as we wanted to compensate inability to detect outliers in first and last k readings for the Hampel filter.

We also correct detected outliers. In case of using Hampel filter, outlying value is replaced by the median of k previous and k subsequent values, where k is a parameter for the filter. Using ARIMA model, we replace an outlying value with its predicted fitted value, provided by the model.

Oscillations and noise

As it has already been mentioned in Chapter 2, during oscillation and noise periods values strongly deviate from the mean. In order to detect such periods in data, we chose to use the simplest but still the most effective method: analysis of the autocorrelation function (see Table 2.6). We use the fact that these fluctuations in noise are uncorrelated, therefore autocorrelation values are closer to zero, whereas measurements with oscillations have oscillatory autocorrelation function. Therefore, we use the following procedure:

- calculate autocorrelation functions for each sensor measurements,
- analyze their behavior,

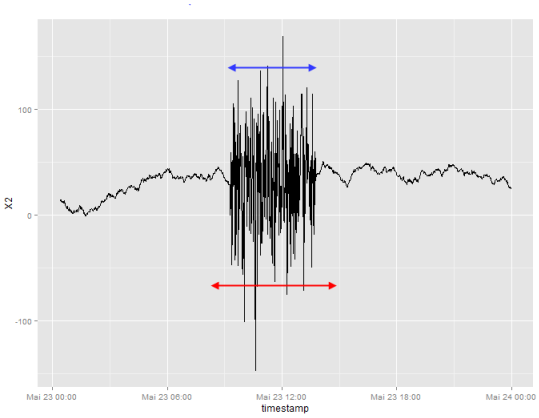


Figure 3.8: Disadvantage of oscillation detection.

- for sensors, whose autocorrelation values are too low (e.g., <0.35, a precise threshold could be additionally set), we search for changes in standard deviations and amplitude of measurements using moving window methods,
- identify detected periods of high variance and high amplitude as noise.

The advantage of this method is its simplicity and effectiveness, however, due to a smoothing effect caused by moving window methods it lacks high precision, and its influence is shown on Figure 3.8. Sensor measurements have a period with heavy oscillations, that have to be detected. Blue arrow shows the actual period of oscillating, whereas red arrow illustrates the detected period of oscillating. Moving window methods detects a change in a parameter a bit earlier, depending on the window size, therefore identifying correct values as belonging to oscillation, i.e., producing bigger amount of false positives. However, the accuracy of the oscillation detection is sufficiently high. In the next chapter we provide test results of oscillation detection, providing evidence for these statements. For a detected oscillation period there are two possibilities depending on the information provided by the semantic model:

1. the sensor does not have any duplicate sensors or correlation values with them are lower than a permissible tolerance: we apply exponential moving average for smoothing the oscillations, or
2. the sensor has one or several duplicate sensors: we treat the noise as a disguised missing data and estimate values using regression model based on measurements of other sensors.

Moreover, in case of presence of only heavy oscillations or noise in data, e.g., more than 90% of measurements, we discard the data.

Difference between duplicating sensors

Measurements of duplicating sensors, i.e., sensors of the same type mounted at the same part of turbine, are checked by the statistic component in order to reveal sensors whose measurements

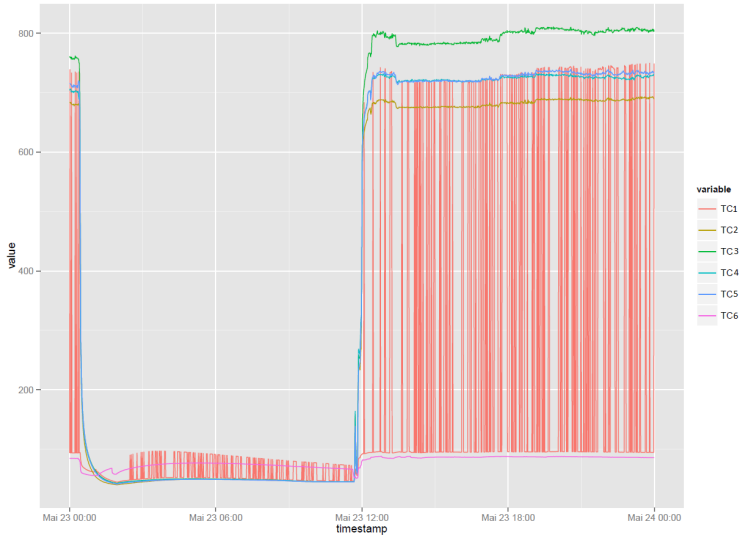


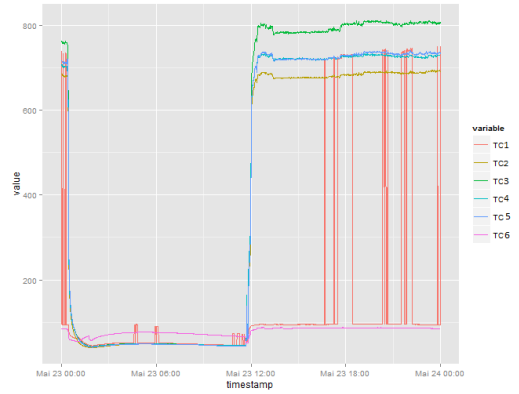
Figure 3.9: Thermocouple measurements

significantly differ from reading provided by other duplicates. We conduct a simple analysis of the main statistical components - mean, maximum, minimum, and standard deviation using moving window methods. If one sensor in a group continuously shows high difference with the other sensors, and the difference exceed one standard deviation, we assume that measurements of this sensor might not be reliable enough. However, in this case we only characterize the data and do not provide any data quality improvement for those measurements. We show a corresponding alert to the user and do not use measurements of this sensor for predicting readings of other duplicates.

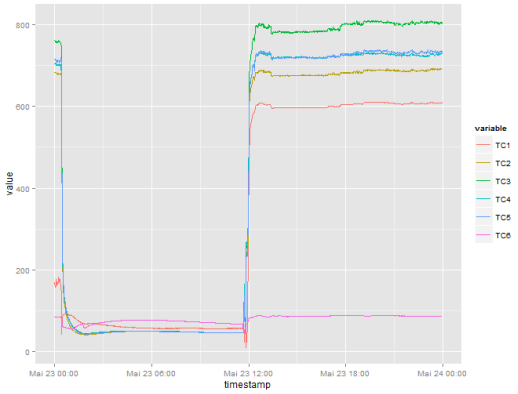
Order of Applying Operations

One of the important aspects for a statistical component is to define the order of applying detection and correction operations for sensor measurements, as one procedure may affect the results of the other procedure. In this section we motivate the preference for applying operations. Supporting examples are based on real measurement data of 6 thermocouple sensors installed in the combustion chamber of a gas turbine. Figure 3.9 shows a fragment of their measurements during a single day (24 hours). Each sensor measures temperature once per minute, therefore there are 1440 values of each sensor.

Outlier detection algorithms often detect a number of outliers during high amplitude fluctuations. Correcting those outliers often gives unsatisfactory results. In Figure 3.9 TC1 measurements contain white noise, i.e., disguised missing data. Results of application of outlier detection and correction procedure are shown on Figure 3.10a. On the other hand, Figures 3.10b shows results of employing available information on duplicate sensors from the semantic model and using prediction. Therefore, it is preferred to detect oscillations and disguised missing data before outlier detection.



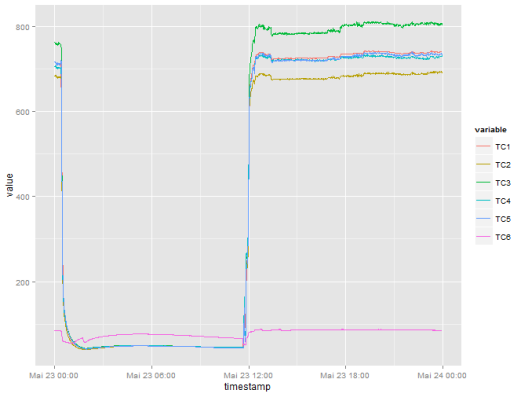
a)



b)

	TC2	TC3	TC4	TC5	TC6
TC2	1.0000000	0.9998073	0.9999066	0.9999228	0.8735512
TC3	0.9998073	1.0000000	0.9998273	0.9999123	0.8736521
TC4	0.9999066	0.9998273	1.0000000	0.9999248	0.8725771
TC5	0.9999228	0.9999123	0.9999248	1.0000000	0.8728469
TC6	0.8735512	0.8736521	0.8725771	0.8728469	1.0000000

c)



d)

Figure 3.10: a) Result of smoothing outliers in TC1 data, b) predicting TC1 data using duplicating thermocouple sensors, c) correlation values of duplicate sensors, d) predicting TC1 data without the usage of TC6 readings.

However, before applying prediction algorithms using measurements of duplicating sensors, we have to ensure, that they all correlate well and none of them has vast difference in comparison with the other duplicates. In Figure 3.9 it is easy to see that TC6 measures unusually low values in contrast with higher values of its duplicating sensor readings, i.e., TC2, TC3, TC4, TC5, the difference between their readings reaches 743^oC. Additionally, TC6 readings correlate with other sensor readings not as well as they correlate with each other (see Figure 3.10c). Detecting that sensor TC6 has suspicious behavior in comparison with its duplicates, and excluding TC6 for predicting TC1 disguised missing values gives us the result shown on Figure 3.10d.

In addition, as it is also stated in Chapter 2 moving-window based algorithms can not be applied to measurements with simple or coded missing data, therefore it is necessary to predict missing values before applying the Hampel filter or oscillation detection algorithm that uses moving standard deviation and amplitude functions.

Therefore, operations are applied to the data in the following order:

- analyze duplicating sensors and their correlation values, identifying possibly unreliable measurements, if there are any;
- predict missing values, if there are any;
- detect oscillations and consult the semantic model, that duplicating sensors do not have the same oscillating behavior in this period;
- detect outliers without consideration of oscillating periods, and exclude false positives using information provided by the semantic model;
- smooth oscillations and outliers, if there are any.

3.4 Component Interaction

In this section we describe the joint operation of the two above-described components applied to sensor data. Figure 3.11 details the procedure of accessing and improving sensor data quality as a UML 2 sequence diagram.

First of all, the semantic component loads the Use Case ontology, traverses it and looks for all measuring devices that are present in the ontology, and the statistical component load the file with sensor measurements. Sensors, whose labels are found in the file, are then matched with individuals from the ontology. Next, we conduct a search for duplicates (if there are any) for each sensor (see Algorithm 3.1). There are two main opportunities to define a sensor cluster:

1. we define a class `SensorCluster` in Sensor ontology, which is intended to store clusters of sensors;
2. we search in an ontology for sensor individuals, that are mounted at the same turbine, deployed at the same component of this turbine, and have the same type of measurements (see Algorithm 3.1).

Based on results of the search, we identify how many sensor clusters are there, which sensors they consist of, and which sensors do not belong to any cluster, this determines, for example, application of the ARIMA models instead of regression model for data prediction (see description of statistical component above in this chapter).

After that, we analyze sensors within each cluster to identify devices having low correlation with its duplicates or vast difference in measured values If there is a sensor that has low (closer to zero) correlations with the other sensors in the cluster and its values and descriptive characteristics continuously differ from the corresponding values of other sensors, then we alert that this sensor is possibly faulty, as it does not show expected behavior. Additionally, we exclude this sensor from the cluster, i.e., do not use its readings for regression modeling: in case we have to predict data of other sensors and we use regression models for that, suspicious sensors are not used in the modeling process.

Next, we conduct univariate analysis of all sensor readings in order to detect and correct DQ deficiencies. Algorithm 3.2 depicts an algorithm for the univariate analysis, whose results

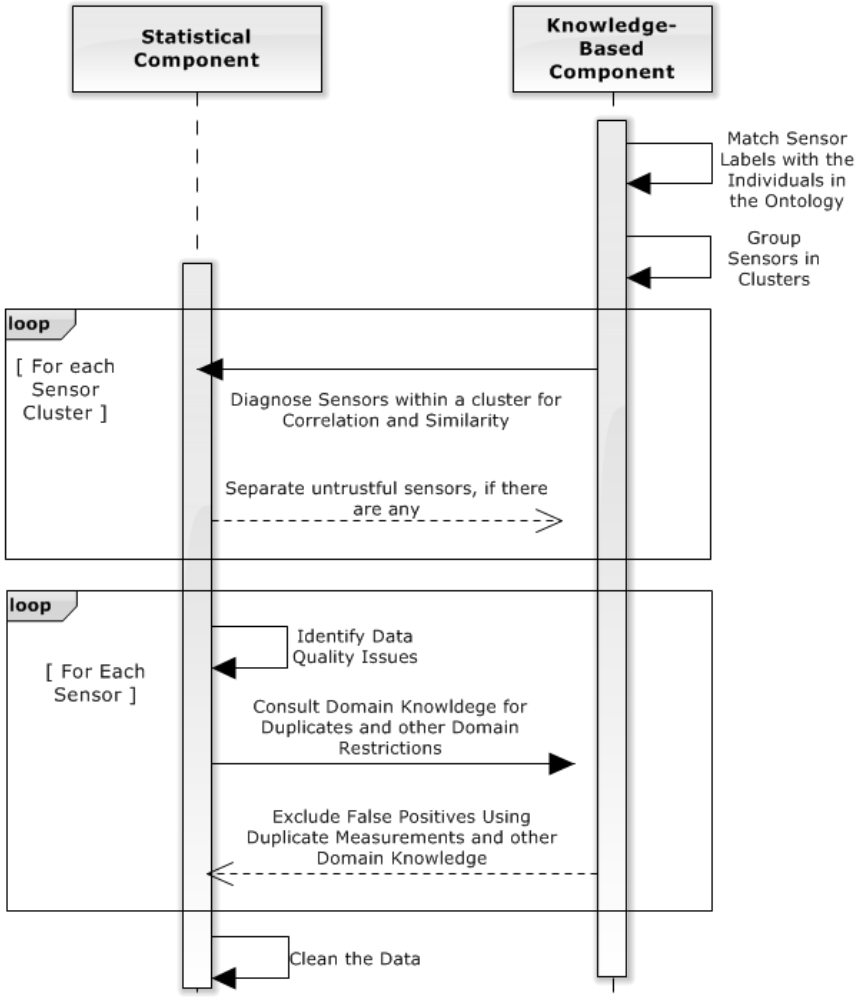


Figure 3.11: Data Quality Assessment and Improvement

are further validated in Algorithm 11. We conduct anomaly detection using statistical methods, described above in this chapter. They include outlier and oscillation detection and smoothing, missing data prediction. Querying knowledge-based component, we receive information on sensor clusters and compare results of anomaly detection conducted by statistics for sensors within clusters. If there are anomalies detected in the same moments of time, i.e., peaks detected at the same minute, we conclude that this is an appliance behavior and this peak is not an anomaly, but a false positive, therefore exclude it from the list. Additionally, for each sensor the ontology might contain measurement range restrictions, e.g., sensors in the combustion chamber can not measure minus temperatures. We retrieve these domain restrictions from the ontology and check, if sensor measurements violate them, e.g., contains values outside of the defined range.

In Chapter 2 we described four possible outcomes of Data Quality Audit. As a result of

Algorithm: DUP-SEARCH

Input : Sensor labels

Output: Groups of duplicates for sensors

```
1 foreach sensor1 in sensor-list do
2     foreach sensor2 in Ontology do
3         if SameType(sensor1, sensor2) AND SameAppliance(sensor1,sensor2)
          AND SameLocation(sensor1, sensor2) then
4             DuplicatesList(sensor1) ← Add(sensor2)
5         end
6     end
7 end
```

Algorithm 3.1: Duplicate search

running the above-described procedure for DQ assessment of sensor measurement data, possible outcomes are:

- DQ is improved, i.e., missing data is predicted using an appropriate algorithm, identified anomalies in data are smoothed;
- DQ does not need an improvement, there are no anomalies in sensor readings, it has high autocorrelation values and good correlated with its duplicate sensors;
- data is characterized and the results of data analysis are provided to the user, but no DQ improvement procedures are applied, i.e., for suspicious sensors identified during comparison with its duplicates, such as TC6 in the example above;
- data is discarded, if the bigger portion of the data is missing, user is informed and no further analysis is performed.

The approach proposed and detailed above has been further implemented. In the next Chapter we provide implementation details, whereas Chapter 5 discusses open issues and improvement possibilities of the approach.

Algorithm: UNIV-ANALYSIS

Input : Sensor measurements

Output: Summary of analysis for anomalies, corrected data

```
1  begin Missing data detection and prediction block
2      CheckForMissingValues();
3      if there is data missing then
           // if too much data is lost, percentage is set in
              parameter threshold, reject the data
4          if missing-data.size() > threshold then
5              reject data;
6          end
7          else RunPredictionAlgorithm(missing-data);
8      end
9  end
10 CalculateAutocorrelation();
11 if acf-values < threshold then
       // if autocorrelation is questionably low, look for
          oscillations
12     CheckForOscillations()
13 end
14 CheckForOutliers();
15 CleanData;
```

Algorithm 3.2: Univariate analysis

Algorithm: EXCL-FP

Input : Sensor anomalies

Output: Sensor anomalies with excluded false positives, corrected data

```
1  CalculateCorrelation(sensor-list);
2  foreach sensor in sensor-list do
3      if outlier-list not empty AND DuplicatesList(sensor) not empty then
4          foreach outlier in outlier-list do
5              if at least one DuplicatesList has outlier then
6                  outlier is a False Positive
7              end
8          end
9      end
10     CleanData();
11 end
```

Algorithm 3.3: Exclude false positives

CHAPTER 4

Implementation and Evaluation

4.1 Implementation

The method proposed in Chapter 3 has been implemented in the programming language Java. All test have been run on a machine with the setup described in Table 4.1. It loads sensor measurements as .csv files that are expected to be structured according to Table 3.2 in Chapter 3. The program requires that the first column necessarily contains timestamp information and has format "%d.%m.%Y %H:%M".

Table 4.1: Computer hardware

Manufacturer	Fujitsu
Processor	Intel®Core™i5-2400 3.10 GHz
RAM	20,0 GB
Operating System	Windows 7 Enterprise (SP1) 64-bit

The implemented program demonstrated the successful functioning of the method for assessing and improving of data quality for sensor measurements, based on the interaction of a semantic and statistical component as described in Chapter 3. Each component is based on an appropriate API: for semantic part we use an ontology API to read an ontology designed in Chapter 3 in order to perform such tasks as search for duplicating sensors or to control fulfillment of its restrictions (data ranges), whereas for the statistical part we use an API for the R language in order to perform statistical computing on our data. For both components there is a number of opportunities, for API choice, which we motivate below.

Statistical Component

As it has already been mentioned in Chapter 3, methods for sensor data validation are implemented with R - an open-source and freely-distributed statistics software package [97]. There exist additionally several hundreds of additional external packages for R which contain collections of functions and data and thus extend the capabilities of R. In the course of this thesis the following packages of R have been used for implementation:

- `ggplot2`[1] - a plotting system based on a grammar of graphics used for data visualization. Plots illustrating quantitative data in Chapter 3 and in this Chapter have been created using this tool.

- `pracma` - contains functions from numerical analysis and linear algebra, numerical optimization, and differential equations. Particularly, this package contains functions for detection of outliers and peaks, e.g., an implementation of the Hampel filter.

- `TSA` - contains methods for time series analysis detailed in [26]. Particularly, it has functions for fitting ARIMA models.

- `caTools` - contains various utility functions, in particular, moving window statistic functions, e.g., running average, running standard deviation, running quantiles, etc.

Other packages used include: `stats` (basic R package containing functions for statistical calculations), `reshape` (transforming the data for the use with `ggplot2`), `FitARMA` (fast implementation of fitting ARMA/ARIMA time series), `outliers` (various methods and tests for outlier detection), `DMwR` (functions and tools for data mining detailed in [99]).

In order to use the statistical component we need an interface which allows to call R functions from a Java application. For this purpose we investigated the following alternatives:

- *JRI* [2] - Java/R Interface, uses Java Native Interface (JNI) to call R from Java. Currently JRI is shipped as a part of *rJava* interface [3] [101] that provides low-level bridge between R and Java, i.e., a mechanism to create objects, call methods and access Java objects from R.

- *RCaller* [4] is an open source (LGPL) library for calling R from Java. It transfers commands from Java to the R interpreter, and handles results as XML documents which are parsed using the standard Java API for XML processing.

- *RServe* [5] [100] acts like TCP/IP server that allows other programs to use R. It supports not only Java applications, but also other languages such as C, C++, PHP.

[1] `ggplot2.org`
[2] `http://rforge.net/JRI/`
[3] `http://www.rforge.net/rJava/`
[4] `http://code.google.com/p/rcaller/`
[5] `http://rforge.net/Rserve/`

- *Rsession* [6] provides a multi-session R engine giving access for Java classes to a remote or local R session.
- *Renjin* [7] is a JVM-interpreter for R. It calls R in Java by implementing the R interpreter from Java. It is intended to offer a good performance, be flexible and completely compatible with the original R interpreter. However, Renjin is still under development.

JRI and rJava require the precise setting of system before execution and their usage appears to be complicated, as they are designed as a very low-level interaction between Java and R. Renjin is currently under development, although it seems to be an interesting alternative, since the developers state that Renjin is designed to be the one of the quickest ways to access R from Java. Rsession capabilities are redundant for our purposes, one session of R is enough for the system. RCaller is the simplest way of using R inside a Java application, it does not require any complex configuration and easy to use. It functions in the following way:

- it prepares and stores the user input, which can be either Java array or object or R code;
- runs an external R process by executing Rscript;
- passes the generated code to Rscript and if needed receives the output as XML documents, if required;
- returned XML documents are parsed using SAX (Simple API for XML) and returned objects are extracted to Java arrays.

Therefore, we used RCaller, although during the implementation and testing we discovered that it is quite slow. We have tried to use Renjin as an alternative - JVM-based interpreter for R, designed to be one of the fastest solution to access R capabilities from Java. Judging on the amount of time it needs to load the data and to calculate basic statistical parameters, such as mean, standard deviation, quantiles, and correlations (see also Table 4.2), we conclude that using Renjin we can significantly accelerate a program execution.

Table 4.2: Comparison of execution time of RCaller and Renjin

Task	RCaller	Renjin
Loading data	5137 ms	2282 ms
Calculate statistical parameters	3381 ms	186 ms

Unfortunately, Renjin lacks a lot of R capabilities and has conflicts with additional packages for R which are written on C, C++ or Fortran, including packages needed for our purpose. For instance, it does not support `caTools` package with running window functions and `TSA` package with time series analysis methods, and also it has no graphical functions yet. Therefore,

[6] `http://code.google.com/p/rsession/`
[7] `http://www.renjin.org/`

we continued working with RCaller, although improving running time of the analysis is one of our goals for future work (see also Chapter 5).

Semantic Component

There is a wide choice of tools which allow working with ontologies from Java. They include:

- *OWL API* [8] [47] is an open source (LGPL) interface that allows to create and manipulate OWL ontologies. Additionally, it provides an interface for working with reasoners.
- *Apache Jena* [9] [20] is an open source framework for building Semantic Web and Linked Data applications. It includes the Jena Ontology API that allows working with OWL.
- *Protégé-OWL API* [10] is another open source Java library for loading, creating, manipulating OWL data models. It supports reasoning as well.

All these APIs are commonly used, however OWL API and the Jena Ontology API are a bit more wide-spread in comparison with Protégé-OWL API. For example, the Protégé ontology editor in version 4.x has OWL API included, whereas commercial editor TopBraid Composer[11] has Jena in its base. We chose to use Jena Ontology API as it is flexible, covers RDF and provides a reasoner interface.

4.2 Evaluation

In this section we evaluate the approach proposed in this master thesis and consider the following hypothesis:
Hypothesis: why the usage of the knowledge-based component and available domain information gives better results in detecting anomalies in sensor measurement data and increases measures of accuracy and precision.

We recall the fragment of real-world data used in Chapter 3 in the description of the statistical component (see Figure 3.9). Analysis of this data using our method gives the following results (see Figure 4.2):

- Sensor TC1 has a bad correlation with the other duplicates and also differs in values in more than 60% values. Moreover, there were detected oscillations in TC1. Based on that, we conclude that TC1 measurements are not reliable and TC1 is excluded from the analysis.
- Sensor TC6 has a good correlation with all other duplicates (excluding TC1), but it significantly differs in value from the other, we inform the user about it with the warning.

[8] `http://owlapi.sourceforge.net/`
[9] `http://jena.apache.org`
[10] `http://protege.stanford.edu/plugins/owl/api/`
[11] `http://www.topquadrant.com`

```
Sensors TC1, [TC2, TC3, TC4, TC5, TC6] are duplicates, but differ in values very significantly
Sensors TC6, [TC2, TC3, TC4, TC5] are duplicates, but differ in values very significantly
Calculating cross-correlations and autocorrelations for further analysis...
Duplicate sensors TC2, [TC3, TC4, TC5, TC6] have high cross-correlations (used threshold is 0.85).
Warning: sensor TC1 is not correlated well enough with its duplicates.

...

Determining exact periods...
Oscillations in TC1 data are detected during periods: 2010-05-23 00:00:00 -- 2010-05-23 00:24:00, 2010-05-23 11:54:00 -- 2010-05-23 23:59:00.

...

Sensor TC2 has 13 false positives: [305, 310, 311, 323, 357, 388, 599, 625, 640, 683, 704, 795, 1242]
Sensor TC3 has 11 false positives: [15, 325, 361, 388, 417, 663, 666, 681, 683, 704, 795]
Sensor TC4 has 12 false positives: [310, 311, 357, 367, 383, 388, 417, 599, 704, 1278, 1289, 1344]
Sensor TC5 has 10 false positives: [305, 311, 364, 599, 625, 640, 664, 666, 681, 683, 704]
Sensor TC6 has 15 false positives: [15, 310, 311, 323, 325, 360, 362, 365, 367, 384, 1243, 1279, 1288, 1290, 1344]
It took 6510 ms
```

Figure 4.1: Analysis of the fragment of real-world data

```
> df[700:707, ]
           timestamp    TC1     TC2     TC3     TC4     TC5    TC6
700 23.05.2010 11:39 46.299  45.099  45.200  45.299  44.799 55.599
701 23.05.2010 11:40 46.400  45.099  45.400  45.400  44.799 53.400
702 23.05.2010 11:41 46.799  45.700  45.799  45.900  45.299 52.500
703 23.05.2010 11:42 58.200  74.300  80.500  75.599  63.799 52.099
704 23.05.2010 11:43 57.299 153.800 158.399 164.000 138.600 51.799
705 23.05.2010 11:44 66.699 115.199 112.300 114.300 111.099 53.099
706 23.05.2010 11:45 61.299  82.099  81.199  81.900  79.800 51.400
707 23.05.2010 11:46 57.299  71.300  71.099  71.400  69.199 50.799
```

Figure 4.2: Outlier common for several duplicating sensors

- Sensors TC2, TC3, TC4, TC5 have high pairwise correlation with each other and also a number of common outliers. For instance, consider an outlier 704 common for all of them (see Figure 4.2 for a database fragment): clearly, there is a peak, but it is common for several duplicating sensors, therefore we conclude, that it is a turbine behavior.

We recall the evaluation metrics introduced in Chapter 2. Classifying sensor readings as oscillating and outlying points, we say that points are classified correctly, if an anomaly is detected as an anomaly (true positive), or a correct value is recognized as a correct value (true negative). On the other hand, classifications are errors if either an anomaly was classified as a normal value (false negative) or a correct value was ranked as an abnormality (false positive). We used these notions and performance metrics based on them (see Formulae (2.10)-(2.18) in Chapter 2) to validate the performance of our approach. However, Accuracy, Precision, Recall and F-measure are used the most frequently used measures to evaluate the performance of algorithms, and some other metrics can be calculated using them (see Chapter 2). Therefore, in this section we give resulting values of these four basic performance metrics.

For evaluation purposes we generated 200 synthetic datasets as sensor readings, introduced outliers and noise to data and ran our approach on it. Each generated dataset contains daily measurements from five sensors, where:

X1 - Inlet Pressure sensor, has duplicates X2 and X5,

X2 - Inlet Pressure sensor , has duplicates X1 and X5,

X3 - Compressor Pressure sensor, has duplicate X4,

X4 - Compressor Pressure sensor, has duplicate X3,

X5 - Inlet Pressure sensor , has duplicates X1 and X2,

We took 1/60 Hz as a frequency for each sensor, i.e., sensors conduct one measurement per minute and 1440 measurements per day.

We used *random walks* to simulate sensor measurements. Random walks are discrete processes, starting from one point and making a random step:

$$X_{i+1} = X_i + e_i, \tag{4.1}$$

where X_i and X_{i+1} are values of the random walk on time points i and $i + 1$, e_i is random step value. Random walks are frequently and successfully used to model and simulate different stochastic processes and time series data in economics [45], computer science [4], physics and biology [40] and other fields [32]. An example of a random walk is shown in Figure 4.3. Random walks can simulate sensor measurements in an ideal way, i.e, in reality sensor measurements are always affected if not with the data quality problems, such as sensor mistakes or data losses, then with the turbine behavior itself - malfunctions such as shutdowns, stops, and trips, changes of fuel or of operational mode, maintenance processes such as purges, or other processes and other events which happen with the appliance, affect sensor measurements, and real-world sensor measurement data is not that smooth as random walk (see Figure 4.4). Ideally, sensor measurements can be modeled as random walks inside some defined ideal ranges (which is of course possible to achieve generating random walk and applying shifts and stretches), e.g., burner tip sensor measurements in a range of perfect combustion temperature. That is why we decided to generate random walks and than randomly introduced anomalies we consider in the current thesis work: outliers and oscillations, to evaluate performance of our combined approach (statistics combined with the ontology knowledge) for their detection.

The process of the generation of simulation datasets consists of the following steps:

1. Create randomly a correlation matrix of dimension 5×5, where corresponding values for duplicating sensors in groups $\{X1, X2, X5\}$ and $\{X3, X4\}$ are necessarily set to be more than 0.8 to ensure their correlation and therefore be able to use duplicating sensors to validate anomalies.

2. Generated 5 random walks of the length 1440 for measurements of 5 above-defined sensors, which comply to the correlation matrix created on the previous step.

3. Randomly introduce a random amount of outliers that differ from neighboring values for at least 3 standard deviations.

4. Similarly, introduce random periods of oscillation that are not longer than 300 time units - we added this condition in order to avoid situations when the whole dataset is replaced by the oscillations. Oscillations are represented as Gaussian white noise.

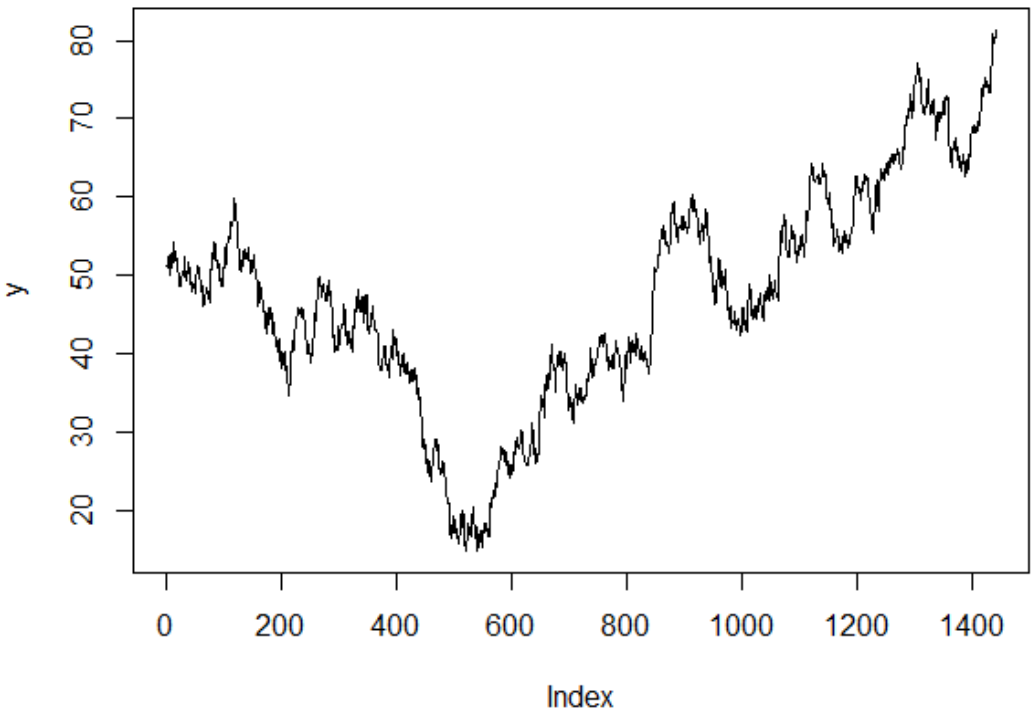


Figure 4.3: Random walk.

5. If for duplicating sensors some generated outliers or oscillations coincide, we do not consider them as anomalies. Additionally, for duplicates we introduce randomly several additional outliers happening at the same time to simulate anomalous turbine behavior detected by these sensors, and these outliers are not considered as anomalies as well.

Next, we introduce the following boolean vectors [57]:

InputOut	-	indicates the outliers introduced to the dataset;
InputOsc	-	indicates the oscillation introduced to the dataset;
OutputOut	-	indicates the outliers detected by our method;
OutputOsc	-	indicates the oscillation points detected by our method.

These vectors are used to count values of FP, PN, TN, TP:

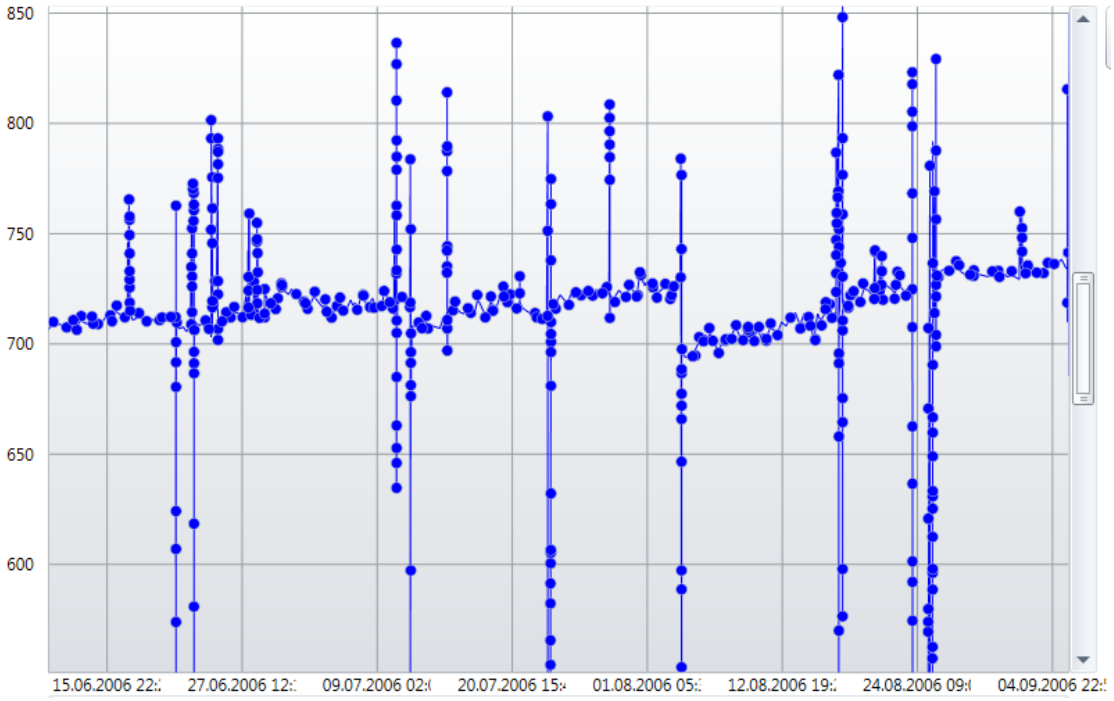


Figure 4.4: Real-world sensor measurements.

$$\text{TP} = \sum \text{Input} \,\&\, \text{Ouput}, \tag{4.2}$$

$$\text{FP} = \sum \neg\text{Input} \,\&\, \text{Output}, \tag{4.3}$$

$$\text{FN} = \sum \text{Input} \,\&\, \neg\text{Output}, \tag{4.4}$$

$$\text{TN} = \sum \neg\text{Input} \,\&\, \neg\text{Output}, \tag{4.5}$$

where Input and Output denote either InputOut and OutputOut for outliers or InputOsc and OutputOsc for oscillations correspondingly.

We ran tests on the generated data; cumulative results of algorithm evaluation on all 200 datasets are shown in Tables 4.3, 4.4: Table 4.3 shows total results of algorithm functioning for 200 datasets, and Table 4.4 shows calculations of performance metrics based of formulas introduced in Chapter 2 and values from Table 4.3. For outlier detection we compared efficiency of several methods and their variance: Hampel filter with different window sizes, outlier detection using ARIMA modeling, and the combination of the two (see Chapter 3 for details). Window sizes are chosen depending on the total number of observations n, in particularly 10%, 5%, 1%, 0.5%, and 0.2% of n. So, for our generated datasets $n = 1440$ for each sensor, and correspondingly window sizes are 144, 72, 36, 14, 7, 3 points. We started our tests from 5% window size and ran tests choosing smaller and bigger windows in order to determine the optimal k value for our case. Based on Table 4.4, the Hampel filter with $k = 7$ or $k = 0.005 \cdot n$ shows the

best efficiency among all. Moreover, as Tables 4.4 and 4.5 proof, the Hampel filter with $k = 7$ shows the best efficiency among all methods for outlier detection. Fitting ARIMA models and identifying outliers as points that do not fit well to the model is efficient enough, especially in comparison with the Hampel filter with bigger values of parameter k. Also, combination of results for ARIMA and the Hampel filter is not efficient enough, it has even worse performance measures than the Hampel filter and the ARIMA model alone.

Table 4.3: Summary for outlier and oscillation detection in synthetic datasets

	TP	FP	FN	TN
Outliers ARIMA	17491	4960	1021	1416528
Outliers Hampel (window size = 10 %)	14017	6236	5035	1414712
Outliers Hampel (window size = 5 %)	15151	3895	2418	1418536
Outliers Hampel (window size = 1 %)	16104	2714	1552	1419630
Outliers Hampel (window size = 0.5 %)	16913	2008	874	1420205
Outliers Hampel (window size = 0.2 %)	16858	2214	1129	1419799
Outliers ARIMA + Hampel	17228	2516	1141	1419115
Oscillations	120732	33337	31540	1255391

Table 4.4: Performance metrics for outliers and oscillations detection in synthetic datasets

	Accuracy	Recall	Precision	F-measure
Outliers ARIMA	0.9959	0.7791	0.9449	0.8540
Outliers Hampel (window size = 10 %)	0.9922	0.6941	0.7348	0.7139
Outliers Hampel (window size = 5 %)	0.9956	0.7883	0.8696	0.8270
Outliers Hampel (window size = 1 %)	0.9978	0.9086	0.9277	0.9181
Outliers Hampel (window size = 0.5 %)	0.9981	0.8987	0.9562	0.9266
Outliers Hampel (window size = 0.2 %)	0.9976	0.9372	0.8839	0.9098
Outliers ARIMA + Hampel	0.9954	0.7685	0.9691	0.8572
Oscillations	0.9550	0.7837	0.7929	0.7883

In order to demonstrate the benefit of our approach, we analyzed generated datasets only with the statistical methods, excluding interaction with the appliance module. In that case detected outlying and oscillating points cannot be validating using the semantic model, and therefore, some false positives can not be excluded. We recall, that in generated data anomalous points common for duplicating sensors are not considered as an outliers, because in that case in the real setting such measurements more likely indicate at the turbine anomalous behavior. How-

ever, statistical analysis alone cannot validate such values as correct ones, an a number of false positives increases, as we show also in Table 4.5. Additionally, as we have already mentioned before, it is important to distinguish between an anomaly as a corrupted data and as measurements taken during malfunctions of an appliance, which are important for a diagnostics, and the statistical methods alone can not provide the possibility of distinguishing these values, as the result in Table 4.5 confirm.

Table 4.5: Performance metrics for outliers and oscillations detection using statistical analysis only

	Accuracy	Recall	Precision	F-measure
Outliers ARIMA	0.9927	0.7552	0.8317	0.7916
Outliers Hampel (window size = 0.5 %)	0.9936	0.8258	0.7984	0.8119
Outliers ARIMA + Hampel	0.9939	0.7451	0.8152	0.7786
Oscillations	0.9373	0.7275	0.6940	0.7124

Therefore, we see that the with usage of existing domain information performance indexes increase:

- Accuracy shows the total amount of correctly classified values: it is defined as a ratio of true positives (correct values identified as correct) and true negatives (correctly identified anomalous data points) to the whole amount of values. When we apply statistical analysis only, these methods can not identify anomalies which we added there to simulate turbine uncommon behavior, and these data points are identified as data quality issues that have to be cleaned, although they are meant to be correct (but anomalous in terms of turbine diagnostics). In this case the amount of true negatives is smaller, whereas false positives - bigger, so the Accuracy measures increased applying our approach.
- Recall shows the completeness of the determined anomalies: it is defined as the ratio of correctly identified anomalies to overall amount of all anomalous points. Statistical methods detect outliers and oscillations well, and we apply our ontology knowledge to extract the information, with which sensors we can compare the current sensor measurements and to exclude false positives (if there are any) that simulate turbine behavior. For both methods recall measures do not differ as much as accuracy and precision, because the amount of true positives depend on the statistical method and its capability to detect outliers and oscillation periods well. False negatives represent errors when an anomaly was accepted as a correct value. This measure depends on statistical methods as well, if we consider only usage of duplicating sensor measurements from the domain. However, not high values of recall measure mean that the statistical methods have to be improved and they perform not as efficient, as we might expect them to.
- Precision shows how correctly the anomaly detection algorithm works: it is defined as a ratio of correctly identified anomalies to a sum of all identified anomalies, true and false

positives. As we mentioned above in the discussion on accuracy, statistical methods can not exclude anomalies simulating turbine uncommon behavior, therefore statistics classifies values, that are intended to be correct in terms of data quality, as anomalies. Therefore the false positives amount is thus significantly larger in case of applying statistical methods and, as a result, precision significantly drops.

- F-measure represents a harmonic mean between recall and precision and is intended to be a more general characteristic of anomaly detection, reflecting its completeness and exactness. Considering that the precision for our approach is bigger than for statistical methods, whereas recall does not differ much, F-measure is for our presented approach is bigger than the measure for statistical methods only, but differs not that much as precision.

However, the percentile of increasing of performance measures in pur approach is highly dependent on generated data and anomalies we introduce, and we introduced random amount of outliers randomly in the dataset. The key parameter is that was mostly influenced evaluating our approach is the false positive rate. The false positive rate in turn dependent on the amount of anomalies we introduced to simulate turbine malfunctions. If there are not many malfunctions introduced, false positive does not change much switching from our combined approach to statistical approach only, and vice versa. Therefore, percentile change for accuracy and precision improvement depends on such simulated malfunctions, and can not be predicted or assessed using random dataset generation. However, it is practically impossible to predict turbine behavior and the amount of its malfunctions (although there are methods for predictive maintenance in the industrial settings), in this case random introduction of such anomalies in our datasets is quite realistic.

Additionally, the approach was tested on several actual measurement dataset from turbines and all known inconsistencies in several datasets were successfully detected. In industrial setting the approach may be used at high-performance server for cleaning 24 hour batches of measurement data.

However, we consider the run time acceptable for a large task and big amount of data, although it is not suitable for streaming data, due to the implementation issues, discussed above in this chapter: the main source of slowness is the statistical part, as it performs a load of computations and R itself is an interpreted language and its programs run slower. The solution for that could be high performance and parallel computing with R and program optimization.

Thus, we see that consulting semantic model of the appliance gives us better results for data quality assessment and improvement in sensor readings.

CHAPTER 5

Conclusion and Future Work

Automatic processing of data for the purpose of determining operating states and identifying faults has become essential for many modern industrial systems. Typical sources for this data include a hundreds of sensors mounted at the device, measuring qualities such as temperatures, movement and vibration, pressure, and many more. However, these sensors are complex technical devices, which means that they can fail and their readings can become unreliable, or "dirty". Potential flaws in data include inliers (values within defined range but deviating largely from previous and following values), outliers (values exceeding defined thresholds), various types of missing data (visible or hidden by noise), oscillations, and many more. Such low quality data makes it hard to solve the original task of assessing system and process status and controlling the system behavior. Overall it considerably reduces reliability of the system and in particular invalidates system analysis results. To reduce cost of misguided decisions and the workload of human operators, it is therefore vital to have effective processes for assessing and improving the quality of sensor data in an automatic or semi-supervised way.

In the current thesis we investigate the topic of data quality assessment and improvement and present an approach that allows to efficiently remove data quality inconsistencies in measurements of industrial sensing devices. The main advantage of the approach is the construction of the comprehensive model of the application domain that covers the topology of the equipment and sensors installed on it. In particular, the application domain of the thesis work is power generation and sensors monitoring functioning of the power machinery. Our technique comprises of two main parts: (i) a statistical part which identifies and removes data anomalies in sensor readings using simple well-known statistical methods, and (ii) a semantic part that has an OWL ontology in its core and is used for extracting additional information on sensors and their settings for supporting and validating results provided by the statistical component. This information includes, for instance, specifications of measurement ranges capturing physical matter of functioning and expressed as data properties in the ontology. Fuel burning temperature slowly decreasing lower than combustion temperature can be detected as a trend in statistics, but considering the knowledge of temperature ranges low temperature is recognized as an anomaly. Moreover, in our application domain sensors mounted at the equipment are often duplicated,

and the information on such duplicated measuring devices contains in the ontology. In our approach we employ this knowledge to distinguish between data quality inconsistency, e.g., caused by sensor failure, and appliance functioning peculiarities which were observed by other devices as well. Moreover, faulty sensors continue producing measurements, often inaccurate and corrupted. Using the information on its duplicate devices and analyzing their correlation values and difference in readings helps to identify possibly malfunctioning sensors, which is not possible using the statistical component alone.

In the thesis work we devised a comprehensive model of the use case and represented it as an OWL 2 ontology. It comprises three modular ontologies, each capturing one aspect of the use case. One of the modules is dedicated to the power generation equipment, i.e. gas turbines, and contains information on their components and properties and functional role of each component. A sensor ontology covers information on various types of measuring devices installed on an appliance: sensors, their deployment in the appliance, their measurement properties and other information. The third ontology is dedicated to monitoring the functioning of the gas turbine and capturing the diagnostics knowledge. It details at which possible diagnosis certain events and anomalies in appliance functioning may indicate. Modular ontologies are connected together using the native mechanism of the OWL ontology language for imports. On the other hand, we selected several well-known statistical methods for analyzing measurement data. Both univariate and multivariate analysis are considered: for each single sensor we use univariate analysis to identify anomalous or missing data, whereas having domain information on duplicate and neighboring sensors we use correlation analysis and compare detected anomalies in data. It allows us to detect possibly faulty sensors, whose measurements continuously deviate from others, as well as to identify so called "false positives": readings of anomalous turbine behavior but classified as bad quality data points.

We implemented and evaluated our approach, and the evaluation showed that the usage of the additional information provided by the domain knowledge increases significantly the performance of outlier and noise detection in measurement data. We simulated sensor measurement data and introduced outliers and noise to these synthetic measurements. We have also simulated appliance malfunctions, i.e., for duplicating sensors introduced several coincident anomalies. After that we compared results of data quality assessment usage statistical methods only to detect outliers and detection and usage of statistics with the support of the domain information, groups duplicating sensors in particular. The results have shown that employment of the domain knowledge for data quality assessment decreases the amount of false positives significantly and thereby increases accuracy, precision and recall characteristics. The main reason is that statistics help to identify anomalous data points, i.e., measurements strongly deviating from the other measurements, but using statistical methods only we can not justify, whether this measurement is anomalous due to sensor failure or it reflects actual turbine behavior, e.g., shutdown. Therefore, anomalies introduced into readings of duplicating sensors to simulate turbine malfunctions are all classified as DQ inconsistencies by statistics, thereby increasing its rate of false positives (*FPRate*). On the other hand, employing knowledge on sensor groups from the ontology allows us to determine that these data points are not a DQ issue.

In the following we consider to develop and improve our approach significantly. We list several directions to advance work:

1. In this thesis work we concentrated on the topological aspect and data restrictions provided by the application domain. However, the domain knowledge contains much more information, and the ontologies designed during the work cover a lot of further information and provide possibilities for extending them, e.g., adding more module ontologies. The usage of further domain rules and restriction is the topic for the future work on the approach.

2. The other direction for an improvement of our approach is to further develop the statistical component. Applying more sophisticated statistical and machine learning techniques to the analysis of sensor measurements would open opportunities of the more comprehensive analysis of readings and more effective detection of anomalies and changes in data.

3. Further development of the framework is another direction of our work. As we showed in Chapter 4, the chosen API for the R language functions too slow and therefore analysis for a large amount of information would take a significant amount of time. Achieving small running time is one of our goals for a future work in order to be able to apply our data quality caccessing and cleaning approach to streaming data.

4. In this work the choice of the method for smoothing oscillations depended on the availability of duplicating or near located sensors: we used the regression model in the presence of duplicates and the ARIMA time series model otherwise. Developing the potential of choosing appropriate data analysis and correction methods based on reasoning results and information provided by the semantic component is one more aspect of the improvement of the proposed approach.

5. The current work concentrates mainly on the quality of sensor measurements, aiming to detect corrupted values, but the usage of more comprehensive data analysis techniques applied to sensor data and combined with semantic knowledge would lead to a strong and thorough fault diagnosis and condition monitoring tool, similar to the ICW-Wind system for wind turbines [58]. Developing such a tool is the main goal of the future development of the proposed approach.

APPENDIX A

Appendix

A.1 Additional Information to the Use Case

Data Quality Assessment

In Chapter 1 we have mentioned factors that badly influence the quality of a data. In the use case presented above in this chapter there also exist factors causing data deficiencies. It could be device failures (e.g., faulty sensor sending erroneous values, data collector or control unit failure), distortion during a data transfer, and other factors. As a consequence, one or more data quality attributes (see Chapter 2) stop being fulfilled. This section presents DQA results and analyzes identified quality insufficiencies. Particular examples of those insufficiencies are illustrated in Figures A.1 and A.2, which plot temperature values of thermocouple sensors mounted in the combustor chamber of a turbine: "Burner Temperature 1", "Burner Temperature 2", "Burner Temperature 3", "Burner Temperature 4", "Burner Temperature 5", "Burner Temperature 6". For the sake of brevity, we label them as "BSignal", "RSignal", "PSignal", "GSignal", "OSignal", and "SSignal" correspondingly, the choice of the first letter is justified by the color of visualization for sensor readings. Parts of the content in this section have already been published in [48]).

Completeness and Accessibility. Data loss is not uncommon in industry for several reasons. These reasons include the inability to access the required data: an appliance might be located in a remote region and the information is unavailable due to a bad or absent connection between the data collector in the unit and the main database. Another reason are device faults.

Depending on causes, there might be absent only some values and tuples or one type of data tables: "raw" or event data, and in that case it is still possible to make use of available information in order to conduct an analysis. For instance, estimation of the missing values using stochastic models or duplicating sensor measurements can be used to reconstruct sensor measurements. Also event data can be considered for approximate estimation: if at some moment of time there is a number of events reporting on a high temperature and fire in a component

of turbine, we conclude that during this period of time sensors located at this component had to measure temperatures exceeding some threshold. In a similar way event data can be reconstructed based on sensor measurements. The worst case is when no data (neither raw sensor measurements nor events) for a particular period is available at all. Figure A.1a illustrate data loss for sensor measurements during some period of time.

Consistent Representation. As it have also been mentioned in Chapter 2, in case when information come from multiple sources, there is a possibility that number of contraventions occur. In the current use case there are mostly syntactical anomalies, which concern a format of records. These violations include:

- Various recordings of timestamps, as date and time sometimes are written in several ways. For instance, devices of one kind write timestamps as *DD/MM/YYYY hh:mm:ss* while another have a format *YYYY-MM-DD hh:mm:ss* and many more of other types of devices having even other date and time formats. This discrepancy occurs due to (i) various device vendors and time settings, or (ii) location of appliances in countries with differing standard time and date notations.
- Data types of some information sources and monitoring devices require conversion from one format to another e.g., from `String` to `Float` or from `String` to `Integer`. It also depends on settings of devices that can differ with each other.
- Lack of standardization occurs due to differing indication of the same event by monitoring systems and control units. That happens due to diverse reasons such as various device vendors, different software versions, or location. For instance, some events have additional information in brackets indicating a sensor ID, which detected this event. Syntax of these events can also vary. Table A.1 illustrates varying indication of a sensing device for the same event.

Table A.1: Examples of different event denotations

Warning	`@TURBINE VIBRATION EQUIPMENT FAULT`
Warning	`@TURBINE VIBRATION EQUIPMENT FAULT [S1]`
Warning	`@TURBINE VIBRATION EQUIPMENT FAULT - S1`
Warning	`@TURBINE VIBRATION EQUIPMENT FAULT [S1]`

Correctness, Believability, Accuracy. Unfortunately, data is not always correct, precise and relevant due to diverse reasons: (i) one or several devices of the appliance faulted and gave inaccurate or wrong measurements; (ii) control unit failure occurred and there was an error during data preprocessing; (iii) poor connection distorted information during data transfer. There are three data transfer segments - from sensors to control unit, from control unit to data collector and between the appliance and data warehouse, and for each frequencies of data transfer and

speeds of data flow differ. As a consequence, the following insufficiencies may occur, which are also illustrated :

- Time synchronization: Timestamps of events and measurements coming from several different devices might slightly differ due to (i) time settings of a particular devices; or (ii) frequency and duration of data transfers between components, control unit, and data collector.

- Oscillations and noise: Signal measurements may be distorted and oscillate with high amplitudes. Furthermore, faulty sensor causes noisy data. Plots in Figures A.1b,c show oscillations and noise in a signal. Plot in Figure A.1c is an illustration of DQ deficiency of type 5. defined in Chapter 2: instead of correct values we observe arbitrary values. Thereafter, periods of noise such as illustrated in Figure 2 be considered as a period of data loss.

- Outliers: Occasionally sensor measurements contain values out of their domain. A particular instance is shown in Figure A.1d - sensor measures minus temperatures of fire in a combustor chamber, where the fuel is burned. Additionally, there are values occur, which belong to the domain but vastly differ from the others, i.e., sudden leaps and sudden changes of values within the domain. Figure A.2a shows a situation, when sensors alternate between showing range minimum and range maximum. Although these temperature values belong to the domain, this behavior is suspicious, as temperature can not change from 0° to 1300° and back that quickly. Referring to data deficiency types defined in Chapter 2, the example illustrated in Figure A.1d clearly is an instance of a gross error - values notably differ from correct values. Whereas an instance in A.2a may also be considered as an instance of disguised missing data, since measured temperature values are arbitrary.

- Vast difference in measurements of comparable sensors. If there are several sensing elements that duplicate each other, and they measure completely different values, then it is difficult to rely on these measurements. We give here several instances of such behavior. Figure A.2b shows that one of the sensors measures temperature up to 150° higher than its 5 duplicates. Figure A.2c illustrates another example: measurements of duplicating sensors have a very high correlation, have no vast difference between their values, but comparing their behavior we mention that when temperature leaps occur some sensors have much higher amplitude than others. Figure A.2d illustrates one more example: two duplicating sensors have difference in values up to 100° and suddenly after a turbine shutdown they their switched positions.

Ease of Manipulation, Data Schemes. Data schemas are highly heterogeneous, depending upon which technique was used to create them, which unit they belong to, or from where they come (historically). Moreover, not all foreign keys between databases are present. If information on the same entity is distributed among several sources (e.g., information concerning a particular malfunction of an appliance should be extracted from tables “Incident Summary”, “Daily Event

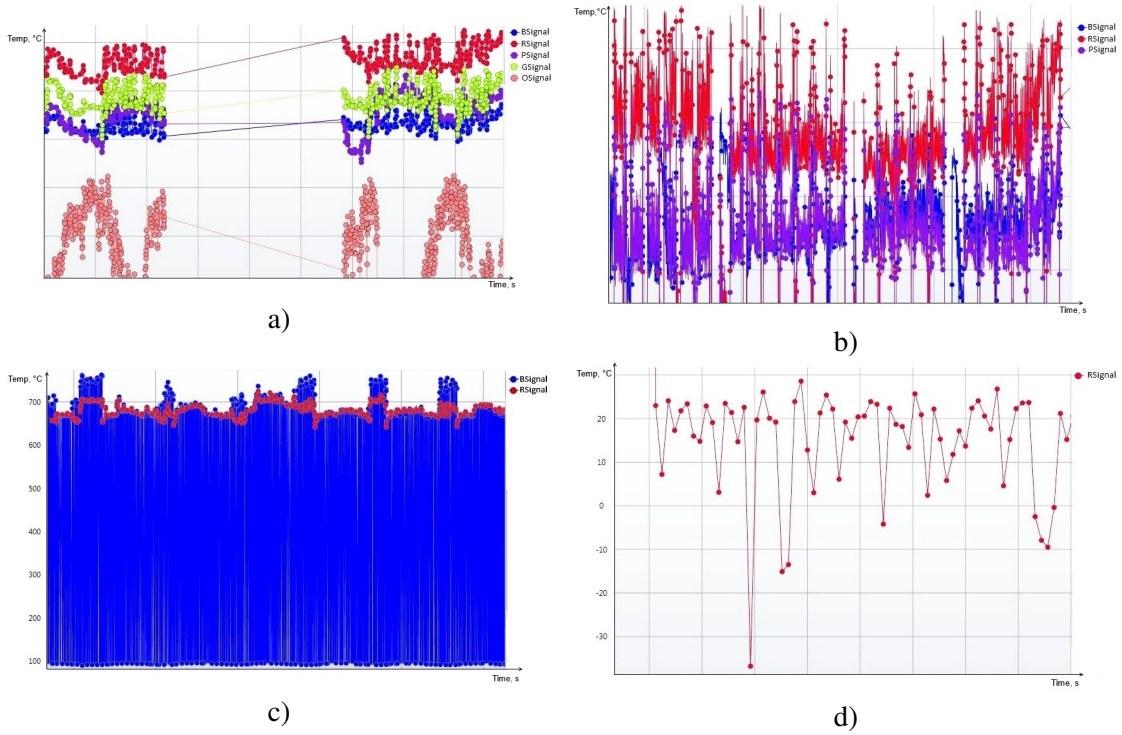


Figure A.1: a) Signal data loss; b) Oscillating signal; c) Noisy data - BSignal measurements look like white noise; d) Values out of range(minus temperatures);

Log", "Burner tip temperature" and others), the problem of missing foreign keys does not allow for easy merging of data.

Up-to-date Information, Appropriate Amount of Information. For each diagnostics case the considered time period always differs: it might be sufficient to consider only the last hour in order to identify a cause of an event, but in other cases one needs to analyze the last several years, for example to detect a deterioration of a particular component. Therefore, usually data does not become irrelevant in several years but has to be stored for decades.

Anonymization

In this work when we mention turbines, observations, sensors, and other above-described entities, we use certain representation for their labeling. This section clarifies notations and labels used in this work, and describes encoding procedure applied to the original Siemens data. Below we list turbines, categories, events and sensors labels that are used within the "Optique" project and therefore during the current thesis research.

Turbine Labels. The turbines are named by "Turbine1" to "Turbine10". In addition, some turbines are assigned a bird name and a number, e.g., "Falcon1", "Swan1".

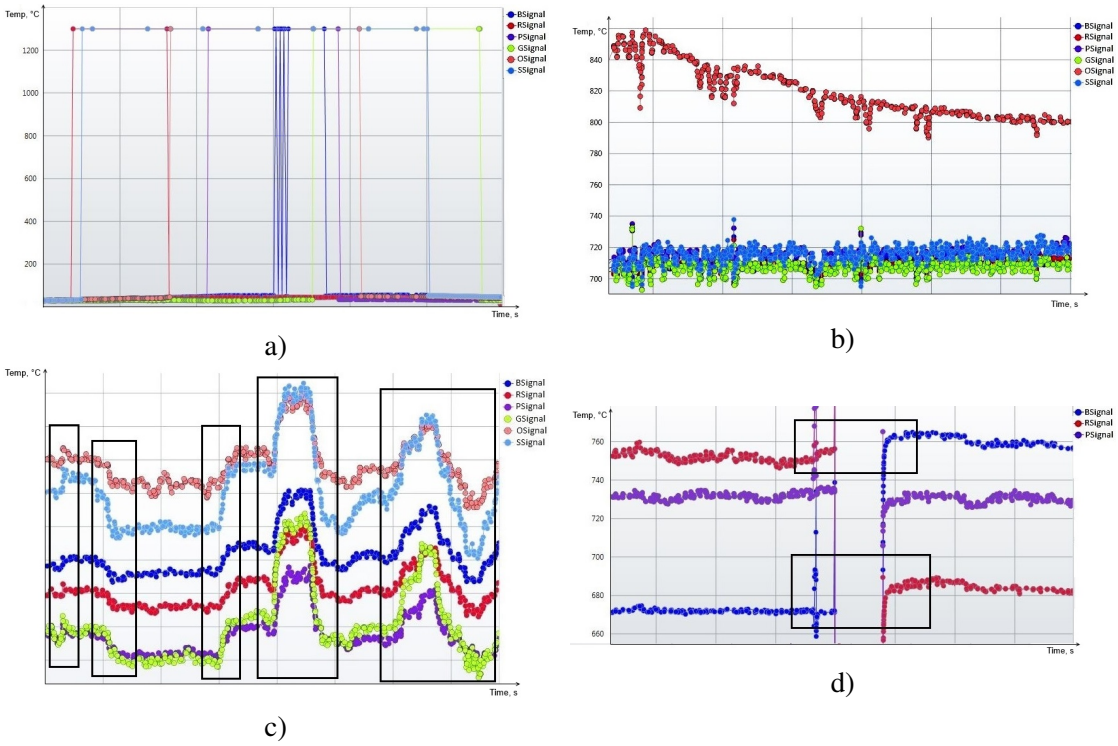


Figure A.2: a) Outliers. b) RSignal shows divergent values; c) SSignal and GSignal have rise/drop amplitudes differing from other duplicate sensors; d) BSignal and RSignal traded places.

Category Labels. Table A.2 provides descriptions for each anonymized category label used within the use case. Each event or message produced by measuring devices and control system is assigned a Category, indicating character and criticality of an information. *Warning* messages belong to Category6, *Error* reports are in Categories 3, 5, 9, 11, and messages from all other Categories report on further *Information*.

Event Labels. There are two different types of event labels:

- Event Type 1: event has no specific event description. In this case, event label is composed of the expression "Event" and a unique ID, e.g., *Event52*. The range of events of type 1 starts at "Event1" and ends at "Event1975".
- Event Type 2: event has a specific event description and denoted by a clarifying name. There are 34 events of this types divided into 3 subgroups:
  - fault events (indicating on devices faults, e.g., *"Boiler fault"*);
  - failure events (reporting on turbine failures, e.g., *"Can flame failure"*);
  - miscellaneous events (reporting on various critical messages, e.g., *"Valve not closed"*).

Table A.2: Category labels

Label	Description
Category1	Event from the control unit, indicating some general information about a status of a unit (e.g. start initiated, etc.)
Category2	Event from the control unit, indicating an operational status of a unit
Category3	Event from the control unit, indicating shutdown at any time (i.e. running, stopped, starting)
Category4	Event from the control unit, indicating, that some maintenance activity is taking place on a unit
Category5	Event generated by some tool, when a unit was stationary
Category6	Event from the control unit, indicating some unwanted situation as a warning
Category7	Event from the control unit, indicating operating status
Category8	Event from the control unit, indicating automatic prevention from a start attempt
Category9	Event generated by some tool to indicate running state of a unit at the beginning of the day
Category10	Event from the control unit, indicating an additional critical event
Category11	Event from the control unit, indicating automatic prevention from a start attempt due to some fault
Category12	Event from the control unit, indicating some general information about the control unit itself and possible problems with it (e.g. communication errors)
Category13	Event from the control unit, indicating an additional non-critical event
Category14	Event from the control unit, indicating a holding event

Sensor Labels. Sensors are labeled each by its measuring capabilities, e.g., *"Inlet Pressure"*. A few sensors with the same measuring capabilities are distinguished by sequence numbers e.g., *"Burner Temperature 1"*, *"Burner Temperature 2"*.

A.2 Detailed Ontology Schemas

This section contains detailed illustration of use case ontologies designed in the course of the current work and for further usage in the OBDA system (see Chapter 3). Figure A.3 shows the turbine ontology scheme providing topology of an appliance. It contains of three main classes:

- `Turbine`: a class of turbines, it contains main families of turbines. Additionally, arcs connecting `Turbine` class with subclasses of `FunctionalUnit` and `Component` define which components and parts the appliance necessarily has, e.g., `LubOilSystem`, `Generator` and `ControlSystem`.

- `FunctionalUnit`: a class of main functional parts of the appliance, i.e., its components having a certain role, e.g., `GasPath`, `LiquedFuelSystem` and `LubOilSystem`.
- `Component`: a class of turbine's components and parts, such as `CombustionCan`, `PressureTransmitter`, different valves (`GasSplitterValve`, `AutoDrainValve` and others), an other parts.

Components are connected with each other using relations `isPartOf` (its inverse `hasPart`), `isDirectPart` (its inverse `hasDirectPart`), and `hasFunctionalSuccessor` (its inverse `hasFunctionalPredecessor`). These relations define the partonomy of a turbine: hierarchy of components is defined with relations defining parts, whereas relations defining functional order define functional properties of the components, for example gas path through the turbine.

Figure A.4 illustrates the sensor ontology with measuring devices information. Its main class is `Sensor`, that first of all contains types of sensors as subclasses, e.g., `GasDetector` devices, sensors measuring `Temperature`, `Speed`, and others. Some of these type in turn have subclasses that give more specification on measurements they conduct. For instance, `Temperature` sensors may measure `FiringTemperature` or `ExhaustTemperature`; `Speed` sensors measure `GasGeneratorSpeed` or `PowerTurbineSpeed`. The relation `isMountedAt` connects sensors with components of turbine they are mounted at. Also, sensors might belong to one of the `SensorCluster`, so the relation `belongsTo` connects sensors and their clusters. Each sensor conducts measurements, so the relation `measures` connects a sensor with an `Information` it produces.

Figure A.5 shows ontology expressing the diagnostics knowledge. One of its classes is `Observation` that contains `Measurement`s provided by sensors and divided to categories of measured physical qualities, such as `Pressure`, `Temperature` and others. Additionally, `Observation` contains `Event` messages that in turn can be `Information`, `Error` or `Warning` messages of different `Categories`, that are detailed above in the Anonymization section of the Appendix. `Symptom` is the observation connected with the `Diagnosis` class with the arc `indicatesAtDiagnosis` (and its inverse). `Diagnosis` is assigned to a `System` with the arc `hasDiagnosis`, indicating that the appliance have a certain diagnosis of malfunctions.

The use case ontology connects all them with the OWL import mechanism and `EquivalentClass` relation. For instance, in the sensor ontology `Sensor` class is connected via `isMountedAt` relation with the class `SubDevice` which is equivalent to `Component` in the turbine ontology. Thus, individual sensors from the sensor ontology are linked to corresponding components in turbine ontology. Similarly, `SensingDevice` in the diagnostics ontology is equivalent to `Sensor` class in the sensor ontology to link observations made by sensors from the diagnostics with the individual sensor in the sensor ontology. More information on the ontology see Chapter 3.

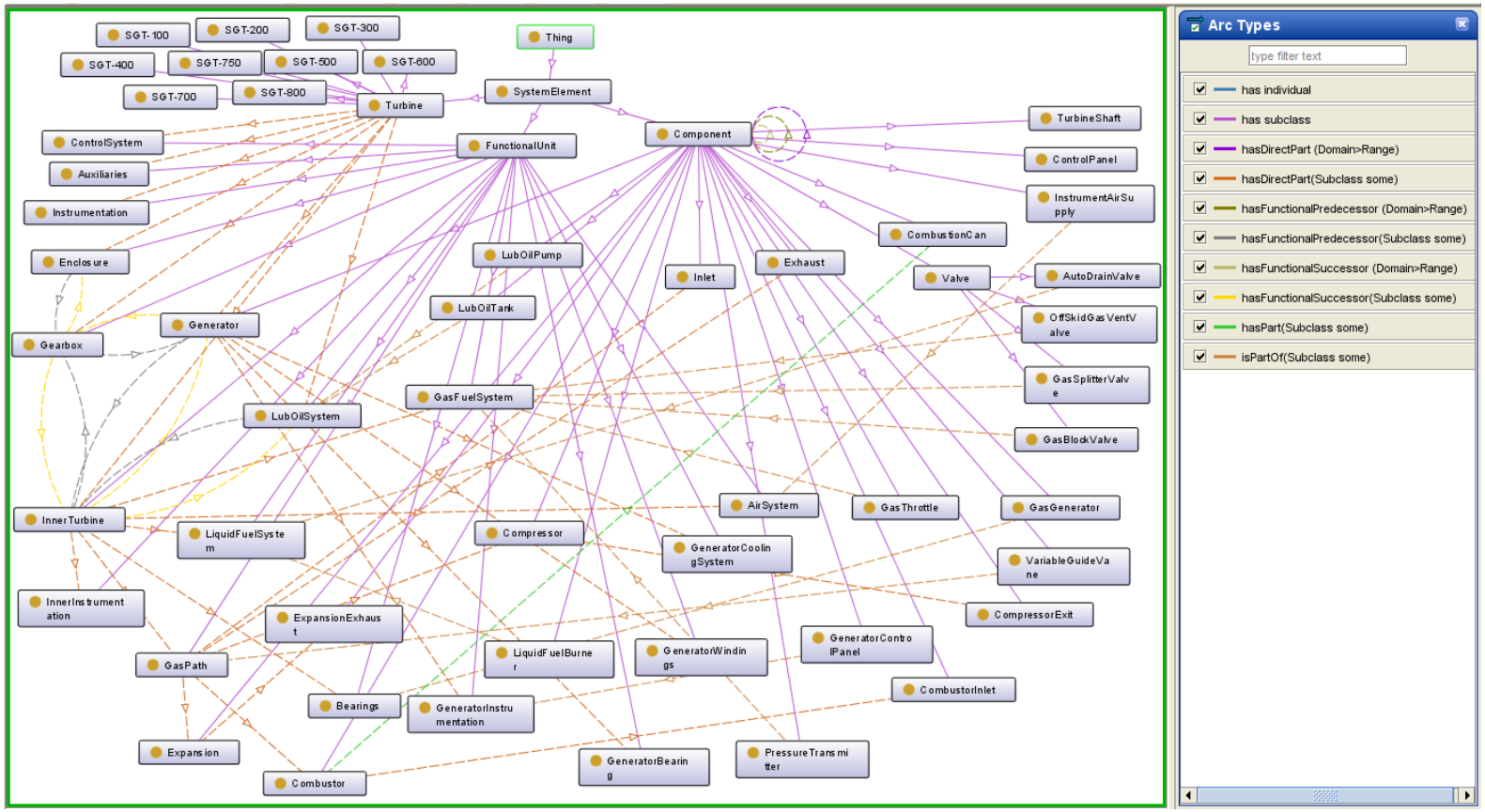


Figure A.3: Turbine Ontology.

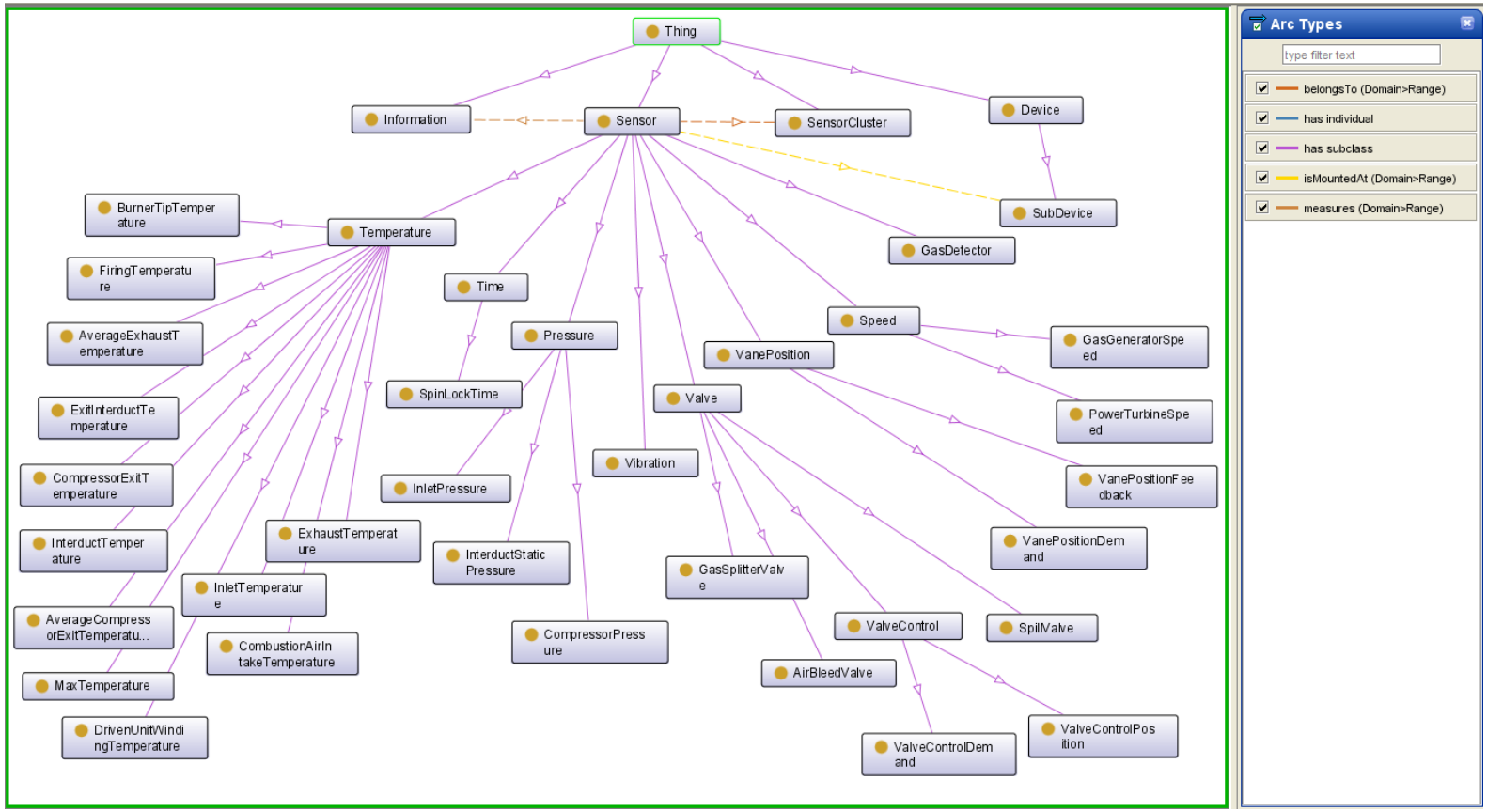


Figure A.4: Sensor Ontology.

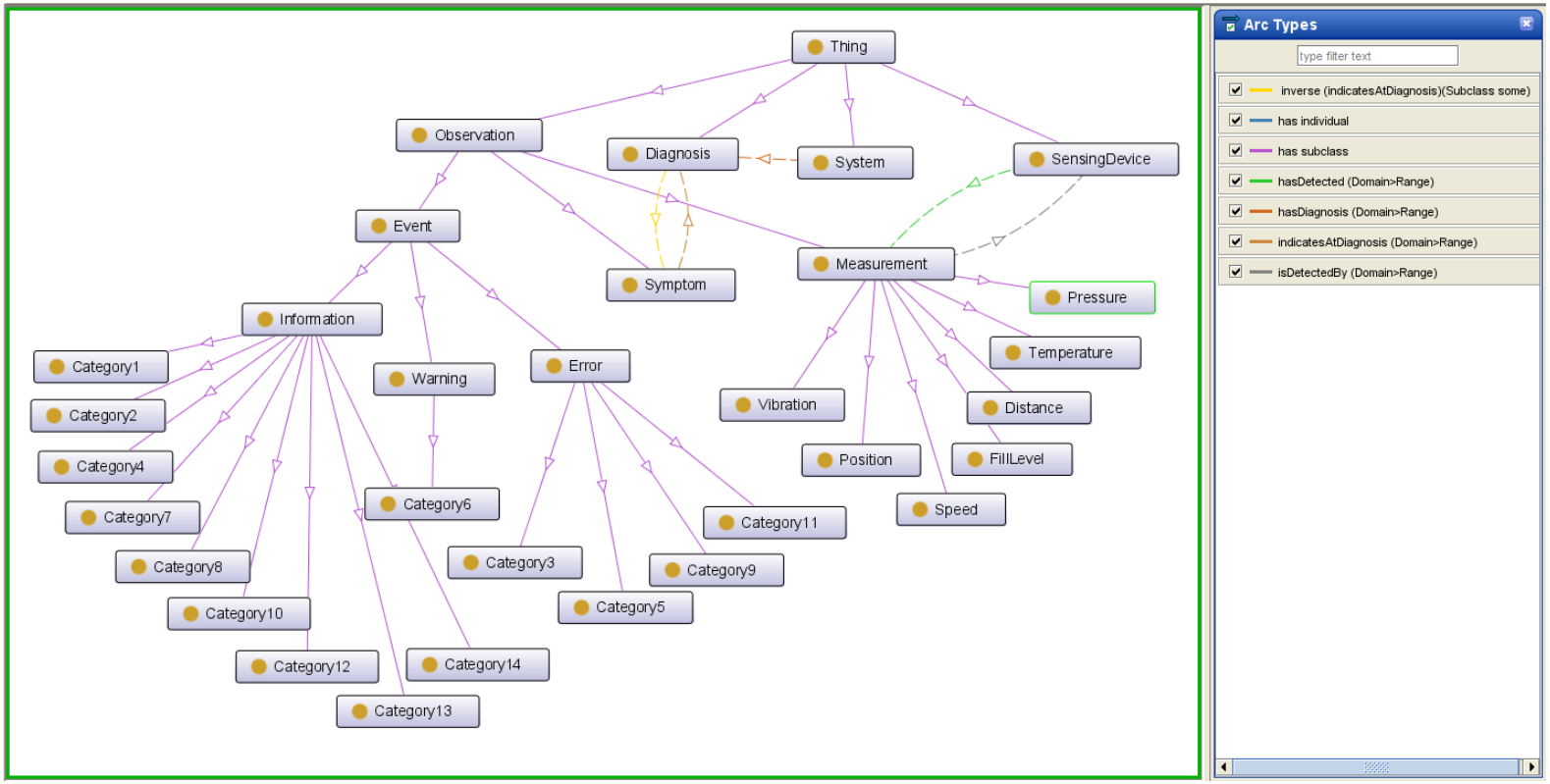


Figure A.5: Diagnostics Ontology.

List of Abbreviations

AI Artificial Intelligence

DBMS Database Management System

DC Data Cleaning

DL Description Logics

DOL Distributed Ontology Language

DQ Data Quality

DQA Data Quality Assessment

ETL Extracting-Transformation-Load

IRI Internationalized Resource Identifier

IQM Interquartile Mean

IQR Interquartile Range

JNI Java Native Interface

kNN k-Nearest Neighbors

KR Knowledge Representation

LOF Local Outlier Factor

MAD Median Absolute Deviation

OBDA Ontology-Based Data Access

OntoIOp Ontology Integration and Interoperability

OWL Web Ontology Language

SOM Self-Organizing Maps

SSN Semantic Sensor Network

SPARQL SPARQL Protocol And RDF Query Language

SQL Structured Query Language

SVM Support Vector Machines

RDF Resource Description Framework

RDBMS Relational Database Management System

RDBS Relational Database System

UML Unified Modeling Language

Bibliography

[1] ISO 19156:2011. *Geographic Information – Observations and measurements*. ISO, Geneva, Switzerland, 2011.

[2] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. The Association for Computing Machinery (ACM), 2001.

[3] Murray Aitkin, Brian Francis, John Hinde, and Ross Darnell. *Statistical Modelling in R*. Oxford University Press, 2009.

[4] Morteza Alamgir and Ulrike Von Luxburg. Multi-agent random walks for local clustering on graphs. In *2010 IEEE 10th International Conference on Data Mining (ICDM)*, pages 18–27. The Institute of Electrical and Electronics Engineers (IEEE), 2010.

[5] Fabrizio Angiulli and Fabio Fassetti. Detecting distance-based outliers in streams of data. In *Proceedings of the 16th ACM conference on Conference on information and knowledge management*, pages 811–820. The Association for Computing Machinery (ACM), 2007.

[6] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery*, volume 2431 of *Lecture Notes in Computer Science*, pages 15–27. Springer Berlin Heidelberg, 2002.

[7] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, 28(2):49–60, 1999.

[8] Daniel W Apley and Jianjun Shi. The GLRT for statistical process control of autocorrelated processes. *IIE Transactions from Institute of Industrial Engineers (IIE)*, 31(12):1123–1134, 1999.

[9] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. A linear method for deviation detection in large databases. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 164–169, 1996.

[10] Franz Baader. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2007.

[11] Carlo Batini and Monica Scannapieca. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[12] Stephen D Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 29–38. The Association for Computing Machinery (ACM), 2003.

[13] Irad Ben-Gal. Outlier detection. In *Data Mining and Knowledge Discovery Handbook*, pages 131–146. Springer, 2005.

[14] Elena Botoeva, Diego Calvanese, and Mariano Rodriguez-Muro. Expressive approximations in dl-lite ontologies. In *Artificial Intelligence: Methodology, Systems, and Applications*, pages 21–31. Springer, 2010.

[15] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, volume 29, pages 93–104. The Association for Computing Machinery (ACM), 2000.

[16] Peter J Brockwell and Richard A Davis. *Introduction to Time Series and Forecasting*. Taylor & Francis US, 2002.

[17] Robert Goodell Brown. *Smoothing, Forecasting and Prediction of Discrete Time Series*. Courier Dover Publications, 2004.

[18] Diego Calvanese. Knowledge representation and ontologies, lecture notes. Free University of Bozen-Bolzano, Last visited 25th March, 2013, from: `http://www.inf.unibz.it/~calvanese/teaching/11-12-kro/lecture-notes/`, 2012.

[19] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Ontology of integration and integration of ontologies. volume 49, pages 10–19, 2001.

[20] Jeremy J Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. The Association for Computing Machinery (ACM), 2004.

[21] Ih Chang, George C Tiao, and Chung Chen. Estimation of time series parameters in the presence of outliers. *Technometrics*, 30(2):193–204, 1988.

[22] William G Cochran. The distribution of the largest of a set of estimated variances as a fraction of their total. *Annals of Eugenics*, 11(1):47–52, 1941.

[23] Michael Compton, Payam Barnaghi, Luis Bermudez, Raul García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.

[24] Michael Compton, Cory A Henson, Holger Neuhaus, Laurent Lefort, and Amit P Sheth. Proceedings of the 2nd international semantic sensor networks workshop. In *SSN*, pages 17–32, 2009.

[25] Simon Cox et al. Observations and measurements - xml implementation. *Oper Geospatial Consortium International Standard*, 2011.

[26] Jonathan D Cryer and Kung-Sik Chan. *Time Series Analysis with Applications in R*. Springer, 2008.

[27] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. Nadeef: A commodity data cleaning system. pages 541–552. The Association for Computing Machinery (ACM), 2013.

[28] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[29] Karel Dejaeger, Bart Hamers, Jonas Poelmans, and Bart Baesens. A novel approach to the evaluation and improvement of data quality in the financial sector. In *Proceedings of the 15th International Conference on Information Quality*, 2010.

[30] Ethan W Dereszynski and Thomas G Dietterich. Probabilistic models for anomaly detection in remote sensor data streams. *Computing Research Repository (CoRR)*, abs/1206.5250, 2007.

[31] Floriana Di Pinto, Domenico Lembo, Maurizio Lenzerini, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Optimizing query rewriting in ontology-based data access. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 561–572. The Association for Computing Machinery (ACM), 2013.

[32] Jack F Douglas. Aspects and applications of the random walk. *Journal of Statistical Physics*, 79(1):497–500, 1995.

[33] James Durbin and Siem Jan Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, 2012.

[34] Eiman Elnahrawy and Badri Nath. Cleaning and querying noisy sensors. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 78–87. The Association for Computing Machinery (ACM), 2003.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[36] Thomas Foken, Mathias Göockede, Matthias Mauder, Larry Mahrt, Brian Amiro, and William Munger. Post-field data quality control. In *Handbook of Micrometeorology*, pages 181–208. Springer, 2005.

[37] Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. AJAX: an extensible data cleaning tool. volume 29, page 590. The Association for Computing Machinery (ACM), 2000.

[38] Michael S. Gendron and Marianne J D'Onofrio. Data quality in healthcare industry. *Data Quality*, 7(1):23–31, 2001.

[39] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of distance-based outliers in high-dimensional datasets. volume 16, pages 349–364. Springer, 2008.

[40] Narendra S Goel, Nira Richter-Dyn, et al. *Stochastic Models in Biology.* Academic Press, 1974.

[41] Peter Haase, Ian Horrocks, Dag Hovland, Thomas Hubauer, Ernesto Jimenez-Ruiz, Evgeny Kharlamov, Christoph Pinkel, Johan Klüwer, Riccardo Rosati, Valerio Santarelli, Ahmet Soylu, et al. Optique system: Towards ontology and mapping management in obda solutions. In *Proceedings of the 2nd International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM)*, page 21, 2013.

[42] Tore Hägglund. Industrial implementation of on-line performance monitoring tools. *Control Engineering Practice*, 13(11):1383–1390, 2005.

[43] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. Robust statistics: the approach based on influence functions. *Series in Probability and Mathematical Statistics*, 114, 2011.

[44] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[45] Thomas Hellström. A random walk through the stock market. *Licentiate Thesis, Department of Computing Science, Umeå University, Sweden*, 1998.

[46] David C Hoaglin, Frederick Mosteller, and John Wilder Tukey. *Understanding Robust and Exploratory Data Analysis*, volume 3. Wiley New York, 1983.

[47] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for working with OWL 2 ontologies. In *Proceedings of the 6th Workshop on OWL: Experiences and Directions.*, volume 529, pages 11–21, 2009.

[48] Thomas Hubauer, Steffen Lamparter, Mikhail Roshchin, Nina Solomakhina, and Stuart Watson. Analysis of data quality issues in real-world industrial data. In *Poster Presentation at the 2013 Annual Conference of the Prognostics and Health Management Society*, 2013.

[49] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[50] Shawn R Jeffery, Gustavo Alonso, Michael J Franklin, Wei Hong, and Jennifer Widom. Declarative support for sensor data cleaning. In *Proceedings of the 4th International Conference on Pervasive Computing*, pages 83–100. Springer, 2006.

[51] Nan Jiang and Zhiqiang Chen. Model-driven data cleaning for signal processing system in sensor networks. In *Proceedings of 2nd International Conference on Signal Processing Systems (ICSPS)*, volume 1, pages V1–237. the Institute of Electrical and Electronics Engineers (IEEE), 2010.

[52] Theodore Johnson, Ivy Kwok, and Raymond T Ng. Fast computation of 2-dimensional depth contours. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 224–228, 1998.

[53] Elsa M Jordaan and Guido F Smits. Robust outlier detection using svm regression. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 3, pages 2017–2022. IEEE, 2004.

[54] Beverly K Kahn, Diane M Strong, and Richard Y Wang. Information quality benchmarks: product and service performance. *Communications of the ACM*, 45(4):184–192, 2002.

[55] Alan F Karr, Ashish P Sanil, and David L Banks. Data quality: A statistical perspective. *Statistical Methodology*, 3(2):137–173, 2006.

[56] Evgeny Kharlamov, Ernesto Jiménez-Ruiz, Dmitriy Zheleznyakov, Dimitris Bilidas, Martin Giese, Peter Haase, Ian Horrocks, Herald Kllapi, Manolis Koubarakis, Özgür Özçep, et al. Optique: Towards obda systems for industry. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 125–140. Springer, 2013.

[57] Samson Sifael Kiware. Detection of outliers in time series data. Master's thesis, Marquette University, Wisconsin, the United States, 2010.

[58] Matthias Klusch, Ankush Meshram, Patrick Kapahnke, and Andreas Schuetze. ICM-Wind: Semantics-empowered fluid condition monitoring of wind turbines. to appear on the 29th Symposium On Applied Computing, March 24 - 28, 2014, Korea, 2014.

[59] Edwin M Knorr, Raymond T Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The International Journal on Very Large Data Bases*, 8(3-4):237–253, 2000.

[60] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Outlier detection techniques. In *Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2009.

[61] Hans-Peter Kriegel, Arthur Zimek, et al. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452. ACM, 2008.

[62] Jeremy Kubica and Andrew W Moore. Probabilistic noise identification and data cleaning. In *ICDM*, pages 131–138, 2003.

[63] Christoph Lange, Till Mossakowski, Oliver Kutz, Christian Galinski, Michael Grüninger, and Daniel Couto Vale. The distributed ontology language (DOL): Use cases, syntax, and extensibility. In *Proceedings of the 10th Terminology and Knowledge Engineering Conference*, pages 33–48, 2012.

[64] Lee S Langston, George Opdyke, and E Dykewood. Introduction to gas turbines for non-engineers. *Global Gas Turbine News*, 37(2), 1997.

[65] Mong Li Lee, Tok Wang Ling, and Wai Lup Low. IntelliClean: a knowledge-based intelligent data cleaner. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 290–294. ACM, 2000.

[66] Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[67] Xinong Li, Jiandong Wang, Biao Huang, and Sien Lu. The DCT-based oscillation detection method for a single time series. *Journal of Process Control*, 20(5):609–617, 2010.

[68] Alexander Maedche, Boris Motik, Nuno Silva, and Raphael Volz. Mafra - a mapping framework for distributed ontologies. In *Knowledge engineering and knowledge management: ontologies and the semantic web*, pages 235–250. Springer, 2002.

[69] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. Eracer: a database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 75–86. The Association for Computing Machinery (ACM), 2010.

[70] Deborah L McGuinness, Frank Van Harmelen, et al. OWL web ontology language overview. *The World Wide Web Consortium Recommendation*, 10(2004-03):10, 2004.

[71] Tina Miao and Dale E Seborg. Automatic detection of excessively oscillatory feedback control loops. In *International Conference on Control Applications, Hawaii, USA*, volume 37, page 38, 1999.

[72] Till Mossakowski, Christoph Lange, and Oliver Kutz. Three semantics for the core of the distributed ontology language. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 3027–3031. AAAI Press, 2013.

[73] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 web ontology language: Profiles. *The World Wide Web Consortium Recommendation*, 27:61, 2009.

[74] Rene Mueller, Gustavo Alonso, and Donald Kossmann. SwissQM: Next generation data processing in sensor networks. In *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research*, volume 7, pages 1–9, 2007.

[75] Heiko Müller and Johann-Christoph Freytag. *Problems, Methods, and Challenges in Comprehensive Data Cleansing*. Informatik-Berichte, Institut für Informatik, Humboldt Universität zu Berlin. Humboldt-Univ. zu Berlin, 2005.

[76] Alberto Muñoz and Jorge Muruzábal. Self-organizing maps for outlier detection. *Neurocomputing*, 18(1):33–60, 1998.

[77] S Muthukrishnan, Rahul Shah, and Jeffrey Scott Vitter. Mining deviants in time series data streams. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 41–50. The Institute of Electrical and Electronics Engineers (IEEE), 2004.

[78] The Seventh Framework Program (FP7) of the European Commission. Optique: Scalable end-user access to big data, project description. Last visited 15th March, 2013, from: `http://www.optique-project.eu/about-optique/about-optique/`.

[79] David L Olson and Dursun Delen. *Advanced Data Mining Techniques*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[80] GIS Open. Sensor model language (sensorml) implementation specification. *OGC® Open Geospatial Consortium*, 2007.

[81] Jeff Z Pan and Edward Thomas. Approximating owl-dl ontologies. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1434. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[82] Ronald K Pearson. Outliers in process modeling and identification. *Control Systems Technology, IEEE Transactions on*, 10(1):55–63, 2002.

[83] Ronald K Pearson. *Mining Imperfect Data: Dealing with Contamination and Incomplete Records*. Siam, 2005.

[84] Alfredo Petrosino and Antonino Staiano. A neuro-fuzzy approach for sensor network data cleaning. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 140–147. Springer, 2007.

[85] Leo L Pipino, Yang W Lee, and Richard Y Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[86] Yuzhong Qu and Zhiqiang Gao. Interpreting distributed ontologies. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 270–271. The Association for Computing Machinery (ACM), 2004.

[87] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[88] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *Proceedings of the 27th International Conference on Very Large Data Bases*, volume 1, pages 381–390, 2001.

[89] Ida Ruts and Peter J Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, 23(1):153–168, 1996.

[90] Dana Gelb Safran, Mark Kosinski, Alvin R Tarlov, William H Rogers, Deborah A Taira, Naomi Lieberman, and John E Ware. The primary care assessment survey: Tests of data quality and measurement performance. *Medical care*, 36(5):728–739, 1998.

[91] Kai-Uwe Sattler, Stefan Conrad, and Gunter Saake. Adding conflict resolution features to a query language for database federations. In *Proceedings of the 3rd Workshop on Engineering Federated Information (Database) Systems*, pages 41–52, 2000.

[92] Ranjit Singh and Kawaljeet Singh. A descriptive classification of causes of data quality problems in data warehousing. *International Journal of Computer Science Issues*, 7(3):41–50, 2010.

[93] EPA Quality Staff. *Data Quality Assessment: Statistical Methods for Practitioners*. Environmental Protection Agency, 2006.

[94] M Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Springer, 2011.

[95] Yee Lin Tan, Vivek Sehgal, and Hamid Haidarian Shahri. Sensoclean: Handling noisy and incomplete data in sensor networks using modeling. Technical report, Technical report, University of Maryland, The United States, 2005.

[96] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining*, pages 535–548. Springer, 2002.

[97] R Developement Core Team et al. R: A language and environment for statistical computing. *R foundation for Statistical Computing*, 2005.

[98] NF Thornhill, B Huang, and H Zhang. Detection of multiple oscillations in control loops. *Journal of Process Control*, 13(1):91–100, 2003.

[99] Luis Torgo. *Data Mining with R: Learning with Case Studies*. Chapman & Hall/CRC, 2010.

[100] Simon Urbanek. Rserve–a fast way to provide r functionality to applications. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), ISSN 1609-395X, EDS.: Kurt Hornik, Friedrich Leisch & Achim Zeileis, 2003 (http://rosuda.org/rserve)*. Citeseer, 2003.

[101] Simon Urbanek. rjava: Low-level r to java interface. *R package version 0.5-1, URL http://www. rforge. net/rJava*, 2007.

[102] Jan Van den Broeck and Lars Thore Fadnes. Data cleaning. In *Epidemiology: Principles and Practical Guidelines*, pages 389–399. Springer, 2013.

[103] Panos Vassiliadis, Zografoula Vagena, Spiros Skiadopoulos, Nikos Karayannidis, and Timos Sellis. Arktos: A tool for data cleaning and transformation in data warehouse environments. *IEEE Data Engineering Bulletin*, 23(4):42–47, 2000.

[104] Panos Vassiliadis, Zografoula Vagena, Spiros Skiadopoulos, Nikos Karayannidis, and Timos Sellis. Arktos: Towards the modeling, design, control and execution of etl processes. *Information Systems*, 26(8):537–561, 2001.

[105] Yair Wand and Richard Y Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, 1996.

[106] Melanie Weis and Ioana Manolescu. Declarative xml data cleaning with xclean. In *Advanced Information Systems Engineering*, pages 96–110. Springer, 2007.