A concept for a modular sensor-based Predictive Maintenance system for Industry 4.0

Bachelorarbeit
Tsvetelina Milusheva
Betreuer: Sebastian Kauschke

TECHNISCHE
UNIVERSITÄT
DARMSTADT

A concept for a modular sensor-based Predictive Maintenance system for Industry 4.0

Vorgelegte Bachelor-Thesis von Tsvetelina Milusheva

1. Gutachten: Prof. Dr. Max Mühlhäuser
2. Gutachten: Sebastian Kauschke

Tag der Einreichung: 08.08.2017

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt
Hiermit versichere ich, Tsvetelina Milusheva, die vorliegende Master-Thesis / Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden. Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Datum: Unterschrift:

_____ _____

Abstract

This bachelor thesis proposes a general purpose predictive maintenance (PdM) system which is based on real-time sensor data and can evaluate the state of industrial appliances and machinery. Detecting potential failures can, therefore, decrease unpredicted downtime. The proposed architecture for a PdM system is a modular solution that can be applied to any system regardless of the use case. This is achieved through an evaluation of already existing PdM systems and gathering of non-use-case-dependent requirements. This thesis also evaluates possible technologies that can be used in the implementation of the PdM architecture and proposes an optimal solution for its implementation. Since sensors tend to generate a large amount of data the Hadoop ecosystem was chosen as the most suitable big data solution.

Contents

1. Introduction

Predictive Maintenance (PdM) has established itself as an invaluable tool for reducing the cost of unplanned failures in many industries. Through constant observation of machinery and appliances usually by sensors monitoring different components or environmental variables, predictive maintenance allows us to detect first symptoms of degradation or loss in efficiency.
The basic function of a predictive maintenance system is to evaluate the state of a system based on its behavior and predict when and which component could be faulty. Based on this evaluation, the PdM system or a maintenance expert can then schedule a replacement for said part with the consideration that the ideal most cost-effective time for a replacement is right before the failure.
Predictive maintenance is an improvement of the classical run-to-failure maintenance, where a part is only replaced when faulty. This could cause unpredicted down times, stop production and severely increase the cost since unplanned maintenance tends to be up to three times more expensive. [20]
An alternative is using preventative maintenance, where maintenance is scheduled based on the mean-time-to-failure statistic for all machines of a particular classification. [20] Mean-time-to-failure indicates the probability for failure as time passes after the initial installation. Mean-time-to-failure has the form of a bathtub curve, which indicates the two potential times, when the likelihood for a fault to occur are particularly high are right after installation and after a certain period of operation time. In preventative maintenance, parts should be replaced before the second increase in the probability for faults. This, however, implies that perfectly functional parts would be exchanged, which is not a cost-effective solution and a problem that predictive maintenance solves by adequately evaluating the state of a system and only performing maintenance when necessary. [20]
An important aspect when it comes to the core functionality of a PdM system are the prognostic and diagnostic functions. The prognostic function handles the question of when a system would experience a performance degradation beyond the threshold that is acceptable and the diagnostic function asks why such a fault has occurred and aims to find the root cause or responsible component behind it.
The data from the observed machinery can be gathered in several ways. The simplest is by adding sensors to the system based on what variables need to be observed. The methods vary depending on the observed system and can cover electricity consumption analysis, thermography, ultrasonics, tribology along with environmental variables like temperature and humidity. The choice of observed variables is critical when designing such a system and also determines what sensors are needed.
Industry 4.0 represents the concept of a next generation manufacturing where machines are interconnected in order to make use of the advancements in information analytics. This next industrial revolution is expected to be triggered by the Internet, which facilitates connecting machines through large networks into Cyber-Physical-Systems (CPS). The integration of software into the industrial process can aid in an autonomous optimization and management of product service needs. Smart factories are nowadays focused on improving machine performance through interaction with surrounding systems [24]. PdM is a key concept in that process.
Machine learning is another important concept as it is being increasingly often used in PdM systems. Machine learning is the process of gathering knowledge from data through training with examples. In the field of PdM, there is an increasingly large amount of data that can be gathered. Machine learning provides a method for generating information from it. This allows systems to improve with experience and thus refine models for predicting outcomes of questions based on gathered knowledge. There are two types of algorithms in machine learning: supervised and unsupervised learning. Supervised learning algorithms work with labeled training data. Every example of training data has

corresponding input and output object. A form of supervised learning, for example, is classification. [19]
In unsupervised learning the algorithm is supposed to find a pattern in unlabeled data. In this type of machine learning, there isn't a predetermined right or wrong outcome. The goal is instead to find what patterns are being discovered. It is used for example in data clustering. [19]
Which type of machine learning is appropriate is decided by the output that should be produced. Machine learning is used in practice for spam detection, in stock trading, robotics, and e-commerce. [19]
The remainder of this thesis is organized as follows. Section 2 reviews related research on the topic and Section 3 provides a classification of the different types of PdM systems. Section 4 describes the requirements for a generic centralized PdM system. Section 5 describes the proposed architecture that implements those requirements. Section 6 gives an overview of possible technologies for implementing that architecture and evaluates which are most suitable for the use case. Finally, Section 7 suggests future work and Section 8 presents conclusions.

2. Related Work

This section introduces related projects in the field of PdM. It gives an overview of how PdM can be implemented in different industries and how the specific use cases influence the design of the maintenance system. the architecture and components of the maintenance system.

2.1. PreData

This thesis is written in collaboration with Accso GmbH as part of the PreData project. The goal of the project was creating a prototype of a big data PdM system for Industry 4.0 that is implemented for the use case of monitoring a refrigerator. The prototype used Estimote [30] and BlueUp [31] sensor beacons for monitoring key variables. A RaspberryPi is used as a gateway for communicating with the server. The prototype uses Hadoop along with HBase and OpenTSDB for data storage. This thesis focuses on providing a reference architecture for future projects in the field of PdM. More information can be found in [29].

2.2. DB Cargo

DB Cargo uses predictive maintenance to constantly monitor the state of their locomotives. [23] The data is then compared to data gathered immediately before a fault occurred. Within a Proof of Concept, the sensor data was analyzed in order to produce a heat map indicating the location of locomotives whose sensor data indicates a potential fault. The data is transmitted in fixed intervals, whose frequency can also be changed by the user.

2.3. SIMAP

The use of predictive maintenance for wind turbines is particularly convenient since most modern wind turbines already have sensors installed in them. One such system that relies on the use of these sensors is SIMAP [11].
It uses the gathered data to evaluate each component of the wind turbines in a Health Condition Assessment Module as well as scan for possible faults in an Anomaly Detection Module. Both of these functions rely on Normal Behavior Models created by artificial neural networks and the comparison between expected and actual values are how anomalies are detected, specifically by calculating normal behavior deviation degree and estimation certainty degree. SIMAP also offers a Diagnosis Expert Module that indicates the root cause of detected anomalies with the use of a fuzzy expert system that handles the uncertainty in such decision making. A Predictive Maintenance Scheduling Module is responsible for deciding when the maintenance actions should be scheduled. For this calculation, a fuzzy genetic algorithm is used since the task requires multi-objective large-scale dynamic decision-making with variable constraints. Finally, the performed maintenance is evaluated in the Maintenance Effectiveness Assessment Module by comparing the health condition of the exchanged elements before and after maintenance.
SIMAP is an example of how machine learning can be incorporated into PdM and gives an overview of the specific task that should be performed by different components of such a system.

2.4. Watchdog Agent

A system with similar prognostic and diagnostic functionality is the Watchdog Agent. It bases its predictions on trending and statistical modeling of the observed performance signatures that have been measured in the past and model parameters. The performance assessment module of the Watchdog Agent [28] is a modular open architecture toolbox. The algorithms in the toolbox are based on neural networks, time series, wavelet and hybrid joint time-frequency. The Sensory Processing and Feature Extraction Module gathers the sensor data and filters it down to the most significant to the performance information. A Quantitative Health Assessment Module evaluates the degree of similarity between the observed signatures and the established ones for normal behavior. This value is called Confidence Value (CV) in the range between 0 and 1 with 1 being the perfect normal behavior value. The Watchdog Agent uses the gathered historical data of occurred anomalies in order to improve its classification of failure modes in the future. The Performance Prediction Module analyses the behavior process signatures and creates predictions based on them with the help of Autoregressive Moving Average (ARMA) and match matrix methods. The Watchdog Agent also can evaluate which components are in good enough condition to be disassembled and reused.

2.5. Taleris

Taleris is a Joint Venture of GE Aviation and Accenture with the purpose of using big data analytics of sensor data gathered from more than 30 airlines in order to prevent any faults in the operating procedures on an airplane. [23] Such improvement in maintenance would lead to improved flight security, fewer delays and flight cancellations and ultimately better customer satisfaction. The developed Taleris Big Data Platform and Analytics solution consists of the following components:

- A real-time data transmission that is compressed due to the bandwidth constraints of the environment
- The acquisition, storing and processing of sensor data from airplanes and partner airlines
- Integration of maintenance, routing and planning data
- Deployment of Data Science and Services for Predictive Analytics, Intelligent Operations, Root Cause Analytics, Simulation and Intelligent Planning, all with the use of learning algorithms
- Visualization of the gathered findings for different user groups per Push or Pull over different channels and direct integration in the specific systems of the client airlines.

This solution allows for predictions of faults in certain vent systems with the accuracy of one week beforehand. The Taleris system allows for airlines to better optimize their wait cycles, save maintenance and replacement costs and reduce delays.

3. Classification of PdM

This section describes possible classifications of PdM systems according to different criteria. Each subsection includes what the challenges for this specific type are and what further design decisions need to be made in response to it.

3.1. Incoming data source

There can be made a distinction between PdM systems based on the source, from which they receive data. This is often determined by whether sensors are already part of the machinery that the PdM system monitors.

3.1.1. Data directly from the machine

The incorporation of sensors in modern machinery becoming commonplace is a great advantage when it comes to designing PdM systems. This means the maintenance system's scope starts at a higher level – the gathering and transmission to the components responsible for processing the data. Such incorporation of sensors is commonplace in wind turbines, where the sensors supply data for different controllers. That sensor data can also be incorporated in a PdM plan, which is the case for the SIMAP [11] system, which was introduced in the previous section.
Whether the sensor data is easily accessible, however, is an important distinction. It is possible that the observed system generates its own output from its sensors, for example in the form of warnings of malfunction. In this case, the exact cause or sensor measurement that set it off is unknown. This is the case in many modern household appliances like refrigerators and washing machines. Such a set-up would influence further design decisions like the functionality of the data analysis component. Without access to the original sensor data, the PdM system only works with very limited information. An alternative, in this case, would be incorporating additional sensors for monitoring.

3.1.2. External measurements

An important aspect when designing a PdM plan for a system that doesn't incorporate sensors is deciding which parameters are relevant to its maintenance and need to be monitored. There are a number of techniques that are commonplace for PdM systems like vibration monitoring, thermography, tribology, and ultrasonics. [20] Along with that it is possible to incorporate visual inspections by maintenance experts as a data source. The decision which PdM technique is sensible varies between use cases since different machines have specific signs of deterioration. In [34] the different methods for gathering data from machinery are introduced including through external, usually wireless sensors. Such sensors can also be an addition to integrated sensors in cases where they do not provide enough information for accurate assessment on their own. Additional benefits of wireless sensors are introduced in Subsection 4.5.2.

3.2. Amount of generated data

Depending on the amount of data generated by a PdM system there are a number of design decisions that need to be made with regards to the storage and transmission of data. In the next subsections, the different scenarios will be examined based on whether a big data solution is necessary.

3.2.1. Big data

The quality and quantity of data produced are steadily increasing with time and making it more difficult to manage with conventional solutions. This is solved through the introduction of big data tools like Hadoop. Having access to such quantities of data would mean that it can be used for analysis and machine learning purposes. Greater intelligence can be used in optimization for smart factories and cities through the introduction of prognostics and health management algorithms. [24] Two examples of big data systems were introduced in the related work section namely Taleris and DB Cargo.

3.2.2. Small data

There are still numerous cases though where the introduction of big data architectures would be unnecessary. In scenarios where only a limited amount of data is gathered in more contained use cases, a light-weight solution would be more suitable. This is the case in systems where only a small number of parameters need to be monitored. An example of such a system is SIMAP. This is also the case for smart buildings and in particular for energy monitoring as described in [37].

3.3. Type of observed system

There are a number of design decisions that are made depending on whether the monitored system is stationary or mobile including communication set-up, updating process and location of the data analysis.

3.3.1. Stationary

Smart factories and buildings fall under the umbrella of stationary systems and as such represent a more traditional use case for PdM systems. This scenario allows for more reliable communication between components.

3.3.2. Mobile

In the case of trains and airplanes, for example, a more distributed approach to the design of a PdM system is usually necessary. This means the anomaly detection modules need to be set up locally since a centralized module would usually have a too slow response time. This would also influence how often the system can be updated. Such a system is, for example, the Distributed Aircraft Maintenance Environment (DAME). [27] Since mobile systems usually require a distributed solution, further details are introduced in Subsection 4.4.1.

3.4. Location of the data analysis

Based on where the fault definitions are stored and where the data processing happens, PdM systems can be divided into two categories: systems that perform data analysis in a distributed manner and such that use a centralized predictive analytics component for that purpose.

3.4.1. Distributed

In this case transmission of large amounts of data is difficult, so the data processing is distributed to local subsystems. This is usually necessary for mobile systems where decision-making is time-sensitive and a delay caused by the data transmission to a centralized component would be too impractical. A distributed approach to data analysis is used for example in the case of DAME. [27] Another significant project is PROTEUS, which was financed by the Federal Ministry of Education and Research of Germany and the French Ministry of Economy, Finance and Industry. It provides an architecture for integrating subsystems that have the task of performing remote maintenance. [35]

3.4.2. In a centralized component

Implementing data processing in a centralized component has a number of advantages like a larger amount of available data and an easier to update system. This is usually implemented in stationary systems that generate large quantities of data. In such use cases, the introduction of cloud-based solutions would be suitable.

3.5. Topology

The topology for the PdM system is an important aspect as it influences the communication protocols that would be necessary

.

3.5.1. Centralized topology

In a traditional set-up for a PdM system, the sensors would either be integrated into the monitored machinery or the connection would resemble a star network topology, where all the sensors are connected to one central component but not to each other. This more classical topology would mean using simpler communication protocols and therefore easier development of the system that doesn't require highly specific expertise.

3.5.2. Wireless Sensor Networks

Wireless sensor networks (WSN) have many advantages over traditional cabled industrial monitoring systems namely self-organization, intelligent processing capability, and flexibility. [25] The rust, corrosion, steam, dirt, dust, and water in industrial environments can do possible damage to wires, which is not a concern for WSN. Furthermore, a cost of $2000 per foot for wiring can be saved since wireless sensors are more cost-effective ($20 per foot). [34]
In WSN sensors are installed on industrial equipment for the purpose of monitoring critical parameters. The sensor nodes are interconnected wirelessly and can transmit data between each other. These sensor nodes are connected to a sink node, which is responsible for analyzing the data from each sensor. In case that analysis detects abnormal measurements, plant personnel is notified.
There are, however, a number of challenges when developing a WSN namely the need for expertise in several different fields, specifically in the areas of sensor-technology, RF design, and propagation environment, networking as well as general industrial expertise. [25]
WSN are for example incorporated in aircraft health management systems. [26]

3.6. Anomaly detection strategies

This section looks at how the gathered data from the monitored system needs to be interpreted so that early signs of deterioration can be detected.

3.6.1. Predefined fault states

In the case of industrial machinery standard alarm levels can usually be gathered from industry standards (ISO for example), manufacturers or diagnostic system retailers. [21] What common faults that occur in different systems are also already known. For example [36] includes the specific problems that can occur in Heating, ventilation and air conditioning (HVAC) systems. This means that in the process of designing PdM the common signs of deterioration are already known. These standards, however, can be imprecise and are not suited to indicating a particular cause for those measurements. A specific diagnosis in those cases is gained through performing machine inspections. [21]
The standard alarm levels can be used as a basis for developing more precise maintenance systems. This can mean incorporating machine learning to discover more accurate deterioration signs.
Another main concern in the design process is determining the time interval until a failure occurs after those initial signs have been detected.

3.6.2. Unknown fault states

Discovering potential deterioration signs is a non-trivial task since it requires detecting anomalies in the gathered data. Anomaly detection is a type of artificial intelligence that can be useful in such a scenario. This would mean that a normal behavior needs to be defined through a model first. One approach is creating a probabilistic model. Another is using a threshold for acceptable values. A potential issue, in this case, is balancing the ratio between false positives and false negatives. This depends on which is more acceptable in the specific use case. A data structure that can be used in this case is t-digest, which is used for the estimation of extreme quantiles in large datasets. [22]

3.7. Overview

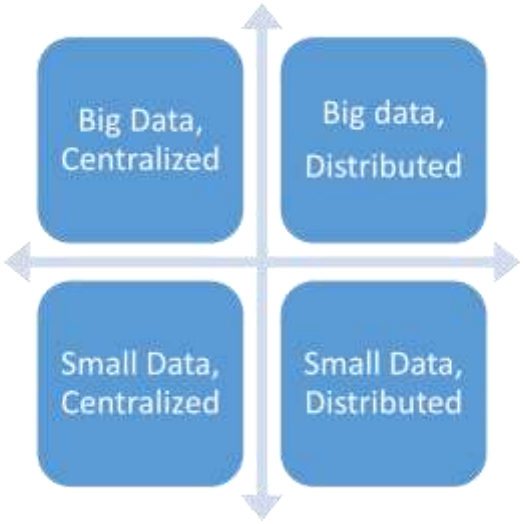

Figure 1: Classification of PdM systems.

Figure 1 represents a possible way of classifying PdM systems. The two main important characteristics that define the architecture of a PdM system are the size of the data that should be handled and whether or not the system is distributed. The distinction of whether a system is stationary or moving is relevant to the design as far as it determines a distributed or a centralized design. Predefined fault states influence what functionality would be necessary, specifically anomaly detection as a part of a machine learning module.

From the given examples SIMAP would fall under the category of centralized small data systems, while PROTEUS has a distributed small data architecture. Most PdM systems for trains and airplanes would fall under the distributed big data quadrant due to the amount of data they generate.

4. Requirements for a modular centralized PdM system

This section introduces the requirements for a generic PdM system that performs all its evaluation in a centralized component. Since they are intended for the definition of the problem domain of the reference architecture, the requirements are formulated for a non-specific scenario in order to ensure the applicability of the reference architecture in a larger number of cases.
The requirements describe the core functionality that a PdM system needs without specifying an use case. In this case we are describing a system where it is appropriate for the analysis of the data to be done by a central component. The distinction between the centralized and distributed manner of analysis was introduced in the previous section. The main functionality of a PdM system is predictive - based on the incoming data the system should be able to predict the time remaining until a potential failure. This would require an accurate overview of the machinery state, which should be available to maintenance experts and can be taken into account when scheduling maintenance. A PdM system should also have diagnostic functionality - since the sensors are monitoring different components, in the case of faulty behavior it should be indicated which component is the cause behind it. The system should allow for scalability through adding new sensors for monitoring new variables.
The system is intended for a generic user. This can include machinery or product owners or maintenance experts. Their goal is to have an accurate and up-to-date overview of the state of the machinery that is being observed by the PdM system. They should, therefore, have access to all gathered data along with scheduled maintenance activities and evaluations made by the PdM system. The user should also be able to report maintenance-related data. This can include found reasons for failures or repairs. They should also be able to report state updates, which are used as non-sensor gathered sources of information on the state of the observed system. This knowledge will then be used in the later evaluation of the system. The user should also be able to schedule maintenance. The classes of data stored in the database, as well as the writing rights to those types, are represented in Figure 2. The user is supposed to have reading rights to all data.
The requirements rely on variables such as t_1 for the allowed delay and t_2 for reaction time to anomalies. S_1 indicates the amount of data the system should be able to store. These values remain undefined for the reference architecture since they vary depending on the specific scenario, for which the PdM system is implemented.
An implementation of the proposed reference architecture would need to define these variables.
As the main purpose of a PdM system is a timely notification of degradation or faulty behavior the user needs to be notified of the remaining time until a failure occurs. Similar to the already introduced mean-time-to-failure for the purpose of the proposed PdM system a variable actualMeanTimeToFailure is introduced, which signifies a prediction based on the accurate assessment of the monitored machinery. Similarly one of the great advantages of PdM is having an overview of the performance and efficiency of the system, which is described by the variable lossInProcessEfficiency.

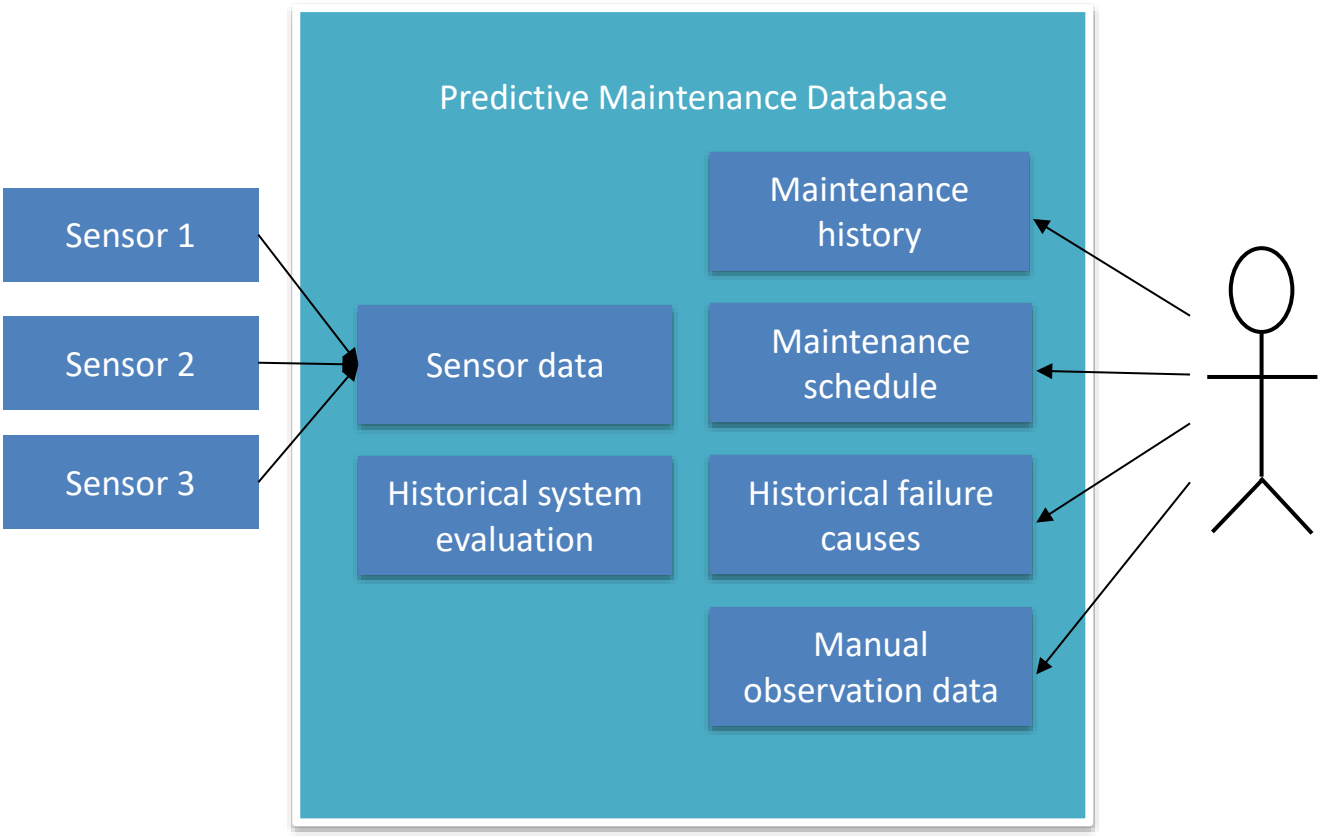

Figure 2: Writing rights to the PdM database.

4.1. Functional

The implementation of the PdM system should fulfill the following functional requirements.

1. **The user should be able to add or remove observed machinery.**

Since one of the possible applications for the PdM system is in the case of large factories, the user should be allowed to adjust the number of monitored machines.

2. **The user should be able to indicate the type of observed machinery.**

Since the fault state definitions are specific to a certain type of machinery, they only need to be defined once when such type is introduced to the observed system. This categorization is useful in later stages since known fault states would not have to be redefined.
Def.: We define the set of all machinery types as T and the set of all components as C.

3. **The integration of new type of machinery should include the definition of fault states.**

This should be performed by a maintenance expert during the set-up of the system or while extending the number of monitored machinery.

4. **The user should be able to add sensors monitoring new variables.**

As different types of systems require a specific set of sensors for their observation, it is necessary for the PdM system to allow the addition of any sensors regardless of the type of variable they are

monitoring. This implies that the storage and processing of the data should allow the addition of new types of monitoring data.

Def.: We define the set of all possible measurable parameters as $P = \{parameter_1, \ldots parameter_n\}$. For each parameter exists a unit u, and an interval I of permitted values [a, b]. Then a sensor measurement would be a set of a parameter $parameter_i$ where $1 \leq i \leq n$, the value x that was measured, a timestamp t and a $machineID \in N$:
$M := \{parameter_i, x, machineID, t\}$.

Such a measurement for example can be $\{temperature, 5, 0002143, 2016-11-28\ 07:27:35\}$. In this case we have temperature as $parameter_1$, 5 as the value that was measured, a $machineID$ – 0002143 and a timestamp 2016-11-28 07:27:35.

5. **The system should be compatible with any type of sensor.**

In order to facilitate the extensibility of the PdM system, its design should allow the adding of new sensors including such that monitor new variables. This also implied that the new data needs to be integrated into the evaluation process. In the case of new variables, this should be achieved through adjusting the failure definitions.

6. **The user should get updates on the values of the monitored variables.**

This visualization can be done in the form of graphs. The allowed delay between when the measurements are taken and the visualization is t_1.

7. **The user should get receive notifications of declining efficiency or detected anomalies.**

These notifications are the result of the real-time data analysis that has been done by other components of the system.

8. **The user should be able to update the fault state definitions.**

These notifications are the result of the real-time data analysis that has been done by other components of the system.

9. **The user should be able to report manual measurements.**

These manual measurements that can be used along with the sensor data to improve the analytical aspect of the PdM system.

10. **The system should filter invalid sensor data.**

It is critical that all calculations are performed with accurate data and therefore a filtering of noise and outliers should occur before that stage. We define the filtering function as

Def.: $filter: M \rightarrow M$

where M is the initial set of measurements. This function can be implemented through Kalman filtering for example.

11. **The system should detect anomalies in the newest sensor data received in the last t_2 seconds.**

Since anomalies in the sensor measurements can indicate possible deterioration of the observed system, their detection can be crucial in preventing failures. Therefore this requirement covers the basic predictive functionality of a PdM system.

12. **In the case of a detected anomaly, the system should calculate the probability of the potential reasons and indicate which parts are most likely to have caused it.**

This information can be used later during maintenance.

Def.: We define

$$\{\text{anomalyDetection} := (m_1, m_2, \dots m_n), F\} \rightarrow (c_i, p_i)$$

Where $m_1, m_2, \dots m_n \in M, F$ is the set of all fault state definitions and the result of the function is a tuple (c_i, p_i) of a component and the probability of it having a fault.

13. **The system should store a record of all maintenance that was performed.**

Information on replaced parts or what the causes behind regularly occurring anomalies can greatly improve the accuracy of both the diagnostic and prognostic functionality. In the case of irregular behavior the system should take into account that newer parts are less likely to cause faults.

14. **The system should be able to compute the value for the KPI actualMeanTimeToFailure.**

Evaluation based on the monitored parameters should be performed and the user should be notified of the remaining lifetime of the machine. The value for actualMeanTimeToFailure can be used in the scheduling of maintenance since the most optimal and cost-effective time for replacement of parts is right before a failure occurs. [1] The result of the actualMeanTimeToFailure function is a time and it is calculated based on the fault state definitions F and all historical measurements M:
actualMeanTimeToFailure: $F, M \rightarrow N$.

15. **The PdM system should be able to compute the value for the KPI lossInProcessEfficiency .**

An analysis of the observed behavior and use of resources of the monitored system should give an evaluation of its performance. For the purpose of this calculation, we use the definition of electrical efficiency

$$\text{Efficiency} = \text{Useful power output} / \text{Total power input} \quad (1)$$

Def.: The value for lossInProcessEfficiency is the ratio between efficiency and ideal performance:

$$lossInProcessEfficiency = Efficiency / Ideal\ Performance \quad (2)$$

16. **The system should store the original unfiltered data.**

This ensures that no data is lost and can later be restored and used. It also allows for improvements in the filtering function to be tested with original data.

17. **The system should store the definitions for fault states.**

Since all systems consist of a finite number of parts that can cause faults, we have a predetermined number of possible outputs. For all appliances and machinery, there exists a knowledge base on what the causes for most common faults are that the automation of this process can rely on.

The advantage of predictive maintenance in this case is the recognition of the reason behind a fault through the sensor observation. This means that specific sensor values can be connected with fault states. This valuable information is behind each definition. A fault state definition should, therefore, consist of a value interval for the sensor data, a part that is responsible for this observation and the probability of that exact fault occurring based on data from the other sensors, historical information and the state of all parts as reported by the maintenance expert including replacements.
The definitions must indicate what measurements for which parameters indicate a potential fault.

Def.: We define a fault state as a set of parameter1, … parametern ∈ P, a corresponding interval $[a_i, b_i]$ for 1 ≤ i ≤ n that the measurements need to be in to fulfill that fault state definition, machinery type t ∈ T, components $c_1, \dots c_m$ ∈ C and probabilities $p_1, \dots p_m \in [0, 1]$:

$$\{(\text{parameter}_1, a_1, \text{b}_1,), \dots (\text{parameter}_n, a_n, \text{b}_n), \text{t}, (c_1, \text{p}_1), \dots (c_m, \text{p}_m)\}$$

Such a fault state for example could be
$\{(\text{temperature}, 0, -\infty), \text{compressor refrigerator}, (\text{temperature control}, 30\%)\}$.
In this scenario, a measurement $(\text{temperature}, -5, 0002143, 2016-11-28\ 07{:}27{:}35)$ should trigger a notification that a fault state has been entered.
We indicate F as the set of all fault state definitions.

18. The system should update the fault state definitions based on historical data.
The gathered historical data on the behavior of the monitored system especially right before a failure occurs can be a great source for identifying patterns and improving the predictive functionality of the PdM system. This can potentially be achieved through machine learning.

Def.: The fault state optimizing function is

defUpdate: F → F.

4.2. Non-functional

Along with those, there are further non-functional requirements that must be fulfilled.

- **The system should require a solution that is not feasible with one machine.**

This requirement ensures the scalability of the system. As sensors can generate large amounts of data it is vital that the PdM system can process it. This is often not realistic when using a single machine for the implementation. This requirement, therefore, establishes scalability through the use of a number of machines.

- **The system should able to support the storage of S_1 GB of data.**

As sensors can generate large amounts of data it is vital that the PdM system can process it. Since the amount of data can vary based on the use case, the values should be set during the implementation.

- **The system should be designed in a way that allows for scalability as additional sensors are added.**

This ensures the lack of delays when the system is extended.

5. Proposed Architecture

5.1. Centralized architecture

This section introduces the reference architecture for a PdM system that performs all its analytical functions in a central component. This architecture is designed according to the requirements introduced in the previous section.
The architecture is introduced in three levels, starting at Level-1 with a black box representation of the PdM system and its interactions with the surrounding environment. Each next level goes into a more detailed white-box description of components – Level-2 describes the general architecture of the system while still having a black-box view of the Predictive analytics module, which is explained in detail in Level-3.

5.1.1. Level-1

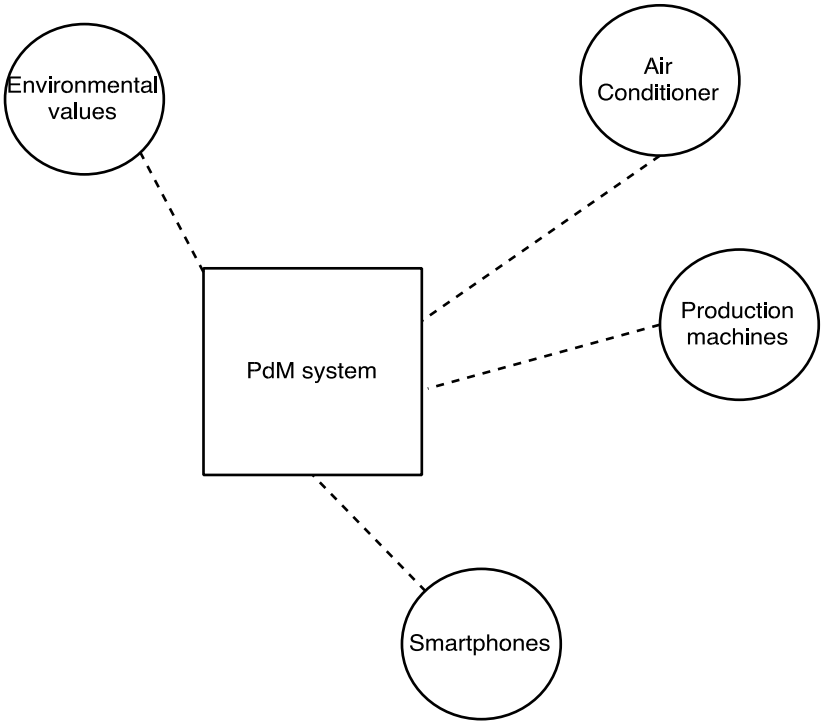

Figure 3: PdM System, Level-1.

We first consider the proposed system in the context of the Internet of Things. Figure 3 shows the interaction between the proposed systems and its environment. In Internet of Things (IoT) the concept of “things” can be any number of information sources like smartphones or factory machines with integrated sensors.
The system is supposed to interact with the sensors that are installed in the observed machines and gather information from them. At this level of representation, the PdM system is one entity interacting with the monitored system.

5.1.2. Level-2

The following UML diagram describes the components that the reference architecture consists of.

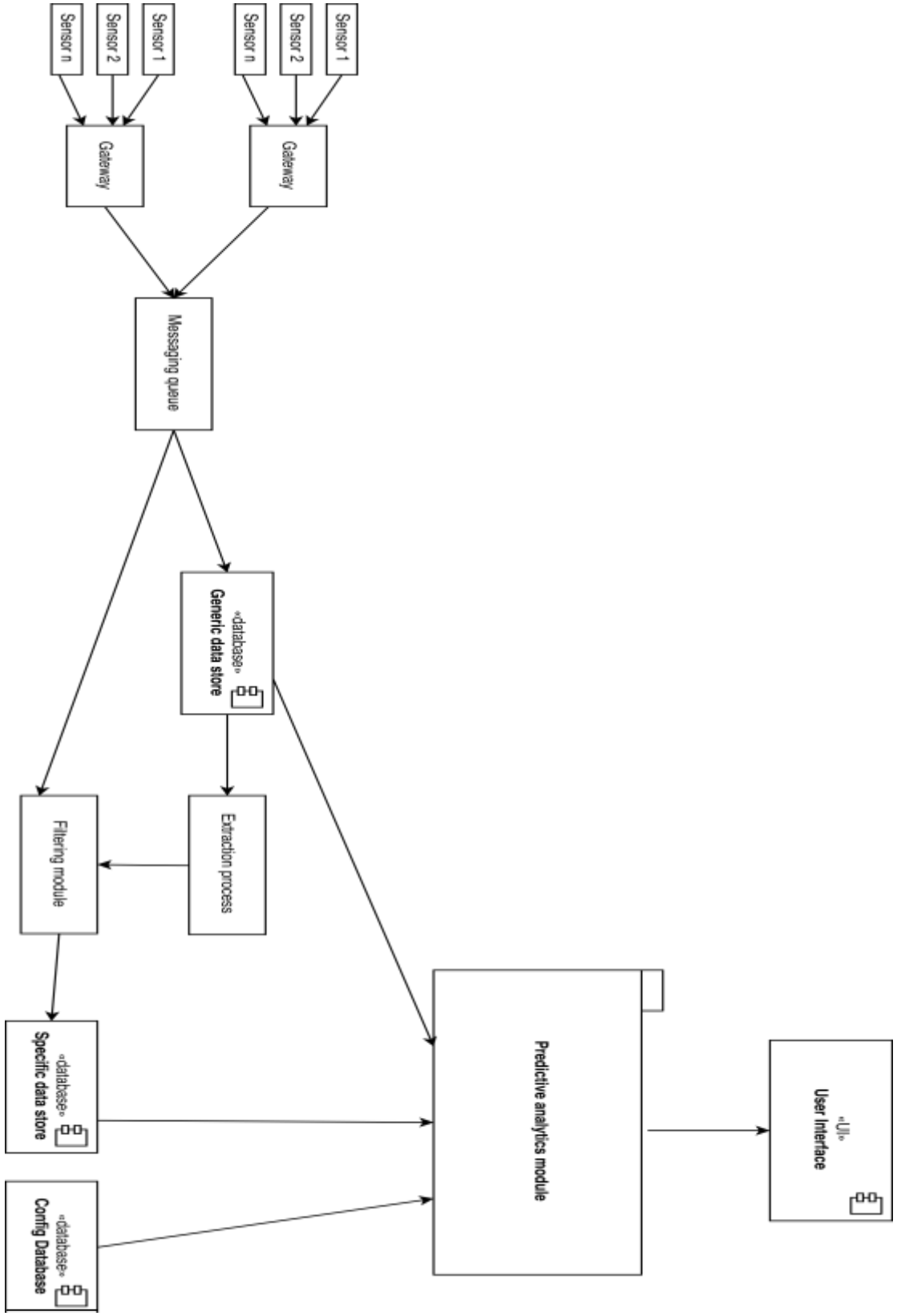

Figure 4 White-box representation of the data stores, Level-2.

The proposed architecture would have the following components:

- The sensors that gather data on the observed system are considered a black box since any implementation can require the measurement of different parameters. The sensors can vary from IoT devices to ones integrated into the machinery like the ones most modern wind turbines contain.
- The gateway is the component responsible for gathering the data from the neighbouring sensors and sending it to the messaging queue.
  Since the designed architecture is a modular one and therefore independent from the types of sensors used and the format they send updates we can only define the gateway as a black box. Its function is to receive those updates and send them to the messaging queue for further redistribution.
- The messaging queue receives the sensor data from the gateways and redistributes it to two channels. The first one is directly to the Filtering module for real-time analysis. The data is also sent to the Generic data store in an unfiltered state. This allows for the later use of machine learning on the original data.
- The proposed architecture would have three main data stores: a Generic Data Store, Specific Data Store and a Config Database. The Generic Data Store contains the gathered sensor data in its original unfiltered state as specified in Requirement 16. The Specific data store contains the data that has been cleared of noise in the Filtering module. The Config database contains the fault state definitions implementing requirement 17 as well as a record of all the machines that are under observation through the PdM system. It also contains data for configuring the gateways.
- The Filtering module has the purpose of clearing noise in the data so that these inaccuracies don't lead to false evaluations in the Predictive Analytics module. The filtered data is then stored in the Specific data store.
- The data meant for batch-analysis undergoes the Extraction process where the unstructured and unfiltered data is gathered for further analysis in the components of the Predictive Analytics module responsible for machine learning.
- The Predictive Analytics module contains all real-time and batch-analysis components that are responsible for the evaluation of the sensor data. The results are then displayed by the User Interface. A more in-depth view of this module is introduced in the next subsection.
- The UI component should provide the user with a visualization of the sensor data as well as the current values for actual-mean-time-to-failure and loss-in-process efficiency. In case those values fall below a certain threshold the user should receive a notification.
  The UI should also provide the user with options for adding or removing sensors and machinery, specifying or updating the fault definitions for a type of machinery as well as adding manual measurements. The UI component thus fulfills requirements 1-4 and 6-9.

5.1.3. Level-3: Predictive Analytics module

The predictive analytics module should have the following components:

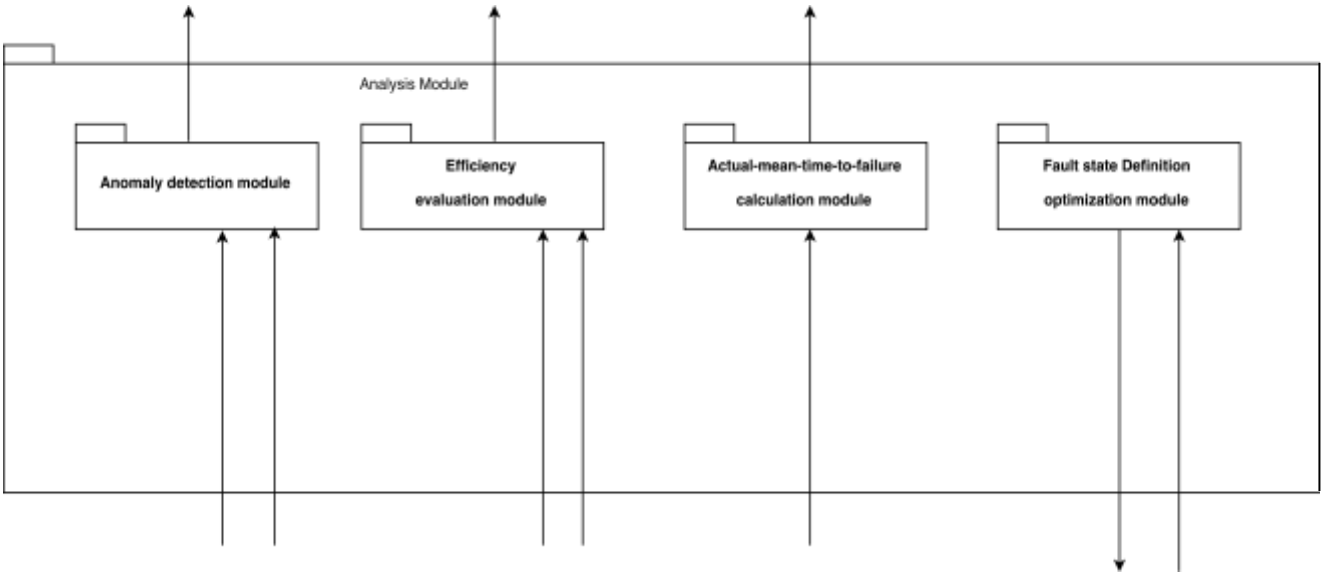

Figure 5 White-box structure of the Predictive Analytics Module, Level-3.

- The Fault State Definition Optimization module has the purpose of improving the fault definitions based on historical data stored in its unfiltered state in the Generic Data Store. The output for it is said definitions. This module's purpose is to fulfill Requirement 18.
  The main purpose of the Fault definition optimization module is to improve the definitions that are used by the real-time analysis module for detecting anomalies. This can be achieved through the use of machine learning algorithms on all the historical data that has been gathered. This can include a re-evaluation of the indicators for specific fault states and the discovery of new related variables.
- The Anomaly Detection Module is responsible for detecting possible faults based on the newest sensor data and thus implementing requirements 11 and 12. This module relies on data from the Specific Data Store and the Config Database where the fault state definitions are saved. When the module evaluates that a fault state is entered, it sends a notification to the UI.
- The Efficiency Evaluation Module calculates the value for the loss-in-process-efficiency variable as defined in the previous section in requirement 15. The calculation uses the current sensor data from the Specific data store. In case the value for loss-in-process-efficiency variable falls beneath a certain threshold t, a notification is sent to the UI.
- The Actual-mean-time-to-failure calculation module implements the calculation for the variable of the same name introduced in the previous section in requirement 14. This module relies on the historical data saved in the specific data store. This includes previous maintenance that was performed on the system, how long it has been in a fault state for along with Config data such as the service life of the specific machinery. The evaluation also uses the current values for loss-in-process-efficiency as well as the results of the Anomaly Detection Module, which indicates whether a fault state has been entered or not. Depending on the amount of data sources available at any point the value for the Actual-mean-time-to-failure variable can be calculated through different methods:
  - In any newly installed system that hasn't gathered enough historical data and has not yet entered a fault state the value is based on the efficiency. If the value for loss-in-process-efficiency is stable, we take the average lifespan of such machinery due to lack of enough data for a more accurate assessment. In case the efficiency is declining this

development can be mathematically modelled through a function that can predict when a failure of the system should occur.

- In the case where there is enough historical data gathered and a fault state is not currently entered, the behavior of the system can be compared to past states that have produced similar sensor measurements and the value should be based on how the system has behaved previously.
- In a scenario where the Anomaly Detection module indicates that a fault state has been entered, historical data can be a good indication of how the deterioration process would look and this can be taken into account. If, however, there is no such data available the Actual-mean-time-to-failure variable would default to the default time-to-failure for a specific fault state.

When the value for the variable falls under a certain threshold a notification is sent to the UI.

6. Implementation of a centralized big data architecture

This section introduces an overview of potential technologies that can be used for the implementation of the suggested architecture. A comparison between the possible implementation options is also made based on the suitability for the given use case.
In this thesis Hadoop is chosen as the base big data framework for the implementation and therefore compatibility with it will be used as a base criteria for the other tools and technologies that would be used.

6.1. Hadoop

Hadoop is a distributed big data storage framework based on the concepts introduced by Google's Big Table. Hadoop provides shared storage (the Hadoop Distributed Filesystem) and analysis (MapReduce) that is resistant to hardware failure. [6]
Hadoop consists of four modules: the Hadoop Distributed Filesystem (HDFS), MapReduce, YARN and Common. [7] The Hadoop ecosystem covers numerous projects that are related to and extend the functionality of those core modules. [6]

- The Hadoop Distributed Filesystem (HDFS) is responsible for the data storage. HDFS's design focuses on three main points: the storage of very large data files (up to hundreds of terabytes in size), streaming data access (a write-once, read-many-times data processing pattern) and the use of commodity hardware. [6] HDFS splits files into blocks (64 MB by default) in order to manage their storage. The filesystem achieves fault-tolerance through storing three copies of each data block. In HDFS there are two types of nodes: namenodes (master) and datanodes (workers). In each cluster there is one namenode that manages the filesystem tree and metadata and multiple datanodes, which store the blocks of data. The namenode is also responsible for storing the information about which blocks store a particular file.
- MapReduce is Hadoop's data processing engine. MapReduce deals with the computation of reads and writes through abstraction and instead works through computation over sets of keys and values. [6] There are two main phases of computation: the map and the reduce function. In the map phase the raw data is sorted into key/value pairs and in the reduce phase the actual processing of the already organised data happens.
- YARN ("Yet Another Resource Negotiator") provides resource management and thus facilitates the separation between the programming and infrastructure models.
- Common is a set of utilities necessary for other Hadoop modules.

While MapReduce is the default processing engine for Hadoop, it can be replaced by other options in the Hadoop ecosystem due to the modularity of Hadoop. In this section, we will take a look at Spark and other more suitable solutions for the proposed PdM architecture.

When it comes to big data analysis platforms Hadoop is considered the de facto choice. [33] The main advantages that contribute to that are its scalability, fault-tolerance, cost-efficiency and flexibility. Next to other big data solutions, Hadoop proves to be a more appropriate choice. A comparison between Hadoop and the Oracle database's performance found that Hadoop was easier to configure and scaled better. [32]

6.2. Databases

Relational Database Management Systems (RDBMS) like MySQL are the most commonly used type of databases. The relational model is a key concept – a relation is a table with rows and columns. [12] A table contains a set of tuples with the same attributes, which describe the same object. A relational database consists of a set of such tables, where data from one can relate to another by key.
The main issue with RDBMS in the context of big data is the performance – queries become increasingly slow. This is the main reason why NoSQL databases are a preferred solution since they facilitate the horizontal scaling of data.
In this section two such databases will be described in more detail – HBase and Cassandra.

6.2.1. HBase

HBase is a column-oriented database that is part of the Hadoop ecosystem. A column-oriented database stores its data by column and not by row as is traditionally the case in MySQL.
In HBase all columns of one row are grouped into column families and have a common prefix as indication. During the definition of table schema all column families of a table need to be specified. New columns, however, can be added on demand. All column family members are also stored together physically on the HDFS. Unlike tables in RDBMS, in HBase, it is possible to add new columns to an already existing table. Cells are versioned in HBase through a timestamp. The tables are also divided into regions – a region is a subset of rows. In the beginning each new table consists of only one region and as it grows it is split into multiple regions.

6.2.2. Cassandra

Cassandra is Facebook's second generation distributed key-value store. [13] Like HBase, Cassandra is designed to handle large amounts of data and provide high availability. One of Cassandra's main advantages over HBase is its lack of a single point of failure since every single node in a cluster occupies the same role. Though the addition of new machines, Cassandra's read and write throughput can be increased linearly.
Similarly to HBase columns can be added to specific keys but Cassandra also allows for columns to be put into column families in a nested way.

6.2.3. OpenTSDB

The data gathered by PdM systems is usually time-series data so databases specifically designed for storing it are a sensible solution for implementing the proposed system. This type of data has certain specifics that define it: it is collected in pairs of timestamp/value, recorded once and rarely changed after and data is usually accessed through a time range.
OpenTSDB is a database for storing time-series data that uses HBase for storage. It can use REST API or client side collection library. OpenTSDB also has a GUI that can be used for plotting and basic analytics. OpenTSDB receives data as a set of four values: timestamp, value, metric and tags. [14]
There are a number of advantages to OpenTSDB as specified in [15] for the use case of smart meters that would still be valid for any PdM system like scalability provided by HBase, fault tolerance and the facilitation of storing an increasing number of metrics.

As OpenTSDB is considered the most popular time-series database due to its simple and generic architecture [16] it is better suitable for the proposed implementation.
Since both HBase and Cassandra are considered fairly competitive databases [15] and OpenTSDB has better integration with HBase, an implementation with OpenTSDB and HBase is the optimal solution.

6.3. Streaming data analytics tools

6.3.1. Spark

Apache Spark is a large-scale data processing engine that is based on MapReduce but provides a considerate improvement on its functionality. In order to provide fault tolerance Spark uses Resilient Distributed Datasets (RDD), which store data in-memory. This allows for better efficiency.
As an iterative batch processing engine Spark does not provide real-time processing. An alternative is Spark Streaming which uses micro-batching, an approach where the data is broken down into smaller packages in order to be processed by the batch system. While not being a true real-time processing system, Spark Streaming provides easier load balancing and resistance to node failure. In the context of the proposed system, however, Spark Streaming is an adequate solution since its minimal latency would not be a problem in most projects. [7]

6.3.2. Flink

Apache Flink provides both batch and stream processing. Unlike Spark's micro-batch approach, Flink's streaming API is event-based. Flink also has connectors for receiving data from Kafka, RabbitMQ, Twitter, and Flume as well as user-defined sources.
FlinkML - Flink's machine learning library was introduced in 2015. Flink is also compatible with SAMOA as an alternative machine learning library for streaming.

A comparison between Spark and Flink done in [8] states that Spark provides better fault-tolerance and is more suitable for iterative algorithms. Flink, however, is said to be easier to integrate with other projects and provide more optimization mechanisms. A comprehensive comparison between the two frameworks done in [9] states that due to its larger developer community Spark has better usability. While Spark does not provide true stream processing its latency is minimal while providing higher fault-tolerance and better usability which is what makes it more suitable for the implementation of the proposed PdM architecture.

6.4. Messaging systems

Publish-subscribe is a distributed interaction paradigm suitable for loosely coupled and scalable systems. It allows the senders of messages to write to a persistent store and the receivers to get the changes to a class they have subscribed to at a system-appropriate time-frame. [17] The most popular open-source implementations are Apache Kafka and RabbitMQ.

6.4.1. Kafka

Apache Kafka is a publish-subscribe messaging system developed at LinkedIn. One of the main aspects considered in its design is delivering large volumes of log and event data with minimal latency.

Publishers submit messages to a specific topic. Each topic holds a feed of all messages sent to it and is spread over a number of Kafka brokers. Each broker holds zero or more partitions of a topic and each partition is a write-ahead log of messages. [18]
Kafka utilises some very effective optimisation strategies like using batching at all stages of the pipeline and using persistent data structures and OS page cache, which leads to significantly better throughput.

6.4.2. RabbitMQ

RabbitMQ is an implementation of the Advanced Message Queuing Protocol (AMQP). AMQP is designed to solve the problem of interoperability in asynchronous messaging middlewares. The protocol divides message brokering into two main concepts: exchanges and message queues. An exchange is a router that receives messages and decides which message queues they should be sent to. The message queue stores the messages and then sends them to their respective receivers. Bindings join together the exchanges and message queues by defining the rules for the routing done by the exchanges. [18]
Beyond the scope of the AMQP protocol RabbitMQ also provides a more efficient mechanism for acknowledging publishers, better flow control and better defined transactional behavior.
RabbitMQ is also considered closer in design to classic messaging systems than Kafka. [18]

When compared Kafka and RabbitMQ have similar low-latency results [18]. In a basic set-up, RabbitMQ performs better in terms of throughput. Kafka's performance can, however, be greatly improved by increasing the partition count on a single node, which demonstrates its superior scalability.
Kafka is also stated to be more suitable for streaming use cases [18] since it has a light-weight stream processing library (Kafka Streams), which makes it better suited for the implementation of the proposed PdM system.

6.5. Predictive analytics tools

This subsection introduces machine learning libraries that are suitable for big-data analysis within the Hadoop ecosystem.

6.5.1. Mahout

Mahout is a machine learning library that covers a large variety of algorithms for classification, clustering and batch based collaborative filtering. [10] Mahout is based on MapReduce which is why its main disadvantage is slow runtimes. Alongside that Mahout has been reported to be difficult to set up for Hadoop projects [7]. Mahout's algorithms, however, scale well for larger datasets. While its extensibility is considered a major advantage, a high proficiency in Java is required in order to make use of that extensibility.

6.5.2. MLlib

MLlib provides a significant improvement in computation time through making use of Spark's in-memory computation. This, however, also means that it is dependable on Spark and therefore not compatible with other platforms. [7] It also provides tools for feature extraction, basic statistics and

regression models, which in particular are not covered by Mahout. MLlib's set up is also reportedly easy compared to Mahout.

Since Spark is the chosen streaming and batch-analysis framework for the proposed implementation, this allows a compatibility with MLlib as the chosen machine learning library. MLlib's main advantage for this implementation is its faster runtimes.

6.6. Proposed implementation

After an evaluation of the possible solutions that can be used when implementing the architecture proposed in the previous chapter this section describes an overview of the most suitable for the generic use case components.
Figure 6 displays the interaction between those components. Apache Kafka's messaging queue receives data from the sensors and sends it to Spark for stream processing. Spark is also responsible for batch analysis through the use of the machine learning library MLlib. Data is stored in OpenTSDB which is based on HBase. All of these components work within the Hadoop framework and use its core components as basis.

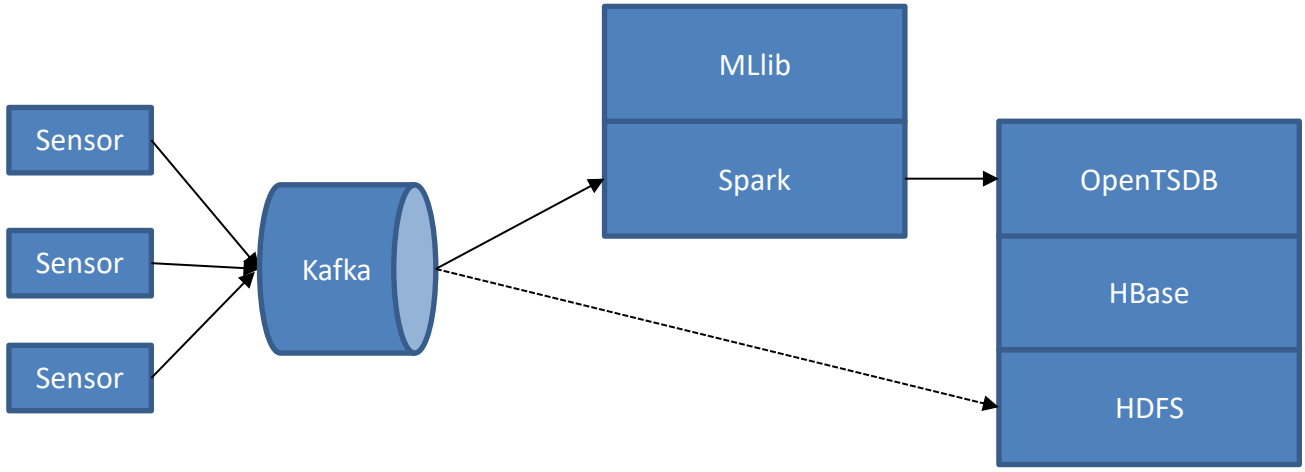

Figure 6 Proposed implementation of the PdM architecture.

7. Future work

A possible continuation of this work could cover the development of a similar architecture and proposal of suitable technologies for implementing it for a scenario, where a more distributed approach in data analysis is necessary. There are real-world scenarios where a reaction time is critical and the delay caused by a central analytical component is too impractical like for example in airplanes and trains.
In such situations a different approach in the development of a PdM system is necessary. The mentioned difficulties can be avoided by delegation the analytical and evaluation modules to a local level. This can be achieved through the introduction of components that provide evaluation only for smaller subsystems. In such a solution, a consideration should also be given to how machine learning can be introduced and at which level in the hierarchy that would be sensible. That also determines how the process of updating the fault state definitions in all subsystems should be set up.

8. Conclusion

In this thesis a proposal for a centralized reference architecture for a PdM system is made. A main aspect of the architecture is the predictive analytics module which contains components responsible for detection of potential failures in streaming data as well as for optimization through using machine learning in batch analysis. This reference architecture can be used as a basis for future projects.
The architecture was designed on the basis of functional and non-functional requirements that define a general use case.
This work also contains an evaluation of possible technologies that can be used to implement the PdM architecture based on the requirements for such a system. The proposed implementation uses projects within the Hadoop ecosystem. The solution would use the HBase and OpenTSDB databases and Apache Spark for batch and stream processing.
In addition, this thesis contains a classification of PdM systems according to incoming data source, amount of generated data, type of observed system, location of the data analysis, topology and anomaly detection strategies.

List of figures

References

[1] Tim Berners-Lee, Larry Masinter, Mark P. McCahill [Hrsg.]: Uniform Resource Locators (URL). Request for Comments 1738, Network Working Group <http://rfc.net/rfc1738.txt>, Dezember 1994. Zugriff am 29. November 2007.

[2] Robert Braden [Hrsg.]: Requirements for Internet Hosts -- Communication Layers. Internet Standard 3, Network Working Group <http://rfc.net/std3.html>, Oktober 1989. Zugriff am 29. November 2007.

[3] Robert Hinden, Stephen Deering: IP Version 6 Addressing Architecture. Request for Comments 23
73, Network Working Group <http://rfc.net/rfc2373.txt>, Juli 1998. Zugriff am 29. November 2007.

[4] Bruno Buchberger: Thinking, Speaking, Writing. Basic Working Techniques for Students of Mathematics and Computer Science. Begleitmaterial zu seiner Vorlesung „Praktische Beweistechnik und wissenschaftliches Arbeiten im Bereich des Symbolic Computation“, Universität Linz, 1992.

[5] Plain English Campaign. <http://www.plainenglish.co.uk/>. Zugriff am 29. November 2007.

[6] White, Tom. Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[7] Landset, Sara, et al. "A survey of open source tools for machine learning with big data in the Hadoop ecosystem." Journal of Big Data 2.1 (2015): 24.

[8] Ni Z. Comparative Evaluation of Spark and Stratosphere. Thesis, KTH Royal Institute of Technology; 2013

[9] Zukunft, Olaf. "Big Data Systeme: Konzeptioneller und experimenteller Vergleich von Apache Flink mit Apache Spark anhand eines Anwendungsszenarios."

[10] Zheng, Jiang, and Aldo Dagnino. "An initial study of predictive machine learning analytics on large volumes of historical data for power system applications." Big Data (Big Data), 2014 IEEE International Conference on. IEEE, 2014.

[11] Garcia, Mari Cruz, Miguel A. Sanz-Bobi, and Javier del Pico. "SIMAP: Intelligent System for Predictive Maintenance: Application to the health condition monitoring of a windturbine gearbox." Computers in Industry 57.6 (2006): 552-568.

[12] Abramova, Veronika, and Jorge Bernardino. "NoSQL databases: MongoDB vs cassandra." Proceedings of the international C* conference on computer science and software engineering. ACM, 2013.

[13] Rabl, Tilmann, et al. "Solving big data challenges for enterprise application performance management." Proceedings of the VLDB Endowment 5.12 (2012): 1724-1735.

[14] OpenTSDB Homepage: http://opentsdb.net

[15] Prasad, Srikrishna, and S. B. Avinash. "Smart meter data analytics using OpenTSDB and Hadoop." Innovative Smart Grid Technologies-Asia (ISGT Asia), 2013 IEEE. IEEE, 2013.

[16] Wlodarczyk, Tomasz Wiktor. "Overview of time series storage and processing in a cloud environment." Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on. IEEE, 2012.

[17] Thein, Khin Me Me. "Apache kafka: Next generation distributed messaging system." International Journal of Scientific Engineering and Technology Research 3.47 (2014): 9478-9483.

[18] Dobbelaere, Philippe, and Kyumars Sheykh Esmaili. "Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper." Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems. ACM, 2017.

[19] Bell, Jason. Machine learning: hands-on for developers and technical professionals. John Wiley & Sons, 2014.

[20] Mobley, R. Keith. An introduction to predictive maintenance. Butterworth-Heinemann, 2002: 1-6

[21] Mann, Lawrence, Anuj Saxena, and Gerald M. Knapp. "Statistical-based or condition-based preventive maintenance?." Journal of Quality in Maintenance Engineering 1.1 (1995): 46-59.

[22] Dunning, Ted, and Ellen Friedman. Practical machine learning: a new look at anomaly detection. " O'Reilly Media, Inc.", 2014.

[23] „Big Data und GeschäftsmodellInnovationen in der Praxis: 40+ Beispiele”, http://www.consultingregion.de/sites/default/files/bitkom-leitfaden_big_data_und_gm-innovationen_06febr2015.pdf

[24] Lee, Jay, Hung-An Kao, and Shanhu Yang. "Service innovation and smart analytics for industry 4.0 and big data environment." Procedia Cirp 16 (2014): 3-8.

[25] Gungor, Vehbi C., and Gerhard P. Hancke. "Industrial wireless sensor networks: Challenges, design principles, and technical approaches." IEEE Transactions on industrial electronics 56.10 (2009): 4258-4265.

[26] Yedavalli, Rama K., and Rohit K. Belapurkar. "Application of wireless sensor networks to aircraft control and health management systems." Journal of Control Theory and Applications 9.1 (2011): 28-33.

[27] Austin, Jackson, et al. "Predictive maintenance: Distributed aircraft engine diagnostics." The Grid, 2nd ed, I. Foster and C. Kesselman, Eds. San Mateo, CA: Morgan Kaufmann (2003).

[28] Zhou, Xiaojun, Lifeng Xi, and Jay Lee. "Reliability-centered predictive maintenance scheduling for a continuously monitored system subject to degradation." Reliability Engineering & System Safety 92.4 (2007): 530-534.

[29] "Big Data im Kühlschrank: Wie Predictive Maintenance die Accsonauten vor warmer Cola warnt" http://accso.de/magazin/big-data-im-kuehlschrank

[30] Estimote Homepage: https://estimote.com/

[31] BlueUp Beacon Sensors: http://www.blueupbeacons.com/index.php?page=sensor

[32] Baranowski, Zbigniew, Luca Canali, and Eric Grancher. "Sequential data access with Oracle and Hadoop: a performance comparison." Journal of Physics: Conference Series. Vol. 513. No. 4. IOP Publishing, 2014.

[33] Hu, Han, et al. "Toward scalable systems for big data analytics: A technology tutorial." IEEE access 2 (2014): 652-687.

[34] Hashemian, Hashem M., and Wendell C. Bean. "State-of-the-art predictive maintenance techniques." IEEE Transactions on Instrumentation and measurement 60.10 (2011): 3480-3492.

[35] Bangemann, Thomas, et al. "PROTEUS—Creating distributed maintenance systems through an integration platform." Computers in Industry 57.6 (2006): 539-551.

[36] Katipamula, Srinivas, and Michael R. Brambley. "Methods for fault detection, diagnostics, and prognostics for building systems—a review, part II." Hvac&R Research 11.2 (2005): 169-187.

[37] Seem, John E. "Using intelligent data analysis to detect abnormal energy consumption in buildings." Energy and buildings 39.1 (2007): 52-58.