Entwicklung eines Referenzmodells für eine echtzeitfähige zustandsorientierte Instandhaltung von komplexen technischen Systemen

von Abderrahim Ait Alla

Dissertation

vorgelegt am Fachbereich 3 der Universität Bremen
zur Erlangung des Grades
eines Doktors der Ingenieurwissenschaften

- Dr.-Ing. -

Tag der mündlichen Prüfung: 16.04.2020

Gutachter:
Prof. Dr. Hans-Jörg Kreowski
Prof. Dr.-Ing. Jens Heger (Leuphana Universität Lüneburg)

Universität Bremen
Fachbereich 3 Mathematik/Informatik
Arbeitsgruppe Theoretische Informatik
Prof. Dr. Hans-Jörg Kreowski

Danksagung

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Bremer Institut für Produktion und Logistik GmbH (BIBA) an der Universität Bremen. Im Rahmen dieser Tätigkeit hatte ich die Möglichkeit, aus der Bearbeitung zahlreicher Forschungsprojekte mein Thema zu entwickeln.

Zahlreiche Personen haben mich bei der Erstellung meiner Dissertation in vielfältiger Art und Weise unterstützt und begleitet, denen ich an dieser Stelle danken möchte.

Mein persönlicher Dank gilt zunächst meinem Doktorvater Herrn Hans-Jörg Kreowski für die Übernahme des Erstgutachtens sowie seine konstruktiven Anmerkungen und engagierte, hervorragende Unterstützung.

Ebenfalls möchte ich mich bei Herrn Prof. Dr.-Ing. Jens Heger, für die freundliche Übernahme des Zweitgutachtens, Prof. Dr. Rainer Malaka, Prof. Dr.-Ing Michael Freitag, Markus Kreutz und Vanessa Beckmann als Mitglieder meiner Prüfungskommission bedanken.

Auch bedanke ich mich bei meinen Kollegen Daniel Rippel, Aljoscha Sander und Michael Lütjen für ihre konstruktive Kritik und den regen Austausch, die erheblich zur Qualität dieser Arbeit beigetragen haben.

Schließlich danke ich meiner Ehefrau Tea Chitauri für ihre Liebe und Motivation. Mein besonderer Dank gilt meinen Eltern für ihre Förderung auf meinem Lebensweg.

I. Zusammenfassung

Instandhaltungsmanagementsysteme sind eines der kritischen Geschäftsfelder in Unternehmen, die eine zentrale Rolle bei der Verringerung der Ausfallzeiten von Maschinen und Anlagen spielen. Der Einsatz eines Instandhaltungssystems ist eine komplizierte Aufgabe. Der Grund für diese Komplexität ist, dass Systeme aus vielen Komponenten bestehen, welche miteinander im Zusammenhang stehen. Dieser Zusammenhang zwischen den Komponenten macht es kompliziert, ein Systemmodell zu erstellen und somit ein Instandhaltungssystem zu optimieren. Dieser Zusammenhang kann in Form von Ausfallsabhängigkeit oder Ausfallkorrelation dargestellt werden, d. h., dass der Ausfall einer Komponente zum Ausfall anderer funktionsfähiger Subsysteme führen kann. In diesen Fällen muss in Echtzeit entschieden werden, welche Maßnahmen zu ergreifen sind, um dies zu verhindern bzw. um die gestörte Ressource wieder in Stand zu setzen. In diesem Kontext bildet das Complex Event Processing (CEP) einen neuartigen Ansatz, welcher die auftretenden Systemereignisse (engl. events) auf Basis von Ereignismustern miteinander in Beziehung setzt und diese (neu) entstandenen komplexen Ereignisse (engl. complex events) in Echtzeit analysiert.

Im Rahmen dieser Dissertation wird ein Zustandsüberwachungssystem (engl. condition monitoring system) entwickelt, welches unter Verwendung des Komplexereignisverarbeitungsansatzes (engl. complex event processing) eine Überwachung eines Systemzustands in Echtzeit erreicht, Störungsursachen identifiziert und den Zustand technischer Komponenten bewertet. Auf Basis dieser Informationen können anschließend im Rahmen der Instandhaltungsplanung verschiedene Instandhaltungsmaßnahmen angestoßen werden. Der Einsatz der CEP-Technologie setzt die Entwicklung neuer Informationslandschaften, basierend auf ereignisgesteuerten Architekturen, voraus. Innerhalb dieser Arbeit wird in diesem Sinne unter anderem die Forschungsfrage bearbeitet, wie das Zustandsüberwachungssystem im Hinblick auf seine Struktur entworfen werden muss, damit ein erfolgreicher Einsatz des CEP-Ansatzes realisiert werden kann. Dies erfolgt mittels eines Referenzmodells. Zudem adressiert die Arbeit die Forschungsfrage, wie das System konfiguriert werden muss, damit Sensordaten verschiedener Komponenten in Echtzeit zueinander in Beziehung gesetzt werden können, um globale Bedingungen richtig einschätzen zu können. Dies wird in dieser Arbeit als Konfigurationsmethodik adressiert.

Die Evaluation an zwei Szenarien hat gezeigt, dass einerseits die Übertragung des CEP-Ansatzes durch die Entwicklung eines Referenzmodells zur echtzeitfähigen Detektion bzw. Vorhersage von Störungen im Vergleich zu herkömmlichen Zustandsüberwachungssysteme geführt hat. Andererseits konnte in beiden Anwendungsszenarien gezeigt werden, dass die Herleitung und die Anpassung von Ereignisregeln stark von den entwickelten ML-Modellen abhängig ist.

II. Abstract

Maintenance management systems are one of the critical business areas in companies that play a central role in reducing machine and plant downtime. Using a maintenance system is a complicated task. The reason for this complexity is that systems consist of many components, which are connected to each other. This connection between the components makes it complicated to create a system model and thus optimize a maintenance system (Nicolai et al. 2008). This relationship can be represented in the form of failure dependency or failure correlation, i.e. the failure of one component can lead to the failure of other functional subsystems. In these cases, it must be determined in real time which measures are to be taken to prevent this or to repair the failed resource. In this context, Complex Event Processing (CEP) is a novel approach that correlates system events on the basis of event patterns and analyzes these complex events in real time.

The objective of this thesis is the development of a Condition Monitoring System which enables surveillance of the system state in real-time using CEP-Technology, identifies fault causes and evaluates the condition of the technical components. On the basis of this information various maintenance measures can be initiated within the framework of the maintenance planning. In general, CEP was used in the finance environment to begin with but its employment can also be of use in other areas (especially in Business Activity Monitoring, and Sensor Networks). The use of the CEP-Technology requires the development of new information architectures based on event-driven architecture. Within this dissertation various research questions are addressed to this end. On the one hand this is the question of how the Condition Diagnosis System should be conceived in regards to its structure, to enable the realization of a successful employment of the CEP approach. To this end a reference model is being developed in the context of this work. On the other hand, there is the question of how the system must be configured so that sensor data of different components can be related to each other in real time in order to correctly assess global conditions. This is addressed in this dissertation as a configuration methodology.

The evaluation of the concept in two scenarios showed that the transfer of the CEP approach through the development of a reference model led to the real-time detection and prediction of disturbances in comparison to conventional condition monitoring systems. On the other hand, it could be shown in both application scenarios that the derivation and adaptation of event rules is strongly dependent on the developed ML models.

III. Inhaltsverzeichnis

I. Zusammenfassung I
II. Abstract II
III. Inhaltsverzeichnis III
IV. Abbildungsverzeichnis VI
V. Tabellenverzeichnis VIII
VI. Abkürzungsverzeichnis IX
1. Einleitung 1
1.1. Motivation 1
1.2. Problemstellung 2
1.3. Zielsetzung 4
1.4. Forschungsfragen und Vorgehensweise 5
1.5. Aufbau der Arbeit 7
2. Grundlage der Instandhaltung 9
2.1. Definitionen der Begriffe im Rahmen der Instandhaltung 9
2.2. Grundlage der Instandhaltung 10
2.2.1. Entwicklungsstufen der Instandhaltung 10
2.2.2. Vierte Phase der Instandhaltung 11
2.2.3. Instandhaltungsstrategien 12
2.2.4. Auswahl der Instandhaltungsstrategie 13
2.3. Zustandsorientierte Instandhaltung – CBM 15
2.4. Grundlagen der Zuverlässigkeitsanalyse von technischen Systemen 19
2.5. Andere Varianten von Instandhaltungsstrategien 21
2.5.1. Total Productive Maintenance (TPM) 22
2.5.2. Reliability-Centered Maintenance (RCM) 22
2.5.3. E-Maintenance 22
3. Grundlagen des Data-Mining und des maschinellen Lernens 24
3.1. Data-Mining-Evolution über die Zeit 24
3.2. Data-Mining-Prozess 26
3.3. Analyse-Verfahren im Data-Mining 29
3.4. Aufgaben und Methoden des Data-Mining 30
3.4.1. Klassifikation 32
3.4.2. Regressionsanalyse 33
3.4.3. Segmentierung/Clustering 35
3.4.4. Assoziationsanalyse 36

3.5. Data-Mining-Methoden: Einsatz von Methoden der KI 37
3.5.1. Neuronale Netze 38
3.5.2. Entscheidungsbaum 42
4. Complex Event Processing 45
4.1. Architekturen und Konzepte zur Verarbeitung von Datenereignissen 45
4.1.1. Konzepte zur Verarbeitung von Daten – Paradigmenwechsel 45
4.1.2. Allgemeiner Aufbau einer ereignisorientieren Architektur 46
4.2. Verarbeitung komplexer Ereignisse in ereignisorientierten Systemen 48
4.3. Beispiel eines ereignisorientierten Instandhaltungssystems 51
4.4. Zentralisiertes und verteiltes CEP-System 54
4.5. Syntax zur Beschreibung von Ereignissen, Ereignisregeln und Ereignismustern 55
4.5.1. Ereignisdarstellung, -definition und -begriffe 55
4.5.2. Semantik von Ereignisregeln 57
4.5.3. Ereignismuster – Event Patterns 58
4.5.4. Sliding Windows innerhalb einer WITHIN-Klausel 60
4.6. Anwendungsgebiete von CEP 63
5. Entwicklung des Konzepts eines echtzeitfähigen, zustandsorientierten Zustandsüberwachungssystems 67
5.1. Konzipierung und Entwurf 67
5.1.1. Anforderungen an ein CEP-Referenzmodell zur Integration des CEP-Ansatzes in das zustandsorientierte Instandhaltungssystem 67
5.1.2. Entwicklung einer Referenzarchitektur 70
5.1.3. Entwicklung des Ereignismodells (Event Model) 77
5.2. Konfiguration und Optimierung 82
5.2.1. Konzeption der Offline-Datenanalyse – Aufbauphasen des ML-Modells 83
5.2.2. Konzept der Regelextraktion aus der Offline-Datenanalyse 85
5.2.3. Störungsbewertung und Ursachendiagnose 89
6. Konzeptumsetzung und Evaluation 91
6.1. Durchführung des Evaluationsprozesses 91
6.2. Szenario 1: Kühlkreislauf der elektrischen Radnabenmotoren eines Van Carriers 92
6.2.1. Beschreibung des Funktionsmodells 93
6.2.2. Situationsanalyse – Datenspezifikation 94
6.2.3. Erstellung der ersten Version des Ereignismodells 95
6.2.4. Konfigurationsmethodik zur Umsetzung des CEP-Systems für das erste Szenario 96
6.2.5. Implementierung und Evaluation des entwickelten CEP-CM-Systems 105
6.3. Szenario 2: Vereisung der Rotorblätter von Windkraftanlagen 110
6.3.1. Beschreibung des Vereisungsproblems von Windkraftanlagen 110

6.3.2. Situationsanalyse – Datenspezifikation 112
6.3.3. Erstellung einer ersten Version des Ereignismodells 115
6.3.4. Konfigurationsmethodik zur Umsetzung des CEP-Systems für das zweite Szenario 117
6.3.5. Implementierung und Evaluation des entwickelten CEP-CM-Systems 123
6.4. Erweiterung und Anpassung des entwickelten CEP-CM-Systems 124
6.4.1. Struktur der Ereignisdatenbank 125
6.4.2. Anwendungsbeispiel für die Anpassung eines CEP-CM-Systems 125
7. Fazit 127
7.1. Zusammenfassung 127
7.2. Beantwortung von Forschungsfragen 128
7.3. Ausblick 131
Literaturverzeichnis 133
Anhang 147

IV. Abbildungsverzeichnis

Abbildung 1: Struktur der Arbeit 8
Abbildung 2: Historische Entwicklung der Instandhaltungsstrategien (in Anlehnung an ISKE 2009) 11
Abbildung 3: Instandhaltungsstrategien (in Anlehnung an Schenk 2010) 13
Abbildung 4: Selektionslogik einer Instandhaltungsstrategie 14
Abbildung 5: Zusammenhang zwischen dem Abnutzungsvorrat und den Störungsauslösungszeiten in einer periodischen bzw. zustandsorientierten Instandhaltungsstrategie (in Anlehnung an Ryll 2008) 16
Abbildung 6: Open System Architecture for CBM (in Anlehnung an Lebold et al. 2003) 17
Abbildung 7: Verlauf der Ausfallrate – aufgrund der graphischen Darstellung oft als „Badewannenkurve“ bezeichnet (in Anlehnung an Ryll 2008) 21
Abbildung 8: KDD-Data-Mining-Prozess (in Anlehnung an Fayyad et al. 1996) 27
Abbildung 9:CRISP-Data-Mining-Referenzmodell (in Anlehnung an Chapman et al. 2000) 28
Abbildung 10: Verschiedene Datentypen im Data-Mining (in Anlehnung an Cleve et al. 2016) ..31
Abbildung 11: Zuordnung der Data-Mining-Aufgaben zu Methoden (in Anlehnung an Alpar et al. 2000) 32
Abbildung 12: Aufbauprozess eines Modells für die Klassifikationsaufgabe 33
Abbildung 13: Beispiel einer linearen Regression bzw. Approximation eines Temperatursensors 34
Abbildung 14: Beispiel einer K-means Cluster-Klassifikation 36
Abbildung 15: Datenverarbeitung eines technischen Neurons (in Anlehnung an Kriesel 2008) ...39
Abbildung 16: Beispiel einiger Aktivierungsfunktionen (in Anlehnung an Nielsen 2015) 40
Abbildung 17: Beispiel eines dreischichtigen Feed-Forward-Netzes (in Anlehnung an Nielsen 2015) 41
Abbildung 18: Beispiel eines Entscheidungsbaumes 43
Abbildung 19: Architektur eines Datenbankmanagementsystems 46
Abbildung 20: Architektur eines Datenstrommanagementsystems 46
Abbildung 21: Aufgabe eines ereignisorientierten Systems 48
Abbildung 22: CEP-Grundkonzept (in Anlehnung an Bruns et al. 2010) 50
Abbildung 23: Prozessablauf eines Instandhaltungssystems mit Ereignisbeispielen 52
Abbildung 24: Beispiel eines Ereignistyps und dessen Ereignisinstanzen 52
Abbildung 25: Ereignismodell für das Instandhaltungsbeispiel der Abbildung 23 53
Abbildung 26: Beispiel eines Ereignisverarbeitungsnetzwerks 54
Abbildung 27: Beispiel eines Längenfensters mit einem Puffer der Länge 3 (Regel 1) 61
Abbildung 28: Beispiel eines Längenfensters mit einem Puffer der Länge 3 (Regel 2) 62
Abbildung 29: Beispiel eines Zeitfensters mit einer Länge von 3 Zeiteinheiten (Regel 3) 63
Abbildung 30: Event-Processing-Konzept eines zentralen, ereignisbasierten Zustandsüberwachungssystems 72

Abbildung 31: Event-Processing-Konzept eines verteilten ereignisbasierten Zustandsüberwachungssystems 76
Abbildung 32: Struktur des Ereignismodells für das CEP-CM-System und dessen grundlegenden Abstraktionsebenen von Ereignissen 80
Abbildung 33: Beispiel eines einfachen Kühlsystems 80
Abbildung 34: Beispiel eines UML-Ereignismodells für das Kühlsystem 81
Abbildung 35: Grundprinzip der Konfigurationsmethodik 83
Abbildung 36: Aufbauphasen des ML-Modells 84
Abbildung 37: Konzept zur Extraktion von Ereignisregeln aus dem ML-Modell 86
Abbildung 38: Regelextraktion aus der Datenvorverarbeitungsphase des ML-Modells 87
Abbildung 39: Regelextraktion aus der Datenaufbereitungsphase des ML-Modells 88
Abbildung 40: Regelextraktion aus der Aufbauphase des ML-Modells 89
Abbildung 41: Gesamter Ablauf des Evaluationsprozesses 92
Abbildung 42: Schematischer Überblick über das Funktionsmodell 94
Abbildung 43: Aufbau der ersten Version des Ereignismodells 96
Abbildung 44: Beispiel von Druckwerten mit jeweils ungültigen Werten und ersetzten Werten 97
Abbildung 45: Struktur des Entscheidungsbaums zur Störungsklassifikation 99
Abbildung 46: Ereignismodell und entsprechende Ereignisströme des betrachteten Funktionsmodells 105
Abbildung 47: Entwicklung der Sensorwerte beim Auftreten einer Leckage 107
Abbildung 48: Verlauf der Sensorwerte beim Auftreten eines Pumpendefekts 108
Abbildung 49: Vergleich der durchschnittlichen Reaktionszeit zwischen dem CEP- und dem konventionellen Ansatz. 109
Abbildung 50: Vereisung auf einem Rotorblatt © SPITZNER ENGINEERS GmbH 111
Abbildung 51: Ergebnisse der Korrelationsanalyse 113
Abbildung 52: Histogramm der Vereisungsalarmzeiten aus SCADA-Daten mehrerer Windparks 114
Abbildung 53: Variation der Eisalarmdauer innerhalb desselben Windparks 115
Abbildung 54: Skizze des Zeitfensters der verwendeten Parameter für das Modell 118
Abbildung 55: Ereignismodell für die Vereisungsvorhersage an Rotorblättern 123

V. Tabellenverzeichnis

Tabelle 1: Anforderungsliste von herkömmlichen CM-Systemen, echtzeitfähigen Systemen und CEP-CM-System (X: erforderlich, O: optional) 68
Tabelle 2: Funktionsbeschreibung der Komponenten des ereignisbasierten Zustandsüberwachungssystems 73
Tabelle 3: Struktur der Trainingsdaten und Validierungsdaten 84
Tabelle 4: Spezifikation der verbauten Sensoren 95
Tabelle 5: Gesamter Überblick über die Anzahl von Trainingsdaten bzw. Validierungsdaten 98
Tabelle 6: Validierungsergebnisse der Störungserkennung des Entscheidungsbaums 99
Tabelle 7: Ausschnitt aus den Trainingsdaten einer Leckage 106
Tabelle 8: Sensordatenausschnitt eines Pumpendefekts 107
Tabelle 9: Definition der Evaluierungsszenarien 109
Tabelle 10: Gültigkeitsbereiche und tatsächliche Wertebereiche der ausgewählten Parameter ... 114
Tabelle 11: Ein- und Ausgabeparameter des Modells 118
Tabelle 12: Gesamter Überblick über die Anzahl der Trainings- und Validierungsdaten 119
Tabelle 13: Validierungsergebnisse der Vereisungsvorhersage des NNs 119
Tabelle 14: Vergleich der Reaktionszeit bei der Vereisungsvorhersage 124
Tabelle 15: Struktur der Datenbanktabelle der Ereignisregeln 125
Tabelle 16: Struktur der Datenbanktabelle der Ereignisströme 125

VI. Abkürzungsverzeichnis

ARMA	Autoregressive-Moving Average
CBM	zustandsorientierte Instandhaltung (engl. condition based monitoring)
CEP	Complex Event Processing
CM-System	Zustandsüberwachungssystem (engl. condition monitoring system)
CPS	Cyber-physisches System
CRISP-DM	Cross-Industry Standard Process for Data-Mining
DBMS	Datenbankenmanagementsysteme
DM	Data-Mining
DNN	Deep neuronale Netze
DSMS	Datenstrommanagementsystem (engl. data stream management system)
DSPS	Data Stream Processing System
EDA	Event-Driven Architecture
EPA	Ereignisverarbeitungsagent (engl. event processing agent)
EPE	Event Processing Engine
EPL	Ereignisverarbeitungssprachen (engl. event processing language)
EPN	Ereignisverarbeitungsnetzwerk (engl. event processing network)
FIFO	First-In-First-Out
FMEA	Fehlermöglichkeits- und Einflussanalyse (engl. failure mode and effect analysis)
FN	False negative
FP	False positive
ID	Identifikationsbezeichnung
IT	Informationstechnologie
IKT	Informations- und Kommunikationstechnologien
KDD	Knowledge Discovery in Databases
KNN	künstliche neuronale Netze
ML	maschinelles Lernen
OSA-CBM	Open System Architectur for Condition Based Maintenance
REST-API	Representational State Transfer – Application Programming Interface
RCM	Reliability-Centered Maintenance

RFID	Radio-Frequency Identification
SCADA	Supervisory Control And Data Acquisition
SOA	Service-orientierten Architektur
SPA	Stream Processing Analysis
TN	True negative
TP	True positive
TPM	Total Productive Maintenance
UML	Vereinheitlichte Modellierungssprache (engl. unified modeling language)
VC	Van Carrier
WEA	Windenergieanlage

1. Einleitung

Dieses Kapitel führt in die Thematik der Arbeit ein, indem zunächst Motivation und Problemstellung erörtert werden. Darauf aufbauend wird die Zielstellung abgeleitet und die Forschungsfragen präsentiert. Als Abschluss dieses ersten Kapitels wird ein Überblick zum Aufbau der Arbeit gegeben und die einzelnen Kapitelinhalte werden vorgestellt.

1.1. Motivation

Die Instandhaltung hat sich von einem reinen Kostenverursacher zu einem unternehmensübergreifenden Geschäftsprozess gewandelt, der aktiv an der Wertschöpfung eines Unternehmens beteiligt ist, sowohl im Sinne eines Herstellers als auch eines Betreibers (Eickemeyer et al. 2014), (Nadj et al. 2016). Um wirtschaftlich im weltweiten Wettbewerb bestehen zu können, ist eine hohe Zuverlässigkeit von Maschinen und Anlagen mit einhergehender Verringerung von Ausfallzeiten für die heutige Produktion unerlässlich (Austerjost et al. 2016). Die Instandhaltung als Form der Produktionsunterstützung und unterstützende Funktion zur Sicherstellung von Anlagenverfügbarkeit, Produktqualität, Termintreue und Anlagensicherheit hat daher an Bedeutung gewonnen (Bengtsson 2007). Dennoch gilt die Instandhaltung immer noch als Kostenstelle und somit als kostenintensive unterstützende Funktion (Alsyouf 2004), (Oelker et al. 2017), (Austerjost et al. 2016). In diesem Zusammenhang, berichtet Bengtsson (2007), dass bis zu 70 % der gesamten Produktionskosten für die Instandhaltung aufgewendet werden können (Bengtsson 2007). In der Windenergiebranche entstehen sogar bis zu einem Drittel unnötiger Instandhaltungskosten während der Betriebsphase. Die Kosten wären durch eine optimierte Planung der Instandhaltungsaktivitäten und Vermeidung unnötiger, vorbeugender Instandhaltungsaktivitäten vermeidbar (Oelker et al. 2017). Eine optimale Instandhaltung zeichnet sich dadurch aus, dass die Anlage stabil arbeitet und dabei so wenig korrektive Maßnahmen und so wenig vorbeugende zyklische Instandhaltungsmaßnahmen wie möglich durchgeführt werden. Das allgemeine Prinzip der optimalen Instandhaltung besteht somit zusammenfassend darin, alle notwendigen Maßnahmen zu ergreifen, die sicherstellen, dass eine Anlage die erforderlichen Funktionen ordnungsgemäß erfüllt, und somit die Anlageverfügbarkeit zu maximieren (Fumeo et al. 2015). In diesem Kontext haben sich in den letzten Jahrzehnten Strategien und Konzepte zur Instandhaltungsunterstützung entwickelt. Eine davon ist die zustandsorientierte Instandhaltung (engl. condition based maintenance – CBM). Bei der zustandsorientierten Instandhaltung werden kritische Systemparameter überwacht (z. B. durch Vibrations- oder Temperaturüberwachung), um frühzeitig Hinweise auf einen beginnenden Ausfall zu erhalten (Schenk 2010), (Bengtsson 2007). Zur Detektion eines beginnenden Ausfalls müssen mögliche zukünftige Probleme bzw. Störungen vorhergesagt werden, indem bspw. ein Prognosemodell für die Verschlechterung des Anlagenbetriebs

erstellt wird (Fumeo et al. 2015). Derartige Degradationsmodelle können entweder auf den zugrundeliegenden physikalischen Gesetzen basieren, die das Verhalten des technischen Systems beschreiben, oder datengetrieben sein (Fumeo et al. 2015). Letztere basieren auf Algorithmen des maschinellen Lernverfahrens, die darauf abzielen, Modelle zu erstellen. Hierbei werden Daten genutzt, die den Zustand des überwachten Systems über seine gesamte Lebensdauer darstellen. In diesem Fall stellen Algorithmen des überwachten Lernens, bei denen das Modell aus Trainingsdaten abgeleitet wird, eine bevorzugte Wahl dar.

Seit einigen Jahren kann der Trend beobachtet werden, dass immer mehr technische Systeme mit Sensoren ausgestattet werden. Durch die Entwicklung von Industrie 4.0 wird sich dieser Trend weiter verstärken (Austerjost et al. 2016). Die Einführung von eingebetteten Systemen bzw. cyber-physischen Systemen (CPS) bietet große Möglichkeiten, Daten zu sammeln und zu verarbeiten, auf Basis derer das Konzept der zustandsorientierten Instandhaltung effektiv angewendet werden könnte. Mit der ständig wachsenden Verbreitung von Sensoren entstehen viele Daten, die viele Unternehmen bereits im Vorfeld haben (Austerjost et al. 2016). Die Transformation dieser großen Datenmengen in verwertbares Wissen in einem realisierbaren Zeitrahmen ist eine Schlüsselaufgabe (Fumeo et al. 2015). Der Zugriff auf diese wachsenden Datenmengen aus verschiedenen Maschinen, Technologien und Rechenressourcen ermöglicht beispielsweise, neue Erkenntnisse zu gewinnen, die für die Instandhaltungsentscheidungsfindung in nahezu Echtzeit analysiert werden müssen, und so die Instandhaltungsprozesse zu optimieren (Mayer-Schönberger et al. 2013), (Alias et al. 2016), (Thoben et al. 2018). Daten und aktuelle Messungen werden korreliert, um daraus Prognosen zu erzeugen. Auswertungen werden nahezu in Echtzeit ermittelt. Dadurch können Reparatur- und Wartungsarbeiten zeitlich priorisiert werden. Somit können aktuelle Risiken frühzeitig erkannt und die notwendigen Informationen an die richtigen Personen weitergeleitet werden. Anschließend lassen sich so schnell und proaktiv die richtigen Entscheidungen treffen, bevor teure Betriebsstörungen eintreten.

1.2. Problemstellung

Eine Grundvoraussetzung für die Implementierung einer zustandsbasierten Instandhaltung ist die Entwicklung eines funktionierenden Zustandsüberwachungssystems (engl. condition monitoring system – CMS). Um ein Zustandsüberwachungssystem operativ einsetzen zu können, muss im Vorfeld geklärt werden, wie die Maschinendaten interpretiert und verarbeitet werden sollen, um daraus einen Mehrwert unter konkreten Zielparametern zu generieren (Thoben et al. 2018). Hierfür müssen intelligente Methoden zur Verarbeitung der Daten eingesetzt werden. Darüber hinaus ist die Datenverarbeitung mit kontinuierlicher Analyse und kurzer Latenzzeit in Echtzeit für Zustandsüberwachungssysteme eine sehr wichtige Anforderung (Akbar et al. 2015). Die Entwicklung eines Zustandsüberwachungssystems ist eine herausfordernde und komplizierte Auf-

gabe, da sich eine hohe Systemkomplexität mit dynamischen Prozessen und probabilistischen Störereignissen, wie Maschinenausfälle, verbindet. Dies erschwert die Erstellung eines ganzheitlichen Systemmodells und somit auch die Optimierung der Entscheidungsfindung im Instandhaltungssystem (Eickemeyer et al. 2014). Zudem kann der Ausfall einzelner Komponenten Auswirkungen auf andere Komponenten, Subsysteme und sogar das Gesamtsystem haben. In diesem Zusammenhang muss in Echtzeit entschieden werden, welche Maßnahmen zu ergreifen sind, um dies zu verhindern bzw. um die gestörte Ressource wieder in Stand zu setzen. Hierzu existiert mit der zustandsorientierten Instandhaltung ein konzeptioneller Ansatz, der darauf abzielt, das dynamische Verhalten eines Systems oder einzelner Teile mittels Zustandsüberwachungssystemen zu überwachen und somit auf den Systemzustand bzw. einen entsprechenden Grad der Abnutzung rückzuschließen. Anschließend kann im Sinne der zustandsorientierten Instandhaltung die Planung der Instandhaltungsmaßnahmen unter Berücksichtigung von Prioritäten, Wartungs- und Arbeitsumfängen sowie vorhandener Ressourcen vorgenommen werden (Jacobs et al. 2009), (Kröning et al. 2013). Für einzelne Komponenten ist dies bislang schon gut gelöst, wobei verschiedene methodische Ansätze zum Einsatz kommen (z. B. Signal Processing und Dynamic Scheduling) (Klemm et al. 2010), (Ryll 2008). Für die Betrachtung komplexer Systeme stellt jedoch die Berücksichtigung aller Interdependenzen zwischen Teilsystemen ein wesentliches Kriterium dar (Koochaki 2009), (Denkena et al. 2012). Die Erkenntnisse im Hinblick auf eine ganzheitliche Systembetrachtung, mit sämtlichen Wechselwirkungen einzelner Teilsysteme bzw. Komponenten, sind heute als nicht ausreichend zu bewerten (Eickemeyer et al. 2014). Es liegen lediglich Konzepte für einzelne Komponenten vor. Des Weiteren fehlt es an einem Ansatz, der sich stets an sich ändernde Bedingungen anpasst. Schwellwerte bestimmter Ereignisse werden heute meist manuell festgelegt (Mehdiyev et al. 2016). Verschleiß, Materialalterung oder der Austausch von Komponenten erfordern folglich einen hohen Konfigurationsaufwand. Ein adaptiver Ansatz, der Zustandsindikatoren in Form von Ereignisregeln, Degradationsmodellen oder Schwellwerten kontinuierlich anpasst und somit eine bedeutende Flexibilität erlaubt, kann als entscheidender Baustein der intelligenten Instandhaltung aufgefasst werden. Eine andere zentrale Herausforderung bei der Entwicklung von Zustandsüberwachungssystemen (CMS) liegt in der Verarbeitung großer Datenmengen in Echtzeit (Bruns et al. 2010), (Canizo et al. 2017). Die konventionellen Zustandsüberwachungssysteme stoßen dabei aufgrund hoher Systemkomplexität an ihre Grenzen, weshalb neue Ansätze entwickelt werden sollen, welche eine echtzeitfähige Überwachung des Systems ermöglichen. In diesem Kontext bildet die Verarbeitung komplexerer Ereignisdaten (engl. complex event processing – CEP) einen neuartigen Ansatz, welcher die auftretenden Systemereignisse (engl. events) auf Basis von Ereignismustern mittels vordefinierter Ereignisregeln miteinander in Beziehung setzt und diese (neu) entstandenen komplexen Ereignisse (engl. complex events) in Echtzeit analysiert (Widder 2011). Mithilfe der Ereignisregeln können die Ereignisse gefiltert, aggregiert, korreliert

und neue komplexe Ereignisse generiert werden. Weder in der Praxis noch in der Theorie ist ein solcher CEP-Ansatz in der Instandhaltung von technischen Systemen bekannt und implementiert. Hierzu besteht ein großer Bedarf an der Entwicklung formaler Methoden und Konzepte, die das Gesamtsystem betrachten, einzelne Informationsstränge systematisch zusammenführen und daraus weitergehende Informationen ableiten.

Das CEP wird zu einem aktuellen Forschungsthema, da alle drei Big-Data-Eigenschaften (Volume, Variety und Velocity) an der Verarbeitung von Datenströmen beteiligt sind (Wang et al. 2018), (Flouris et al. 2017). CEP bezieht sich auf große Datenmengen (Volume) aus verschiedenen heterogenen Quellen (Variety), die sich durch eine hohe Datenankunftsrate (Velocity) auszeichnen und gesammelt werden, um effektiv nützliche Informationen über technische Systeme zu gewinnen. Die Ereignisverarbeitung (engl. stream processing analysis – SPA) mit CEP im Kontext der zustandsorientierten Instandhaltung ist ein hochmodernes Forschungsgebiet (Fumeo et al. 2015). Dieses Forschungsfeld umfasst die rechnergestützte Analyse, Modellierung und Extraktion von Informationen aus großen Datenmengen, die kontinuierlich aus mehreren Quellen in Echtzeit stammen. Daten werden in Form von Echtzeit-Ereignissen erzeugt, die komplexe Muster bilden, wobei jedes komplexe Muster ein einzigartiges Ereignis darstellt (Thoben et al. 2018). Diese Ereignisse müssen mit minimaler Zeitverzögerung interpretiert werden, um sie für die Entscheidungsfindung im Kontext der aktuellen Situation zu nutzen. Die Notwendigkeit, diese komplexen Muster in nahezu Echtzeit zu verarbeiten, zu analysieren und daraus Schlüsse zu ziehen, bildet die Grundlage für das Forschungsgebiet SPA bzw. CEP (Akbar et al. 2015).

1.3. Zielsetzung

Diese Arbeit verfolgt das Ziel, ein echtzeitfähiges, zustandsorientiertes Instandhaltungskonzept durch die Integration von Methoden zur Verarbeitung komplexerer Ereignisdaten, die aus kontinuierlich erfassten Zustandssensordaten bestehen, zu entwickeln. In der Arbeit werden dazu ein Referenzarchitekturmodell für die Datenverarbeitung und eine aufbauende Konfigurationsmethodik entwickelt. Die Entwicklungen werden durch modellbasiertes Testen am Beispiel von Realsystemen validiert und evaluiert. Hierfür wird das zu entwickelnde Referenzarchitekturmodell auf eine prototypische Erprobung eines Modells an einem Fuhrpark von Portalhubwagen (engl. straddle carrier) und Windkraftanlagen evaluiert. Beim ersten Anwendungsfall liegt der Fokus auf der Erkennung von Systemstörungen; während der zweiten Fallstudie steht die Detektion von Vereisung an den Rotorblättern der Windkraftanlagen im Fokus. Bereits heute werden in vielen Anwendungsgebieten Sensordaten gesammelt und verarbeitet. Daraus werden auch heute schon Systemereignisse (Verschleißgrenzen, kritischen Füllstandmengen, Zustandsstörungen, Komponentenausfälle) und komplexere Informationen (Reichweitenberechnung, dynamische Serviceintervalle) generiert

und protokolliert. Diese sind für den Nutzer in der Regel jedoch nicht sichtbar. Zum Beispiel sind bereits heute multifunktionale Ölzustandssensoren verfügbar, die ausgehend von den Betriebsbedingungen und physikalischen Parametern kontinuierlich die Qualität des Öls überwachen.

Ziel dieser Arbeit ist die Entwicklung eines echtzeitfähigen Zustandsüberwachungssystems, welches unter Verwendung des CEPs eine Überwachung von Systemzuständen in Echtzeit erreicht, Störungsursachen identifiziert und den Zustand technischer Komponenten bewertet. Auf Basis dieser Informationen können im Rahmen der Instandhaltungsplanung gezielt Instandhaltungsaktivitäten eingeleitet werden. Diese Arbeit begründet sich durch die steigende Konfrontation mit hohen Maschinenkomplexitäten, die ein adäquates Instandhaltungsmanagement erschweren (Eickemeyer et al. 2014). Störereignisse wie der Ausfall einer bestimmten Komponente haben stets Auswirkungen auf andere Komponenten, Subsysteme und/oder das Gesamtsystem (Eickemeyer et al. 2014). In diesem Zusammenhang muss in Echtzeit entschieden werden, welche Maßnahmen zu ergreifen sind, um dies zu verhindern bzw. um die gestörte Ressource wieder in Stand zu setzen. Die Berücksichtigung sämtlicher Interdependenzen wird bisher nicht erreicht (Cordes 2018). Hierzu wird beabsichtigt, das im Finanzsektor entwickelte CEP auf die zustandsorientierte Instandhaltung zu übertragen. Der Einsatz der CEP-Technologie im Kontext der Zustandsüberwachung setzt die Entwicklung neuer Informationslandschaften, basierend auf ereignisgesteuerten Architekturen, voraus. Dazu wird einerseits ein CEP-konformer Entwurf eines Zustandsüberwachungssystems entwickelt. Darüber hinaus wird andererseits eine Konfigurationsmethodik als systemischer Rahmen entworfen. Hiermit können die Sensordaten verschiedener Komponenten in Echtzeit zueinander in Beziehung gesetzt werden. Aus diesem Grund fokussiert diese Dissertation ein klares Forschungsziel:

> Das Konzept des Complex Event Processing auf das Zustandsüberwachungssystem zu übertragen, um insbesondere komplexe technische Systeme in Echtzeit besser in Stand halten zu können.

1.4. Forschungsfragen und Vorgehensweise

Zur Erreichung dieses Gesamtzieles werden zwei Unterziele verfolgt: (a) Entwicklung eines Referenzarchitekturmodells bzw. eines Referenzmodells zur Implementierung des CEP in den zustandsorientierten Instandhaltungssystemen, (b) Entwicklung eines adaptiven Ansatzes zur Erstellung bzw. Anpassung von Ereignisregeln.

Das Referenzmodell (a) definiert den systemischen Rahmen und stellt einen strukturellen Gesamtüberblick über das zu betrachtende technische System zur Einbindung und Erstellung von Ereignisregeln unter Betrachtung aller involvierten Systeme (z. B. SCADA (engl. supervisory control and sata acquisition) System, Wartungsdaten etc.) dar. Die Beschreibung von komplexen technischen

Systemen erfolgt zumeist durch Modelle, welche das Verhalten des Systems beschreiben und sein Funktionalität darstellen. Unter einem Modell wird ein vereinfachtes, abstraktes Abbild des Systems verstanden, das die interessierenden Eigenschaften so genau wie nötig und möglich wiedergibt. Das Modell sollte helfen, Erklärungen für Vorgänge zu finden, die am realen System zu beobachten sind (Cordes 2018). Am Ende des Modellierungsprozesses steht ein Modell, dessen Verhalten als Vorhersage für das Verhalten des realen Systems dienen kann. Ziel der Arbeit ist es, die Prozesse so zu definieren, dass Transparenz geschaffen sowie ein erster Schritt in der Standardisierung und Übertragung des CEP-Konzeptes in die Instandhaltung vollzogen wird. Ein solches Referenzmodell existiert im Kontext des CEP-Einsatzes im ereignisorientierten Instandhaltungsbereich noch nicht. Dabei wird auf Modelle aus dem Ereignisverarbeitungsbereich sowie die Architektur aus dem OSA-CBM (Open System Architecture for Condition Based Maintenance (Lebold et al. 2003)) zurückgegriffen, um das ereignisorientierte Referenzmodell zu entwickeln. In Anlehnung an diese Modelle bzw. Architekturen wird das Referenzarchitekturmodell entwickelt. Vor diesem Hintergrund ist das erste Teilziel:

Modellierung von ereignisorientierten Instandhaltungsprozessen im Hinblick auf die Realisierung eines Referenzmodells für ein ereignisorientiertes Instandhaltungssystem.

Daraus ergibt sich die erste Forschungsfrage:

> Forschungsfrage 1: Wie muss das zustandsorientierte System im Hinblick auf seine Struktur entworfen werden, damit ein erfolgreicher Einsatz des CEP-Ansatzes realisiert werden kann?

Zur Erkennung von komplexen Ereignissen mithilfe von CEP (b) ist zudem die Generierung von Ereignisregeln erforderlich. Die technischen Systeme bestehen aus verschiedenen Komponenten bzw. Bauteilen verschiedener Hersteller, welche ständig Betriebsdaten aufnehmen und viele Daten erzeugen. Diese Daten enthalten Informationen über den Zustand des technischen Systems und deren Verarbeitung kann den Zustand ermitteln. Obwohl Funktionsinformationen der Komponenten jeweiliger Hersteller in den meisten Fällen in Form von Regelwerken oder Schwellenwert-Systemen vorliegen, sind diese aufgrund der Systemkomplexität und der Wechselwirkungen verschiedener Komponenten während der Betriebsphase nicht nutzbar oder schwer anwendbar. Diese vorgegebenen Schwellenwerte sind nicht mehr gültig oder brauchen Anpassungen, damit sie wieder die Funktionen der jeweiligen Komponenten erfüllen. Eine entsprechende Konfigurationsmethodik wird als wesentliche Voraussetzung für den Einsatz von CEP im Bereich der Instandhaltung angesehen, um aufgrund von natürlicher Alterung oder auch neu eingebauter Komponenten eine kontinuierliche Anpassung der Ereignisregeln zu ermöglichen (Peterson et al. 2016), (Şimşek at al. 2018), (Thoben et al. 2018). Aus diesem Grund muss gewährleis-

tet werden, dass die Ereignisregeln immer aktuell sind und bei Systemänderungen immer neu generiert werden. Vor diesem Hintergrund ist das zweite Teilziel:

Entwicklung eines adaptiven Ansatzes zur Generierung der Ereignisregeln mithilfe der Methoden der maschinellen Lernverfahren.

Daraus ergibt sich die zweite Forschungsfrage:

> Forschungsfrage 2: Wie muss das System konfiguriert werden, damit Sensordaten verschiedener Komponenten in Echtzeit zueinander in Beziehung gesetzt werden können, um globale Bedingungen richtig einschätzen zu können?

1.5. Aufbau der Arbeit

Diese Arbeit ist in sieben Kapitel aufgeteilt. Ausgehend von diesem ersten einleitenden Kapitel, in dem die Motivation und Problemstellung, Zielsetzung, Forschungsfragen und Vorgehensweise sowie die Struktur der Arbeit beschrieben sind, führt das Kapitel zwei die relevanten Instandhaltungsgrundlagen ein. Der Fokus wird dabei auf die zustandsorientierte Instandhaltung gelegt. Das nachfolgende Kapitel drei stellt die Grundlage des Data-Mining und Methoden der maschinellen Lernverfahren vor, welche für die erfolgreiche Implementierung der zustandsorientierten Instandhaltung erforderlich sind. Daran anschließend beschreibt Kapitel vier den Complex-Event-Processing-Ansatz und definiert den theoretischen Rahmen. Des Weiteren wird die Event-Driven Architecture (EDA) als Architekturstil eingeführt und das CEP als Technologie für die Verarbeitung von komplexen Ereignissen in einer EDA-Architektur ausführlich erläutert. Hierbei wird auf die wesentlichen Eigenschaften, Begriffe und Fachterminologien von CEP eingegangen. Innerhalb des fünften Kapitels wird das Konzept zur Integration bzw. Verknüpfung des CEP mit dem CM-System erarbeitet. Dies umfasst die Entwicklung eines Referenzmodells, die Anforderungsdefinition für die Implementierung des CEP-Systems in die Zustandsüberwachungssysteme sowie die Entwicklung der Methoden zur Erstellung der Ereignisregeln auf Grundlage der Anwendung von maschinellen Lernverfahren. Der Aufbau der Evaluation sowie deren Durchführung und die Darstellung dieser Ergebnisse stellen den Gegenstand des Kapitels sechs dar. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick im Kapitel sieben. Die nachfolgende Abbildung skizziert die Struktur der Arbeit und die Inhalte der einzelnen Kapitel.

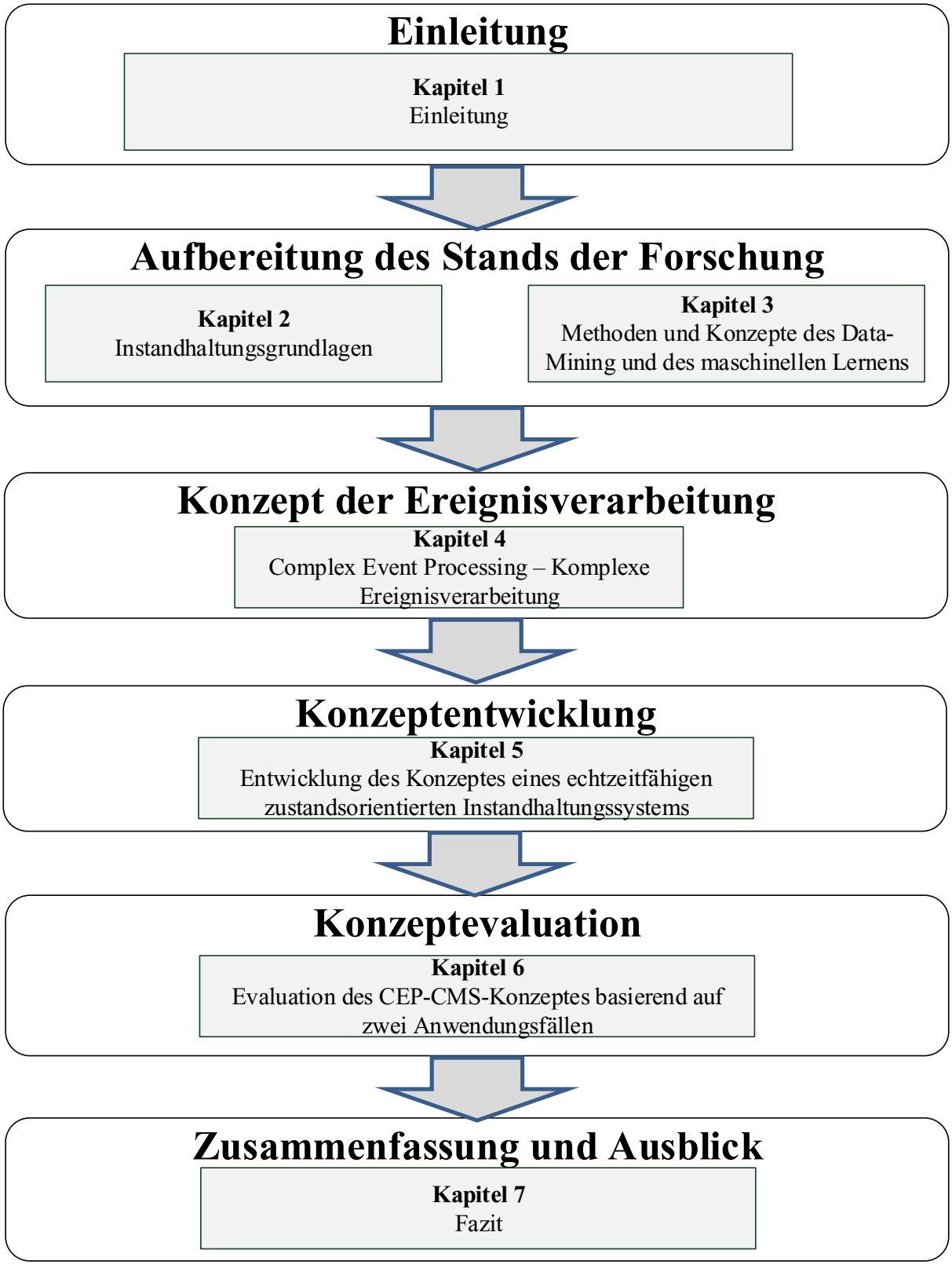

Abbildung 1: Struktur der Arbeit

2. Grundlage der Instandhaltung

Dieses Kapitel beginnt mit einer Einführung in die Instandhaltungsgrundlagen. Die historische Entwicklung der Instandhaltung vom Anfang der Industrialisierung bis heute wird erläutert. Zudem werden die verschiedenen Formen von Instandhaltungsstrategien vorgestellt. Der Fokus wird dabei auf die zustandsorientierte Instandhaltung gelegt. In diesem Zusammenhang werden auch die Grundlagen des Einsatzes der CBM vorgestellt.

Im folgenden Abschnitt werden die Definitionen der Begriffe vorgestellt, die im Rahmen dieses Kapitels verwendet werden.

2.1. Definitionen der Begriffe im Rahmen der Instandhaltung

- Betrachtungseinheit/Komponente
  - Ein diskretes Element eines technischen Systems, das sich im Laufe der Zeit entwickelt und eine definierte Funktion erfüllt. Es ist mit anderen Betrachtungseinheiten verbunden oder tritt in Wechselwirkung mit anderen Komponenten (DIN 31051).
- Degradierung
  - Ein irreversibler Entwicklungsprozess einer oder mehrerer Eigenschaften einer Betrachtungseinheit in Bezug auf Zeit, Nutzungsdauer oder andere externe Ursachen (Zemouri 2003).
- Defekt/Störung
  - Verlust der Fähigkeit einer Betrachtungseinheit, eine gewünschte Funktion mit definierter Leistung in der technischen Spezifikation durchzuführen. Eine gravierende Störung bringt das System zum Ausfallzustand (Walther 2011).
- Ausfall/Fehler
  - Anormaler Zustand, in dem eine Betrachtungseinheit bzw. Komponente nach Auftreten einer Störung unfähig ist, eine erforderliche Funktion durchzuführen (DIN 31051).
- Symptom/Störungszeichen
  - Ein Zeichen, das auf eine Störung bzw. einen Defekt hinweist. Die Überschreitung eines definierten Temperatur-Schwellenwertes oder die Erfassung von starken Vibrationswerten anhand von Schwingungssensoren sind einige Beispiele von Störungszeichen (Walther 2011).
- Diagnostik
  - Umfasst die Maßnahmen zur Erkennung der Störungen, der Lokalisierung der Komponenten am Ursprung der Störung und der Identifikation von Ursachen (DIN 31051).
- Prognostik

- Die Vorhersage von Störungen, bevor sie auftreten. Der Prognostik-Prozess ist als einer der wichtigeren Prozesse in der Instandhaltung zu betrachten. Anstatt nach einem Ausfall zu reagieren, hat es sich als ratsam erwiesen, den Ausfall zu antizipieren und entsprechend geeignete Reaktionen auszuführen (Muller et al. 2008).

2.2. Grundlage der Instandhaltung

Die Instandhaltung hat in der Produktion und Industrie aufgrund des rasant wachsenden Drucks durch Wettbewerb und der enormen Weiterentwicklung der Anlagen in den letzten Jahren stark an Bedeutung gewonnen, wobei es immer schwieriger wird, den Zustand einzelner Bauteile oder Baugruppen zu erfassen. Das Ziel der Instandhaltung ist, eine hohe technische Verfügbarkeit der Anlage zu gewährleisten bzw. Ausfälle früher zu erkennen und zu verhindern.

Die Instandhaltung in der Produktion zeichnet sich durch verschiedene Definitionen aus. Das British Standards Institute definiert „Instandhaltung“ als eine Kombination aller technischen und damit verbundenen administrativen Maßnahmen, die erforderlich sind, um Anlagen bzw. materielle Güter auf dem gewünschten Betriebszustand zu halten oder in den normalen Zustand zurückzuführen (BSI 1984). Die DIN 31051 definiert „Instandhaltung“ als die „Kombination aller technischen und administrativen Maßnahmen sowie Maßnahmen des Managements während des Lebenszyklus einer Betrachtungseinheit zur Erhaltung des funktionsfähigen Zustandes oder der Rückführung in diesen, so dass sie die geforderte Funktion erfüllen kann.“ (DIN 31051)

2.2.1. Entwicklungsstufen der Instandhaltung

Die Instandhaltung als Disziplin hat sich in den letzten Jahrzehnten enorm entwickelt. Nach ISKE (2009) hat die Entwicklung der Instandhaltung drei Phasen bzw. Generationen erlebt (siehe Abbildung 2). Während der ersten Generation (Anfang der Industrialisierung im 19 Jahrhundert bis 1960) wurde nicht viel Aufmerksamkeit auf die Instandhaltung gerichtet. Die Herstellung von Waren war nicht stark mechanisiert und die Anlagen bzw. Anlagenteile waren relativ einfach, in vielen Fällen sogar überdimensioniert. Des Weiteren wurden überwiegend korrektive Instandhaltungsaktivitäten bei Ausfällen durchgeführt. Dies ergab wenig oder keinen Bedarf an Instandhaltung im heutigen Sinn außer einer einfachen Reinigung, Wartung und Schmierung.

In der zweiten Generation (1960–1980) führte eine erhöhte Nachfrage nach Gütern zu einer verstärkten Mechanisierung und Komplexität der Anlagen. So kam es zu einer deutlichen Fokussierung auf die Ausfallzeiten, welche eine gravierende Auswirkung auf die Produktion aufwiesen. Dies führte zum Konzept der vorbeugenden Instandhaltung in Form von Instandsetzungen in festen Intervallen.

Mit zunehmender Komplexität der Fertigungsanlagen erhöhten sich auch die Erwartungen und Anforderungen an die Instandhaltung. Eine höhere Zuverlässigkeit, Verfügbarkeit und Sicherheit, eine längere Lebensdauer der Anlagen und erhöhte Anforderungen an die Wirtschaftlichkeit sind unter anderem das, was Instandhaltungsabteilungen in den letzten Jahren und in nahezu allen Bereichen zu gewährleisten hatten. Es bestand daher ein großer Bedarf an der Entwicklung von Techniken und die Anpassung der Instandhaltung an die neuen Anforderungen, was die Instandhaltung in der dritten Generation (1980–heute) vorantrieb (Quandt et al. 2017). In dieser Generation hat das Bewusstsein für die Bedeutung der Instandhaltung zugenommen. Zusammen mit der Forderung nach größerer Zuverlässigkeit bei geringeren Kosten kam es zu einem neuen Bewusstsein für Ausfallprozesse, verbesserte Managementtechniken und neue Technologien, um ein Verständnis für die Maschinen- und Komponentenzustände zu ermöglichen. Die Untersuchung des Risikos ist sehr wichtig geworden. Umwelt- und Sicherheitsfragen haben oberste Priorität. Neue Konzepte sind entstanden, wie State-of-the-Art-Zustandsüberwachung, Just-in-Time-Fertigung, Qualitätsstandards, Expertensysteme oder zustandsorientierte Instandhaltung, um nur einige zu nennen (Quandt et al. 2017).

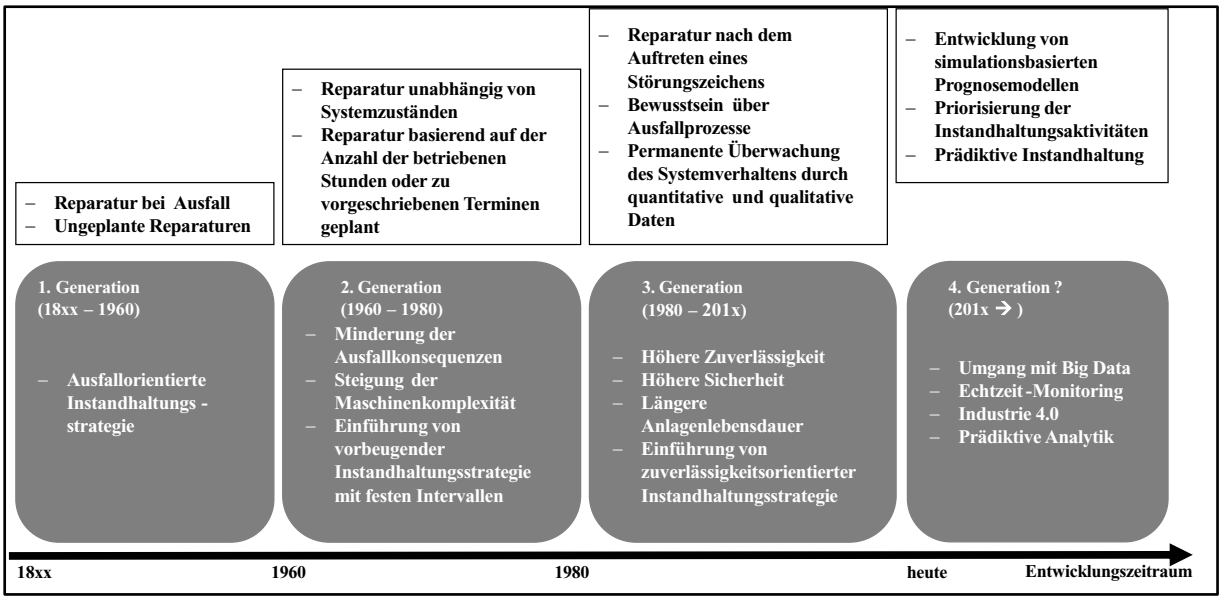

Abbildung 2: Historische Entwicklung der Instandhaltungsstrategien (in Anlehnung an ISKE 2009)

2.2.2. Vierte Phase der Instandhaltung

Der Aufstieg von Industrie 4.0 hat die Instandhaltungs-Community dazu veranlasst, über die vierte Instandhaltungsgeneration nachzudenken. Die Einführung neuer IT-Technologien und die Weiterverbreitung des Einsatzes von eingebetteten Systemen stellen einen weiteren Schritt in Richtung einer flexibleren Instandhaltung dar (Canizo et al. 2017). Im Zuge von Industrie 4.0 werden technische Systeme zu Cyber-physischen Systemen weiterentwickelt, die physikali-

sche Prozesse mit der Berechnung von eingebetteten Sensoren und Aktoren kombinieren (Reinhart et al. 2013). In dieser Vision werden die eingebetteten Systeme einen signifikanten Beitrag zur Entwicklung der digitalen und vernetzten Fabrik im Instandhaltungskontext leisten (Quandt et al. 2017). In diesem Kontext stellen Cyber-physische Systeme die Basis für den Aufbau von vernetzten Maschinen, Speichersystemen, Sensoren und IT-Systemen dar. Diese CPS-Systeme interagieren digital mit verschiedenen IT-Systemen, Sensoren und Maschinen, um die Entscheidungsunterstützung bei den Instandhaltungsaktivitäten zu optimieren. Dadurch können Wissen und Informationen am Ort der Instandhaltung dezentral bereitgestellt werden (Khalid et al. 2014).

Während die Aufgabe der zustandsorientierten Instandhaltung lange Zeit im Austausch von Verschleißkomponenten nach Analyse des Zustandes über Sensoren und Zustandsüberwachungssysteme bestand, liegt der Trend aktuell in der Entwicklung von simulationsbasierten Prognosemodellen. Diese Modelle kombinieren Daten aus Sensoren, Informationen aus Logdateien, im Voraus erhobene Betriebsdaten, Protokolldaten und Reparaturinformationen miteinander unter Zuhilfenahme von künstlicher Intelligenz und von maschinellem Lernen, um eine Prognose der Lebensdauer der Komponente zu ermitteln und die gegenseitige Beeinflussung von Ausfällen zu untersuchen (siehe Abbildung 2). In diesem Fall können zukünftige Störungen antizipiert werden (Austerjost et al. 2016).

2.2.3. Instandhaltungsstrategien

Grundsätzlich kommen verschiedene Instandhaltungsstrategien zum Einsatz (siehe Abbildung 3), die sich anhand des Planungsanlasses in die Kategorie reaktiv oder die Kategorie präventiv einordnen lassen (Austerjost et al. 2016), (Schenk 2010), (SS-EN 13006, 2001). Bei einer reaktiven Instandhaltungsplanung, die auch als schadensbezogene bzw. ausfallsorientierte Instandhaltung bezeichnet wird, werden die Instandhaltungsmaßnahmen nach dem Auftreten der eigentlichen Störung bzw. des Ausfalls geplant. In diesem Fall können die Instandhaltungsmaßnahmen entweder sofort nach Erkennen der Störung durchgeführt oder um einige Zeit aufgeschoben werden – solange die Störung nicht die Gesamtfunktionalität des Systems beeinflusst. Bei einer präventiven Instandhaltung werden hingegen vorbeugende Instandhaltungsmaßnahmen durchgeführt, um etwaige Ausfälle zu vermeiden. Des Weiteren kann die präventive Instandhaltung in eine zyklische-periodische, auch als zeitbezogene Instandhaltung bekannt, eine zustandsorientierte und eine vorausschauende Instandhaltung unterteilt werden. Bei einer zustandsorientierten Instandhaltung werden kritische Komponenten und Bauteile überwacht (z. B. durch Vibration oder Temperaturüberwachung), um den Zustand zu ermitteln und entsprechend notwendige Instandhaltungsmaßnahmen auszulösen. Die vorausschauende Instandhaltung stellt eine Weiterentwicklung der zustandsorientierten Instandhaltung dar, deren Ziel es ist, frühzeitige Hinweise auf einen Ausfall zu erkennen und entsprechend deren Auswirkungen zu verhindern (Schenk 2010).

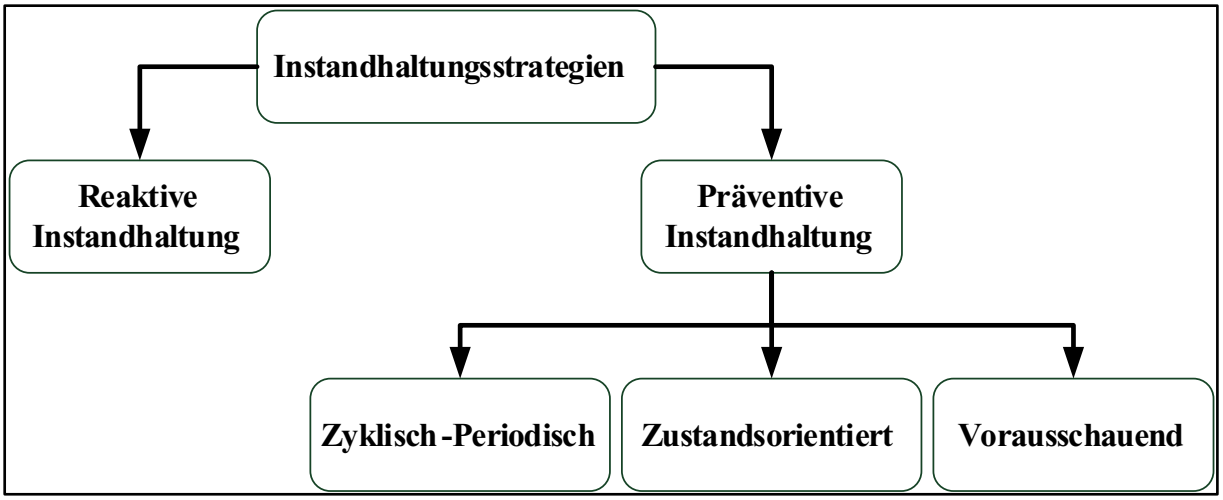

Abbildung 3: Instandhaltungsstrategien (in Anlehnung an Schenk 2010)

2.2.4. Auswahl der Instandhaltungsstrategie

In der Regel wird für komplexe technische Systeme immer noch eine präventiv-periodische Instandhaltungsstrategie durchgeführt (Lewandowski 2016). Bei einer präventiv-periodischen Instandhaltungsstrategie erfolgen die Wartungen bzw. die Inspektionen der Maschine nach einem bestimmten Zeitintervall. Obwohl die Verfügbarkeit der Maschine bei der präventiv-periodischen Instandhaltungsstrategie gewährleistet ist, stellt die Bestimmung der Intervalldauer eine Herausforderung dar. Des Weiteren kann die Maschine ohne Grund repariert und die Verschleißteile können unnötigerweise ausgetauscht werden. Dementsprechend entstehen Kosten für Wartungsszenarien, die eigentlich gar nicht notwendig sind. Zudem führen diese Instandhaltungsaktivitäten zum Maschinenstillstand und zu mehr Instandhaltungskosten. Das Auftreten von Störungen zwischen den festgelegten Perioden kann zudem nicht rechtzeitig detektiert werden, was zum Totalschaden führen kann. Durch die Entwicklung von technischen Systemen, die ein permanentes Monitoring der Sensordaten ermöglichen, ist es möglich, einen Wechsel zur zustandsorientierten Instandhaltungsstrategie vorzusehen.

Die Entscheidung, welche Instandhaltungsstrategie zu welchem System passt, hängt stark von den eingesetzten IT-Systemen, den Auswirkungen der auftretenden Störungen und dem Schwierigkeitsgrad einer Störungserkennung ab (Ryll 2008). Bei Störungen an kritischen Funktionen muss sofort eine korrektive Wartung durchgeführt werden. Für die Störungen jedoch, die keine oder nur geringe Auswirkungen auf die Gesamtsystemfunktion aufweisen, können die Wartungen zeitlich auf eine passendere Gelegenheit verlegt oder verschoben werden (Bengtsson 2007). In diesem Zusammenhang gibt (Starr 1997) an, dass eine korrektive Instandhaltung nur in nicht kritischen Bereichen angewandt werden sollte, also in Bereichen, in denen die Kapitalkosten und die Konsequenzen bzw. Schlussfolgerungen der Störungen gering sind und in denen keine Sicherheitsri-

siken vorliegen und eine schnelle Fehlererkennung bzw. -behebung möglich ist (Abbildung 4).

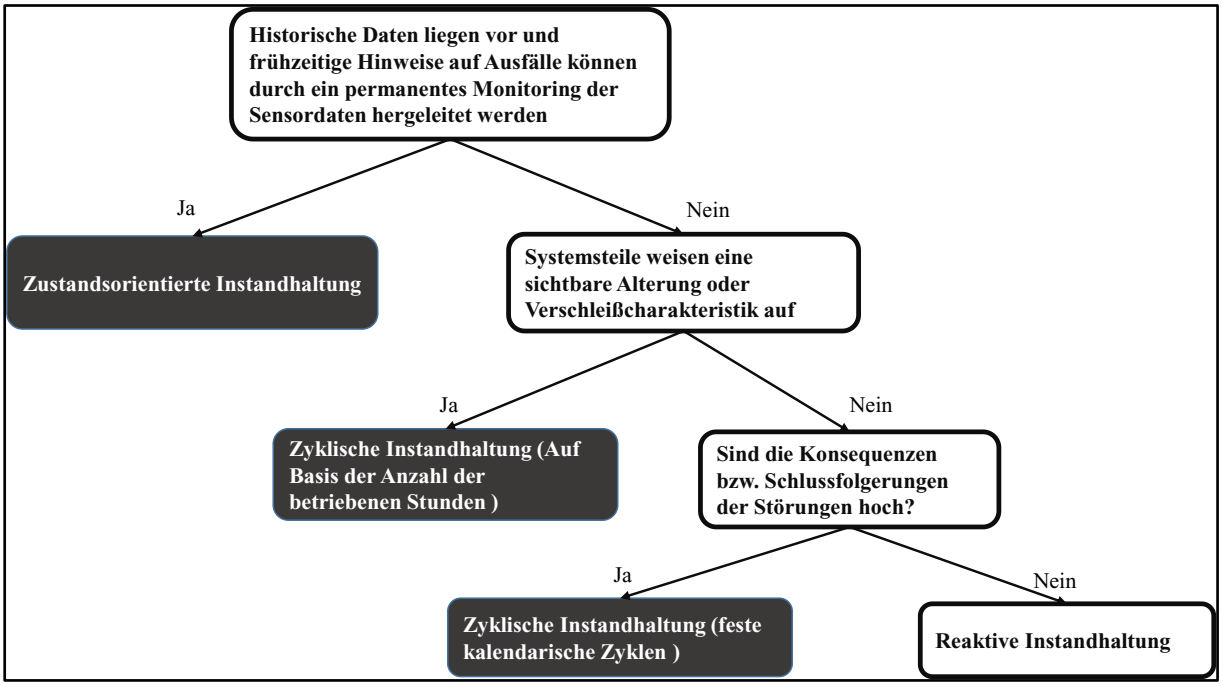

Abbildung 4: Selektionslogik einer Instandhaltungsstrategie

Darüber hinaus ist die präventive zyklische Instandhaltung als eine Wartung definiert, die nach vorgeschriebenen Kriterien in bestimmten Zyklen durchgeführt wird, um die Ausfallwahrscheinlichkeit oder Verschlechterung der Systemfunktion zu verringern. In diesem Kontext werden häufig die zyklischen Instandhaltungsmaßnahmen ohne das Auftreten eines Störungszeichens angestoßen. Die Instandhaltungsaktivitäten werden dann basierend auf der Anzahl der Betriebsstunden, der gefahrenen Kilometer oder vorgeschriebenen Terminen geplant. Dadurch eignet sich die zyklische Instandhaltung am besten für eine Betrachtungseinheit, die eine sichtbare Alterung oder eine Verschleißcharakteristik aufweist. In diesem Zusammenhang lassen sich die Ausfälle meistens verhindern, wenn die Wartungsaktivitäten rechtzeitig durchgeführt werden können (Bengtsson 2007). Wenn viele unnötige zyklische Instandhaltungsaktivitäten festgestellt werden, historische Daten vorliegen und durch Anwendung von Methoden des Data-Mining und maschinellen Lernens mittels Zustandsüberwachungssystemen die Systemzustände bzw. die Störungszeichen erkennen lassen, eignet sich die zustandsorientierte Instandhaltung am besten.

Diese Selektionslogik einer Instandhaltungsstrategie soll nicht unbedingt nahelegen, dass andere Strategien, die nach der Logik nicht in Frage kommen, ausgeschlossen sind und nicht weiter in Betracht gezogen werden sollten, sondern sie gibt an, welche Instandhaltungsstrategie für bestimmte Maschinen oder Anwendungsbereiche mehr Gewicht hat bzw. theoretisch am besten geeignet ist (Ryll 2008).

Untersuchungen haben jedoch gezeigt, dass die zustandsorientierte Instandhaltung nicht auf breiter Basis durchgeführt wird (Lewandowski 2016). In diesem Zusammenhang wird in der Regel für komplexe technische Systeme immer noch eine präventiv-periodische Instandhaltungsstrategie in Kombination mit einer zustandsorientierten Instandhaltung eingesetzt (Lewandowski 2016).

2.3. Zustandsorientierte Instandhaltung – CBM

Mit der rasanten Entwicklung der modernen Technik und der Informationstechnologie (IT) sind die Anlagen und dementsprechend die Instandhaltungsprozesse komplexer geworden, während die Anforderungen an Qualität und Zuverlässigkeit gestiegen sind. Das hat zur Folge, dass die Kosten für eine zyklische Instandhaltung sehr hoch und schließlich zu einem der wichtigsten Kostenfaktoren vieler Unternehmen geworden sind (Canizo et al. 2017). Aus diesem Grund wurden effizientere Instandhaltungsstrategien umgesetzt, um diese Mängel der zyklischen Instandhaltungen zu bewältigen. Bei vielen industriellen Anwendungen ist die Nachfrage danach aufgekommen, die zyklische Instandhaltungsstrategie durch eine zustandsorientierte zu ersetzen (Koochaki 2009). Dieser Wechsel erfordert einige technologische Verfahren, moderne Informations- und Kommunikationstechnologien (IKT) und gute Kenntnisse über geeignete Analyseverfahren. Eine permanente Überwachung des Systemverhaltens durch quantitative (Sensordaten) und qualitative (Reparaturdaten, Erfahrung der Experten, usw.) Daten ermöglicht somit den Aufbau eines robusten und flexiblen Entscheidungssystems, mit dem Störungen früh erkannt und Fehlfunktionen verhindert werden können, bevor diese auftreten. Anschließend wird eine Alarm-Meldung ausgelöst, wodurch die Produktion ggf. verlangsamt werden kann (Basseville et al. 1996). Die zustandsorientierte Instandhaltung erlebt nicht zuletzt aufgrund der rapiden Entwicklung der Sensortechnik und der Informationstechnologie einen erheblichen Aufschwung (Bengtsson 2007), (Wenzel et al. 2011). Die Vernetzung der Maschinen hat zugenommen und somit auch die Möglichkeiten einer Online-Überwachung verschiedener instandhaltungsrelevanter Betriebsparameter. Trotz höherer Investitionskosten erlaubt dies eine Ablösung von zyklischen Instandhaltungsstrategien oder manuellen Zustandserfassungen, da sich Maschinenverfügbarkeiten und Instandhaltungsaufwände deutlich reduzieren lassen. Dieser Wechsel von einer „Ausfall erleidenden“ zu einer „Ausfall kontrollierenden“ Instandhaltung erfordert die Analyse sowohl quantitativer als auch qualitativer Daten. Jardine et al. (2006) stellen fest, dass sich unter der Voraussetzung einer durchdachten Implementierung und Systemkonfiguration die Instandhaltungskosten bei einer zustandsorientierten Instandhaltung deutlich reduzieren lassen, indem überflüssige zyklische Instandhaltungsaktivitäten vermieden werden (Jardine et al. 2006).

Idealerweise sollen die Störungen mit dem Einsatz von CBM rechtzeitig detektiert und die Instandhaltungsmaßnahmen dementsprechend frühzeitig angestoßen werden können. In diesem Kontext werden die Ersatzmaterialien und die

benötigten Ressourcen vorbereitet und die Instandhaltungsaktivitäten gut geplant, um mögliche lange Stillstände der Maschinen zu vermeiden. Die folgende Abbildung 5 veranschaulicht den Zusammenhang zwischen Abnutzungsvorräten (bzw. Systemzuständen) und den Auslösungszeiten der Instandhaltungsmaßnahmen für CBM bzw. periodische Instandhaltung.

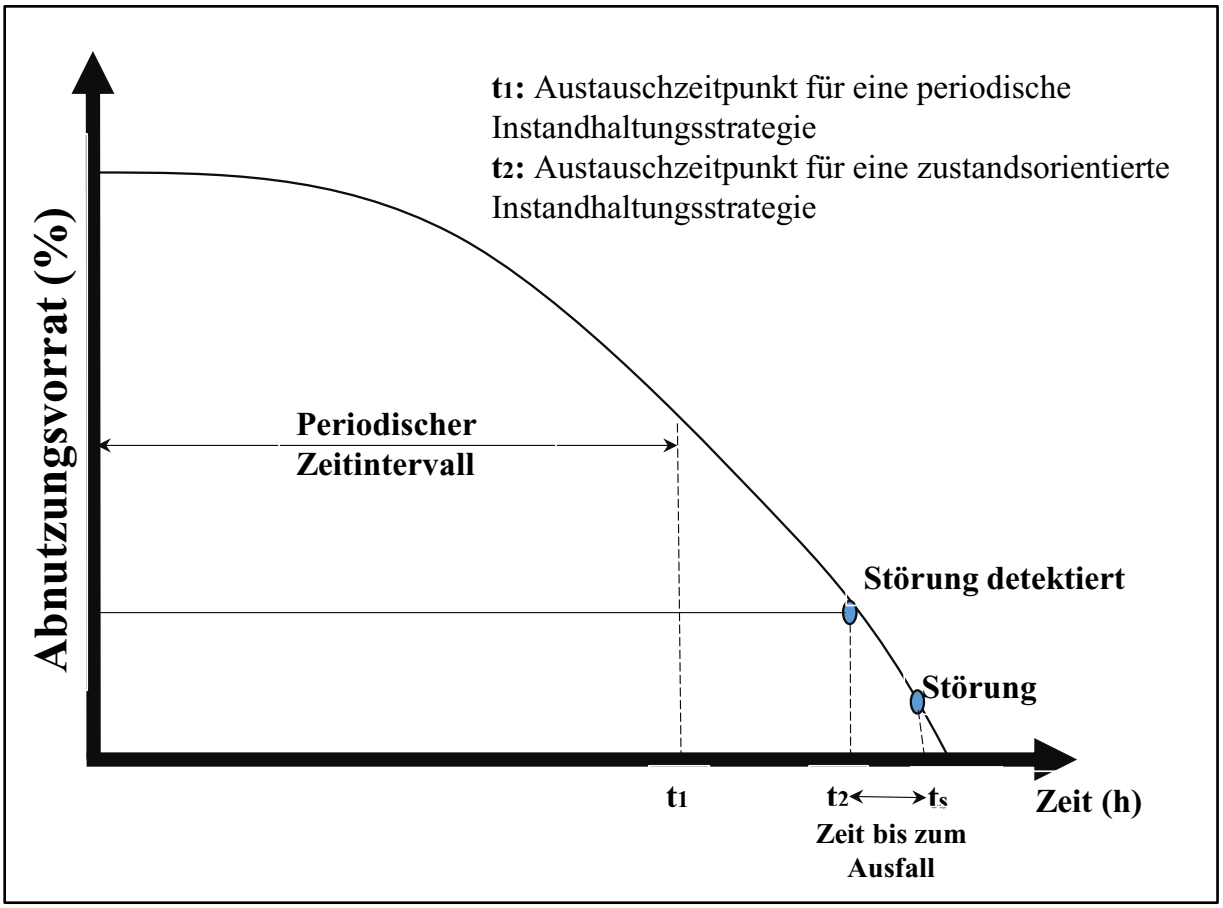

Abbildung 5: Zusammenhang zwischen dem Abnutzungsvorrat und den Störungsauslösungszeiten in einer periodischen bzw. zustandsorientierten Instandhaltungsstrategie (in Anlehnung an Ryll 2008)

Eine zustandsorientierte Instandhaltung soll so ausgelegt sein, dass die Zeit zwischen der Störungsdetektion und dem Ausfall der Maschine nicht zu kurz ist, um dann wiederum die Nachteile der präventiv-periodischen Instandhaltungsstrategie zu haben, und nicht zu lang, womit wieder eine korrektive Instandhaltungsstrategie vorliegt. Außerdem ist bei einer zyklischen Instandhaltung die Zeit bzw. die Anzahl der betriebenen Stunden die Steuergröße für den Anstoß der Inspektionen, während bei einer CBM der Zustand die Steuergröße für die Durchführung der Instandhaltungsaktivitäten ist (Ryll 2008).

Jardine et al. (2006) haben in einem Review die relevanten Forschungsarbeiten, die sich mit zustandsorientierter Instandhaltung beschäftigen, zusammengefasst. Sie berichten, dass eine vernünftige Implementierung von zustandsorientierter Instandhaltung die Instandhaltungskosten durch Vermeidung von überflüssigen präventiven periodischen Instandhaltungsaktivitäten bedeutend reduzieren kann.

Sie definieren drei wesentliche Schritte für einen erfolgreichen Einsatz der zustandsorientierten Instandhaltung:

1. Datenerfassung (Daten sammeln und speichern): Heutzutage sind Sensornetzwerke ein wichtiger Bestandteil von Instandhaltungssystemen. Durch den Einsatz von Sensoren zur Messung von physikalischen Werten verschiedener Komponenten kann ein kontinuierliches Monitoring der Komponenten durchgeführt werden. In diesem Schritt wird zwischen zwei Typen von Daten unterschieden: zustandsorientierte Daten (condition monitoring data) und Ereignisdaten (event data). Bei zustandsorientierten Daten handelt es sich um Sensordaten, während es sich bei Ereignisdaten um Informationen bezüglich der letzten durchgeführten Installationen, präventiven bzw. korrektiven Wartungen, Ölwechsel, Ursachen der Ausfälle usw. handelt.
2. Datenverarbeitung (der Umgang mit den Informationen): Dieser Schritt besteht aus der Filterung und Analyse der Daten. Bei der Filterung werden fehlerhafte und doppelte Einträge aus den erhobenen Daten entfernt. In der Datenanalyse werden Methoden und Ansätze für die Verarbeitung und Auswertung der gefilterten Daten entwickelt. Jardine et al. (2006) präsentieren einen Überblick über verschiedene Methoden für die Verarbeitung von Daten für verschiedene Datentypen (einzelner Datentyp, multidimensionaler Datentyp).
3. Entscheidungsprozess: Die Verfahren für Entscheidungsprozesse werden in zwei Kategorien aufgeteilt: Diagnostik und Prognostik. Bei der Diagnostik geht es um die Detektion, Isolation und Identifikation von Störungen, wenn diese auftreten. Bei der Prognostik hingegen handelt es sich um die Vorhersage von Störungen, bevor diese auftreten. Die Autoren geben hier eine Übersicht über statistische Ansätze, Methoden der künstlichen Intelligenz und über modellbasierte Ansätze, die im Entscheidungsprozess angewendet werden können.

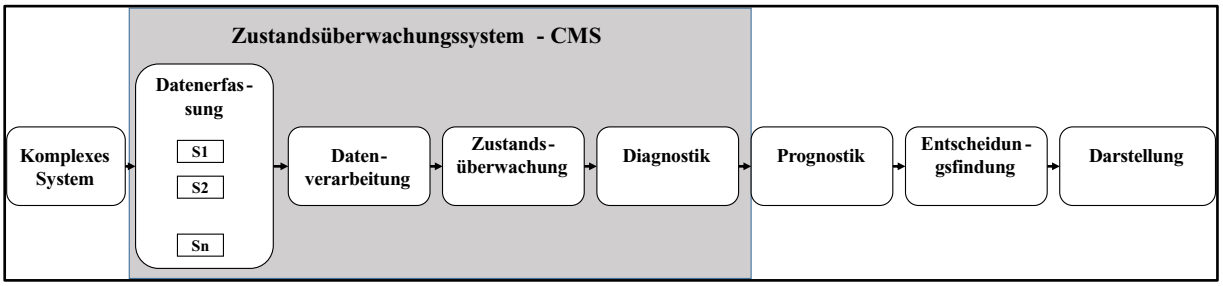

Abbildung 6: Open System Architecture for CBM (in Anlehnung an Lebold et al. 2003)

Für die Umsetzung von CBM hat sich u. a. die OSA-CBM Architektur (Open System Architecture for Condition Based Maintenance) etabliert (Abbildung 6) welche eine Referenz-Implementierung der ISO-13374 darstellt (Lebold et al. 2003). Die OSA-CBM besteht aus sieben Funktionsmodulen, die bezüglich der verwendeten Methoden weiter ausgestaltet werden können und sowohl Daten-

strukturen als auch Schnittstellen definieren, wobei die Vereinheitlichte Modellierungssprache (engl. unified modeling language – UML) zur Spezifikation verwendet wird.

Ausgehend von der Datenerfassung in der OSA-CBM-Architektur werden dem OSA-CBM-basierten Wartungssystem kontinuierlich Sensordaten zur Verfügung gestellt. Im Rahmen der Datenverarbeitung kommen Methoden der Signalverarbeitung (engl. signal processing) zum Einsatz, um spezifische, aus den Rohdaten abgeleitete Merkmale zu extrahieren. Es schließt sich die Zustandsüberwachung an, wobei die extrahierten Merkmale hinsichtlich der Schwellwertüberschreitung analysiert und mit anderen Zustandsindikatoren verglichen werden, um gegebenenfalls entsprechende Warnungen zu generieren. Im Rahmen der Diagnostik werden die Ursachen für die Degradation ermittelt und Vorschläge für eine Instandsetzung des Systems gemacht.

Um die Störungen frühzeitig zu erkennen und dementsprechend das Instandsetzen zu einem früheren Zeitpunkt zu planen, ist eine Prognose erforderlich. Bei der Prognostik geht es um die Vorhersage des Systemszustands in der Zukunft (Muller et al. 2008). Dabei wird auf Methoden der künstlichen Intelligenz und mathematischer Verfahren zurückgegriffen, um Systemverhalten in der Zukunft vorherzusagen. Das Gesamtsystem soll damit in der Lage sein, die Störungen früh zu melden. Bei der Berechnung der Prognose handelt es sich allgemein um eine Zeitreihenanalyse auf Basis der Sensorsignale. Die Prognose basiert auf vorherigen, vergangenheitsbasierenden Daten, um die zukünftigen Daten zu berechnen. Damit haben die Quantität und die Qualität der vorher betrachteten Daten großen Einfluss auf die Qualität der Prognose. Im Rahmen der Entscheidungsfindung wird die rechtzeitige Einleitung von Instandhaltungsmaßnahmen auf Basis der Daten aus dem Diagnostikmodul und dem Prognostikmodul ermittelt. Abschließend geht es um die Benutzerschnittstelle, bei der eine Aufbereitung der Daten für den Menschen erfolgt und verschiedene Systemzustände möglichst prägnant visualisiert werden. Aufbauend auf der OSA-CBM-Architektur wird die Kernfunktion dem Zustandsüberwachungssystem (engl. condition monitoring system) entsprechend als Kombination der Funktionsmodule „Datenerfassung, Datenverarbeitung, Zustandsüberwachung und Diagnostik“ identifiziert (Lebold et al. 2003).

Eine Reihe von Arbeiten zeigt die erfolgreiche Anwendung von CBM für einfache bzw. nicht komplexe Szenarien mit klaren Systemgrenzen, wie es bei einzelnen Komponenten oder Maschinen der Fall ist.

Bey-Temsamani et al. (2009) haben eine zustandsorientierte Instandhaltung implementiert, wobei die Autoren Data-Mining-Methoden und eine Wahrscheinlichkeitsanalyse auf die historischen Daten angewandt haben, um die relevanten Features für die Bestimmung des Zustandes zu identifizieren. Darüber hinaus wurde ein Vorhersage-Modell mit den Live-Daten der Anlage erstellt, welches die Vorhersage der zukünftigen Entwicklung der Merkmale und Prognose des Zeitintervalls bis zur nächsten Instandhaltungsaktion ermöglichte.

Koochaki (2009) hat die Vorteile der Integration von kollaborativem Lernen innerhalb der Asset-Life-Cycle-Prozesse zwischen verschiedenen Betrieben und sämtlichen Disziplinen der Instandhaltung in dem Erfolg von CBM-Implementierung untersucht. Er berichtet, dass die in der Literatur erwähnte Erfolgsgeschichte von CBM-Implementierungen in der Realität nicht anwendbar oder finanziell gerechtfertigt ist. Sein Argument beruht auf dem Fakt, dass die Verarbeitung des Wissens und der Informationen fehlt, die durch die Interaktion mit allen beteiligten CBM-Abteilungen und Institutionen zustande kommt. Aus diesem Grund führt er den Begriff des kollaborativen Lernens ein, um diese Mängel zu bewältigen. Der Lernprozess findet dann im Datenerfassungsschritt und im Entscheidungsprozessschritt statt.

In der Fehlerdiagnose kommen häufig Methoden der Künstlichen Intelligenz (KI) zum Einsatz, um den Systemzustand in „normal“ bzw. „anormal“ klassifizieren zu können. Hayashi et al. (2002) haben hierfür ein überwachtes Lernverfahren basierend auf künstlichen neuronalen Netzen (KNN) verwendet, mit dem sich Störungen anhand von gelernten normalen bzw. anormalen Systemzuständen relativ sicher identifizieren ließen (Hayashi et al. 2002). Nicht lineare Regressionsmethoden wurden ebenfalls verwendet, um verschiedene Submodelle mit Support-Vector-Maschinen zu erlernen (Odenthal et al. 2014). Sofern es sich um noch nicht klassifizierte Daten handelt, kann auf unüberwachte Lernverfahren zurückgegriffen werden. Zum Beispiel ließen sich Verfahren der Clusteranalyse verwenden, um verschiedene Störungskategorien auf Basis von Eigenschaftsähnlichkeiten zu ermitteln (Lieber et al. 2013). Des Weiteren können auch andere Verfahren des Data-Mining verwendet werden, um relevante Merkmale für die Bestimmung des Systemzustands zu identifizieren (Bey-Temsamani et al. 2009) oder die Zuverlässigkeitsanalyse des Systems durchzuführen (Rieker et al. 2015). Mittels Zeitreihenprognosen können zudem einzelne Merkmale für die Zukunft vorausberechnet werden, um eine Vorhersage des zu erwartenden Systemzustands zu erhalten (Denkena et al. 2012).

2.4. Grundlagen der Zuverlässigkeitsanalyse von technischen Systemen

Die Zuverlässigkeit eines Systems ist die Eigenschaft eines solchen, eine definierte Funktion in einem Zeitintervall zu erfüllen. Zur Beschreibung der Zuverlässigkeit eines Systems wird je nach dessen Komplexität zwischen quantitativen bzw. qualitativen Zuverlässigkeitsanalysen unterschieden (Ryll 2008), (Lewandowski 2016).

Bei der qualitativen Zuverlässigkeitsanalyse werden zur Beschreibung von möglichen Folgen eines auftretenden Fehlers häufig die Methoden der Fehlermöglichkeits- und Einflussanalyse (engl. failure mode and effect analysis – FMEA), Fehlerbaumanalyse und Ereignisbaumanalyse (engl. event tree) angewendet (Lewandowski 2016). Daher ermöglicht eine qualitative Zuverlässigkeitsanalyse eine funktionelle und strukturelle Systemanalyse (Beugin 2006).

Bei der quantitativen Zuverlässigkeitsanalyse geht es um eine empirische Analyse auf Basis einer statistischen Analyse mittels verschiedener metrisch messbarer Größen. In diesem Kontext wird die Zuverlässigkeit oft als Wahrscheinlichkeit verschiedener Größen gemessen und dadurch bewertet. Um das Ziel einer zustandsorientierten Instandhaltung zu erreichen, werden verschiedene Zuverlässigkeitsgrößen definiert, die für die Zuverlässigkeitsanalyse besonders von Interesse sind. Aus dem Bereich der Statistik wurden im Rahmen der Instandhaltung folgende Größen zur Beschreibung der Zuverlässigkeitsanalyse eingeführt: Ausfallwahrscheinlichkeit, Überlebenswahrscheinlichkeit, Ausfalldichte und Ausfallrate (Ryll 2008).

Die Ausfallwahrscheinlichkeit F(t) gibt an, wie groß die Wahrscheinlichkeit eines Komponentenausfalls bis zum Zeitpunkt t ist.

Hingegen gibt die Überlebenswahrscheinlichkeit R(t) an, mit welcher Wahrscheinlichkeit kein Komponentenausfall bis zum Zeitpunkt t auftritt.

$$R(t) = 1 - F(t)$$

Eine andere Darstellungsform der Zuverlässigkeitsanalyse stellt die Ausfalldichte f(t) dar. Die Ausfalldichte ist die Ableitung der Ausfallwahrscheinlichkeit F(t). Das folgende Integral berechnet die Wahrscheinlichkeit, dass ein Ausfall zwischen den Zeitpunkten t_1 und t_2 auftritt.

$$\int_{t1}^{t2} f(t)dt = F(t2) - F(t1)$$

Eine weitere interessante Größe der Zuverlässigkeitsanalyse ist die Ausfallrate λ(t). Sie gibt an, mit welcher Wahrscheinlichkeit ein zum Zeitpunkt t1 funktionierendes System zum Zeitpunkt t1 + Δt mit Δt → 0 ausfällt. Die Formel zur Berechnung der Ausfallrate λ(t) ist:

$$\lambda(t) = \frac{f(t)}{1-F(t)} = \frac{f(t)}{R(t))}$$

Der Verlauf der Ausfallrate wird öfter aufgrund der graphischen Darstellung als „Badewannenkurve“ bezeichnet (Abbildung 7).

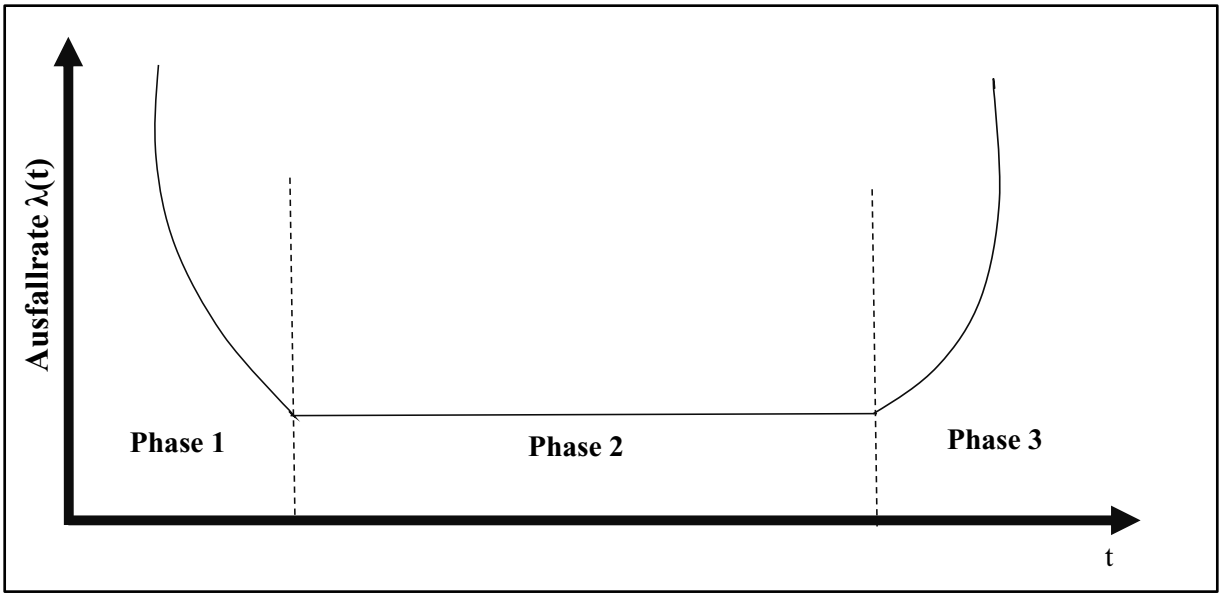

Abbildung 7: Verlauf der Ausfallrate – aufgrund der graphischen Darstellung oft als „Badewannenkurve“ bezeichnet (in Anlehnung an Ryll 2008)

Der Verlauf der Ausfallrate ist durch drei Phasen gekennzeichnet. In der ersten Phase fällt die Ausfallrate drastisch ab. Frühfehler, grundsätzlich verursacht durch Fertigungs- bzw. Konstruktionsfehler, treten am Anfang vermehrt auf, werden aber in der ersten Phase nach und nach behoben, wodurch das System in einen akzeptablen Betriebszustand befördert wird. Daran schließt sich die Phase 2 an, die ein normales Betriebsverhalten eines Systems darstellt und eine relativ konstante Ausfallrate aufweist. In dieser Phase werden die Ausfälle durch Umweltbeanspruchungen, unzulässige Belastungen oder Bedienungsfehler verursacht (Ryll 2008). In der letzten Phase treten neben den bereits aufgeführten Ausfällen der Phase 2 noch abnutzungsbedingte Ausfälle durch den Alterungsprozess der Bauteile auf, was zur Steigerung der Ausfallrate führt.

Für die Bestimmung der bereits genannten Zuverlässigkeitsgrößen werden einige typische Verteilungsfunktionen zur Beschreibung des Ausfallverhaltens verwendet. Diese Verteilungen sind die Normalverteilung, auch als Gauss-Verteilung bekannt, die Exponential- und Weilbullverteilung (Ryll 2008), (Lewandowski 2016), (Warnecke 2008). Je nach Ausfallverhalten eines Systems eignet sich eine der Verteilungsfunktionen zur Beschreibung der Ausfallwahrscheinlichkeit. In diesem Zusammenhang eignet sich die Exponentialverteilung für die Beschreibung der Zuverlässigkeit von elektronischen Bauelementen. Die Weibullverteilung findet ihre Anwendung in der Beschreibung der Zuverlässigkeit von mechanischen Bauelementen (Ryll 2008).

2.5. Andere Varianten von Instandhaltungsstrategien

Es existieren neben den oben genannten Instandhaltungsstrategien noch andere Varianten von Strategien, welche die Ziele der vorbeugenden Instandhaltung

verfolgen und durch verschiedene Kontexte entstanden sind. Im folgenden Abschnitt werden diese Strategien bzw. Instandhaltungskonzepte eingeführt.

2.5.1. Total Productive Maintenance (TPM)

Total Productive Maintenance (TPM) ist eine Variante der vorbeugenden Instandhaltung, die 1971 in Japan entstanden ist. TPM ist eine Weiterentwicklung der damals bekannten Instandhaltungsmethoden, insbesondere der präventiven Instandhaltung aus Amerika, mit dem Ziel, die Maschinenleistung durch einen proaktiven Ansatz zu verbessern. Zu diesem Zweck bemüht sich die TPM um die Effektivität der Maschinen mittels folgender Ziele: i) Keine Defekte ii) Keine Maschinenstillstände oder Leerläufe iii) Keine Störungen iv) Keine Prozessfehler oder Unfälle, v) Keine Qualitätsverluste. Die Implementierung bzw. die Umsetzung der TPM erfordert ein autonomes Instandhaltungsprogramm, welches eine gemeinsame Verantwortung der Instandhaltungsziele schafft und starkes Mitarbeiterengagement bzw. starke Mitarbeiterbeteiligung fördert (Bengtsson 2007), (ISKE 2009).

2.5.2. Reliability-Centered Maintenance (RCM)

Zuverlässigkeitsorientierte Instandhaltung (engl. reliability-centered maintenance – RCM) ist ein strukturierter Ansatz für die Einrichtung eines Instandhaltungsprogramms (Bengtsson 2007). Es ist als ein Prozess definiert, der dafür verwendet wird, festzustellen, was getan werden muss, um zu gewährleisten, dass ein technisches System weiterhin den gewünschten Betriebszustand hält und seine Funktion in seinem derzeitigen Betriebskontext erfüllt (Moubray 1997), (Bengtsson 2007). Des Weiteren ist RCM ein Ansatz, der sich mit folgenden Fragestellungen befasst (Moubray 1997):

1. Was sind die Funktionen und die damit verbundenen Leistungsstandards des Systems in seinem derzeitigen Betriebskontext?
2. Unter welchen Umständen werden die Aufgaben nicht erfüllt?
3. Was sind die Ursachen einer jeden Funktionsstörung?
4. Was geschieht, wenn die jeweilige Störung auftritt?
5. Wie wirkt sich eine Störung aus?
6. Was kann getan werden, um jeden Ausfall vorherzusagen oder zu verhindern?
7. Was ist zu tun, wenn keine geeignete proaktive Aufgabe gefunden werden kann?

2.5.3. E-Maintenance

E-Maintenance, auch eMaintenance, steht für exzellente, effektive, effiziente und Enterprise Maintenance (Levrat et al. 2008). Sie stellt sich als ein aufstre-

bendes Konzept für die Integration der neuen Informations- und Kommunikationstechnologie in die Instandhaltungsstrategien und Instandhaltungsplanung dar (Muller et al. 2008). Nach Candell et al. (2009) umfasst E-Maintenance die Überwachung, Erfassung, Speicherung und Verteilung von Echtzeit-Systemen, Instandhaltungs-generierten Daten sowie anderer Entscheidungs- und Performance-Support-Informationen an verschiedene Interessengruppen, unabhängig von Organisations- oder geographischer Lage. Der Einsatz von E-Maintenance bietet durch die rasante Entwicklung von Hard- und Software, Wireless-Sensoren und IKT einen schnelleren Austausch von Informationen zwischen verschiedenen Interessengruppen und somit die Vorteile der Bearbeitung dieser Menge von Informationen in einer kollaborativen Art und Weise, um zu einer effizienten Instandhaltung beizutragen (Candell et al. 2009). Diese Kollaboration und Kooperation zwischen verschiedenen Interessengruppen wird meistens durch die Anwendung eines Web-Services als Kommunikationsschnittstelle und durch Einsatz von serviceorientierter Architektur als Plattform-Architektur realisiert.

3. Grundlagen des Data-Mining und des maschinellen Lernens

Im vorherigen Kapitel wurden die Grundlagen der verschiedenen Instandhaltungsstrategien vorgestellt. Der Fokus lag auf der zustandsorientierten Instandhaltung. Voraussetzungen der Umsetzung der zustandsorientierten Instandhaltung sind die Entwicklung moderner IT-Technologien und Methoden zur Erfassung von Mess- und Betriebsdaten aus komplexen Systemen sowie leistungsfähige Zustandsüberwachungssysteme, die ein permanentes Monitoring der erfassten Sensordaten ermöglichen. In diesen Zustandsüberwachungssystemen werden die von Maschinen und Anlagen erfassten Daten verarbeitet, ausgewertet und anschließend daraus Informationen und Wissen über den Anlagenzustand gewonnen. Dadurch lassen sich permanente Informationen über den aktuellen Systemzustand und die Betriebsbedingungen ermitteln und somit die Instandhaltungsprozesse optimieren. Die gewünschten Kostenreduzierungen, die mithilfe der Anwendung von künstlicher Intelligenz und maschinellem Lernen erzielt werden könnten, werden nicht in einem Schritt erreicht, denn der Datenverarbeitungsprozess muss die verschiedenen Stufen der Datenauswertung Schritt für Schritt durchlaufen. Das bedeutet, dass die Daten erst mal aufgezeichnet und aufgenommen werden müssen. Durch die Analyse und Weiterverarbeitung dieser Daten werden daraus Key-Performance-Indikatoren entwickelt, die helfen, bei der Steuerung und Planung der Instandhaltungsmaßnahmen einzugreifen. Für die Verarbeitung und Auswertung von verschiedenen erfassten Daten nimmt die Entwicklung von Methoden und Verfahren des Data-Mining (DM) und von maschinellen Lernverfahren eine entscheidende Rolle ein.

In diesem Kapitel werden die Grundlagen des Data-Mining vorgestellt. Des Weiteren wird ein allgemeiner Überblick über das Thema Data-Mining und ein kurzer Einblick in Data-Mining-Prozesse und Analytics gegeben. Verschiedene Methoden der maschinellen Lernverfahren werden eingeführt, welche im Rahmen dieser Arbeit verwendet werden. Dabei wird zwischen drei Hauptklassen von Lernverfahren unterschieden: Supervised, Unsupervised und Reinforcement Lernmethoden.

3.1. Data-Mining-Evolution über die Zeit

Im Allgemeinen ist Data-Mining der Prozess der Analyse von Daten aus verschiedenen Perspektiven mit dem Ziel, die Daten in nützliche Informationen zusammenzufassen, um beispielsweise Anomaly Detektion, verdächtige Situationen und das Verhalten des Systems festzustellen (Bramer 2013). DM ermöglicht, Daten aus verschiedenen Dimensionen zu analysieren und zu kategorisieren, um abschließend die Zusammenfassung von identifizierten Beziehungen zu ermöglichen. Data-Mining verfolgt dabei das Ziel, eine automatische und nicht triviale Suche nach Wissen in Massendaten durchzuführen.

Der Begriff Data-Mining wurde in den 1980er Jahren eingeführt, jedoch hat das Konzept des DM seine Wurzel schon vor vielen Jahren mit dem Einsatz von statistischer Analyse etabliert. Data-Mining hat seine jetzige Form erreicht, nachdem es mehrere Phasen Forschung und Studien durchlaufen hatte. Die richtige Evolution des Data-Mining begann, als Daten auf Computern bzw. in Datenbanken gespeichert wurden. Grundsätzlich wurde der Prozess mit den Fortschritten in der Informations- und Kommunikationstechnik einschließlich Datenspeicher (Datenbanken), Computer-Hardware, Rechenleistung und neuer Algorithmen fortgesetzt.

Der Einsatz von Data-Mining begann mit der Erhebung und Speicherung von Daten Anfang der 1960er Jahre, wobei die konventionellen Methoden der Statistik (Regressionsanalyse, verschiedene Standard-Distributionen/Varianz, Kovarianz) zum Einsatz kamen.

Der nächste evolutionäre Schritt im DM fand in den 1980er Jahren mit der Einführung von relationalen Datenbanken statt. Basierend auf strukturierten Abfragesprachen (Structured Query Language) gelang es, Erkenntnisgewinne aus den in Datenbanken gespeicherten Daten abzuleiten. Andere Methoden wie Fuzzy-Logik, nicht lineare Methoden, künstliche Intelligenz und maschinelles Lernen wurden zudem angewendet, da die zurzeit erreichte Rechenleistung solche Analysen oftmals unterstützen kann. In diesem Zusammenhang erlangte in den 1980er Jahren die Nichtlineare Dynamik als ein weiterer Bereich der Datenanalyse Popularität. Die Nichtlineare Dynamik entstand aus der Physik und Chaostheorie, wobei die komplexen Systeme, insbesondere Produktionssysteme, als dynamische Systeme betrachtet wurden, deren zeitliche Entwicklung durch innere Systemgrößen und Einflüsse ihrer Umgebung bestimmt sind (Kantz et al. 2004), (Freitag et al. 2015a).

In den frühen 1990er Jahren wurde DM allgemein als Unterprozess in einem größeren Prozess namens Knowledge Discovery in Databases (KDD) definiert (Fayyad et al. 1996). Data-Mining beschränkte sich daher auf die Anwendung von spezifischen Algorithmen für die Mustererkennung (engl. pattern extraction) aus Daten. Die weiteren Schritte des KDD-Prozesses, wie Daten-Akquisition, -Vorbereitung, -Filterung, und –Interpretation, gehörten nicht zu den Kernaufgaben des Data-Mining (Fayyad et al. 1996). Das Data-Warehousing wurde dabei als eine Erweiterung der relationalen Datenbanken Ende der 1990er Jahre eingeführt.

Mit der Einführung des Begriffs „Internet der Dinge“ und der Entstehung von großen Datenbanken ist eine neue Ära von komplexen Systemen und Anwendungen entstanden. Allerdings stoßen die traditionellen statistischen Methoden an ihre Grenzen. Neue Anforderungen und Anwendungen von DM-Methoden wurden definiert. Hier zeigten Verfahren der Künstlichen Intelligenz und des maschinellen Lernens oftmals ihre Zuverlässigkeit und Anwendbarkeit mit dem

Umgang großer Datenmengen. Dabei stellte die Skalierbarkeit ein riesiges Problem im DM dar.

Allgemein ist die Evolution von DM mit dem technologischen Fortschritt gekoppelt. Die Aufgaben von DM haben sich auch evolutionär entwickelt. Wenn jeder Schritt der DM-Entwicklung analysiert wird, wird klar, dass jeder Schritt auf dem vorherigen Schritt aufgebaut ist. Während sich in den 60er und 70er Jahren die DM-Methoden mit statistischen Fragen auseinandergesetzt haben, sind beim Erscheinen von relationalen Datenbanken, Data-Warehousing und dem „Internet der Dinge“ andere Kernaufgaben entstanden. Die zu verarbeitenden Daten waren überwiegend Zahlen, welche in der Form von Tabellen oder Standard-Datensätzen zur Verfügung gestellt wurden. Die Herausforderung stellte dabei die damalige verfügbare Rechenleistung dar, die von signifikanter Bedeutung war. Als sich jedoch die Rechenleistung mit der Zeit verbesserte und die Verarbeitungszeit weniger problematisch wurde, kam der Wunsch nach erhöhter Genauigkeit durch Verarbeitung von immer größeren Datenmengen auf (Coenen 2011). Heutzutage existieren verschiedene DM-Anwendungen für die Verarbeitung von Tabellendaten und Standard-Datensätzen (Rapid Miner, Weka, Python, etc.). Ein weiterer Aspekt von DM repräsentiert seine Anwendung für Nicht-Standard-Datensätze, welche heutzutage weit verbreitet sind. Beispiele dieser komplexen Datensätze sind Bilder, Videos und Text-Dokumente. Das erforderte auch eine Erweiterung der Methoden des DM.

3.2. Data-Mining-Prozess

Für die Umsetzung von Data-Mining haben sich verschiedene DM-Prozesse etabliert. Der von Fayyad (1996) vorgestellte DM-Prozess besteht aus fünf Arbeitsphasen, welche je nach Zielsetzung erhebliche Iterationen und Schleifen zwischen zwei Stufen beinhalten können (Abbildung 8). Ausgehend von Datenerhebung und Datenselektion aus einer Datenbank werden im Rahmen der Datenaufbereitungsphase die selektierten Daten auf Fehler und Ausreißer gefiltert. Es schließt sich die Transformationsphase an, wobei die in den vorherigen Schritten vorverarbeiteten Daten in ein passendes Analyseformat transformiert und gegebenenfalls diskretisiert werden. Im Rahmen der vierten Data-Mining-Phase findet die eigentliche Anwendung der DM-Methoden statt. Abschließend geht es in der finalen Phase um die Interpretation der gefundenen Muster durch den Experten und die graphische Darstellung von Ergebnissen.

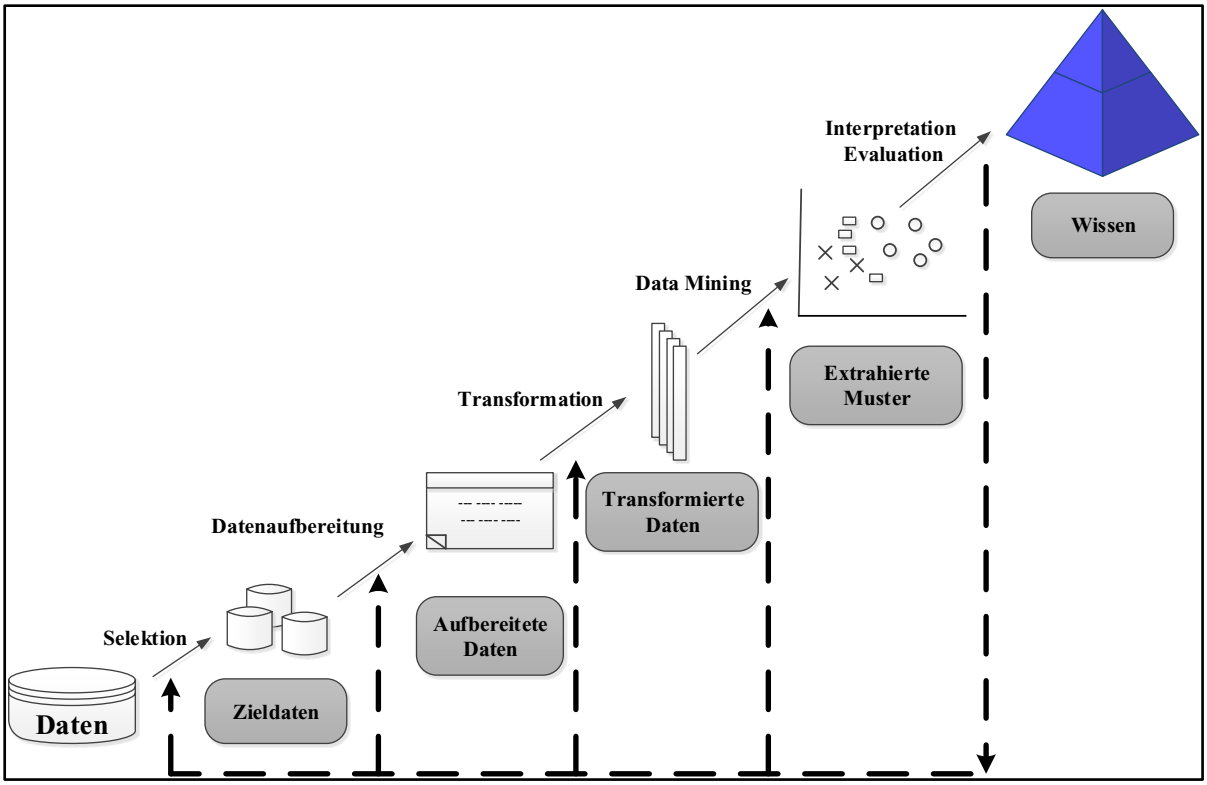

Abbildung 8: KDD-Data-Mining-Prozess (in Anlehnung an Fayyad et al. 1996)

Das „Cross-Industry Standard Process for Data-Mining"-Modell (CRISP-DM) hat sich als ein DM-Prozessmodell etabliert, das sich aus sechs Phasen zusammensetzt (Abbildung 9). Im Gegensatz zum DM-Prozess von Fayyad ist CRISP-DM ein Industriestandard, der im Rahmen eines EU-Projekts entstanden ist (Wirth et al. 2000). Die erste Phase befasst sich mit dem Verständnis der Geschäftsmodelle bzw. Prozesse, wobei es in diesem Schritt um die Aufstellung von Geschäftszielen, Erfolgskriterien und die Festlegung der Zielsetzung des DM-Projekts geht. Die zweite Phase befasst sich mit dem Daten-Verständnis, wobei der Fokus auf Datenerhebung, -qualität und -entdeckung für die Lösung des Problems liegt. Es schließt sich die Datenaufbereitungsphase an, die alle Aktivitäten der Datensammlung, Datenfilterung und Datentransformation umfasst. In der vierten Phase gilt es, auf Basis der Problem-Zielstellung und der in der vorherigen Phase aufbereiteten Daten verschiedene Modellierungstechniken bzw. Data-Mining-Algorithmen einzusetzen, um ein Modell zu bilden. Manche Modellierungstechniken haben bestimmte Anforderungen an das Datenformat, deswegen sind Rücksprünge zur vorherigen Phase „Datenaufbereitungsphase" möglich. Darauf aufbauend wird in der fünften Phase das generierte Modell evaluiert. Die Evaluation erfolgt allgemein durch Verwendung verschiedener Evaluationsmethoden und Kriterien für Testdaten. In der finalen Phase wird das evaluierte Modell eingesetzt. Dabei wird weiterhin die Gültigkeit des Modells überwacht und verschiedene Überarbeitungsstrategien werden festgelegt.

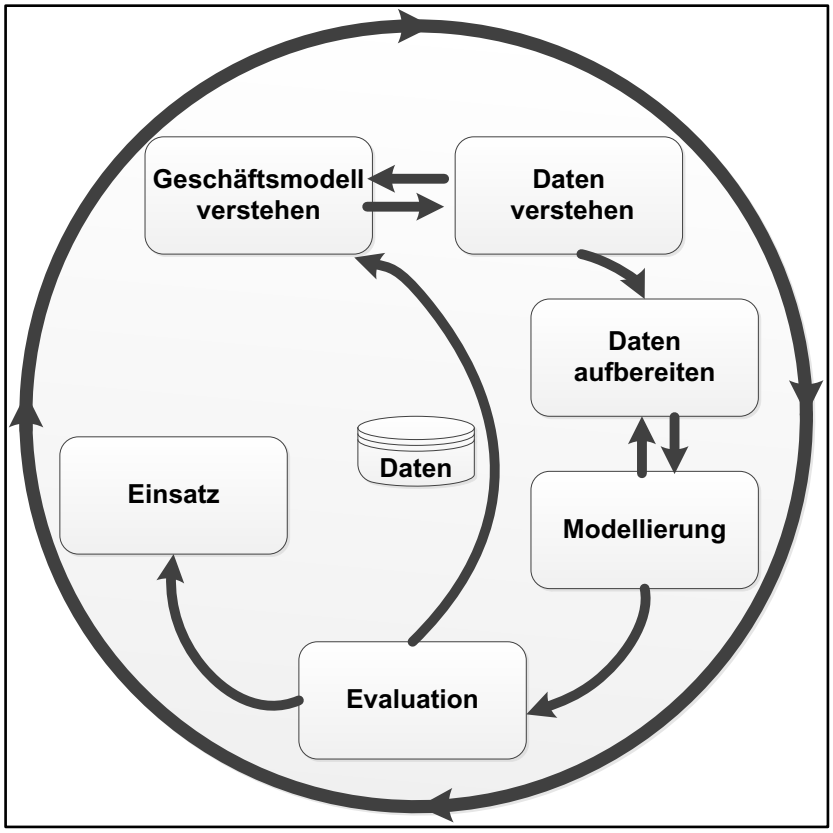

Abbildung 9:CRISP-Data-Mining-Referenzmodell (in Anlehnung an Chapman et al. 2000)

Aufbauend auf den beiden aufgeführten Data-Mining-Prozessen kann der Ablauf eines Data-Mining-Projektes wie folgt definiert werden: Zunächst geht es darum zu formulieren, welche Informationen mit den Data-Mining-Methoden gewonnen werden sollen. Im zweiten Schritt müssen die Daten importiert und bereinigt werden. Nur selten sind Daten sofort einsatzbereit. In diesem Zusammenhang sollen Eingabefehler, fehlende Werte und andere Inkonsistenzen beseitigt werden, bevor Data-Mining-Methoden darauf angewandt werden. Anschließend werden die Daten auf Zusammenhänge untersucht. Dazu können Tabellen, statistischen Auswertungen und Grafiken benutzt werden. Basierend auf den gewonnenen Erkenntnissen werden im dritten Schritt Verfahren ausgewählt. Je nach Fragestellung und Datensituationen eignen sich unterschiedliche Verfahren. Alle geeigneten Verfahren werden mit entsprechenden Parametern, die sich aus sinnvollen Annahmen ergeben, auf die Daten angewandt. Nachdem die Verfahren Ergebnisse berechnet haben, können diese im vierten Schritt verglichen werden. Dies kann graphisch oder tabellarisch geschehen. Es können dabei auch hier verschiedene Metriken, wie die mittlere quadratische Abweichung, verwendet werden, um die Ergebnisse des Modells zu evaluieren. Die Modellauswahl und die Konfiguration sollten basierend auf den Ergebnissen ein weiteres Mal angepasst werden. Wenn die gewünschten Modelle validiert wurden, können sie im letzten Schritt auf neue Daten angewendet werden.

3.3. Analyse-Verfahren im Data-Mining

Data-Mining-Prozesse verfolgen das Ziel, Muster in Daten, die nicht einer wohldefinierten Vorstellung von normalem Verhalten entsprechen, zu entdecken. Für die Erkennung solcher Muster werden Data-Mining-Techniken verwendet, um Situationen bzw. Anomalien in einer Datenmenge zu identifizieren. Die Erkennung von Anomalien ist eine wichtige Thematik, die in verschiedenen Forschungsbereichen und Anwendungsdomänen untersucht wurde, zum Beispiel im System-Health-Management, in der Intrusion-Detektion, Betrugserkennung und mechanischen Fehlererkennung. Ziel der Anomalie-Erkennung ist es, Muster zu identifizieren, die nicht dem erwarteten Verhalten entsprechen, um Anomalien frühzeitig zu erkennen (Leger et al. 1998), (Chandola et al. 2009). Insbesondere stellt die Disziplin des Data-Mining, die sich mit Operationen und Analysen von Signalen in kontinuierlicher oder diskreter Zeit beschäftigt, Techniken zur Erkennung von Anomalien nach dem Stand der Technik bereit. Chandola et al. (2009) klassifizierten die Anomalie-Erkennungstechniken in verschiedene Bereiche, wie klassifikationsbasierte Anomalie-Erkennungstechniken (Neuronales Netzwerk, Bayes-Netzwerk und Support-Vektor-Maschine), statistische Anomalie-Erkennungstechniken und Clustering-basierte Anomalie-Erkennungstechniken (Gauß-Modell-basiert, Regressionsmodell-basiert und Mischung aus parametrischen Verteilungen).

Es gibt dabei drei allgemeine Konzepte des maschinellen Lernens, die verschiedene Arten von Data-Mining-Verfahren verwenden (Hodge et al. 2004):

1) Überwachtes Lernen erfordert Datensätze, bei denen die zu lernenden Ausgaben im Vorhinein bekannt sind. Ein Beispiel für die Überwachung eines Systemzustandes wäre, wenn die zu lernenden Ausgaben aus erfassten Sensordaten als „normal“ und „abnormal“ gekennzeichnet wären. Beim Lernen geht es darum, ein Modell zu generieren und zu trainieren, welches die Zuordnungsfunktion zwischen den Eingabe- und Ausgabedaten lernt.
2) Beim unüberwachten Lernen wird versucht, Modelle zu entwickeln, die Muster in den Eingabedaten erkennen können, ohne dass die Ausgabenwerte im Voraus bekannt sind. Die Muster werden dann aufgrund der Ähnlichkeiten zu den Eingabedaten gebildet. Im Beispiel der Überwachung eines Systemzustandes wird dann das Modell nur unter der Annahme gebildet, dass die Mehrheit der Instanzen in den Eingabedaten dem normalen Systemzustand angehört und dass Eingabedaten, die keine große Ähnlichkeit zum Rest der Daten vorweisen, einen abnormalen Systemzustand abbilden.
3) Das bestärkende Lernen stellt eine Kombination zwischen überwachtem und unüberwachtem Lernen dar. Anders als beim überwachten Lernen, bei dem die zu lernenden Ausgaben im Vorhinein schon bekannt sind, versucht das Modell im bestärkenden Lernen

durch Belohnungen bzw. Bestrafungen (negative Belohnungen) die Ausgabewerte zu approximieren. Dabei vergibt ein Agent nur zu bestimmten Zeitpunkten die Belohnungen. Diese Belohnungen geben an, wie gut die approximierten Ausgabewerte des Modells waren.

Durch die Anwendung der Methoden des DMs können verschiedene Optimierungen und Verbesserungsmöglichkeiten umgesetzt werden. Es existierten dabei drei Analytik-Werkzeuge, die diese Ziele verfolgen. Der Fokus dieser Analytik-Werkzeuge besteht darin, mit dem Einsatz von DM-Methoden die Daten zu Information und danach zu Wissen umzuwandeln und anschließend Entscheidungen zu treffen. Die Diagnostik/deskriptive Analytik versucht auf Basis von historischen Daten, die Fehleranalyse und Systemzustände über das, was passiert, zu liefern. Die prädiktive Analytik nutzt die Analyse von historischen Daten, um aus Datenmustern der Vergangenheit einen Trend für die Zukunft zu ermitteln, um dann die Frage „Was wird passieren?“ zu beantworten. Die präskriptive Analytik erweitert die prädiktive Analytik und liefert neben den Vorhersagen auch Handlungsanleitungen mit. Hierbei wird die Frage „Was soll getan werden, wenn etwas passiert?“ beantwortet (Freitag et al. 2015a).

Durch die Anwendung richtiger Werkzeuge und Methoden können leistungsfähige, aussagekräftige Erkenntnisse aus den gesammelten Rohdaten, insbesondere Big Data, gewonnen werden. Ausgehend von diesen Daten kann mit der Datenanalyse begonnen werden. In der deskriptiven Analytik geht es darum, Modelle zu generieren, die einen deskriptiven Zusammenhang zwischen Daten und Systemverhalten abbilden. Die prädiktive Analytik ist der nächste Schritt in der Datenanalyse. Diese Analyse nutzt eine Vielzahl von statistischen Methoden, Regressionen und Methoden des maschinellen Lernens, um eine Prognose des aktuellen Zustands in der Zukunft zu liefern. Darauf aufbauend kann die präskriptive Analytik als eine Art von prädiktiver Analytik betrachtet werden. Dabei werden Simulationswerkzeuge und Optimierungsalgorithmen verwendet, um die beste Entscheidung zu treffen, wenn einen Zustand auftritt (Freitag et al. 2015a).

3.4. Aufgaben und Methoden des Data-Mining

Wie bereits erwähnt, stellt die Verarbeitung der erfassten Daten aus Produktion und Maschinen die treibende Kraft für die Optimierung der Instandhaltungsprozesse dar. Dabei können die Daten in verschiedenen Datenformaten bzw. Datentypen vorliegen. In diesem Zusammenhang wird zwischen strukturierten, semistrukturierten und unstrukturierten Daten unterschieden (Cleve et al. 2016). Unter unstrukturierten Daten werden zum Beispiel Bilder, Videos, aber auch Texte verstanden, die keine Struktur aufweisen. Beispiele für semistrukturierte Daten sind Webseiten und Log-Dateien, die unstrukturierte Daten besitzen, aber aufgrund der aufgewiesenen Grundstruktur als semistrukturierte Daten bezeichnet

werden. Anders als unstrukturierte Daten weisen alle Daten, die in einer relationalen Datenbank vorliegen, eine Struktur auf und werden deswegen als strukturierte Daten bezeichnet. Aus diesen drei Datenkategorien ist der Begriff des Data-Mining um zwei weitere Teilgebiete erweitert worden. In diesem Kontext werden die Methoden des DM, welche sich mit unstrukturierten Daten auseinandersetzen, dem Teilgebiet „Text Mining“ zugeordnet. Für die semistrukturierten Daten hat sich das Teilgebiet „Web-Mining“ und für die strukturierten Daten das Teilgebiet „Data-Mining“ etabliert. Grundsätzlich ist daher das eigentliche Data-Mining der Prozess der Suche nach Mustern und Korrelationen in großen relationalen Datenbanken (Cleve et al. 2016).

Daten sind bekannte Tatsachen, Zahlen oder Texte, die von einem Computer verarbeitet werden können. Heutzutage sammeln Unternehmen umfangreiche und wachsende Datenmengen in unterschiedlichen Datenformaten und legen sie in Datenbanken ab. Neben den vielfältigen Datenkategorien (unstrukturierte, semistrukturierte und strukturierte Daten) liegen die erfassten Daten in verschiedenen Datenformaten vor, welche die Verarbeitung solcher Daten zu einer herausfordernden Aufgabe macht. In diesem Zusammenhang wird zwischen folgenden Datenformaten unterschieden (Abbildung 10):

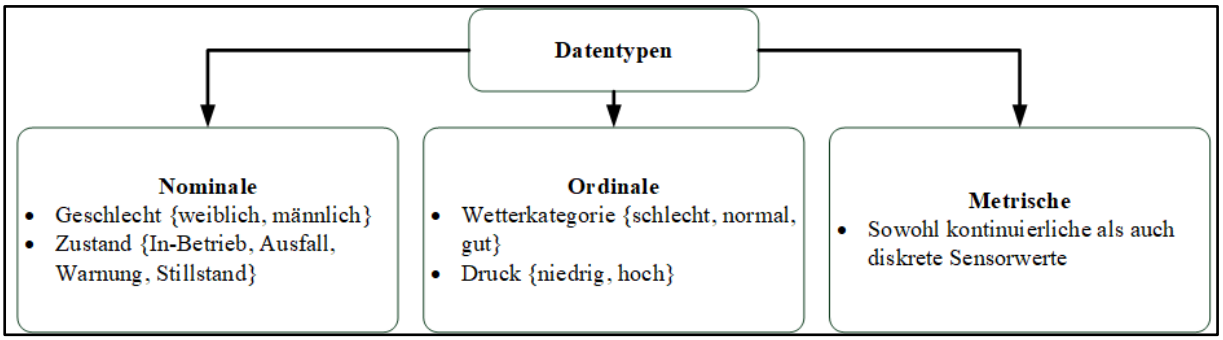

Abbildung 10: Verschiedene Datentypen im Data-Mining (in Anlehnung an Cleve et al. 2016)

Unter nominalen Datentypen werden alle Daten verstanden, die nicht numerisch sind und keiner Rangfolge unterliegen. Systemzustand {In-Betrieb, Ausfall, Warnung} ist ein Beispiel für einen nominalen Datentyp. Hingegen sind ordinale Daten nominale Daten mit einer Rangfolge. Zum Beispiel lassen sich die Werte einer Wetterkategorie {sehr gut, gut, normal, schlecht, sehr schlecht} in einer Reihenfolge anordnen. Metrische Daten besitzen eine Ordnungsrelation und können jeweils als diskrete (z. B. natürliche Zahlen) oder kontinuierliche Werte (z. B. reelle Zahlen) vorliegen. Darüber hinaus können die metrischen Daten in verschiedenen numerischen Berechnungsmethoden verwendet werden.

Allgemein lassen sich die Methoden des Data-Mining in die Aufgaben „Klassifikation“, „Segmentierung (Cluster-Analyse)“, „Assoziationsanalyse“ und „Regressionsanalyse“ einteilen (Alpar el al. 2000). Auf Basis dieser Aufgaben werden den einzelnen Aufgaben verschiedene Methoden zugeordnet (Abbildung 11).

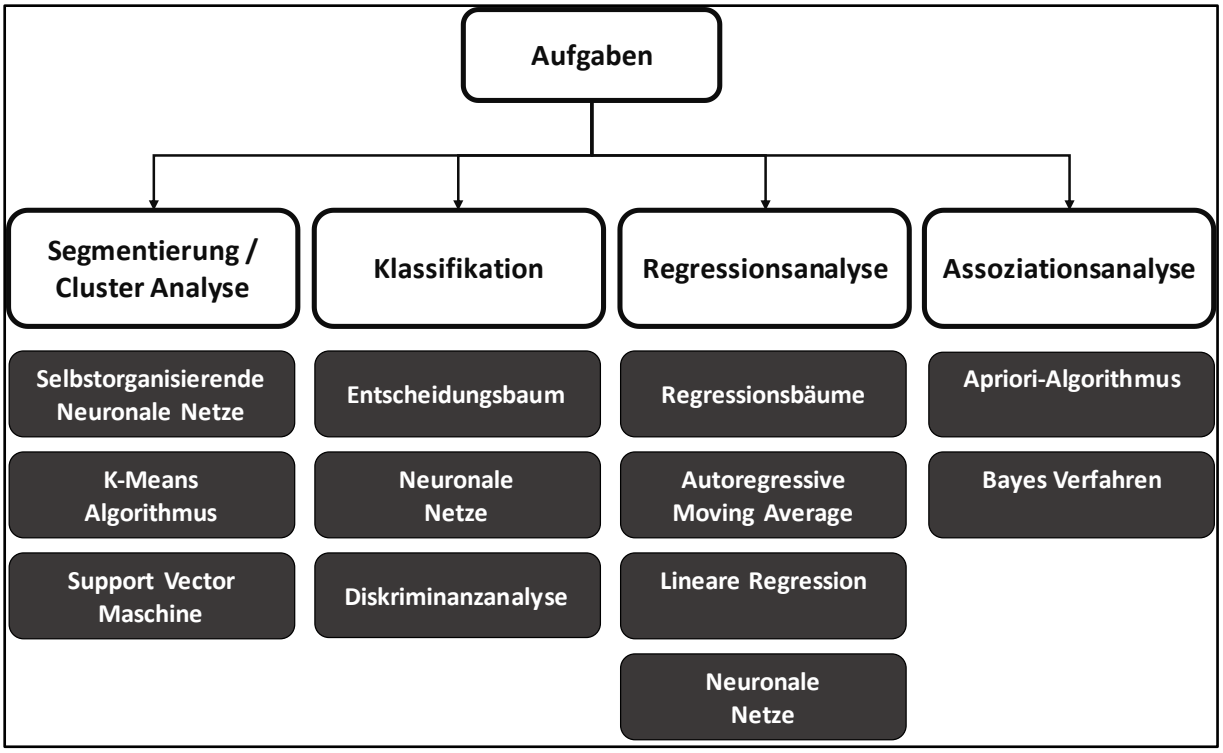

Abbildung 11: Zuordnung der Data-Mining-Aufgaben zu Methoden (in Anlehnung an Alpar et al. 2000)

3.4.1. Klassifikation

Die Klassifikation ist eine typische Anwendung des Data-Mining, wobei die vorherzusagende Zielvariable Kategorie alle Natur ist (Choudhary et al. 2009). Beispiele hierfür in Anwendungsfällen wären etwa, bei der Betrugserkennung mögliche betrügerische Transaktionen aufzudecken, in der Medizin Patienten mit hohem Infektionsrisiko oder in der Instandhaltung kritische Maschinezustände zu identifizieren. Jedes dieser Beispiele kann mit Data-Mining-Methoden unterstützt werden. Eine präzise arbeitende Klassifikationsroutine kann helfen, Zeit und Geld zu sparen. Die Ziele der Klassifikationsverfahren umfassen dabei, Merkmale mit starkem Einfluss auf die Zielvariable zu identifizieren und ein Modell zur Klassifikation zu erstellen (Windt et al. 2011). Hierbei können Daten in bereits vordefinierten Gruppen bzw. Klassen eingeteilt werden. Solche Modelle sagen kategoriale (diskrete, ungeordnete) Klassenbezeichnungen voraus. Weiterhin müssen die Zugehörigkeiten der Trainingsdaten zu einer der Klassen aufgrund ihrer Eigenschaften vorab bekannt sein (Bramer 2013). Zum Beispiel können Klassifizierungsmodelle in der Fehlerdiagnose zum Einsatz kommen, um den Systemzustand in „normal“ oder „anormal“ zu klassifizieren. Für die Klassifikation spielt die Qualität der Trainingsdaten eine entscheidende Rolle. Im Allgemeinen eignen sich die Methoden des maschinellen Lernens für die Klassifikation, besonders die Methoden des überwachten Lernens, wie beispielsweise neuronale Netze oder Entscheidungsbaumverfahren. Formal lässt sich die Klassifikation wie folgt beschreiben:

Gegeben sei ein Satz von n Attributen (ordinal oder metrisch) A= $\{a_1, a_2, \ldots, a_n\}$ und ein Satz von k Klassen K = $\{c_1, c_2, \ldots, c_k\}$. Ausgehend von Trainingsdaten T bestehend aus m Datensätzen $T_i = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$ und $i \in [1, m]$, besteht die Klassifikation darin, ein Modell als Funktion zu finden, sodass gilt:

$$\forall\, j \in [1, k]\ \forall\, i \in [1, m], \qquad C_j = F(T_i) = F(a_{i1}, \ldots, a_{im})$$

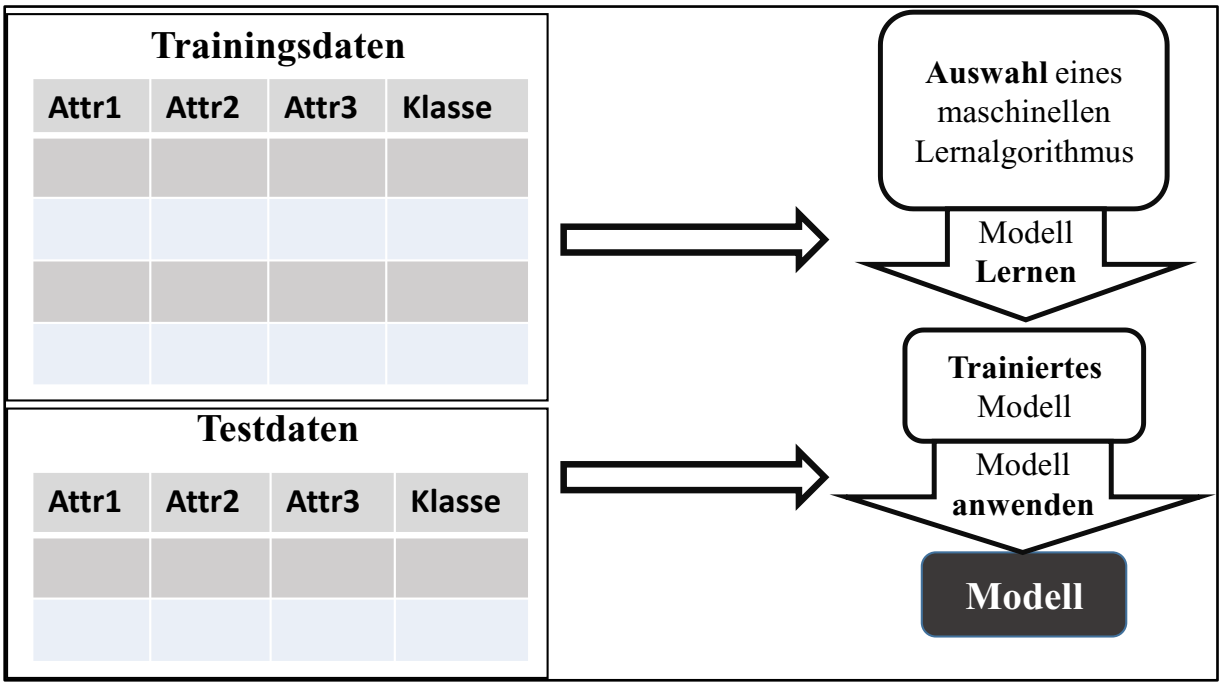

Abbildung 12: Aufbauprozess eines Modells für die Klassifikationsaufgabe

Der Prozess für den Aufbau eines Modells für die Klassifikationsaufgabe besteht aus folgenden Schritten (Abbildung 12) (Choudhary et al. 2009):

- Bildung der Trainingsdaten aus bereits bekannten Zugehörigkeiten. Dabei werden auf den gesamten Datensätzen 80 bis 90 % als Trainingsdaten und 10 bis 20 % als Testdaten benutzt
- Aufbau eines Modells, das die Zugehörigkeit der Inputdaten zu vordefinierten Klassen lernt
- Einsatz des trainierten Modells auf unbekannte Testdaten zum Zweck der Validierung des Modelles, bis die Klassifikation der Testdaten eine akzeptable Fehlerrate aufweist
- Sind die Ergebnisse der Klassifikation zufriedenstellend, wird das Modell für die Klassifikation bereitgestellt

3.4.2. Regressionsanalyse

Die Regression ist ein weiterer Anwendungsfall des Data-Mining. In diesem Fall ist die zu ermittelnde Zielvariable kontinuierlich. Die Regression verfolgt das Ziel, durch Verwendung mathematischer und statistischer Verfahren eine approximative Vorhersage für bestimmte numerische Werte für die Zukunft zu be-

rechnen (Freitag et al. 2015a). Mögliche Anwendungsfälle der Regression sind zum Beispiel, Sensorwerte vorherzusagen, eine Abschätzung der Restlebensdauer einer Komponente zu berechnen, eine Messgröße im Produktionsprozess zu optimieren oder Prognosemodelle im Rahmen der zustandsorientierten Instandhaltung zu generieren (Lewandowski 2016). Die Ziele bei Anwendung der Regression sind ähnlich wie bei der Klassifikation. Es sollen Merkmale vom Einfluss auf die Zielvariable gefunden und ein Prognosemodell zur Ermittlung der Höhe eine Zielvariable erstellt werden (Bramer 2013). Die einfachste und älteste Form der Regression ist die lineare Regression, die eine Beziehung zwischen zwei Variablen in Form einer linearen Funktion darstellt.

Im Allgemein wird aus einer Menge von Attributen $X = \{x_1, x_2, \ldots, x_n\}$ und einer Menge von Ausgabewerten $Y = \{y_1, y_2, \ldots, y_n\}$ eine Funktion $F(X) = Y$ gesucht, die einen möglichst guten quantitativen Zusammenhang der Attribute x_i zu den einzelnen Ausgabewerten y_i zulässt.

Exemplarisch ist in Abbildung 13 der Verlauf einer Außentemperatur abgebildet, welche von einem Temperatursensor erfasst wurde. An den Sensordaten lässt sich über den Testzeitraum ein saisonal bedingter Trend erkennen, der mit dem Verlauf der Außentemperatur einhergeht. Wie in dieser Abbildung erkennbar, ist es schwer, mithilfe der linearen Regression die Außentemperatur zu approximieren.

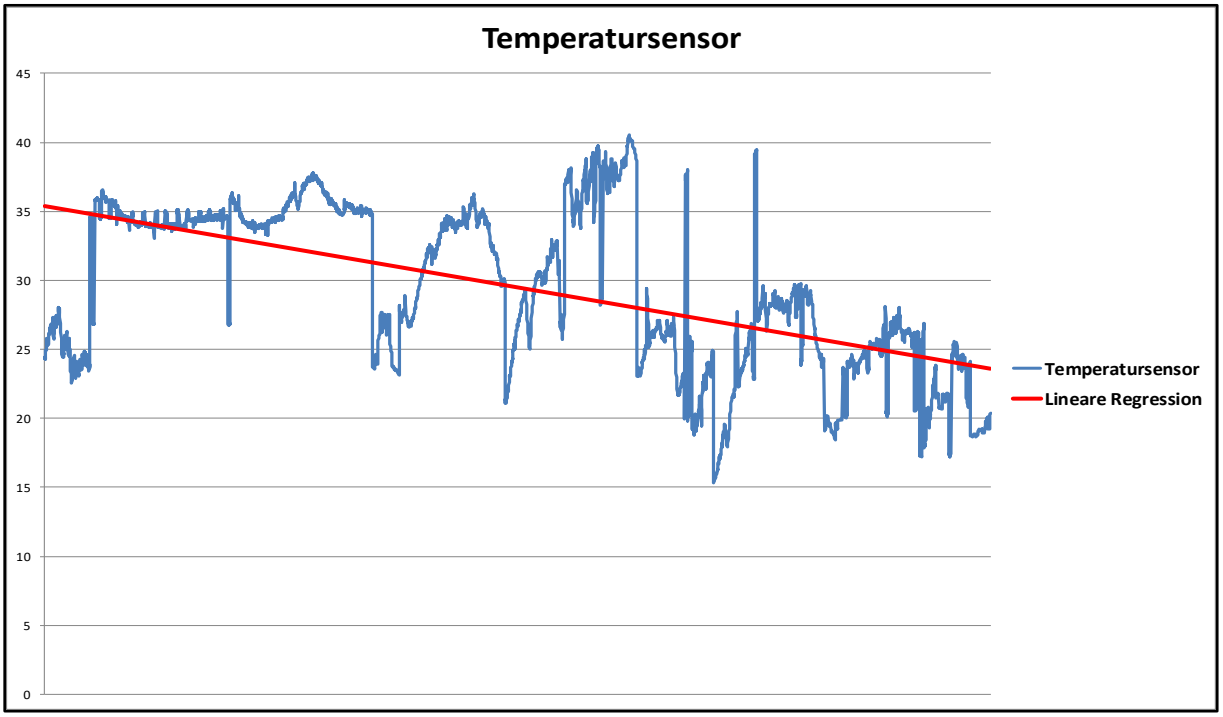

Abbildung 13: Beispiel einer linearen Regression bzw. Approximation eines Temperatursensors

Zur Regressionsanalyse eignen sich unter anderem die Methoden der Zeitreihenanalyse wie **A**uto**r**egressive-**M**oving **A**verage ARMA (Freitag et al. 2015b).

Ein weiteres Verfahren, das für Regressionsaufgaben eingesetzt werden kann, sind künstliche neuronale Netze (Freitag et al. 2015b).

Die Schritte für den Aufbau eines Regressionsmodells sind ähnlich wie bei der Klassifikation. Das Schema einer Regressionsanalyse umfasst die Aufbereitung der Trainingsdaten, das Lernen des Modells, die Modellvalidierung und letztendlich den Einsatz des validierten Modells.

3.4.3. Segmentierung/Clustering

Die Segmentierung bzw. das Clustering ist eine weitere typische Data-Mining-Anwendung, unterscheidet sich aber deutlich von Regression und Klassifikation. In diesem Fall gibt es keine Zielvariable, stattdessen sollen die Daten in Clustern bzw. Gruppen zusammengefasst werden. Ziel ist es, unbekannte Zusammenhänge in den Daten zu entdecken und zu verstehen (Choudhary et al. 2009). Die Segmentierung beschäftigt sich mit der Aufteilung von Daten in verschiedene Gruppen bzw. Cluster. Die Ziele der Segmentierung umfassen dabei den Vergleich von Clustern hinsichtlich der Merkmalausprägungen, die Identifikation von Merkmalen, welche die Gruppenzugehörigkeit beeinflussen, die Zuordnung neuer Fälle zu Clustern und die Messung der Stärke einer Klassezugehörigkeit. Anders als bei der Klassifikation liegt bei der Segmentierung keine Zugehörigkeit der Daten zu den Gruppen vor und die Anzahl der Gruppen ist nicht bekannt. Die Daten werden aufgrund ihrer Ähnlichkeiten in verschiedene Cluster gruppiert (Windt et al. 2011). Die Ähnlichkeit zwischen den Daten wird mittels Einsatz einer Distanzfunktion als Metrik formuliert. Daher ist die Distanz zwischen Daten eines Clusters kleiner als die Distanz zwischen Daten aus verschiedenen Clustern (Xu et al. 2005). Häufig wird die Segmentierung für die Ausreißer-Erkennung verwendet, wenn kein Wissen über die Ausreißer vorliegt. In diesem Kontext werden die Ausreißer keinem Cluster zugeordnet oder bilden selber ein kleines Cluster. Meistens eignen sich die Methoden des unüberwachten Lernens wie K-means, Support Vector Maschine oder Selbstorganisierende neuronale Netze für die Segmentierung bzw. das Clustering (Xu et al. 2005). Im folgenden Beispiel in Abbildung 14 wurde ein Clusterverfahren angewendet. Dazu wurde K-means Clustering mit der Vorgabe, 4 Klassen zu suchen, eingesetzt. Das Verfahren hat jeden Punkt aus den Datensätzen je einem Cluster zugeordnet.

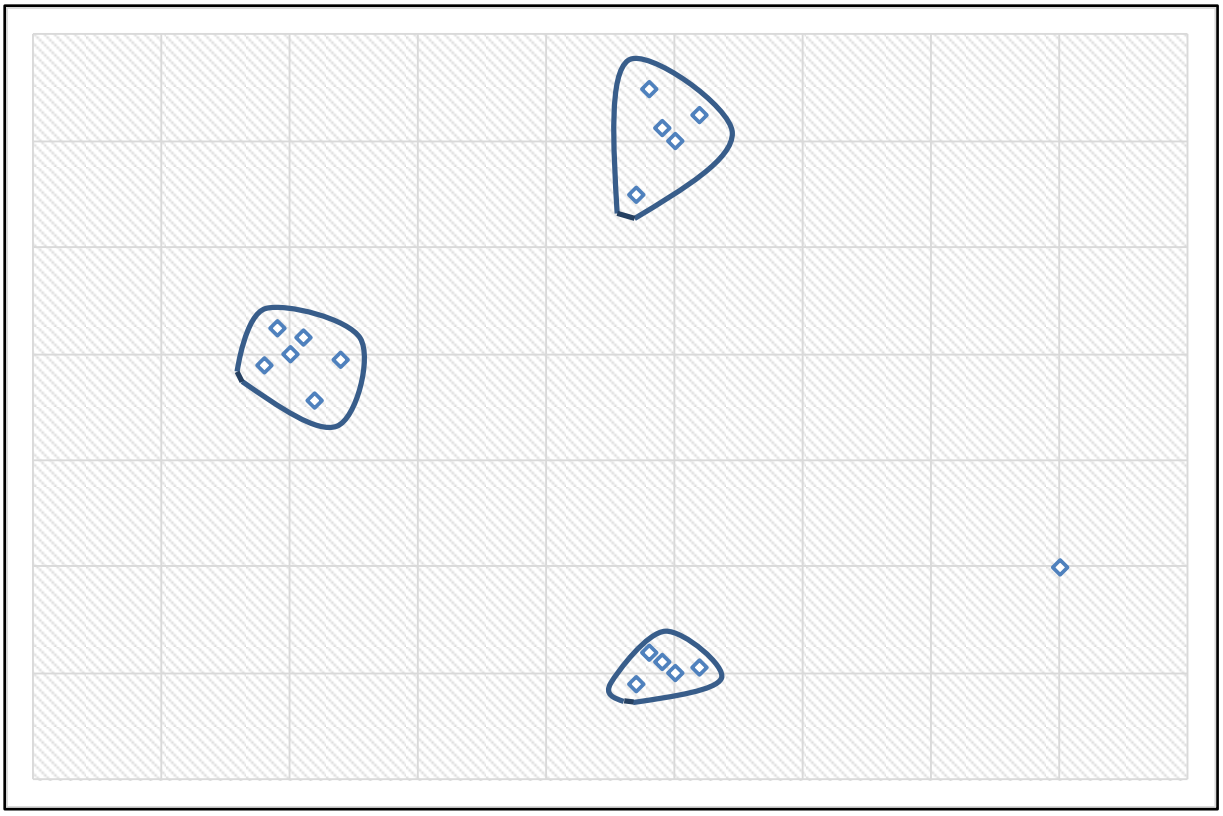

Abbildung 14: Beispiel einer K-means Cluster-Klassifikation

3.4.4. Assoziationsanalyse

Eine andere interessante Data-Mining-Aufgabe stellt die Assoziationsanalyse dar. Ziel der Assoziationsanalyse ist, Zusammenhänge bzw. Abhängigkeiten zwischen Elementen aus historischen Daten in der Form von Assoziationsregeln darzustellen (Choudhary et al. 2009). Dabei werden diese Zusammenhänge nach bestimmten Regeln abgestimmt. Die Assoziationsanalyse wurde erstmals Anfang der 1990er Jahre eingeführt. Sie wird verwendet, um Beziehungen zwischen einer Reihe von Elementen in einer Datenbank zu identifizieren (Agrawal et al. 1994). Diese Beziehungen basieren nicht auf inhärenten Eigenschaften der Daten selbst wie bei funktionalen Abhängigkeiten, sondern auf dem gemeinsamen Auftreten der Datenelemente. Diese Beziehungen können als Entscheidungsregeln verfeinert werden, die in einem Wissensbasissystem verwendet und gespeichert werden. Das Ziel der Assoziationsanalyse besteht darin, Gruppen von Ereignissen hervorzuheben, die zusammen auftreten können, um beispielweise eine Ursache-Wirkungs-Beziehung abzuleiten. Durch die Analyse der resultierenden Regeln, die in der Form „wenn Bedingung, dann Ergebnis“ ausgegeben werden, können diejenigen Regeln ausgewählt werden, die eine hohe Signifikanz und ein hohes Interesse ausdrücken (Buddhakulsomsiri et al. 2006).

Die Assoziationsanalyse wird in vielen Bereichen angewendet. Beispielweise kann für Customer Relationship Management die technische Analyse von Kunden anhand von Assoziationsregeln das Kaufverhalten darstellen, sodass das

Layout von Produkten angepasst werden kann, um die Dienstleistungen und Gewinne zu verbessern (Chen et al. 2005), (Harding et al. 2006). Klassisches Einsatzgebiet der Assoziationsanalyse in der Instandhaltung ist die Ursache-Wirkungs-Analyse. In der Instandhaltung können Assoziationsregeln zur Vorhersage von Fehlern in technischen Systemen verwendet werden, indem identifiziert wird, welche Ereignisse vor einem Fehler auftreten (Choudhary et al. 2009). Dabei werden Methoden wie Apriori-Algorithmen verwendet, um eine Ursache-Wirkungs-Analyse durchzuführen, die untersucht, welche nacheinander auftretende Störungen sich nach Auftreten einer Ursache identifizieren lassen.

Formell lässt sich die Assoziationsanalyse wie folgt beschreiben:

Sei $I = [i_1, i_2, \dots, i_n]$ eine Menge ausgewählter Elemente bzw. Attribute (auch Störungen) und T eine Menge von Trainingsdatensätzen oder Transaktionen (Störungen, die in den historischen Daten in verschiedenen Zeitpunkten beobachtet wurden), wobei jeder Datensatz bzw. jede Transaktion in T aus mehreren Attributen bzw. Elementen von I besteht und mit einer eindeutigen Kennung versehen ist. Eine Assoziationsregel ist eine Implikation des Typs $A \Rightarrow B$, mit $A \subset I$ und $B \subset I$ und $A \cap B = \emptyset$.

A und B sind jeweils als Antezedens und Folge der Regel definiert. Der Ausdruck bedeutet, dass wenn A in T eintritt, B wahrscheinlich auch in T eintreten wird.

Diese Wahrscheinlichkeit der Assoziationsregeln lässt sich durch „Support und Confidence“ der Regeln formulieren. Hierbei gibt „Support“ an, wie häufig die beteiligten Attribute einer Regel in der Menge von Trainingsdatensätzen T auftreten, während die „Confidence“ einer Regel aussagt, wie oft die Regel als wahr erkannt wurde (ausgelöst wurde).

$$S(A \Rightarrow B) = \frac{|\{t \in T; A \cup B \subseteq t\}|}{|T|} \quad \text{und} \quad C(A \Rightarrow B) = \frac{|\{t \in T; A \cup B \subseteq t\}|}{|\{t \in T; A \subseteq t\}|}$$

3.5. Data-Mining-Methoden: Einsatz von Methoden der KI

Wie im vorgängigen Kapitel in Anlehnung an das Zustandsüberwachungssystem von OSA-CBM beschrieben, ist für die Verarbeitung der erfassten Daten der Einsatz verschiedener Methoden in den verschiedenen Funktionsmodulen des OSA-CBM erforderlich. In diesem Zusammenhang werden zunehmend die Methoden der KI bzw. maschinelle Lernverfahren verwendet, wenn die Modelle, die für die Bewertung des Anlagenzustandes zuständig sind, nicht als Erkenntnisse a priori in Form von Formeln oder Funktionen vorliegen (Ryll 2008). Dieser Abschnitt stellt einige dieser Verfahren vor, die öfter in der Literatur vorzufinden sind.

3.5.1. Neuronale Netze

Das menschliche Gehirn besteht aus Milliarden Zellen, die Neuronen genannt werden. Ihr Verhalten ist einfach: Sie reagieren auf Reize, die durch Dendriten empfangen werden. Durch die Axone führt diese Reaktion zum Senden einer elektrischen Energie. Neuronen sind durch Synapsen miteinander verbunden. Die Reaktion eines Neurons kann auf andere Neuronen übertragen werden. Durch die Vernetzung verschiedener Neuronen wird aus dem einfachen lokalen Verhalten von einzelnen Neuronen ein wesentlich komplexeres globales Gesamtverhalten aufgebaut (Kriesel 2008).

Künstliche neuronale Netze orientieren sich an biologischen Prozessen und Neuronen des Gehirns, um das Erkennen von Mustern in Daten zu erlernen. Aus biologischer Sicht empfängt und produziert ein Neuron Signale und besteht aus dem Zellkern, den Dendriten, die für das Empfangen und Sammeln alle Informationsanteile zuständig sind, und dem Axon, das die ausgehenden Impulse und Ausgaben des Neurons nach außen übermittelt (Kriesel 2008).

Künstliche neuronale Netze stellen ein hervorragendes mathematisches Werkzeug für die Modellierung von dynamischen nicht linearen Systemen dar. Hierbei kann jedes neuronale Netz mit mindestens einem versteckten Layer (engl. hidden layers) eine nicht lineare Funktion approximieren (Patan 2008). Eine andere wichtige Eigenschaft von KNN besteht in ihrer Lernfähigkeit. Ein KNN kann in historischen Daten Muster erkennen und extrahieren, ohne im Voraus Kenntnisse über Systemprozesse zu haben. Ein KNN verhält sich wie eine Black-Box. Diese Lernfähigkeit ermöglicht die Modellierung von nicht linearen Systemen mit einer großen Flexibilität.

3.5.1.1. Bestandteile künstlicher neuronaler Netze

Wie auch beim realen Vorbild von biologischen neuronalen Netzwerken bilden die Neuronen auch beim KNN den Hauptbestandteil eines künstlichen bzw. technischen neuronalen Netzes. Zwei Hauptelemente bilden ein künstliches neuronales Netz: erstens das Neuronen-Modell (Abbildung 15), das verwendet wird, um das Netzwerk zu bilden, und zweitens die Architektur des Netzwerks, auch als Topologie bezeichnet. Ein technisches Neuron j nimmt die Ausgaben $x_{i1}, x_{i2}, \dots x_{in}$ von anderen mit ihm verbundenen Neuronen $i_1, i_2 \dots i_n$ entgegen und verarbeitet diese in Betracht auf die Verbindungsgewichte $w_{i1,j}, w_{i2,j} \dots w_{in,j}$ zur Netzeingabe Net_j. Die Verbindungsstärke zwischen zwei verbundenen Neuronen gibt an, wie stark die Beziehung zwischen den beiden Neuronen ist. Diese Beziehung bzw. das Gewicht zwischen zwei Neuronen i und j wird als w_{ij} bezeichnet. Die gewichtete Summe der Netzeingabe, die von der Propagierungsfunktion berechnet wird, wird von der Aktivierungsfunktion verwendet, um die Ausgabe des Neurons zu erzeugen. Die Propagationsfunktion

des Neurons j ist die gewichtete Summe der Ausgabe der Neuronen (I: Mengen dieser Neuronen), von denen eine Verbindung zu j ausgeht:

$$Net_j = f_{prop} = \sum_{i \in I} (x_i . w_{i,j})$$

Nach der Aufnahme der Ausgabe der Neuronen, von denen eine Verbindung zu j ausgeht, erfolgt im nächsten Schritt die Berechnung der Reaktion des Neurons, auch als Aktivierungszustand bezeichnet, auf diese Eingabe. Der Aktivierungszustand gibt also den Grad der Aktivierung eines Neurons an und wird oft kurz als Aktivierung bezeichnet (Kriesel 2008). Der neue Aktivierungszustand wird aus der Netzeingabe, dem Schwellenwert und dem aktuellen Aktivierungszustand des Neurons berechnet. Da der aktuelle Aktivierungszustand des Neurons nicht immer in die Berechnung des neuen Aktivierungszustandes miteinbezogen werden muss, ist die Formel für die Aktivierungsfunktionen wie folgt definiert:

$$a_j = f_{act}(Net_j\ , \Theta_j, a_i)$$

Der Schwellwert Θ_j gibt an, ob das Eingangssignal mit der verwendeten Aktivierungsfunktion zu einer Aktivierung führt, wobei dieser Wert von Neuron zu Neuron individuell definiert werden kann.

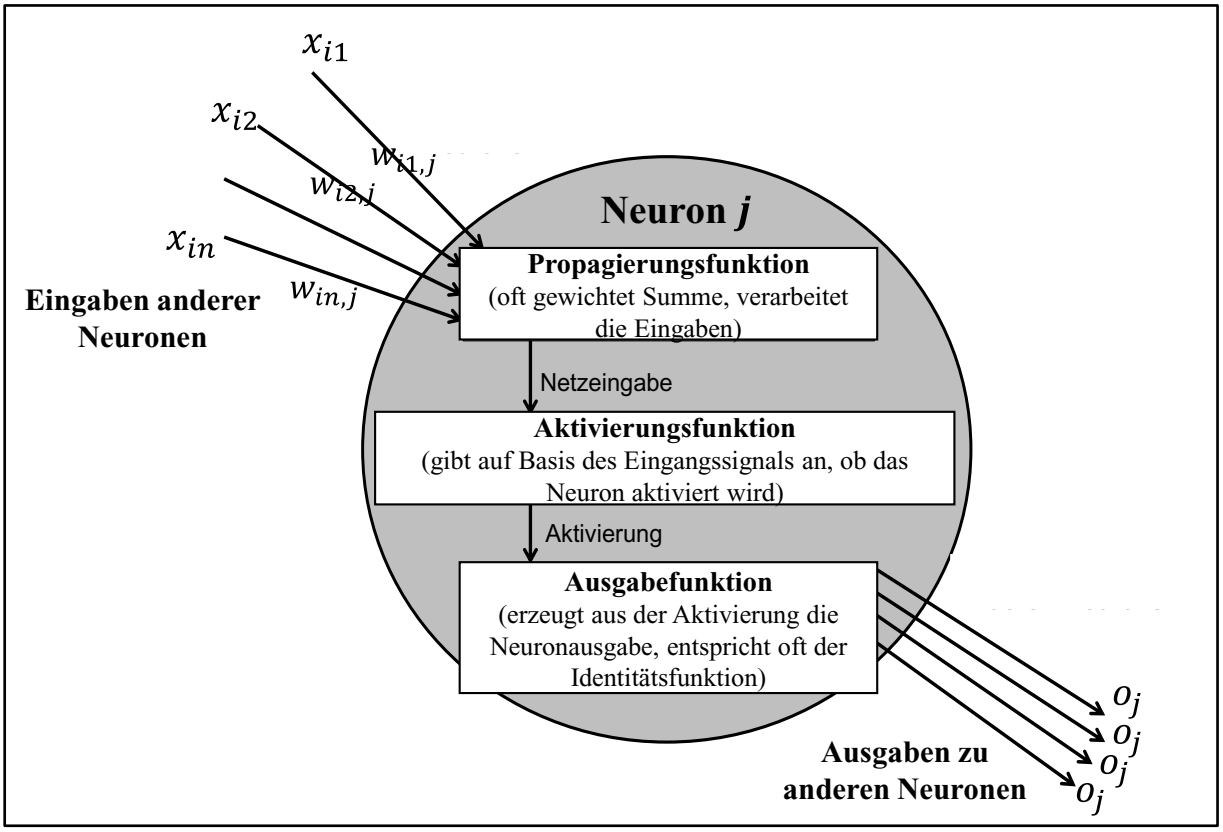

Abbildung 15: Datenverarbeitung eines technischen Neurons (in Anlehnung an Kriesel 2008)

Beispiele von oft verwendeten Aktivierungsfunktionen sind in Abbildung 16 dargestellt.

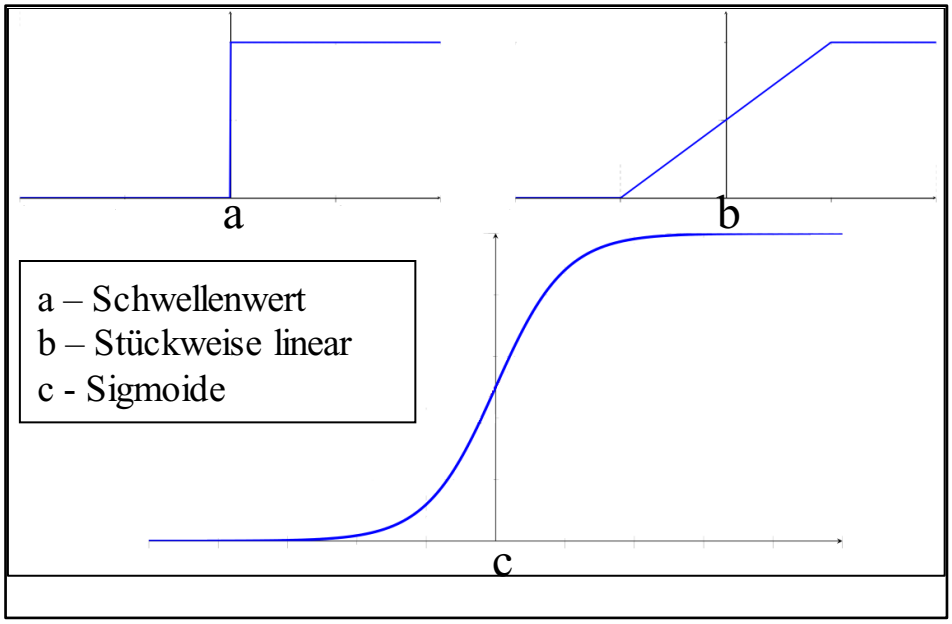

Abbildung 16: Beispiel einiger Aktivierungsfunktionen (in Anlehnung an Nielsen 2015)

Die Schwellwertfunktion gilt als die einfachste Aktivierungsfunktion:

$$a_j = \begin{cases} x_1 \text{ für } Net_j > \Theta_j \\ x_2 \text{ für } Net_j \leq \Theta_j \end{cases}$$

Die am häufigsten verwendete Aktivierungsfunktion ist die Sigmoid-Funktion (Nielsen 2015):

$$a_j = \frac{1}{1 + e^{(-Net_j)}}$$

Schließlich lässt sich der Ausgabewert des Neurons j aus der eigenen Aktivierung a_j mithilfe der Outputfunktion f_{out} berechnen. Dieser Ausgabewert wird an abgehende Kanten weiterübermittelt. Häufig ist die Ausgabefunktion die Identitätsfunktion $f_{out} = f_{act}$.

3.5.1.2. Neuronale Netztopologien – Feed-Forward-Netze

Aus mathematischer Sicht ist ein KNN ein Graph, welcher aus der Vernetzung von verschiedenen technischen Neuronen gebildet ist. In diesem Kontext besteht ein technisches neuronales Netz aus simplen Knoten oder Recheneinheiten, „den Neuronen“, sowie gerichteten, gewichteten Verbindungen zwischen diesen (Kriesel 2008), wobei das Netz aus einer Eingabeschicht, mindestens einer versteckten Schicht und einer Ausgabeschicht aufgebaut ist (Abbildung 17). Die Neuronen sind in diesem Zusammenhang in Schichten angeordnet. Jedes neuronale Netz ist mit der Außenwelt durch Eingabe- oder Ausgabe-Neuronen verbunden. Die Inputneuronen bzw. Ausgabeneuronen bilden die Eingabeschicht bzw. Ausgabeschicht und dienen als Schnittstelle sowohl für die Eingabedaten als auch die Ausgabewerte des künstlichen neuronalen Netzes.

Je nach der Art der Verbindungen zwischen Neuronen wird zwischen verschiedenen Topologien bzw. Bauarten des KNNs unterscheiden. Wenn die Verbindungen zwischen den Neuronen des Netzes keine Schleifen bilden und eine Verbindung eines Neurons immer ausschließlich zu einem Neuron der nächsten Schicht in Richtung Ausgabeschicht gerichtet ist, wird über eine Feed-Forward-Netztopologie gesprochen. Der prinzipielle Aufbau eines Feed-Forward-Netzes ist in Abbildung 17 dargestellt.

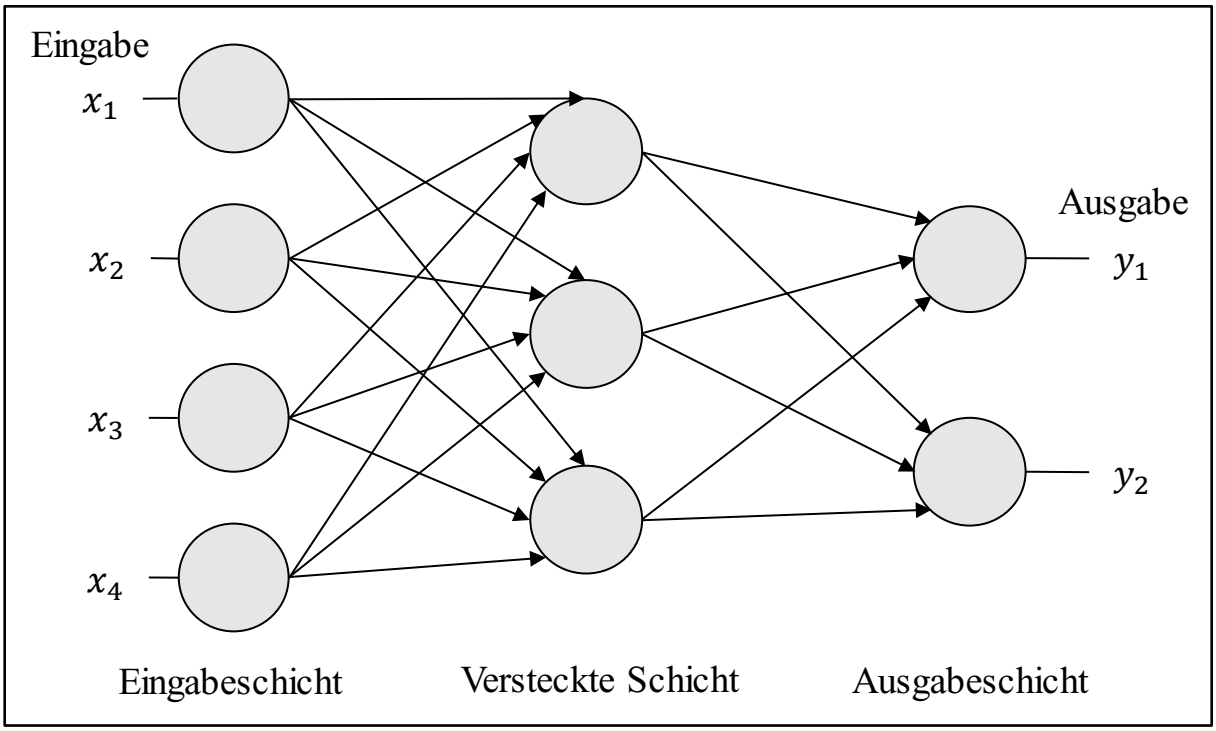

Abbildung 17: Beispiel eines dreischichtigen Feed-Forward-Netzes (in Anlehnung an Nielsen 2015)

Bei der Verwendung eines Feed-Forward-Netzes ist die Realisierung einer Funktion *F* mit Abhängigkeit vom Eingangsvektor *E* zu definieren. Für die Lernphase werden in der Regel mithilfe des sogenannten Backpropagation-Algorithmus die Gewichte des Netzes angepasst. Dazu werden die Trainingsbeispiele mit der korrekten Ausgabe genutzt. Dabei wird eine Kostenfunktion, die als Metrik bzw. Performance-Funktion zwischen den Trainingsdaten und der Netzausgabe dient, berechnet. Die am häufigsten verwendete Kostenfunktion ist die quadratische Fehlersumme (engl. mean squared error – MSE).

$$mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2,$$

Wobei N die Anzahl der Muster ist, die dem Netz vorgestellt werden, t_i die gewünschte Soll-Ausgabe und y_i die errechnete Ist-Ausgabe (Output des Netzes)

Der Backpropagation-Algorithmus besteht aus drei Stufen (Patan 2008), (Nielsen 2015):

1. Vorwärtsberechnung: Aus dem Eingabevektor wird der Ausgabevektor y vorwärts durch das Netz berechnet.
2. Rückwärtsberechnung (Backpropagation): Die quadratische Fehlersumme wird berechnet. Dieser Fehler wird über die Ausgabeschicht zur Eingabeschicht des Netzes zur Berechnung des Gradienten bezüglich der Gewichtsfaktoren des Netzes zurückpropagiert.
3. Anpassung der Gewichtsfaktoren.

3.5.1.3. Deep neuronale Netze (DNN)

Deep Learning (tiefgehendes Lernen) ist eine spezielle Erweiterung von künstlichen neuronalen Netzen. Im Gegensatz zu KNN besteht ein DNN aus mehreren versteckten Schichten, was auch den wesentlichen Unterschied zur KNN darstellt (Tishby et al. 2015). DNNs gehören zu den überwachten maschinellen Lernverfahren, welche die Netze durch ihre Kapazität gut trainieren, wenn große Datenmengen (Big Data) verfügbar sind (LeCun et al. 2015). Anstatt Modellen beizubringen, Daten zu verarbeiten und aus ihnen zu lernen, wie es beim herkömmlichen maschinellen Lernen der Fall ist, trainiert sich das Modell selbst, die Daten zu verarbeiten und aus diesen zu lernen. Ein DNN entdeckt komplexe Strukturen in großen Datensätzen, wobei der Backpropagationsalgorithmus verwendet wird, um die internen Parameter jeder Schicht aus den internen Parametern der vorherigen Schicht zu berechnen (LeCun et al. 2015). Um sicherzustellen, dass das DNN die gewünschten Ergebnisse liefert, werden große Datensätze benötigt. Ein DNN benötigt viele Erfahrungen, um Informationen zu lernen und abzuleiten, entsprechend werden große Datenmengen benötigt. Das erklärt auch, warum DNNs meistens in den Bereichen der Sprach- und Bildverarbeitung eingesetzt wird (LeCun et al. 2015). Die Komplexität eines neuronalen Netzwerks kann durch die Anzahl der Parameter ausgedrückt werden. Bei tiefen neuronalen Netzen kann diese Zahl sehr groß sein (Nielsen 2015). Dies führt dazu, dass das Training von DNN im Vergleich zu KNN sehr zeitaufwendig ist (Nielsen 2015). Darüber hinaus sind die meisten DNNs sehr gut darin, Probleme zu lösen, jedoch erfordert die Lösung eines sehr ähnlichen Problems das neue Training bzw. die neue Evaluierung des DNN (Nielsen 2015). Dies erschwert die Anwendbarkeit von DNN in einem dynamischen System, in welchem die Echtzeitanforderung notwendig ist.

3.5.2. Entscheidungsbaum

Entscheidungsbäume sind ein überwachtes maschinelles Lernverfahren, bei dem die Daten gemäß einem bestimmten Parameter aufgeteilt werden. Das Entscheidungsbaumverfahren ist eine häufig verwendete Data-Mining-Methode für das Erstellen von Klassifizierungssystemen zur Entwicklung von Vorhersage-Algorithmen für eine Zielvariable (Choudhary et al. 2009), (Bramer 2013). Diese Methode klassifiziert Daten in branchenähnlichen Segmenten, die einen in-

vertierten Baum mit einem Stammknoten, internen Knoten und Blattknoten erstellen. Der Algorithmus kann große und komplizierte Datensätze effizient verarbeiten, ohne eine komplizierte parametrische Struktur aufzuweisen (Backus et al. 2006). Der Hauptvorteil von Entscheidungsbäumen besteht darin, dass sie leicht in verständliche Regeln übersetzt werden können (Backus et al. 2006). Somit ist die Logik, die den Baum zu einer Entscheidung bringt, für den Benutzer sehr klar definiert. Wie der Name schon sagt, wird ein Entscheidungsbaum grafisch als Baum dargestellt (siehe Abbildung 18). Ein solcher Baum wird von oben nach unten gelesen. Sobald ein Knoten gekreuzt wird, wird eine Entscheidung getroffen, die durch einen Test auf einem der Systemattribute repräsentiert wird. Für jeden Test sind mehrere Entscheidungen möglich. Wenn das Attribut binär ist, sind zwei mögliche Entscheidungen gegeben. Wenn das Attribut k Modalitäten hat, sind k mögliche Entscheidungen gegeben. Der Baum wird daher entsprechend der Anzahl der Attribute des Systems erweitert, aber auch entsprechend der Anzahl der Modalitäten für jedes Attribut. Die Endknoten des Baumes sind die Blätter des Baumes. Sie stellen die endgültige Entscheidung dar, also die Klasse der Zugehörigkeit, deren Beobachtung für die verschiedenen Tests des Baumes verwendet wurde (Sha et al. 2005).

In dem in Abbildung 18 gezeigten Beispiel tritt eine Störung immer dann auf, wenn der Sensorwert des Sensors 1 kleiner als 25 und des Sensors 2 größer als 12 und des Sensors 3 kleiner als 40 ist.

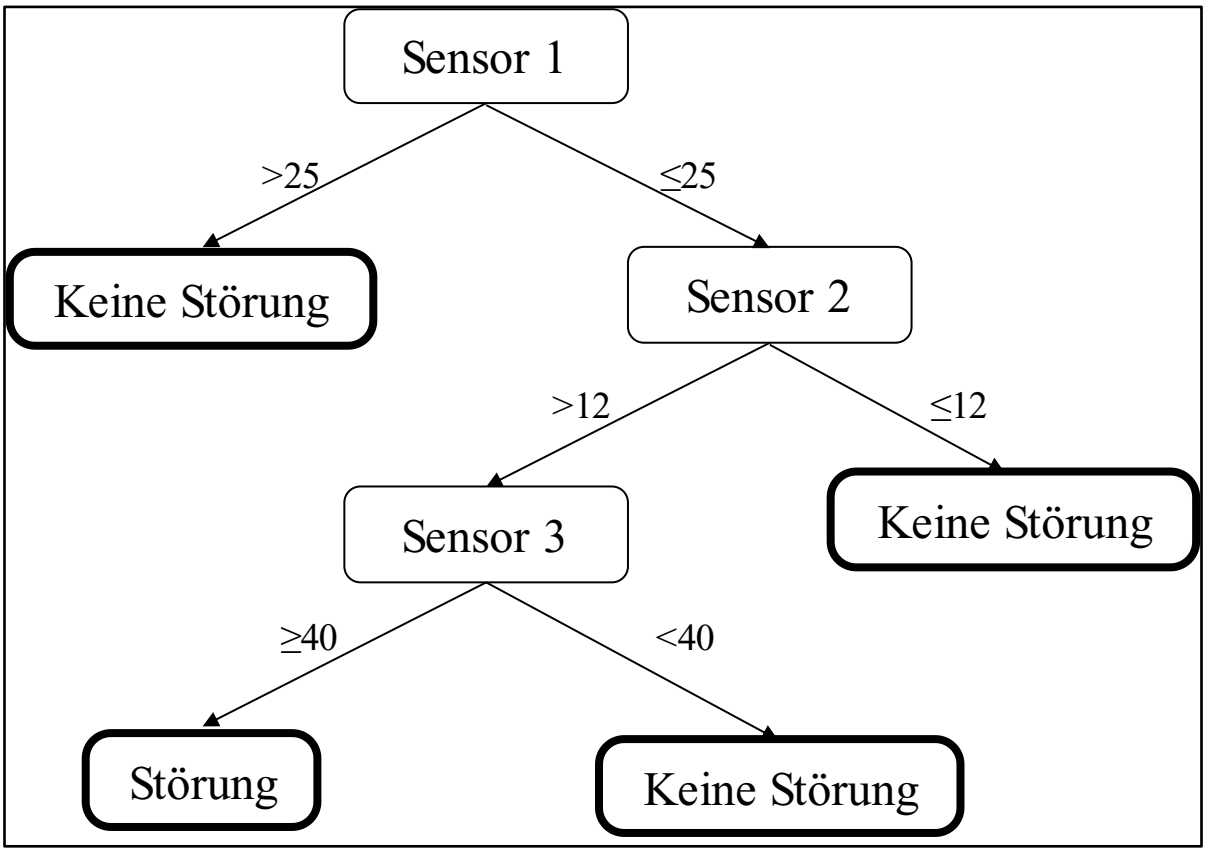

Abbildung 18: Beispiel eines Entscheidungsbaumes

Obwohl die Ausführung eines bereits erstellten Entscheidungsbaums auf neue Daten schnell erfolgt, ist das Erstellen bzw. Erzeugen des Entscheidungsbaums sehr zeitaufwendig. Es gibt eine Vielzahl von Algorithmen zur Konstruktion eines Entscheidungsbaums, wobei die Hauptalgorithmen CART (Classification And Regression Tree) und ID3 (Iterative Dichotomiser 3) sind.

Zusammenfassend ist der Gegenstand des Data-Mining, nach Mustern in Daten zu suchen. Unter Mustern können Zusammenhänge in Form von Korrelation oder Abhängigkeiten verstanden werden. Auf Basis dieser Muster werden beispielsweise Situationen in komplexen Systemen identifiziert. Deswegen kommt eine Vielzahl von Methoden zum Einsatz, um diese Muster festzustellen. Im Anhang A werden einige Methoden dargestellt, welche beispielsweise für die Diagnose und Prognose von Systemzuständen verwendet werden können.

4. Complex Event Processing

Die letzten Jahre haben die Entstehung neuer Klassen von datenintensiven Anwendungen, wie Sensordatenverarbeitung, Verkehrsüberwachung und Aktienhandel, erlebt. In diesen Anwendungen müssen die Daten mit einer hohen Eingangsrate und kontinuierlich über lange Zeiträume verarbeitet werden. Typischerweise liegen die Daten in Form eines Datenstroms vor. Eine Herausforderung liegt in der Entwicklung von Verarbeitungsansätzen für die kontinuierliche Berechnung und Aggregation eingehender Daten, um interessante Änderungen oder Muster rechtzeitig zu identifizieren und zu implementieren (Chakravarthy et al. 2009).

Dieses Kapitel beschreibt die Grundlage von CEP. Des Weiteren wird EDA als Architekturstil eingeführt und das CEP als Technologie für die Verarbeitung von komplexen Ereignissen in einer EDA-Architektur ausführlich vorgestellt. Hierbei wird auf die wesentlichen Eigenschaften, Begriffe und Fachterminologien von CEP eingegangen.

4.1. Architekturen und Konzepte zur Verarbeitung von Datenereignissen

4.1.1. Konzepte zur Verarbeitung von Daten – Paradigmenwechsel

Mit dem Data Stream Processing System (DSPS) bzw. Datenstrommanagementsystem (engl. data stream management system – DSMS) existiert ein konzeptioneller Ansatz, der darauf abzielt, die Analyse von Datenströmen mittels kontinuierlicher Abfragen (engl. continuous queries) durchzuführen (Chakravarthy et al. 2009). Dabei stellt die ereignisorientierte Architektur eine leistungsfähige Architektur zur Ereignisverarbeitung dar. Die CEP-Technologie liefert die Art von Analysen, die normalerweise von relationalen Datenbankenmanagementsystemen (DBMS) bereitgestellt werden können, allerdings in Echtzeit und in einer ereignisgesteuerten Implementierung. Ein traditionelles relationales DBMS wurde entworfen, um eine Anfrage auf einem Block von Daten, die in einer persistenten Datenbank zuerst gespeichert und indexiert werden, mittels einmalig gestellten SQL-Abfragen auszuführen. Durch diese Abfragen können beispielweise eine Regressionsanalyse bzw. eine statistische Analyse durchgeführt werden, um Informationen aus Prozessen abzuleiten und nach Mustern zu suchen. Dabei erfolgen diese Analysen offline nach Prozessende und nicht als Reaktion auf eingehende Daten. Ein Datenstrommanagementsystem nimmt dagegen eingehende Daten und führt sie durch einen Satz vordefinierter kontinuierlicher Anfragen aus, um abgeleitete Datenströme oder Datensätze zu erzeugen. Der grundlegende Unterschied zwischen einem DBMS und einem DSMS ist in Abbildung 19 und Abbildung 20 zusammengefasst. Abbildung 19 veranschaulicht die Verwendung eines herkömmlichen DBMS für die Verarbeitung von Daten-

strömen. Die Daten werden zunächst in einer Datenbank persistent gespeichert und über gespeicherte Abfragen werden Daten verarbeitet und analysiert. Dieser Ansatz führt zu einer erhöhten Reaktionszeit für Abfrageergebnisse bzw. Verarbeitungsergebnisse, hauptsächlich wegen der hohen Ankunftsraten von Daten, welche mit dem Speicher und der Abruflatenz gekoppelt sind (Chakravarthy et al. 2009). Im Gegensatz dazu zeigt Abbildung 20 eine Architektur, in der die eingehenden Datenströme direkt von einem DSMS verarbeitet werden. Die Archivierung wird in diesem Zusammenhang parallel zur Datenverarbeitung durchgeführt, jedoch ohne Beeinträchtigung der Datenanalyse und mit der Durchführung von kontinuierlichen Abfragen, wodurch die Reaktionszeit der Abfrageergebnisse verringert wird (Chakravarthy et al. 2009).

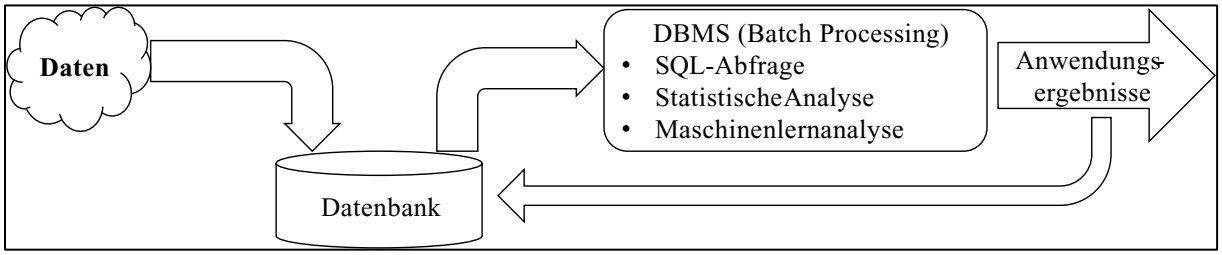

Abbildung 19: Architektur eines Datenbankmanagementsystems

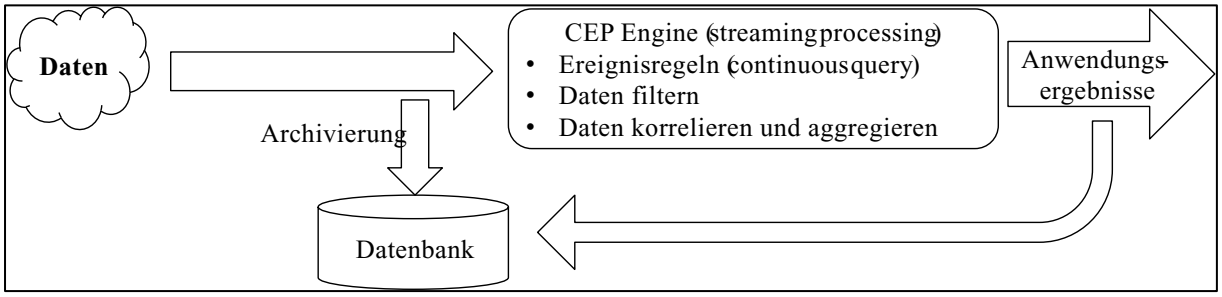

Abbildung 20: Architektur eines Datenstrommanagementsystems

Zusammengefasst liegt der kritische Unterschied zwischen einem Datenbankmanagementsystem und einem Datenstrommanagementsystem in der Art und Weise der Datenanalyse. Während DBMS auf zuvor erworbene Daten zurückblickt, um daraus Information bzw. Wissen abzuleiten (Batchverarbeitung), wird diese Datenanalyse in DSMS (Streaming-Verarbeitung) in Echtzeit und online durchgeführt. Folglich können die traditionellen DBMS jedoch nicht bei Anwendungen eingesetzt werden, in denen kontinuierliche große Datenmengen eintreffen werden.

4.1.2. Allgemeiner Aufbau einer ereignisorientieren Architektur

Ereignisorientierte Architektur ist ein Oberbegriff, der einen Ansatz für Informationssysteme beschreibt, der auf das Lauschen (engl. listening) und die Reaktion auf Ereignisse angewiesen ist. Diese Ereignisse können beispielweise das Senden eines Sensorwertes oder des Einscannens eines Barcodes oder RFID-Tags

sein. Allgemein besteht ein EDA-System aus einem Ereignishörer (engl. event listener) und Ereignisbehandlungsprozessen, die auf Ereignisse reagieren. Während EDA einen Ansatz für Software-Design darstellt, geht es bei CEP darum, Regeln zu verwenden, um auf mehrere Ereignisströme zu reagieren (Bruns et al. 2010).

EDA folgt dem publish/subscribe-Modell, das asynchrone Kommunikation zwischen Softwarekomponenten bereitstellt. In diesem Zusammenhang stellt die Ereignisquelle die „Publisher“ und die Ereignissenke die „Subscriber“ dar. Auf diese Weise werden die Ereignisquelle und die Ereignissenke verknüpft und die CEP-Anwendung übernimmt das Abhören, die Interpretation und die Verarbeitung der Ereignisinformationen. Anders als bei einer ablauforientierten Serviceorientierten Architektur (SOA), in der die Kommunikation zwischen Softwarekomponente synchron und nach dem Prinzip request/response für den Aufruf eines Dienstes erfolgt, wird eine asynchrone Kommunikation zwischen Softwarekomponenten nach dem publish/subscribe-Modell erfolgen (Tarkoma 2012), (Mousheimish 2017). Dadurch wird sichergestellt, dass eine Softwarekomponente nicht durch einen Dienstaufruf blockiert wird. In diesem Fall wird von einer entkoppelten Kommunikation zwischen Softwarekomponenten gesprochen. In einer EDA-Architektur werden die Ereignisse zwischen Teilsystemen in der Regel in Form von Nachrichten ausgetauscht. Hierbei sind sich die Ereignisquellen, welche die Ereignisse erzeugen, nicht der Existenz von Ereignissenken bewusst, die das Interesse an bestimmten Ereignistypen erfragen. Jedoch ist für einen erfolgreichen Austausch der Nachrichten das Teilen bzw. die Verständigung über die Semantik der Nachrichten zwischen den Ereignisquellen und Ereignissenken erforderlich. Hauptsächlich haben die ereignisorientierten Systeme die Aufgabe, die Ereignisse zu erkennen, zu verarbeiten und anschließend darauf zu reagieren (Abbildung 21). Dabei werden wieder neue Ereignisse erzeugt und an weitere Teilsysteme gesendet (Bruns et al. 2010).

CEP verfolgt dabei das Ziel, Muster in stetig einhergehenden Daten online und in Echtzeit zu entdecken (Thoben et al. 2018). Dies geschieht durch: i) Kombination von Daten aus mehreren Datenquellen und anschließendem Erzeugen eines neuen abgeleiteten Ereignisstroms mit reichen Informationen, ii) Generierung von High-level-Informationen wie Zusammenfassungsdaten, Statistiken und Trends, um einen Einblick in verschiedene Einzelereignisse zu schaffen, iii) Suche nach bestimmten Mustern, um eine sofortige Reaktion auf der Grundlage einer komplexen Analyse der eingehenden Daten zu ermöglichen.

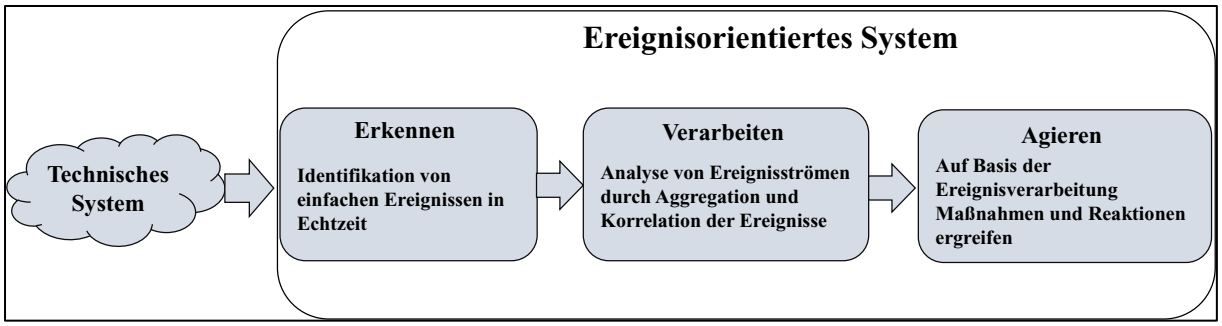

Abbildung 21: Aufgabe eines ereignisorientierten Systems

Erkennen: Das EDA nimmt Daten aus verschiedenen internen oder externen Quellen entgegen. Es können sowohl Sensor- als auch historische oder fachliche Geschäftsdaten sein. Die Aufgabe des Erkennens liegt in der Extraktion von Informationen aus diesen Daten. Diese gewonnenen Informationen werden in Form von Ereignissen vorgelegt. Diese Ereignisse werden für die weitere Verarbeitung als komplexe oder komposite Ereignisse definiert (Gillani et al. 2017).

Verarbeiten: Allgemein lassen sich einige Zustände des Systems nicht durch Eintreten einzelner Ereignisse erkennen, sondern durch das Zusammenspiel bzw. den Zusammenhang verschiedener Ereignisse. Nach dem Erkennen der ersten Informationen in Form von Ereignissen erfolgt in diesem Schritt die Verarbeitung dieser Ereignisse. Dabei werden die Muster als Ergebnisse von Korrelation, Aggregation oder Klassifikation zwischen den Ereignissen gesucht (Akbar et al. 2015).

Reagieren: In diesem Schritt können Reaktionen und Maßnahmen als Ergebnis des Verarbeitungsschritts initiiert werden. Das Senden einer Warnmeldung, Stoppen einer Maschine oder Auslösen eines Reparaturauftrags sind Beispiele für Reaktionen (Bruns et al. 2010).

4.2. Verarbeitung komplexer Ereignisse in ereignisorientierten Systemen

Da CEP-Systeme auf dem Prinzip der Verarbeitung der eingehenden Stromdaten beruhen, ist eine Architektur erforderlich, die durch die folgenden Hauptaspekte gekennzeichnet ist: die Fähigkeit, große Mengen an Daten verarbeiten zu können (Datenvolumen), eine schnelle Datenverarbeitung (Geschwindigkeit), heterogene Daten mit verschiedenen Datentypen zu verarbeiten (Vielfalt) (Flouris et al. 2017), (Bajaj et al. 2014).

Eine zentrale Herausforderung bei der Entwicklung von Zustandsüberwachungssystemen liegt in der Verarbeitung großer Datenmengen bei hoher Systemkomplexität in Echtzeit (Bruns et al. 2010). Darüber hinaus ist die Datenverarbeitung mit kontinuierlicher Analyse und kurzer Latenzzeit in Echtzeit für Zustandsüberwachungssysteme eine anhaltende Anforderung (Akbar et al. 2015). Die konventionellen serviceorientierten Architekturen stoßen dabei aufgrund

zunehmender Komplexität technischer Systeme und steigender Anforderungen an deren Zuverlässigkeit an ihre Grenzen, weshalb der Paradigmenwechsel zu ereignisgesteuerten Architekturen vollzogen wird (Bruns et al. 2010), (Bajaj et al. 2014). Mit EDA wurde die Grundlage für die Entwicklung von Informationslandschaften geschaffen, die auf Ereignisverarbeitung ausgerichtet sind. Hieraus entwickelte sich mit dem CEP ein neuartiger technologischer Ansatz, der auf die dynamische Verarbeitung komplexer Ereignisse ausgerichtet ist. CEP hat zur Aufgabe, Muster voneinander abhängiger Ereignisse in einer großen Ereigniswolke zu identifizieren und das Auftreten solcher Ereigniskombinationen zu erkennen (Esposito et al. 2015).

Ein Beispiel für ein einfaches Ereignis ist ein Sensor, der einen Wert generiert. Ein Beispiel für ein anderes Ereignis ist ein Sensor, dessen Wert einen Schwellenwert überschreitet und für ein komplexes Ereignis ein Diagnosesystem, das einen Alarm über eine Störung sendet. Die einfachen bzw. einzelnen Ereignisse verfügen im Allgemeinen nicht über ausreichende Informationen für die Ermittlung des Systemzustands. Jedoch ermöglicht die Kombination bzw. der Zusammenhang von verschiedenen Ereignissen, einen besseren Einblick in das System zu gewinnen, da zwischen Ereignissen komplexe Wechselwirkungen bestehen.

Bruns und Dunkel (2010) stellen die Vorteile von Ereignisorientierung als Architekturstil im Vergleich mit der traditionellen ablauforientierten Architektur vor und wie ereignisgesteuerte Geschäftsprozesse von CEP profitieren können. Darüber hinaus spezifizieren sie für die Implementierung von CEP drei wesentlichen Schichten: Ereignisquellenschicht (Event Producers), Ereignisverarbeitungsschicht und Ereignisbehandlungsschicht (Event Consumers) (siehe Abbildung 22).

Die Ereignisquellenschicht wird aus Sensornetzwerken und Geschäftsaktivitäten gebildet, die stetig Daten an die Ereignisverarbeitungsschicht senden. Da diese Daten in verschiedensten Datenformaten vorliegen, werden sie mittels Ereignisadaptern in ein einheitliches Format für die Weiterverarbeitung transformiert. In diesem Zusammenhang wird zwischen verschiedenen Klassen von Ereignissen unterschieden. Die erste Klasse umfasst Ereignisse, die sich durch die Überschreitung eines Schwellwerts detektieren lassen. Dies bildet in Kombination mit historischen Daten und anderen Quelldaten die Ereigniswolke (engl. event cloud, ein von Luckham geprägter Begriff) (Luckham 2002). Die zweite Ereignisklasse besteht aus Ereignissen, die sich als temporaler Zusammenhang von einfachen Ereignissen darstellen lassen. Ein Beispiel eines Ereignisses der zweiten Klasse wäre: „Innerhalb der letzten 5 Stunden ist der Sensorwert um 30 % gestiegen.“ Die letzte Ereignisklasse umfasst die komplexen Ereignisse, die als Komposition oder Aggregation von vielen Ereignissen bezeichnet werden. Im Rahmen der Ereignisverarbeitung werden zunächst die einfachen Ereignisse auf Fehler und Duplikate hin analysiert. Dann werden die Ereignisse aggregiert und nach auftretenden Ereignismustern abgesucht. Hierfür ist eine Event Processing

Engine (EPE) zuständig, die dabei auf das Ereignismodell und die Ereignisregeln zurückgreift.

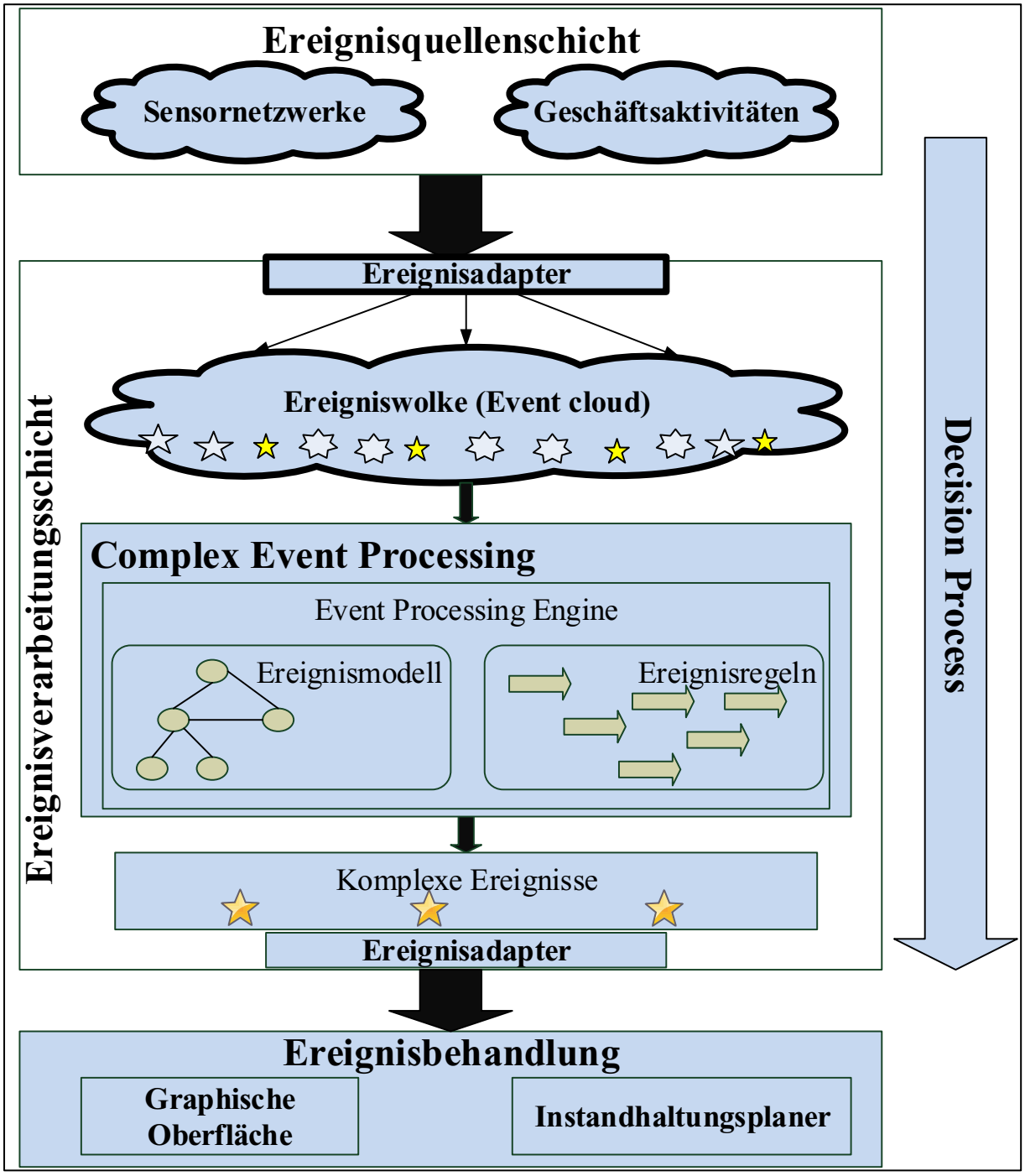

Abbildung 22: CEP-Grundkonzept (in Anlehnung an Bruns et al. 2010)

Das Ereignismodell beinhaltet eine exakte Spezifikation aller relevanten Ereignisse. Hierfür sind Eigenschaften und Abhängigkeiten der betrachteten Ereignisse für die jeweilige Anwendungsdomäne hinterlegt. Ausgehend vom Ereignismodell werden Ereignisregeln zur Verarbeitung der Ereignisse definiert (Bruns et al. 2010).

Mithilfe der Ereignisregeln können die Ereignisse gefiltert, aggregiert, korreliert und neue komplexe Ereignisse generiert werden. Hierzu lassen sich spezielle Ereignisverarbeitungssprachen (engl. event processing languages – EPL) nutzen (Fülöp et al. 2010):

i) EPL basierend auf Algebra (engl. composition operators)
ii) EPL basierend auf SQL-Query
iii) EPL basierend auf Produktionsregeln (engl. production rules)

Hierfür wird ein Abstraktionsmodell für Ereignisse definiert, um zwischen einfachen Ereignissen (engl. low level events), wie „ein Sensor schickt einen Wert“, und die hierarchischen höheren Ereignisse, wie „innerhalb der letzten Stunden ist der Sensorwert um 30 % gestiegen“, unterscheiden zu können. Dieses Abstraktionsmodell basiert auf einer hierarchischen Ebenen-Architektur, typischerweise aufgeteilt in viele Abstraktionsebenen. Dabei besitzen die Low-Level-Ereignisse keine semantische Signifikanz für den Entscheidungsprozess, im Gegensatz zu den hierarchisch höher gestellten Ereignissen, die sich aus einer definierten Kombination von Low-Level-Ereignissen zusammensetzen (Widder 2011). Wichtig beim Erkennen von komplexen Ereignissen ist, die kausalen Zusammenhänge zwischen verschiedenen Ereignissen auf verschiedenen Abstraktionsebenen festzulegen, da ein Wartungsauftrag zum Beispiel von den Ursachen der Ausfälle nach Eintritt des komplexen Ereignisses abhängig eingeleitet wird. Das Nachvollziehen von kausalen Abhängigkeiten bietet die Möglichkeit, eine Erklärung für das Entstehen des Ereignisses und somit auch den Grund der Störungen zu identifizieren (Bruns et al. 2010).

4.3. Beispiel eines ereignisorientierten Instandhaltungssystems

Am Beispiel eines Instandhaltungssystems wird das ereignisorientierte Konzept veranschaulicht. Ein Beispiel für ein ereignisorientiertes Instandhaltungssystem für Windkraftanlagen ist in Abbildung 23 visualisiert. Das Instandhaltungssystem muss mit diversen Systemen bzw. mit deren Prozessen durch den Austausch von Nachrichten kommunizieren. In diesem Zusammenhang entspricht jede Nachricht dem Eintreten eines Vorkommnisses im System. Zum Beispiel analysiert das Zustandsüberwachungssystem bzw. Fehleranalysemodell eines technischen Systems viele Daten aus verschiedenen Sensoren und meldet dem Leitservice eine mögliche Störung. Folglich wird eine Nachricht generiert und gesendet. Diese Nachricht löst einen Prozess beim Leitservice aus, der die Störung auswertet und darauf reagiert. Des Weiteren generiert der Leitservice einen neuen Instandhaltungsauftrag, der in Form einer Nachricht zum Instandhaltungsservice gesendet wird. Der Instandhaltungsservice reagiert auf diese Nachricht und überprüft sowohl die Wettervorhersage als auch die Verfügbarkeit der Ressourcen für eine erfolgreiche Durchführung des Auftrages.

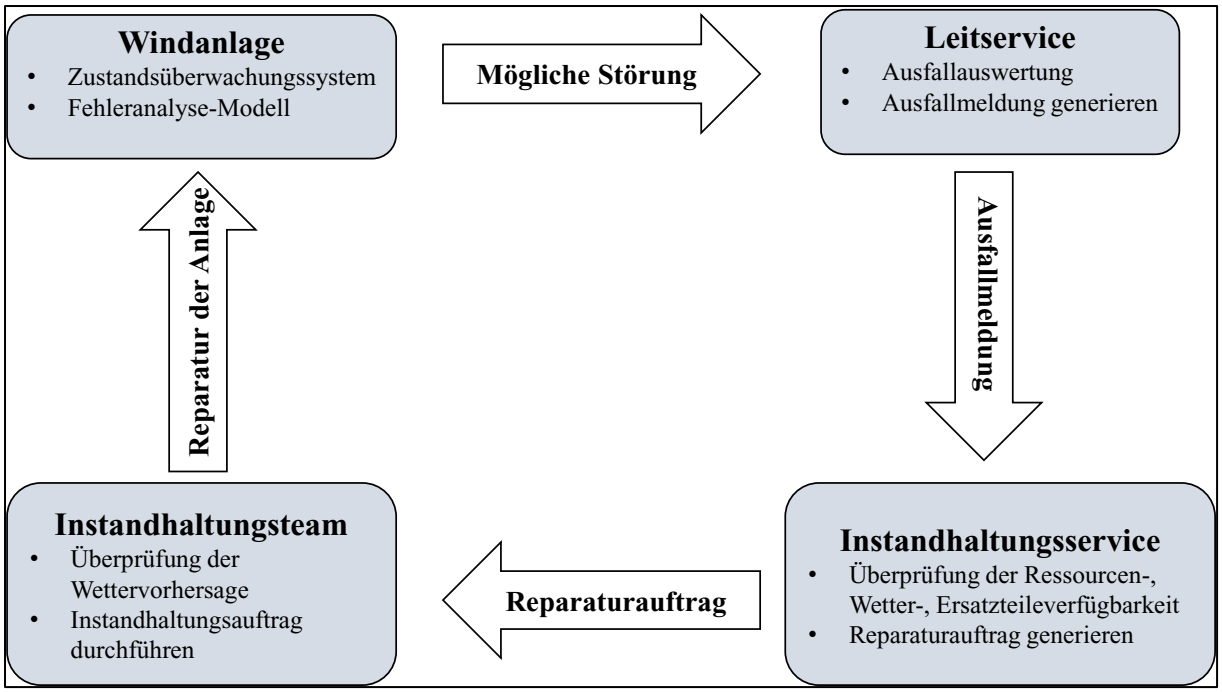

Abbildung 23: Prozessablauf eines Instandhaltungssystems mit Ereignisbeispielen

Es kann festgehalten werden, dass sich jede Nachricht als Repräsentation eines eingetretenen Ereignisses darstellen lässt. Das ganze System lässt sich somit als ereignisgesteuert bezeichnen.

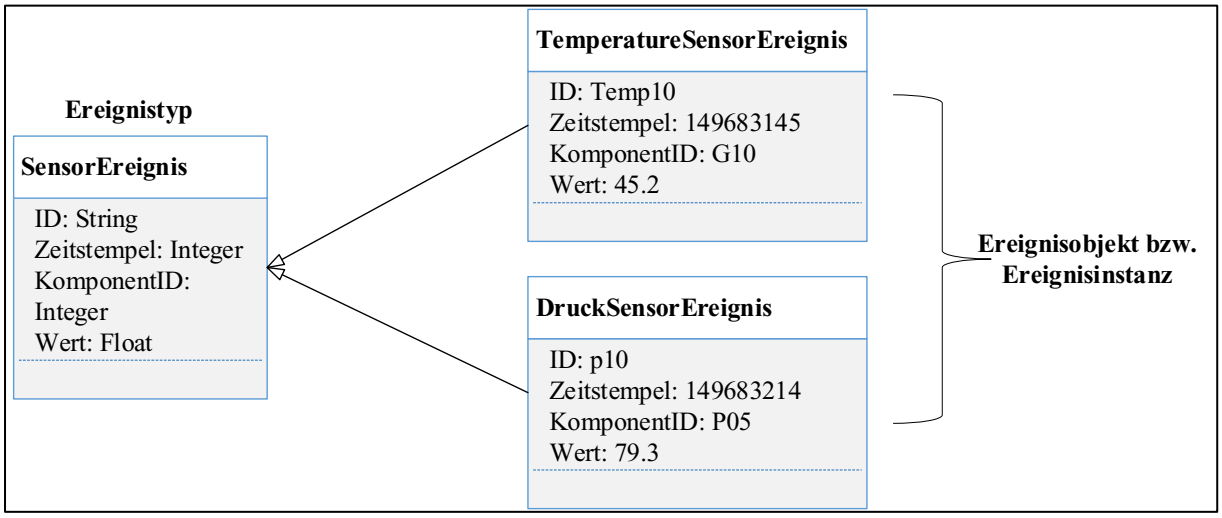

Abbildung 24: Beispiel eines Ereignistyps und dessen Ereignisinstanzen

In dem Beispiel von Abbildung 24 ist die Struktur eines Sensorereignisses definiert. Ein Sensorereignis ist mit einem Zeitstempel versehen und enthält eine eindeutige Identifikationsbezeichnung ID, eine eindeutige Identifikationsbezeichnung der übergeordneten Komponente und den Sensorwert. Für dieses Beispiel sind die beiden Ereignisinstanzen ein konkretes Vorkommnis des Ereignisses vom Typ Sensor.

Die Abbildung 25 zeigt ein vereinfachtes Ereignismodell des Instandhaltungsbeispiels von Abbildung 23.

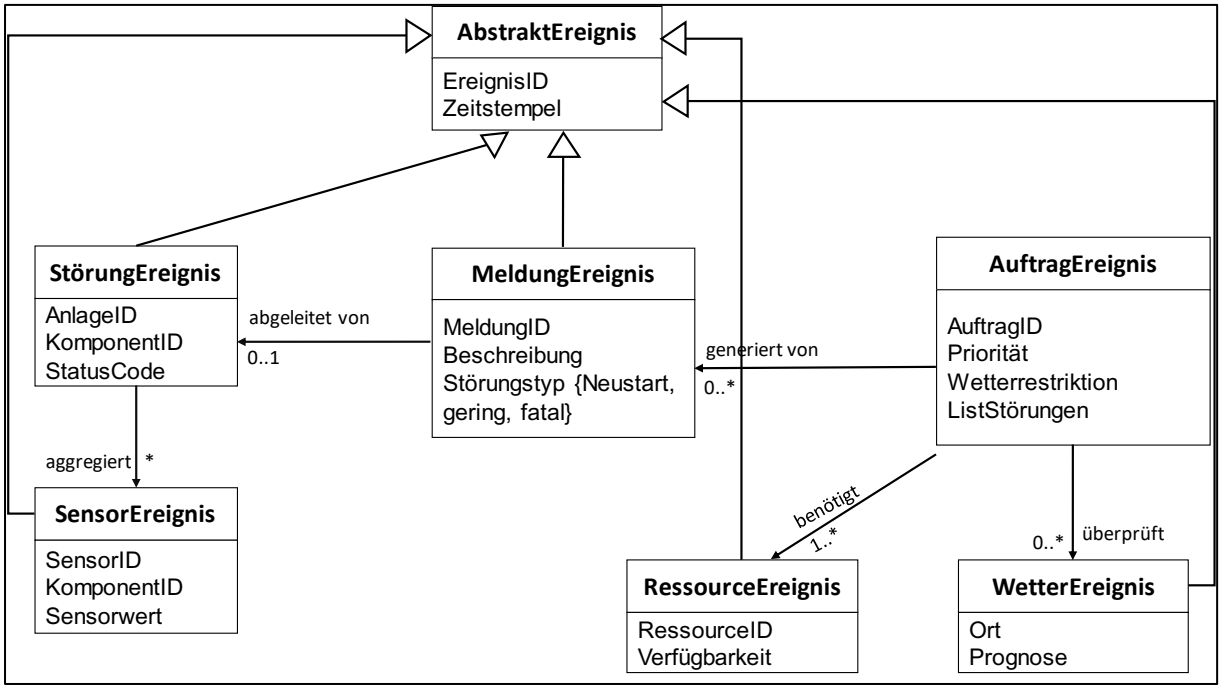

Abbildung 25: Ereignismodell für das Instandhaltungsbeispiel der Abbildung 23

Abstraktes Ereignis bezeichnet die Klasse, die Metadaten enthält und als Strukturelement für alle möglichen Ereignistypen innerhalb des Ereignissystems dient. In diesem Zusammenhang erben alle Ereignisinstanzen die Attribute der abstrakten Klasse. Sie werden durch eine eindeutige Identifikationsbezeichnung *(EreignisID)*, zur Unterscheidung des Ereignisses im System, und den Auftrittszeitpunkt *(Zeitstempel)* repräsentiert.

Sensorereignis steht für die Ereignisse, die unmittelbar von den Sensoren in einer Anlage ausgesendet werden. Diese Sensorereignisse enthalten neben Sensordaten, deren Verarbeitung zur Identifikation von Störungen führt, eine Sensor-Identifikationsbezeichnung (SensorID) und die ID der Komponente (KomponentID), an der sich der Sensor befindet.

Störungsereignis beschreibt mögliche aufgetretene Störungen in einer Anlage. Diese werden erkannt, indem die Ereignisströme der Sensorereignisse auf kritische Muster hin überprüft werden. Ein Störungsereignis kann als aggregiertes Ereignis (komplexes Ereignis) verstanden werden, das eine Vielzahl einzelner Sensorereignisse zusammenführt.

Meldungsereignis repräsentiert eine Ableitung von einem Störungsereignis. Hier werden die Störungsereignisse auf Korrektheit und Redundanz geprüft und gefiltert. Handelt es sich beispielsweise um einen Meldungsereignis-Typ „Neustart“, wird die Anlage ferngestartet, andernfalls wird hier die Ernsthaftigkeit der Störung definiert.

Ressourcen-Ereignis steht für Ereignisse, die Ressourcen für die Durchführung der Instandhaltungsaufträge benötigen. Je nach Zeitpunkt können diese Ereignisse verschiedene Verfügbarkeiten haben.

Wetter-Ereignis repräsentiert die Ereignisse, die beispielweise von verschiedenen Wetterdienstleistern ausgesendet werden. Sie enthalten die Wettervorhersage für einen bestimmten Ort, der mit einem Windparkstandort übereinstimmt.

Auftragsereignis beschreibt den Ereignistyp, der auf Basis von Ressourcenverfügbarkeit und guter Wettervorhersage generiert wird. Ein Auftragsereignis ist ein komplexes Ereignis, das durch die zeitliche Kombination von Wetterereignis und Ressourcenereignis abgeleitet wird.

4.4. Zentralisiertes und verteiltes CEP-System

Für die Bereitstellung bzw. die Umsetzung von CEP-Anwendungen sind Softwarekomponenten erforderlich, die für das Starten und die Initiierung der CEP-Engine verantwortlich sind. In diesem Kontext kommen die sogenannten Ereignisverarbeitungsagenten (engl. event processing agent – EPA) zum Einsatz, die den Ereignisverarbeitungsaufgaben – „das Monitoring des Eingangsstroms, die Mustererkennung und die Reaktion auf komplexe Ereignisse“ (Correira et al. 2015) – nachkommen.

In dem vorgestellten Grundkonzept des CEP in Abbildung 26 übernehmen die Event Processing Agents die Aufgaben der Ereignisverarbeitungsschicht. Dabei können diese Aufgaben von einem einzelnen EPA (zentral) oder von mehreren EPAs (verteilt) durchgeführt werden. Auf diese Weise bilden die unterschiedlichen Ereignisverarbeitungsagenten ein Ereignisverarbeitungsnetzwerk (engl. event processing network – EPN). In diesem Fall stellt jede Ereignissenke eines Agenten die Ereignisquelle Nachfolger EPA dar.

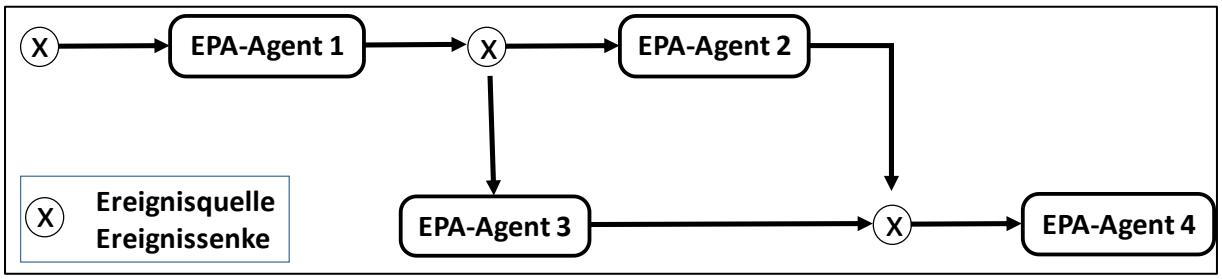

Abbildung 26: Beispiel eines Ereignisverarbeitungsnetzwerks

Bei einem zentralisierten Ansatz übernimmt ein einzelner EPA die Ereignisverarbeitungsaufgaben. Der kann mit einer einzelnen Ereignisquelle oder mehreren -quellen und auch mit einer oder mehreren Ereignissenken verknüpft sein. Die vorhandenen Ereignisregeln sind auf einem leistungsstarken Rechner gespeichert und alle eingetroffenen Ereignisse werden von dem EPA verarbeitet. Der Hauptvorteil des zentralisierten Ansatzes liegt in der Einfachheit der Implementierung des CEP-Systems, da die Bereitstellung der Regeln nicht auf verschiede-

nen Knoten des Netzwerks stattfinden wird. Jedoch stellt die Skalierbarkeit dieses Ansatzes eine große Herausforderung dar, wie es für zentralisierte Systeme üblich ist (Flouris et al. 2016). Steigt die Anzahl der Ereignisquellen, führt das zu einer Überlastung des Systems und somit zu längeren Reaktionszeiten bei der Erkennung der komplexen Ereignisse. Für die Verarbeitung von stetig ankommenden Daten in Echtzeit wird eine skalierbare Architektur benötigt. Dabei wird die Ereignisverarbeitungsaufgabe verschiedenen Hardwarekomponenten in einem verteilten System zugeteilt. In diesem Zusammenhang übernehmen verschiedene Event-Processing-Agenten in einem Event-Processing-Netzwerk eine Aufgabe innerhalb des Netzwerkes. Durch das Hinzufügen mehrerer EPAs für die „vertikale Skalierung“ erhöht sich dabei der Durchsatz der Datenverarbeitung, dementsprechend wird die Skalierbarkeit in der Datenverarbeitung gewährleistet (Meister 2014), (Canizo et al. 2017).

4.5. Syntax zur Beschreibung von Ereignissen, Ereignisregeln und Ereignismustern

4.5.1. Ereignisdarstellung, -definition und -begriffe

Dieser Abschnitt beschreibt die verschiedenen Möglichkeiten, Ereignisse in einem CEP-System zu modellieren und darzustellen.

Luckham (2002) stellt die Grundlagen von CEP in seinem Buch „The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems“ vor, indem er die diversen Vorarbeiten im Kontext der Verarbeitung von Ereignissen zum Konzept des Complex Event Processing zusammenführt. Er definiert ein Ereignis als ein Objekt, das eine Aufzeichnung einer Aktivität darstellt.

Luckham (2002) definiert hierzu grundlegend den Begriff des **Ereignisses** als ein Objekt, das eine Aufzeichnung einer Aktivität darstellt. Im CEP-Kontext ist ein Ereignis also ein **diskretes** Auftreten bzw. Geschehen einer Aktivität.

Für die Beschreibung von Ereignissen in einem CEP-System ist die Definition von Metadaten bzw. Ereignistypen erforderlich. In diesem Zusammenhang wird jede Instanz eines Ereignistyps als Ereignisinstanz bezeichnet (Abbildung 24).

Eine Ereignisinstanz ist eine unveränderliche Aufzeichnung eines vergangenen Auftretens einer Aktion oder Beobachtung einer Zustandsänderung.

Ereigniseigenschaften erfassen die Zustandsinformationen für eine Ereignisinstanz, die eine technische Repräsentation benötigt. Jede Ereignisinstanz e instanziiert eine abstrakte Klasse (auch bezeichnet als **Ereignistyp** E), die als Strukturelement innerhalb einer Klassenhierarchie dient. Allgemein lässt sich eine Ereignisinstanz als $e = < t , A >$ repräsentieren, wobei t der Auftrittszeitpunkt der Ereignisinstanz und $A = (\mathrm{attr}_1, \mathrm{attr}_2, \ldots , \mathrm{attr}_n)$ die Liste der Ereignis-

attribute ist. Die Ereignisattribute beschreiben den Kontext, in dem das Ereignis aufgetreten ist. Diese beinhalten neben einer eindeutigen *ID* zur Unterscheidung des Ereignisses im System auch andere Attribute, die sämtliche Informationen über das Ereignis im Hinblick auf seinen Kontext darstellen. Diese Attribute unterstützen unterschiedliche Datentypen (Boolesch, Zeichenfolge, Zahlen, etc.). Auf Werte einer Ereignisinstanz wird mithilfe der Punkt-Notation zugegriffen. Beispielsweise gibt $e_1.attr_1$ den Wert der Attribute $attr_1$ der Ereignisinstanz e_1 zurück. Die Zeit des Auftretens einer Ereignisinstanz *e* im System wird durch *e.t* (auch *e(t)*) zurückgegeben.

In der Literatur gibt es verschiedene Ereignisdarstellungen. Neben dem Punktereignistyp (engl. point event), bei welchem dem Ereignis für jeden diskreten Zeitpunkt *t* ein Ereigniswert zugewiesen wird, existiert noch eine andere Form der Ereignisdarstellung: das „Intervallereignis“ (engl. interval event). Bei dem Intervall-Ereignis wird dem Ereignis nur ein Wert über einen Zeitraum zugewiesen (Li 2010). In dieser Arbeit werden die Punktereignisse betrachtet.

Durch die stetig eingehenden Daten und die daraus abgeleiteten Ereignisse wird ein unendlicher Ereignisstrom gebildet. In diesem Kontext stellt eine Ereignisfolge einen definierten Abschnitt aus dem Ereignisstrom dar (Bruns et al. 2010).

Eine Ereignisfolge ist heterogen mit Ereignisinstanzen unterschiedlicher Ereignistypen gefüllt. Sie wird durch eine Reihenfolge von Ereignisinstanzen gebildet.

Eine Ereignisfolge *w* kann folgendermaßen notiert werden: $w = a_1b_1a_2c_3b_4c_5c_6$, wobei a_i, b_i *und* c_i Ereignisinstanzen von der jeweiligen Ereignistypen *A, B* und *C* sind. Zur Vereinfachung wird der Index *i* die Zeit des Auftretens eines Ereignisses in der Ereignisfolge bezeichnen. Beispielweise ist $a_1(t) = 1$ und $c_3(t) = 3$. Hierbei wird davon ausgegangen, dass die Zeitstempel der Ereignisse zu einer natürlichen Gesamtordnung führen.

Ein Ereignisstrom wird durch die unendliche eingehende Ereignisfolge definiert.

Ein Ereignisstrom besteht aus unendlichen Reihenfolgen von Ereignissen, die mehrere Datenpunkte im System beschreiben. Diese eingehenden Ereignisse selbst sind nicht komplex, aber die CEP-Logik betrachtet sie in Kombination miteinander, um nützliche Informationen und Erkenntnisse zu erzeugen. Dies geschieht durch die Kombination bzw. Aggregation der einfachen Ereignisse mittels deklarativer, kontinuierlicher Abfragen, um Ereignismuster zu identifizieren (Flouris et al. 2016).

Ein **Ereignismuster** beschreibt anhand spezifizierter Regeln eine bestimmte Situation in einer Ereignisfolge. Die Regel definiert die Bedingung, durch deren Erfüllung in einer Ereignisfolge ein Muster erkannt werden kann.

Beispielsweise ist das Ereignismuster mit der Bedingung „zwei Ereignisse von Typ *C* folgen unmittelbar aufeinander“ in der Ereignisfolge $w = a_1b_1a_2c_3b_4c_5c_6$ erfüllt, da die Ereignisinstanz c_6 auf Ereignisinstanz c_5 folgt.

Für eine Ereignisfolge *w* und einen Ereignistyp *E* bezeichnet *E[w]* die Menge aller Ereignisinstanzen von Typ *E* in *w*. $E[w] = \{e_1, e_2, \dots, e_i\}\ mit\ \forall\ i\ e_i \in w\ ,\ typeOf(e_i) = E$

Zum Beispiel gibt $A[w] = \{a_1, a_2\}$ die Menge aller Ereignisinstanzen des Typs *A* in *w* zurück.

4.5.2. Semantik von Ereignisregeln

Um komplexe Muster in Ereignisfolgen zu detektieren, muss das CEP-System Regeln auf die Ereignisfolgen anwenden. Hierzu werden Operatoren einer **Ereignisalgebra** verwendet, um die Regeln zu formulieren. Basierend auf diesen Regeln, können die primitiven (einfachen) Ereignisse zusammengefügt werden, um komplexere Ereignisse zu erzeugen. Diese komplexen Ereignisse beschreiben komplexe Zusammenhänge der primitiven Ereignisse und entsprechend komplexen Situationen. Der CEP-Datenanalyseansatz verwendet kontinuierliche Abfragen, um einen Ereignisstrom zu analysieren. Die kontinuierliche Abfrage ähnelt der herkömmlichen Datenbankabfrage. Anders als bei der herkömmlichen DBMS allerdings müssen die Abfragen nicht immer erneut vorgelegt werden, um diese weiterhin auszuführen – sie laufen kontinuierlich im CEP-System (Bruns et al. 2010).

Ein **komplexes Ereignis** wird mittels Regeln aus primitiven Ereignissen eines eingehenden Ereignisstroms bzw. mehrerer Ereignisströme durch Korrelationen und Aggregationen unter Verwendung von zeitlichen, räumlichen oder logischen Beziehungen von einem CEP-System erzeugt.

Aus dieser Definition lässt sich ein wichtiger Unterschied zwischen einfachen und komplexen Ereignissen feststellen. **Einfache Ereignisse können nicht durch Regeln abgeleitet werden**.

Die Regeln beschreiben die Beziehungen zwischen Ereignissen in einer Ereignisfolge. Diese Beziehungen können über kausale, zeitliche oder räumliche Zusammenhänge spezifiziert werden. Der Musterausdruck (engl. pattern expression) lässt sich, wie schon vorher erwähnt, durch Operatoren der Ereignisalgebra formulieren. Die Semantik der unterstützten Menge von zusammengesetzten Ereignisoperatoren, beispielsweise **SEQ**, **AND**, **OR, NEGATION, First und Last,** wird nachstehend erläutert (Flouris et al. 2016).

Die logischen Operatoren **AND** und **OR** spezifizieren **unabhängig von der Reihenfolge** in einer Ereignisfolge die Typen von Ereignisinstanzen, die auftreten müssen. Der Operator **AND** setzt voraus, dass alle Ereignisinstanzen der eingegebenen Ereignistypen vorliegen müssen, während der Operator **OR** nur

das Vorliegen einer Ereignisinstanz der eingegebenen Ereignistypen erfordert. Der Sequenzoperator **SEQ** spezifiziert eine bestimmte Reihenfolge, in der die Ereignisinstanzen von Interesse auftreten sollten.

$$\boldsymbol{AND}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = < a_i\, b_j > \cup < b_j\, a_i > mit\ \forall\, a_i\ \in A[w]\ und\ \forall\, b_j\ \in B[w]$$

$$und\ < a_i\, b_j > \in A[w] \text{ x } B[w]\ und\ < b_j\, a_i > \in B[w] \text{ x } A[w]$$

$$\boldsymbol{OR}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = A[w] \cup B[w]$$

$$\boldsymbol{SEQ}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = < a_i\, b_j > mit\ < a_i\, b_j > \in A[w] \text{ x } B[w]\ und\ a_i(t) < b_j(t)$$

$$\boldsymbol{NEG}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = \{e_i\}\ wobei\ e_i\ \notin A[w]\ und\ e_i\ \notin B[w]$$

$$\boldsymbol{FIRST}(\boldsymbol{A})[\boldsymbol{w}] = \{a\}\ wobei\ \forall\, a_i\ \in A[w]\ a(t) \leq a_i(t)$$

$$\boldsymbol{LAST}(\boldsymbol{A})[\boldsymbol{w}] = \{a\}\ wobei\ \forall\, a_i\ \in A[w]\ a(t) \geq a_i(t)$$

Die folgenden Beispiele veranschaulichen die Berechnung von logischen Operatoren AND(A,B), OR(A,B)und SEQ(A,B) unter Berücksichtigung der Ereignisfolge $w = a_1b_1a_2c_3b_4c_5c_6$.

Für diese Ereignisfolge *w* sind $A[w] = \{a_1, a_2\}$ und $B[w] = \{b_1, b_4\}$

$$\boldsymbol{AND}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = \{< a_1b_1 >, < a_1b_4 >, < a_2b_1 >, < a_2b_4 >, < b_1a_1 >, < b_1a_2 >, < b_4a_1 >, < b_4a_2 >\}$$

$$\boldsymbol{OR}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = \{a_1, a_2\} \cup \{b_1, b_4\} = \{< a_1 >, < a_2 >, < b_1 >, < b_4 >\}$$

$$\boldsymbol{SEQ}(\boldsymbol{A},\boldsymbol{B})[\boldsymbol{w}] = \{< a_1b_1 >, < a_1b_4 >, < a_2b_4 >\}$$

Durch die Kombination dieser logischen Operatoren können komplexe Muster zwischen Ereignistypen ausgedrückt werden. Als Beispiel sucht ***FIRST(SEQ(A,B))[w]*** nach der ersten Ereignisfolge in *w*, in der nach einem Ereignis vom Typ *A* ein Ereignis vom Typ *B* auftritt. Die Ereignisfolge $< a_1b_1 >$ und $< a_1b_4 >$ erfüllen das Muster, $< a_2b_4 >$ hingegen nicht.

Das Ereignismuster ***SEQ(A,OR(B,C))[w]*** erfordert eine Ereignisfolge, in der nach einem Ereignis von Typ *A* entweder ein Ereignis von Typ *B* oder Typ *C* auftritt.

Neben den oben aufgeführten Operationen können andere Operatoren semantisch definiert werden, für weiterführende Literatur siehe (Li 2010), (Ksystra et al. 2012).

4.5.3. Ereignismuster – Event Patterns

Die komplexe Ereignisverarbeitung hat drei Kernphasen: (i) einen Filterschritt, in dem versucht wird, jene Listen der relevanten primitiven Ereignisse zu definieren, die als Prädiktoren der Ereignisabfrage zugeordnet werden können, (ii)

einen Übereinstimmungsschritt (engl. matching step), in dem versucht wird, die identifizierten primitiven Ereignisse aus dem vorherigen Schritt dahingehend zu analysieren, ob sie die spezifischen Bedingungen definierter Regelmuster erfüllen können, (iii) einen Ableitungsschritt, in dem komplexe Ereignisse erkannt und abgeleitet werden.

Allgemein haben die Regeln folgende Syntax:

Regel:

```
PATTERN     <Musterausdruck>
WHERE       <Bedingungen>
WITHIN      <Zeitintervall>
GROUP BY    <Ereignisattribute>
INSERT      <Ereignisstrom>
```

- **PATTERN**: Die PATTERN-Klausel gibt zeitliche und logische Beziehungen zwischen Ereignissen an. Der Musterausdruck hat folgende Form: PATTERN $\bigoplus_{i=1}^{n}$Operatoren (Liste von Ereignistypen), wobei die wichtigsten Operatoren in dem Musterausdruck {SEQ, AND, OR, NEGATION, LAST, FIRST} sind, von denen jeder bestimmte Ereignistyp als Eingabe erhalten werden kann, während das Symbol $\bigoplus$ jede mögliche Kombination dieser Operatoren ausdrückt.
- **WHERE:** Die meisten Anwendungen erfordern eine Echtzeitfilterung, bei der die Benutzer an komplexen Ereignismustern interessiert sind. Die Bedingungsklausel legt eine oder mehrere Gleichheitsbedingungen zwischen Ereignisattributen der Eingangsereignisse fest, die durch logische Operatoren verbunden sind.
- **WITHIN:** Die WITHIN-Klausel legt ein Zeitfenster (Zeitintervall) fest, in dem nur die innerhalb dieses Zeitfensters eintreffenden Eingangsereignissen in der Regelausführung betrachtet werden.
- **GROUP BY:** Wie bei SQL-Abfragen wird durch diese Klausel die Ergebnismenge der Abfrage auf Basis von Ereignisattributen gruppiert. Häufig wird diese Klausel in Kombination mit den Aggregatfunktionen verwendet.
- **INSERT:** Die Ergebnisse der Ereignisregel werden in Form eines Ereignisstroms generiert. In dieser Klausel wird dann das Ausgangsereignis definiert.

Theoretisch reichen zuerst nur die Teile PATTERN und INSERT, um Ereignisabfragen zu erzeugen. Die anderen Teile sind optional. Natürlich gibt es noch andere Teile, wie **HAVING** oder **ORDER BY**, die auch in der Definition von Regeln verwendet werden, um noch komplexere Muster zu formulieren (Zhou et al. 2012).

Genau wie bei traditionellen SQL-Abfragen können Aggregationsfunktionen wie Sum(), avg() oder min() verwendet werden, um zeitliche Zusammenhänge

zwischen Ereignissen basierend auf den Aggregationsfunktionen auszudrücken. Darüber hinaus können auch Aliasnamen mit den Schlüsselwort ***AS*** innerhalb der Regeln verwendet werden (Luckham 2002).

Siddhi[1] oder ESPER[2] sind unter anderem einige Ereignisverarbeitungssprachen, die eine SQL-ähnliche Syntax anbieten, um die Ereignisregeln auszudrücken.

4.5.4. Sliding Windows innerhalb einer WITHIN-Klausel

Eine der wichtigsten Aspekte im Complex Event Processing ist die Verwendung von zeitlichen Korrelationen, um anhand der gespeicherten Regeln die primitiven Ereignisse daraufhin zu prüfen, ob ein komplexes Ereignis vorliegt. Aufgrund der kontinuierlichen Entstehung von neuen Ereignissen, die stetig mittels der Ereignisregeln abgeglichen werden, ist es wichtig, einen Puffer zu definieren, in dem die Anzahl der zu prüfenden Ereignisse auf Ausschnitte aus dem Ereignisstrom beschränkt ist. Dies erfolgt durch sogenannte Sliding Windows (Bruns et al. 2010). Hierbei wird das Sliding Window durch Längenfenster (engl. length windows) oder Zeitfenster (engl. time windows) beschrieben. Bei einem Längenfenster wird die Länge (Anzahl) der zu berücksichtigenden Ereignisse des Ereignisstroms festgelegt und die Ereignisse werden so lange im Puffer gespeichert, bis die Länge erreicht wird. In diesem Fall werden nach dem First-In-First-Out-Prinzip (FIFO-Prinzip) die alten Ereignisse durch neue Ereignisse im Puffer ersetzt. Ein Zeitfenster legt einen Zeitraum fest und es werden nur die Ereignisse betrachtet, die innerhalb dieses Zeitraums aufgetreten sind.

Die folgenden Beispiele verdeutlichen, wie die Zeitfenster und Längenfenster mit dem Werkzeug **Siddhi Streaming SQL**[3] in der Formulierung von Ereignisregeln ausgedrückt werden. Für diese Beispiele werden zwei Ereignisströme definiert: ein Eingangsstrom und ein Ausgangsstrom.

Die Syntax der Definition von Ereignisströmen bzw. Ereignisregeln wird auf Basis von **SiddhiQL**, einer SQL-basierten CEP-Engine, formuliert. In **SiddhiQL** werden Sliding Windows durch entsprechende Schlüsselwörter ***#window.time(x)*** bzw. ***#window.length(y)*** in eckigen Klammern spezifiziert.

In folgenden Beispielen besteht der Eingangsstrom aus Werten von einem Temperatursensor und der Ausgangsstrom aus dem berechneten Minimalwert, Maximalwert und Mittelwert des Temperatursensors innerhalb eines Zeitfensters oder Längenfensters. Die Definition der Ströme enthält keine Identifikationsbezeichnung (ID) und keinen Zeitstempel, da jede Ereignisinstanz des Ereignis-

[1] Siddhi: https://wso2.com/analytics-and-stream-processing/

[2] Esper ist eine Sprache, ein Compiler und eine Laufzeit für komplexe Ereignisverarbeitung (CEP) und Streaming-Analysen, verfügbar sowohl für Java als auch für .NET: http://www.espertech.com/esper/

[3] Siddhi CEP ist eine leichte, einfach zu bedienende Open Source Complex Event Processing Engine (CEP) unter Apache Software License v2.0. Siddhi wird als Kern-Engine für WSO2 CEP verwendet: https://wso2.com/analytics-and-stream-processing/

stroms von einer abstrakten Klasse abgeleitet wird, welche die beiden Attribute implementiert.

Als Beispiel für ein Längenfenster ermittelt die folgende Ereignisabfrage (Regel 1) die maximale, minimale und durchschnittliche Temperatur der letzten 3 Ereignisse aus dem Eingangsstrom und fügt die Ergebnisse in den Ausgangsstrom ein.

Regel 1:
define stream Eingangsstrom (name string, temp float);
define stream Ausgangsstrom (name string, minimalwert float, maximalwert float, mittelwert float);
from Eingangsstrom #window.length(3)
select name, min(temp) as minimalwert, max(temp) as maximalwert, avg(temp) as mittelwert
insert into Ausgangsstrom;

Die Abbildung 27 stellt eine Ereignisfolge von Temperaturereignissen bei einem Längenfenster von 3 Ereignissen dar, die im Laufe der Zeit eintreffen. Die Zahl in Klammern ist der Temperaturwert, eine Ereigniseigenschaft, die in dem oben genannten Eingangsstrom definiert wurde. Diese Abbildung veranschaulicht die Ausführung der Abfrage:

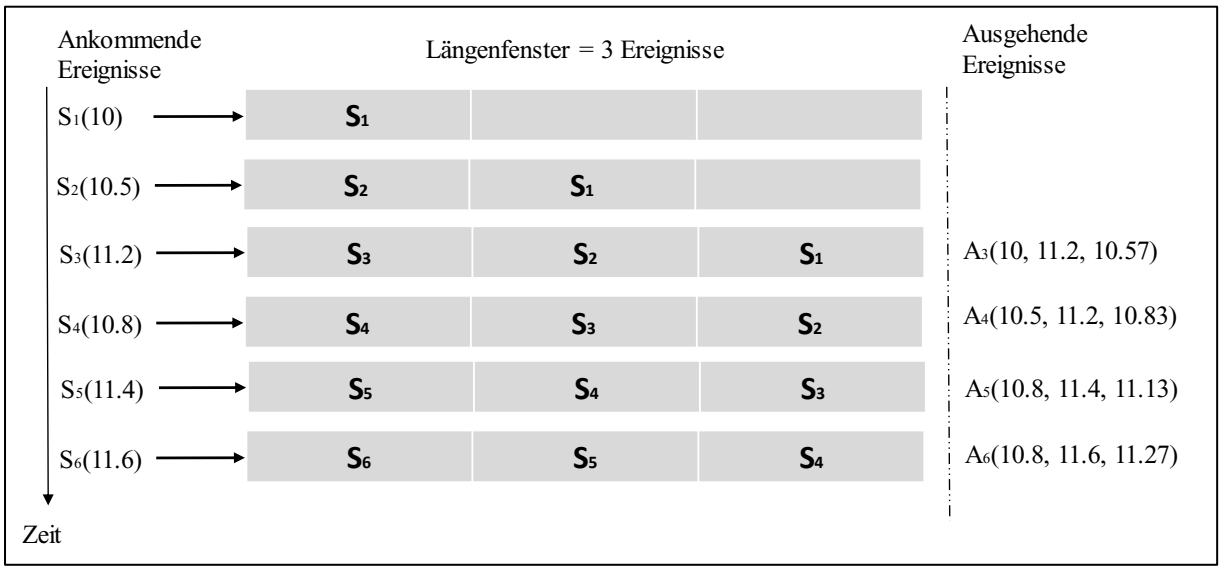

Abbildung 27: Beispiel eines Längenfensters mit einem Puffer der Länge 3 (Regel 1)

Nach Ereignisregel 2 werden, wie bei Regel 1, die maximale, minimale und durchschnittliche Temperatur der letzten 3 Ereignisse aus dem Eingangsstrom, deren Werte 11 Grad überschreiten, ermittelt und die Ergebnisse in den Ausgangsstrom eingefügt. Die Abbildung 28 veranschaulicht die Ausführung der Ereignisregel 2. Das Filtern in der Abfrage ermöglicht die Eingrenzung von Ereignissen aus dem Eingangsstrom, die im Datenfenster (Puffer) eintreten. Mit diesem Filter werden alle Temperatursensoren, die einen Wert von weniger als

11 Grad erfassen, nicht in dem Puffer eintreten und dementsprechend nicht in der Berechnung der Aggregationsfunktionen betrachtet.

Regel2:
from Eingangsstrom[temp >11]#window.length(3)
select name, min(temp) as minimalwert, max(temp) as maximalwert, avg(temp) as mittelwert
insert into Ausgangsstrom;

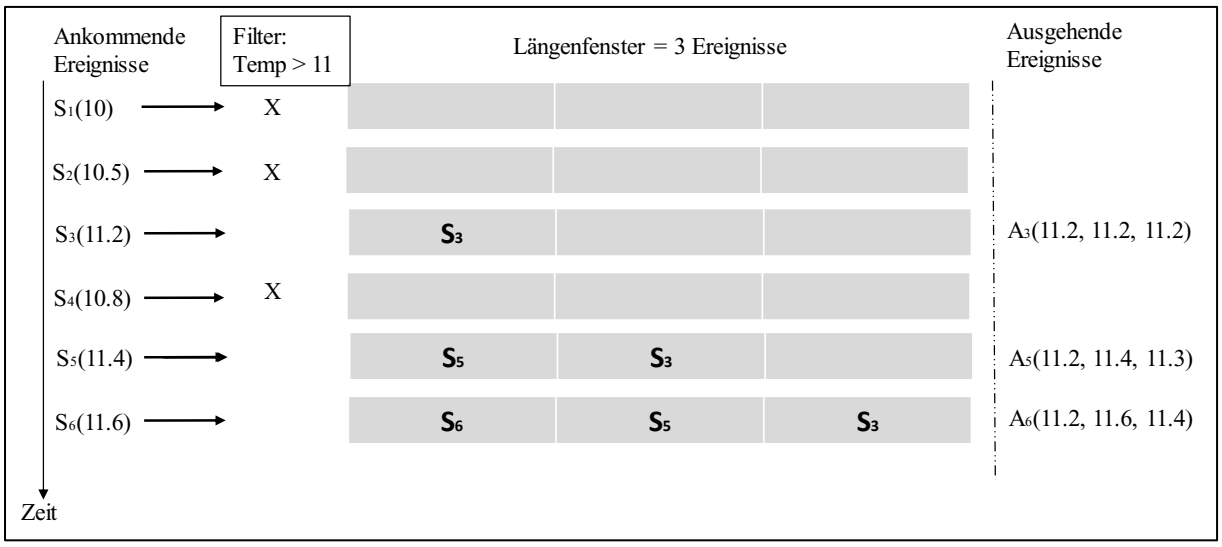

Abbildung 28: Beispiel eines Längenfensters mit einem Puffer der Länge 3 (Regel 2)

Die folgende Abfrage (Regel 3) berechnet die Durchschnittstemperatur pro Temperatursensor für die letzten 3 Minuten und warnt, wenn sie 33 Grad überschreitet. Im Gegensatz zu dem vorherigen Beispiel (Regel 2) findet das Ereignisfiltern nach der Verarbeitung der Select-Anweisung durch den Einsatz der Having-Klausel statt. Wie die Abbildung unten zeigt (Abbildung 29), treten ankommende Ereignisse in dem Puffer ein, jedoch werden nur Ereignisse übergeben, die die Having-Klausel erfüllen.

Regel3:
from Eingangsstrom #window.time(3 min)
select avg(temp) as mittelwert
group by name
Having mittelwert > 33
insert into Alertstrom;

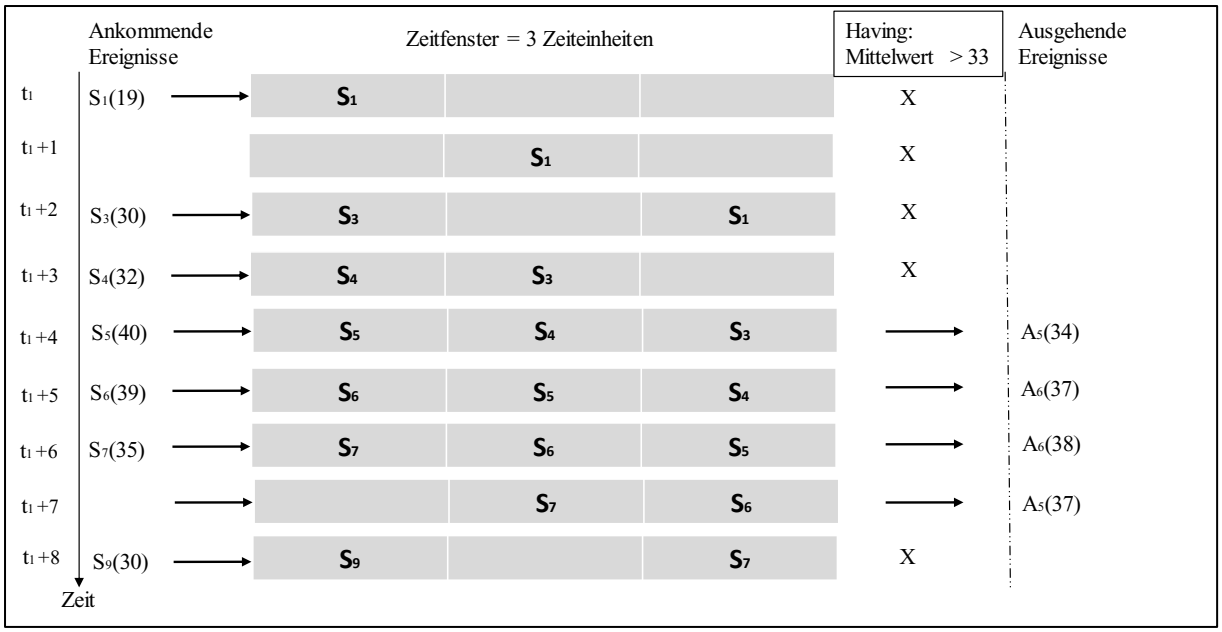

Abbildung 29: Beispiel eines Zeitfensters mit einer Länge von 3 Zeiteinheiten (Regel 3)

4.6. Anwendungsgebiete von CEP

Eckert et al. (2009) definieren eine Reihe von Anwendungsgebieten, in denen der Einsatz von CEP hohes Potenzial aufweist. Diese Anwendungsgebiete sind durch das Aufkommen großer Datenmengen, komplexer Verarbeitungsprozesse und der Notwendigkeit von echtzeitfähigen Reaktionen gekennzeichnet. Diese Anwendungen lassen sich in folgenden Gebieten erkennen (Eckert et al. 2009):

i) Das Business Activity Monitoring umfasst die Überwachung von Geschäftsprozessen, um rechtzeitig auf Problemsituationen reagieren zu können.
ii) Die Überwachung von physikalischen Sensornetzwerken bezweckt eine Kontrolle des Systemverhaltens, um z. B. Störungen frühzeitig erkennen zu können.
iii) Bei der Beobachtung von Marktdaten sollen frühzeitig Trends erkannt werden, um möglichst schnell auf diese reagieren zu können.

Hinsichtlich des Business Activity Monitorings stellen Greiner et al. (2006) beispielsweise ein Modell für die Erteilung eines Kredits (easy credit) bei der Noris Bank vor. Das Modell nimmt an, dass jeder Kreditantrag innerhalb seines gesamten Lebenszyklus mehrere Bearbeitungsschritte durchläuft. Basierend auf CEP können Ereignisse während der Prozessabläufe aggregiert und verarbeitet werden, sodass im Falle einer deutlichen Abweichung vom normalen Ablauf ein Alarm ausgelöst wird. Widder et al. (2007) beschreiben einen Ansatz für die Detektion von verdächtigen Ereignismustern bei Kreditkartentransaktionen und damit verbundenen Betrugsproblemen. Ein Event Processing Agent kommt zum Einsatz, um die Kredittransaktion mittels Einsatzes von Diskriminanzanalyse spezifischen Ereignisgruppen zuzuordnen und gegebenenfalls einen Alarm zu

senden. Auch Correia et al. (2015) setzen sich mit der Betrugsdetektion beim Einsatz von Kreditkarten auseinander und verwenden Komplex Event Processing für die Betrugserkennung.

Beispielhaft für physikalische Sensornetzwerke lassen sich die Arbeiten von Dunkel (2009 und 2010) anführen, die auf ein verbessertes Verkehrsmanagement abzielen. Das Ziel der Arbeit ist die Vermeidung von Verkehrstaus und die Verbesserung der Sicherheit der Verkehrsteilnehmer durch eine zeitaktuelle Analyse der bestehenden Verkehrssituation. Erweiternd wurde an einer Prognose gearbeitet, um die Voraussetzung für eine aktive Verkehrssteuerung zu schaffen. Dabei benutzt der Autor die Web Ontology Language, um auf Basis von semantischen Ereignismodellen die strukturellen Eigenschaften und Zusammenhänge der Ereignistypen darzustellen. In diesem Zusammenhang schlagen Nagappan et al. (2010) für die Reduktion der Reisezeiten in einem dynamischen Verkehrsumlegungssystem (engl. dynamic traffic assignment system) eine ereignisorientierte Architektur, basierend auf Erkennung der Verkehrsdichte, vor. Dabei werden Sensoren für die Überwachung der Verkehrsdichte auf Straßen angebracht. Die erhobenen Sensordaten werden auf Basis des CEP-Ansatzes aus Duplikation gefiltert, verarbeitet und tragen anschließend zur Evaluierung der Verkehrsdichte bei. Dann wird eine Aktion ergriffen und eine angemessene Verkehrsteuerungsmaßnahme initiiert. Auch Sokha et al. (2013) setzen sich mit dem Verkehrsmanagement auseinander. Sie haben ein intelligentes CEP-Verkehrsmanagementsystem entworfen, das, basierend auf der Verarbeitung verschiedener Sensordaten, die Überwachung des Straßenverkehrsflusses ermöglicht. In diesem Zusammenhang werden Sensoren direkt auf den Straßen angebracht, um die Verkehrsdichte zu ermitteln. Mouttham et al. (2009) präsentieren eine Datenintegrationsarchitektur innerhalb eines elektronischen Gesundheitspflege-Netzwerks auf Grundlage der Erweiterung der traditionellen SOA mit Unterstützung für komplexe Ereignisverarbeitung. Ein detailliertes Health-Monitoring-Szenario für die Versorgung von Herzpatienten dient der Veranschaulichung der Systemanforderungen und der Validierung der vorgeschlagenen Architektur. Jeder Patient, der unter Verdacht steht, einen Herzinfarkt erleiden zu können, wird mit Sensoren ausgestattet, deren physikalischen Daten von einem Smartphone kontinuierlich an einen Überwachungsserver gesendet werden. Ihr Ansatz hat eine höhere Versorgungsqualität, geringere Kosten für Gesundheitsdienstleister und eine höhere Lebensqualität für die Patienten erbracht. Vaidehi et al. (2012) präsentieren den Einsatz von CEP zur Gesundheitsüberwachung von geriatrischen Patienten. Die Patienten wurden mit tragbaren (engl. wearable) Körpersensoren ausgestattet, deren physikalische Daten kontinuierlich via Internet auf einen Überwachungsserver gesendet wurden. Im Ergebnis konnte eine höhere Qualität der Versorgung, geringere Kosten für die Leistungserbringer und eine höhere Lebensqualität für die Patienten erzielt werden.

Büllow et al. (2013) beschäftigen sich mit der Überwachung von Geschäftsprozessen. In diesem Zusammenhang beschreiben die Autoren eine Architektur für

den Einsatz von CEP als Unterstützungswerkzeug der Prozessmanagementmodelle. Das Ziel der Arbeit war die Überwachung von Transportprozessen, um frühzeitig unerwünschte Ereignisse zu vermeiden. Cabanillas et al. (2014) setzen sich mit komplexen logistischen Lieferketten auseinander. Sie stellen die Grundlage für die konzeptionelle Erweiterung von Prozessmanagementsystemen mit CEP-Funktionalitäten zur Unterstützung logistischer Prozesse vor. Die Autoren identifizieren drei Hauptherausforderungen für den Einsatz des CEP-Konzepts bzw. die Verarbeitung von Ereignissen. Diese Herausforderungen sind die Diskretisierung von Überwachungszuständen, die als primitive Ereignisse auftreten, die Aggregation von Überwachungsaktivitäten auf Basis von feingranularen Ereignissen und letztendlich die Korrelation von verschiedenen Ereignissen. Im Rahmen der Beobachtung von Marktdaten haben sich die Vorteile vornehmlich in der hohen Verarbeitungsgeschwindigkeit von CEP-Ansätzen gezeigt. Mukherjee et al. (2010) präsentieren einen Ansatz für die Marktbeobachtung von Aktienkursen, der in kürzester Zeit verschiedene Ereignisse bezüglich Kursänderungen detektieren kann. Dabei haben sie herkömmliche Datenbankmanagementsysteme mit einem entsprechenden CEP-Ansatz verglichen und deutliche Leistungssteigerungen aufgezeigt.

Widder (2011) präsentiert ein Hybrid-Modell zur automatischen Entdeckung von Betrugstransaktionen innerhalb der kompletten Transaktionsmenge bei Banken auf Basis von maschinellen Lernverfahren und Complex-Event-Processing-Technologie in Echtzeit. Die CEP-Technologie dient dabei als Daten- bzw. Eventversorgungskomponente für die Betrugserkennungskomponente. Die Betrugsanalyse wird dementsprechend durch die Betrugserkennungskomponente mithilfe von drei maschinellen Lernverfahren (Entscheidungsbaum, Diskriminanzanalyse und neuronales Netzwerk) erfolgen. Im Gegensatz zur Arbeit von Widder verfolgt diese Arbeit das Ziel der Integration von maschinellen Lernverfahren in den CEP-Ansatz mittels einer einzigen Komponente. Hierbei werden die maschinellen Lernverfahren in der Formulierung von Ereignisregeln ausgedrückt und in die CEP-Technologie integriert.

Zusammenfassend kann festgehalten werden, dass der CEP-Ansatz als neuartige Technologie schon in vielen Anwendungsbereichen gute Ergebnisse erzielt hat, jedoch bisher vornehmlich in betriebswirtschaftlichen Bereichen zum Einsatz gekommen ist. Bislang sind noch keine Arbeiten für den Bereich des Instandhaltungsmanagements für komplexe Systeme erschienen, die durchgehend den CEP-Ansatz verfolgen. Obwohl die Gesundheitsüberwachung von Herzpatienten in Mouttham et al. (2009) als eine einfache Form des Instandhaltungsmanagements betrachtet werden kann, erfüllt diese aber nicht alle Anforderungen an komplexe Systeme, wie zum Beispiel die Interdependenzen zwischen Teilsystemen und die Menge der zu verarbeitenden heterogenen Datenmengen. Hier besitzt die CEP-Technologie die Fähigkeit zur kontinuierlichen Verarbeitung großer heterogener Datenmengen und der Ableitung komplexer Verarbeitungsprozesse in Echtzeit (10.000 bis 500.000 Events/Sekunde) (Widder 2011). Ein gro-

ßer Nachteil bei den aktuellen Ansätzen ist zudem die Beschränkung der Ereignisabfragen auf vordefinierte Ereignisregeln, die zumeist manuell erstellt werden (Wang et al. 2011), (Correia et al. 2015). Dies macht die Ansätze unflexibel und erfordert einen hohen Konfigurationsaufwand. Insbesondere bei komplexeren Systemen liegen nicht immer alle Informationen a priori vor, weswegen sich die Ereignisregeln, die beispielsweise das Auftreten von Störungen veranschaulichen, nur schwer beschreiben lassen.

5. Entwicklung des Konzepts eines echtzeitfähigen, zustandsorientierten Zustandsüberwachungssystems

In diesem Kapitel wird das Konzept zur Lösung des im Rahmen dieser Arbeit behandelten Problems vorgestellt und beschrieben. Der erste Abschnitt soll die Voraussetzungen schaffen, den CEP-Ansatz auf die zustandsorientierte Instandhaltung übertragen zu können. Eine ereignisgesteuerte Referenzarchitektur wird in Anlehnung an das Zustandsüberwachungssystem von OSA-CBM (siehe Abbildung 6) entwickelt und es wird sich mit der Erstellung entsprechender Ereignismodelle auseinandergesetzt. Der nächste Abschnitt behandelt die Entwicklung der Methoden zur Merkmalsextration aus Sensordaten. Darauf aufbauend werden die Ereignisregeln ermittelt, die auf komplexe Ereignisse hindeuten.

5.1. Konzipierung und Entwurf

In diesem Kapitel werden die Grundlagen für die Entwicklung einer Referenzarchitektur untersucht. Das Ziel ist die Modellierung von ereignisorientierten Instandhaltungsprozessen im Hinblick auf die Realisierung eines Frameworks für ein ereignisorientiertes Instandhaltungssystem, das die Grundstrukturen der ereignisgesteuerten Architektur-Komponenten vorstellt. Des Weiteren werden die Voraussetzungen geschaffen, um den CEP-Ansatz auf die zustandsorientierte Instandhaltung zu übertragen. Eine ereignisgesteuerte Referenzarchitektur wird in Anlehnung an die Open-Source-Architektur des zustandsorientierten Systems (OSA-CBM) (siehe Abbildung 6) entwickelt. Die Referenzarchitektur wird die Grundlage für die Erstellung entsprechender Ereignismodelle in Form eines Schichtenmodells bilden. Für die Charakterisierung des Modells wird die Typisierung der Referenzarchitektur aus der Literatur verwendet (Cordes 2018), (Schütte 1998). Hierfür wird ein Referenzanwendungssystemmodell adressiert. Das Modell stellt die Merkmale einer Referenzarchitektur zur Einführung einer Software oder zum Business Process Reeingineering (Cordes 2018, Schütte 1998) dar. Darüber hinaus wird ein generisches Referenzmodell adressiert. Das bedeutet, dass das entwickelte Modell mithilfe von Anpassungen und Detaillierung auf andere Anwendungsfälle übertragbar sein wird.

5.1.1. Anforderungen an ein CEP-Referenzmodell zur Integration des CEP-Ansatzes in das zustandsorientierte Instandhaltungssystem

Zustandsüberwachungssysteme haben das Ziel, ein technisches System zu überwachen und durch Auswertung sowie Analyse der Sensordaten auf den Zustand des gesamten Systems bzw. einzelner Komponente zu schließen. Es existieren auf dem Markt schon verschiedene CM-Systeme, die für die zustandsorientierte Instandhaltung von technischen Systemen eingesetzt worden sind. In diesen CM-Systemen werden herkömmliche Methoden wie Schwellenwerte oder Methoden der Statistik verwendet, die häufig nur für die Analyse einzelner

Komponente eingesetzt werden können. Durch Integration Künstlicher Intelligenz und maschineller Lernverfahren konnten diese CM-Systeme komplexe Situationen in den Systemen entdecken. Dabei nimmt jedoch die Reaktionszeit jener CM-Systeme stark zu. Diese CM-Systeme sind so konzipiert, dass sie kontinuierlich Daten aus verschiedenen Quellen ein- und persistent in einer Datenbank ansammeln, zur Verfügung stellen und anschließend Informationen und Wissen aus diesen Daten als einen Block gewinnen. Dabei spielt die Zeit, die vom Eintritt einer Störung bis zur deren Identifikation vergeht, eine entscheidende Rolle. Deswegen besitzen CM-Systeme häufig nicht die notwendige Anforderung, um eine echtzeitfähige Verarbeitung der Daten abzubilden. Die Informationstechnologie (IT) bildet die Basis für die Umsetzung der CM-System-Aufgaben. Um dieser Herausforderung gerecht werden zu können, ist eine neue Art von Softwarearchitektur notwendig, die in der Lage ist, die kontinuierlich großen Datenmengen aus verschiedenen Datenquellen in Echtzeit effizient zu verarbeiten. Zum Erreichen dieses Ziels sollen Anforderungen an das System hinterlegt werden. In diesem Kontext sollen die Anforderungen an das ereignisorientierte CEP-Zustandsüberwachungssystem (CEP-CM-System) sowohl die Anforderungen an ein herkömmliches CM-System als auch die Anforderungen an ein echtzeitfähiges System berücksichtigen. Tabelle 1 listet die wichtigsten Anforderungen auf.

Tabelle 1: Anforderungsliste von herkömmlichen CM-Systemen, echtzeitfähigen Systemen und CEP-CM-System (X: erforderlich, O: optional)

Anforderung	**Traditionelle CM-Systeme**	**Echtzeitfähige Systeme**	**CEP-CM-System**
Detektion kritischer Situationen	X	X	X
Modulare / verteilte Verarbeitung	O	X	X
Implementierung und Integration von ML Algorithmen	X	O	X
Anpassungsfähigkeit	O	O	X
Übertragbarkeit	O	O	X
Erweiterbarkeit	X	X	X
Echtzeitfähigkeit	O	X	X
Skalierbarkeit	O	X	X
Zeitreihendaten	O	X	X

Anforderung	Traditionelle CM-Systeme	Echtzeitfähige Systeme	CEP-CM-System
Wissensrepräsentation durch Ereignisregeln	O	X	X

- **Detektion von kritischen Situationen**: Ein CEP-CM-System soll so ausgelegt sein, komplexe Ereignisse, die auf eine kritische Situation hindeuten, durch den Einsatz von Ereignisregeln zu detektieren und anschließend Maßnahmen herzuleiten.
- **Modulare / verteilte Verarbeitung:** Für die Verarbeitung von kontinuierlich ankommenden Daten in Echtzeit ist eine skalierbare Architektur erforderlich. Dabei werden die Aufgaben des Referenzmodells verschiedenen Hardwarekomponenten in einem verteilten System zugewiesen. In diesem Zusammenhang können verschiedene-Event-Processing-Agenten in einem Event-Processing-Netzwerk eine Aufgabe des Referenzmodells übernehmen.
- **Implementierung / Integration von ML-Algorithmen:** Existierende CEP-Systeme verwenden Ereignisregeln zur Erkennung von komplexen Ereignissen. Diese Ereignisregeln werden auf Basis einer Schwellenwertanalyse oder von Methoden der Statistik generiert. In komplexen technischen Systemen jedoch kommen diese Ereignisregeln an ihre Grenzen. Durch Integration Künstlicher Intelligenz und maschineller Lernverfahren konnten diese CEP-Systeme noch mehr komplexe Situationen in den Systemen entdecken (Meister 2014).
- **Anpassungsfähigkeit:** Die Architektur soll auf spezifischen Anwendungsszenarien problemlos angepasst werden können, ohne die Struktur der Architektur modifizieren zu müssen. Bei den meisten traditionellen CM-Systemen ist eine Anpassungsfähigkeit wegen der starren Systemarchitektur kaum gegeben.
- **Übertragbarkeit**: Durch die Anpassungsfähigkeit der Architektur ist die Übertragung des Modells auf andere Anwendungsszenarien möglich. Bei den meisten traditionellen CM-Systemen hingegen ist eine Übertragung wegen der starren Systemarchitektur nur sehr schwer zu realisieren.
- **Erweiterbarkeit**: Neue Sensoren oder Eingangsquellen können, ohne die Architektur modifizieren zu müssen, hinzugefügt, ausgetauscht oder entfernt werden.
- **Echtzeitfähigkeit**: Das CEP-CM-System soll in Echtzeit auf anerkannte Situationen reagieren. Dabei hängt die Echtzeitfähigkeit von dem erwarteten Durchsatz der Ereignisse, d. h. die Anzahl der zu verarbeitenden Ereignissen pro Zeiteinheiten, und der zulässigen Latenzzeit, d. h. die zeitliche Reaktionszeit bei der Ereignisverarbeitung, ab.

- **Skalierbarkeit**: Durch den Einsatz eines Event-Processing-Netzwerks, in dem jede EPA eine Aufgabe innerhalb des Referenzmodells übernimmt, und durch das Hinzufügen „vertikale Skalierung“ mehrerer EPAs erhöht sich der Durchsatz der Datenverarbeitung und die Skalierbarkeit in der Datenverarbeitung wird gewährleistet. Da die CEP-basierten Lösungen regelbasiert aufgebaut sind, sind sie methodisch auch gut skalierbar.
- **Zeitreihendaten**: Die Eingangsdaten bilden einen kontinuierlichen Ereignisstrom, innerhalb dessen die Ereignisse mit einem Zeitstempel versehen werden. In traditionellen CM-Systemen können neben Zeitreihendaten auch diskrete Daten (nicht unbedingt Zeitreihendaten) verarbeitet werden. In CEP-Systemen bilden die Daten aufgrund des kontinuierlichen Dateneintritts jedoch Zeitreihendaten – dabei spielt der Zeitpunkt des Ereigniseintritts in der Erkennung von Kausalitäten und komplexen Ereignissen eine sehr wichtige Rolle.
- **Wissensrepräsentation durch Ereignisregeln:** Bei traditionellen CM-Systemen erfolgt eine Wissensrepräsentation durch Regeln nur unter Einsatz von Expertensystemen. Werden andere Data-Mining-Methoden verwendet, ist das Wissen dabei in so einer Black-Box-Form gekapselt. Daraus ergibt sich, dass das Wissen in Form von Regeln optional darstellbar ist. Bei CEP-echtzeitfähigen Systemen ist es notwendig, das Wissen durch den Einsatz von Ereignisregeln auszudrücken, da die CEP-Engine auf deklarative Ereignisregeln aufgebaut ist.

5.1.2. Entwicklung einer Referenzarchitektur

Eine Untersuchung der Grundlage für das Erstellen einer Referenzarchitektur basierend auf einer ereignisorientierten Architektur stellt den ersten Schritt bei der Entwicklung einer Referenzarchitektur dar. Hierzu werden die Struktur des Systems, sowohl deren statischen als auch dynamischen Aspekte, die Konzeption der Datenschnittstelle zum Sensorsystem und die Datenschnittstelle zu den anderen Softwarekomponenten untersucht. In diesem Zusammenhang soll auch erarbeitet werden, wie sich die Software-Agenten (ein verteiltes Event-Processing-Netzwerk) in ein derartiges Konzept integrieren lassen. Weiterhin werden die Konzepte der serviceorientierten Architektur und der ereignisgesteuerten Architektur einer Analyse unterzogen, da bestehende Zustandsüberwachungssystemen zumeist serviceorientiert aufgebaut sind. Diese beiden Konzepte stellen in Kombination die Grundlagen für die zu konzipierende Referenzarchitektur. Darüber hinaus sollte die Entkopplung der Kommunikation zwischen verschiedenen Softwarekomponenten in so einer Referenzarchitektur ermöglicht werden. Ausgehend von der Spezifikation einer zustandsorientierten Instandhaltung nach OSA-CBM (siehe Abbildung 6), wird eine Referenzarchitektur entwickelt, die nach dem CEP-Ansatz aufgebaut ist.

5.1.2.1. Architektur eines zentralen CEP-CM-Systems

Bereits in Kapitel drei wurden die wesentlichen Module der OSA-CBM-Architektur vorgestellt. Abgeleitet aus dieser Architektur werden die wichtigsten Module identifiziert, die im Rahmen einer ereignisgesteuerten Architektur in Betracht gezogen werden. Eine geeignete Systemarchitektur für die CEP-zustandsorientierte Instandhaltung sollte spezifische Modulaufgaben enthalten. Diese sind: 1) Datenerfassung/Datenakquisition, die den Zugriff auf die verschiedenen Rohdaten ermöglicht und diese aufzeichnet. 2) Datenmanipulation: kann Signaltransformationen durchführen und spezielle Algorithmen zur Merkmalsextraktion auf die erfassten Daten anwenden. 3) Zustandserkennung: führt eine Zustandsüberwachung durch, wobei die Merkmale mit erwarteten Werten oder Betriebsgrenzen verglichen werden und Zustandsanzeigen und/oder Alarme zurückgegeben werden. 4) Zustandsbewertung: bestimmt, ob der Systemzustand eine Verschlechterung aufweist, wobei Trends in der Zustandshistorie, dem Betriebszustand und der Wartungshistorie berücksichtigt werden. 5) Herleitung von Instandhaltungsempfehlungen: liefert Empfehlungen zu Instandhaltungsmaßnahmen und Änderungen der Anlagenkonfiguration unter Berücksichtigung der Betriebshistorie, aktueller und zukünftiger Aufgabenprofile und Ressourcenbeschränkungen. In diesem Kontext soll die vorgestellte Systemarchitektur alle oben genannten Modulaufgaben in ein funktionales System integrieren, das die Implementierung einer intelligenten und zustandsorientierten Instandhaltung ermöglicht (Cachada 2018).

Die Referenzarchitektur hat weiterhin zur Aufgabe, das Erzeugen, Erkennen und Verarbeiten von Ereignissen in Echtzeit zu ermöglichen. Die Modellierung der Architektur des Instandhaltungssystems erfolgt in UML und beinhaltet neben entsprechenden Use-Cases auch die Spezifikation der Datenschnittstellen zum Sensorsystem und zu den anderen Softwarekomponenten.

Zum besseren Verständnis wird in Abbildung 30 die Grundstruktur der zu entwickelnden Referenzarchitektur veranschaulicht.

In der physikalischen Schicht des CEP-Zustandsüberwachungssystems bilden die Sensoren ein Sensornetzwerk, das zur Messung von physikalischen Werten verschiedener Systemkomponenten bzw. Systeme verwendet wird. In Kombination mit beispielsweise den Statusmeldungen und kontextbezogenes Geschäftsaktivitäten, die beispielsweise Status- und Wartungsdaten umfassen, stellt diese Schicht die sogenannte Ereignisquelle dar und liefert die Eingangswerte für die Ereignisverarbeitungsschicht über eine Datenschnittstelle (Event Adapter). Die Datenquellen können sich in dem Datenformat, in den Kommunikationsprotokollen und in weiteren Eigenschaften unterscheiden.

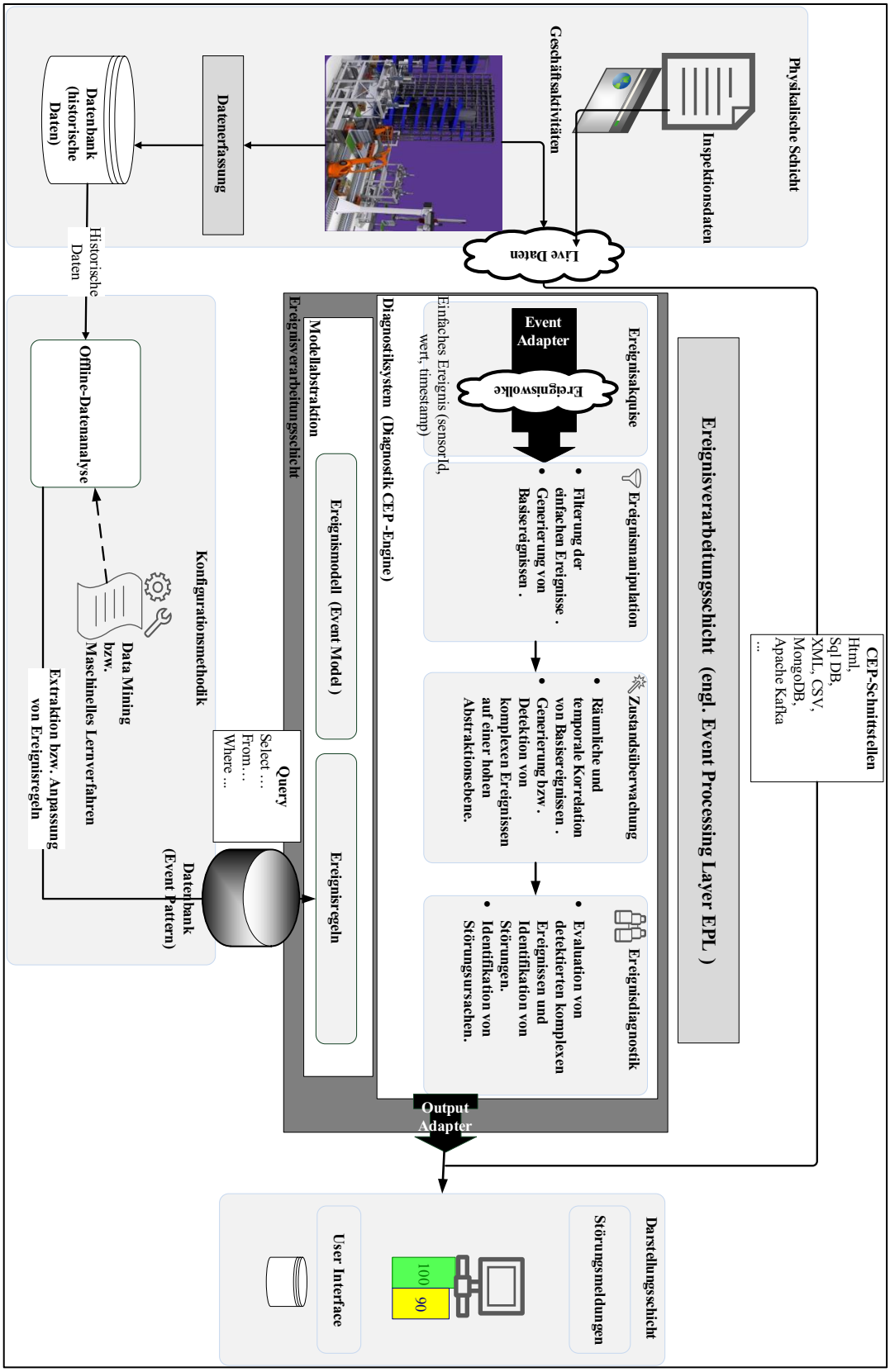

Abbildung 30: Event-Processing-Konzept eines zentralen, ereignisbasierten Zustandsüberwachungssystems

Zur Komplexitätsreduzierung in der Ereignisverarbeitungsschicht sollen Daten aus den heterogenen Datenquellen der Ereignisverarbeitungsschicht in einer ein-

heitlichen Repräsentationsform bereitgestellt werden. Um dies zu gewährleisten, werden Methoden der semantischen Datenintegration durch den Ereignisadapter dahingehend angepasst, dass der Ereignisverarbeitungsschicht logische Sichten auf die Sensorwerte bzw. Eingangswerte bereitgestellt werden. In der Ereignisverarbeitungsschicht erfolgt die Umsetzung des CEP-Ansatzes. Auf Basis von Ereignisregeln wird der Ereignisstrom auf das Vorhandensein von komplexen Ereignismustern hin analysiert. Die Ereignisregeln sind in einer lokalen Datenbank gespeichert und werden bei der Initialisierung des Systems ausgelesen. In Anlehnung an das Zustandsüberwachungssystem der OSA-CBM besteht die Ereignisverarbeitungsschicht aus vier Funktionsmodulen: Ereignisakquisition, Ereignismanipulation, Zustandsüberwachung und Ereignisdiagnostik. In der Ereignisakquisition werden die erfassten Sensordaten mittels eines Adapters auf das benötigte Format der CEP-Engine angepasst und in Ereignisse unterschiedlichen Typs transformiert. Diese Ereignisse enthalten normalerweise abstrakte Informationen (z. B. Sensor-ID, Wert, Zeitstempel), die im Allgemeinen auch fehlerhafte oder redundante Werte haben, die zu Inkonsistenzen in den Daten führen können. Diese Ereignisse bedürfen daher einer Filterung nach Fehlern und Redundanz. Im Modul Ereignismanipulation werden die Ereignisse mithilfe von Ereignisregeln abgeleitet, nach Signalverarbeitungsmethoden gefiltert und anschließend diskrete Ereignisse (verfeinerten Basisereignisse) generiert. Diese Basisereignisse verfügen im Allgemeinen nicht über ausreichende Informationen für die Ermittlung des Systemzustands. Deswegen erfolgen im Zustandsüberwachungsmodul eine Aggregation und Korrelation der Ereignisse sowohl in zeitlicher als auch in räumlicher Dimension, um somit komplexe Ereignisse auf einer höheren Abstraktionsstufe zu generieren. Die Evaluation und Auswertung der detektierten komplexen Ereignisse erfolgen im Diagnostikmodul wie auch die Identifikation der Störungsursachen. Die von der Ereignisverarbeitungsschicht ausgelösten Ereignisse werden in der Darstellungsschicht dann an die Instandhaltungsplanung übertragen, was auch eine Visualisierung über eine Benutzerschnittstelle beinhalten kann.

Folgende Tabelle (siehe Tabelle 2) fasst die wichtigsten Komponenten des Referenzarchitekturmodells zusammen:

Tabelle 2: Funktionsbeschreibung der Komponenten des ereignisbasierten Zustandsüberwachungssystems

Komponente	**Funktionsbeschreibung**
Ereignisquelle	- liefert die Eingangsdaten für die Ereignisverarbeitungsschicht.
Input-Adapter	- nimmt die Daten aus Ereignisquellen entgegen. - stellt die heterogenen Daten in einer einheitlichen Repräsentationsform bereit. - leitet die Ereignisse zur Ereignisverarbeitungsschicht weiter.

Komponente	Funktionsbeschreibung
Ereignismodell	- enthält die wesentlichen Definitionen, Spezifikationen von Ereignissen und die Beziehungen zwischen diesen Ereignissen. - Es werden meistens Meta-Modelle für die Definition von Ereignismodellen verwendet
Ereignisregeln	- werden bei der Verarbeitung von Ereignissen verwendet. - verarbeiten die Ereignisströme durch Filter, Windows, Aggregation, Joins sowie Sequenzoperationen und erzeugen neue Ereignisströme.
CEP-Engine	- ist als Kernkomponente für die Ereignisverarbeitung zuständig. - implementiert eine deklarative Ereignissprache (ähnlich SQL), um die Ereignisregeln auszudrücken – darüber hinaus ermöglicht eine CEP-Engine das Listen (Abhören) der Ereignisse und liefert alle notwendigen Softwarekomponenten, um echtzeitfähige Ereignisverarbeitung zu implementieren.
Output-Adapter	- nimmt die ausgegebenen komplexen Ereignisse aus der CEP-Engine entgegen. - stellt die komplexen Ereignisse in einer einheitlichen Repräsentationsform bereit. - leitet die Ereignisse zur Darstellungsschicht weiter.
Ereignisstrom	- stellt einen kontinuierlichen Strom von Ereignissen dar, der sich an ein definiertes Ereignismodell (Meta-Modell oder Schema) richtet.
Offline-Datenschicht	- Auf Basis von historischen Daten und bekannten Störungen werden maschinellen Lernverfahren entwickelt, um eine Zuordnung von historischen Daten zu Störungen zu erzeugen. - Die ML-Verfahren werden zur Generierung bzw. Aktualisierung der Ereignisregeln beitragen.

5.1.2.2. Eingangs- bzw. Ausgangsereignisse

Das ereignisbasierte Zustandsüberwachungssystem basiert auf einer CEP-Engine, die Ereignisse aus Datenströmen verarbeitet, komplexe Situationen erkennt und schließlich Aktionen auslöst. Die Systemfunktionalität wird mit dem Modul Datenerhebung/Datenakquisition eingeleitet, wobei die Live-Daten aus mehreren Quellen gesammelt und gleichzeitig in einer Datenbank gespeichert werden. Diese Datenbank wird neben der historischen Datenbank, dem Modul Offline-Datenanalyse, mit Daten bereitgestellt, wobei eine erweiterte Datenana-

lyse und ein maschinelles Lernverfahren zur Wissensgenerierung eingesetzt werden. Dabei werden die Eingangsdaten in verschiedenen Formaten vorliegen. Auch die Ausgangsdaten können in verschiedenen Formaten bereitgestellt werden.

Die Kommunikation zwischen Ereignisverarbeitungsschicht und Eingangsschicht (physikalische Schicht) bzw. Darstellungsschicht geschieht durch CEP-Schnittstellen basierend auf dem CEP-Adapter. Dabei packt der jeweilige Adapter die Ereignisse in eine Nachricht, basierend auf dem Ereignismodell, und sendet diese weiter. Der Empfänger liest die Nachricht mithilfe des jeweiligen CEP-Adapters, extrahiert die Ereignisse, was gegebenenfalls dazu führen kann, dass entsprechende weitere Verarbeitungsprozesse angestoßen werden. Die Nachrichten können über mehrere Transportprotokolle, wie HTTP, TCP, Kafka, und in verschiedenen Datenformaten, wie XML, JSON, Text, Binär, etc., zwischen den Modulen versendet bzw. empfangen werden.

5.1.2.3. Event-Adapter

Ein Event-Adapter ist eine Software-Komponente, die eine Mapping-Aufgabe übernimmt. Unter Mapper wird eine Unterkomponente verstanden, die für jede Quelle konfiguriert werden muss, um den eingehenden Datenfluss in das interne Ereignisformat des CEP-Ereignisses zu konvertieren. Bedingt insbesondere durch die Vielfalt der möglichen Ereignisquellen und folglich der Heterogenität der Daten, werden die Adapter als Schnittstellen einerseits zwischen den Ereignisquellen und der Ereignisverarbeitungsschicht, andererseits zwischen der Ereignisverarbeitungsschicht und der Darstellungsschicht, eingesetzt (Bruns et al. 2010).

Aufgrund der erheblichen Datenheterogenität kann der CEP-Adapter verschiedene Methoden zur Kapselung der eingehenden Daten in das interne Format implementieren. Exemplarisch können Methoden zum Ablesen der Daten aus einer SQL/No-SQL-Datenbank, aus CSV- und Text-Dateien, HTML oder einem Webservice implementiert werden. In diesem Kontext wird der Output-Adapter die resultierenden Ereignisse in das gewünschte externe Format, basierend auf den entwickelten Adapter-Methoden, transformieren.

5.1.2.4. Architekturen eines verteilten CEP-CM-Systems

Es ist möglich, alle oder einen Teil dieser Module verteilt und dezentral auszuführen. In diesem Fall wird ein Agentennetzwerk implementiert, wobei jeder Processing-Agent die resultierenden Ereignisse von dem vorherigen Modul entgegennimmt, die Modulaufgabe ausführt und wieder neue Ereignisse generiert und diese zum nachfolgenden Processing-Agent zur Weiterverarbeitung weiterleitet (vertikale Skalierbarkeit). Abbildung 31 veranschaulicht die wesentliche Struktur dieses verteilten Netzwerks mit deren CEP-Schnittstellen.

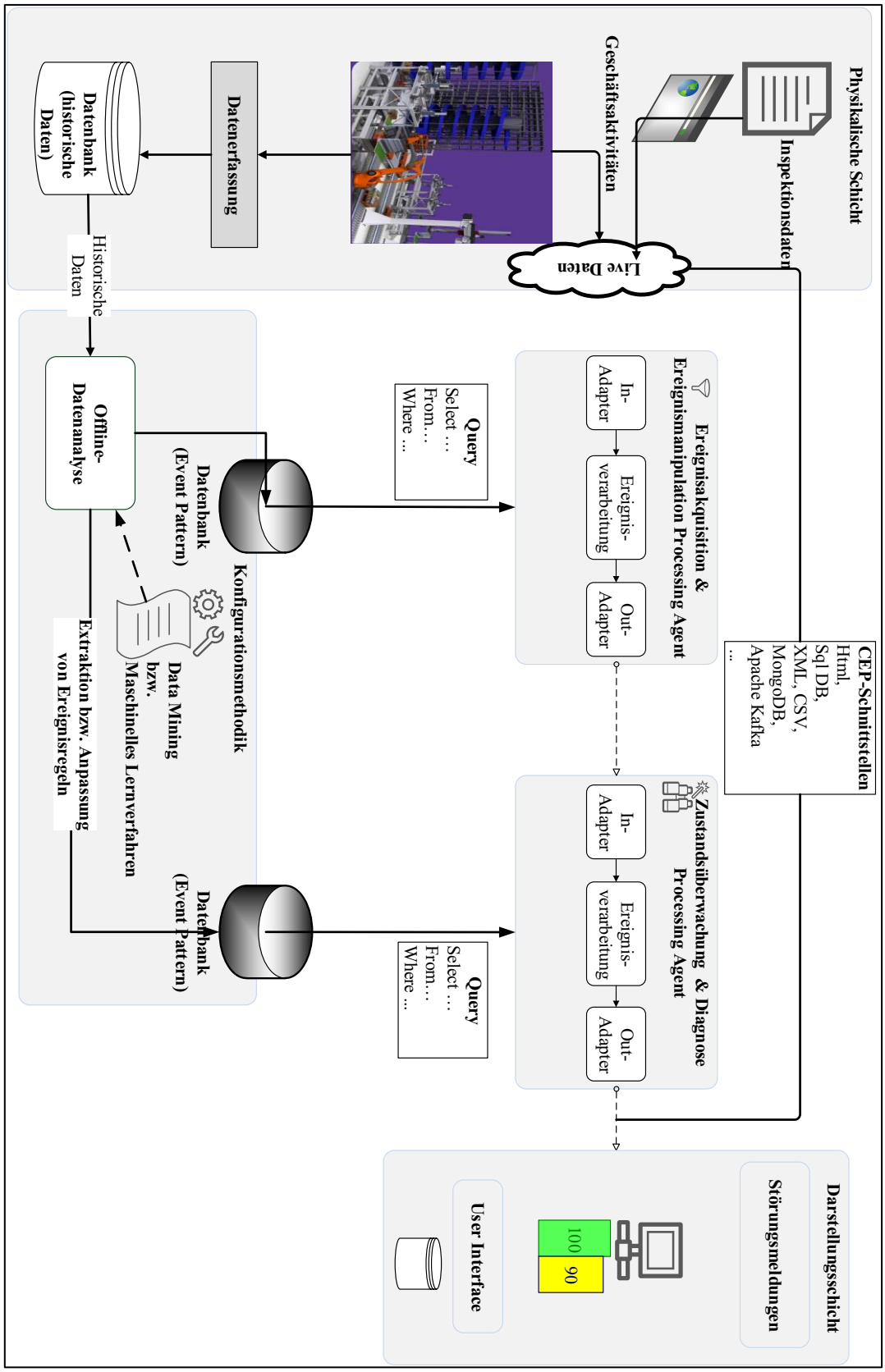

Abbildung 31: Event-Processing-Konzept eines verteilten ereignisbasierten Zustandsüberwachungssystems

Das Beispiel in der Abbildung 31 repräsentiert die Struktur eines verteilten CEP-CM-Systems bestehend aus zwei Ereignisverarbeitungsagenten. In diesem Kon-

text übernimmt ein Verarbeitungsagent die Aufgaben der Ereignisakquisition und der Ereignismanipulation, während der zweite Verarbeitungsagent die Aufgaben der Zustandsüberwachung und der Diagnose ausführt. Darüber hinaus ist es möglich, für dieselbe Aufgabe bzw. dieselben Module mehrere Agenten einzusetzen (horizontale Skalierbarkeit). Im Gegensatz zur zentralen Architektur ist es empfehlenswert, für die Agenten eines bestimmten Moduls eine Ereignisdatenbank zu spezifizieren, da sich die Ereignisregeln für die Ereignismanipulation von Ereignisregeln der Zustandsüberwachung unterscheiden.

Die ereignisgesteuerte Referenzarchitektur enthält die wesentlichen Konzepte für die Entwicklung und den Entwurf sowie detaillierte Vorgaben für die Umsetzung echtzeitfähiger CM-Systeme, die auf die spezifischen Anforderungen von ereignisgesteuerten Systemen zugeschnitten sind. Darüber hinaus stellt die Referenzarchitektur die Leitfäden für die weitere Übertragung des CEP-Konzepts auf weitere Anwendungsszenarien vor.

5.1.3. Entwicklung des Ereignismodells (Event Model)

Relevante Informationen mit Einfluss auf den Instandhaltungsprozess ergeben sich einerseits durch das Erkennen des Auftretens einzelner Basisereignisse, andererseits durch das Erkennen von Zusammenhängen und Korrelationen zwischen mehreren Ereignissen. Aus diesem Kontext werden mittels eines Ereignismodells die Ereignisse präzise und formal definiert. Hierzu ist das Ziel der Entwicklung eines Ereignismodells spezifiziert, das eine semantische Darstellung und eine Beschreibung der Ereignisse verschiedener Typen – mit deren Eigenschaften und Beziehungen auf den verschiedenen Abstraktionsebenen – wiedergibt. Zur formalen Beschreibung von Ereignismodellen existieren noch keine spezifischen Standards (Bruns et al. 2010). Die Entwicklung eines Ereignismodells erfolgt grundsätzlich auf Basis der Analyse der Instandhaltungsprozesse. Darüber hinaus hängt das Ereignismodell vom funktionalen Aufbau des zu untersuchenden technischen Systems ab.

Das Ereignismodell bildet die Grundlage für die spätere Generierung der Ereignisregeln. In diesem Ereignismodell werden alle Ereignisse, die im Rahmen der Ereignisverarbeitung verwendet werden können, zu einem Modell kombiniert. Dabei wird sich mit der qualitativen und semantischen Definition von Datenquellen für die Extraktion von Basisereignissen sowie die Komposition räumlicher und gegebenenfalls auch logischer Ereignisse auseinandergesetzt. Die abschließende quantitative Spezifikation von Schwellwerten und die Komposition temporaler Ereignisse werden später separat bearbeitet. Um einen systematischen Aufbau des Ereignismodells zu gewährleisten, werden bestehende Ansätze zur Funktionsmodellierung (z. B. VDI-Richtlinie 2888 für zustandsorientierte Instandhaltung) verwendet und mit Methoden zur Systemmodellierung (z. B. SysML) zusammengeführt. Das Ziel ist die vereinfachte Systemkonfiguration mittels integrierter grafischer Abbildung von technischen Komponenten, Funkti-

onen, Sensoren und Informationsflüssen. Im Ereignismodell werden alle Ereignisse definiert und hierarchisch strukturiert, ohne zunächst auf die Ereignisverarbeitung einzugehen.

Grundsätzlich müssen hierfür Systemstörungen auf Komponenten- bzw. Funktionsebene und deren Auswirkungen definiert werden, wobei methodisch auf die System-FMEA zurückgegriffen wird. Für die Modellierung der System-FMEA können modellbasierte Testfälle als Vorlage dienen, weil sie zum einen das schon erwähnte Sollverhalten beschreiben und zum anderen das Wissen über kritische Zustände enthalten. Im Ergebnis werden Störereignisse definiert, die idealerweise sensorisch erfasst werden können und über eine Störungsursache verfügen. Inwieweit andere Ereignisse in Beziehung zum Störungsereignis stehen und sich somit zur erweiterten frühzeitigen Detektion nutzen lassen, wird auch erarbeitet.

Ein Ereignis aus dem Ereignismodell beschreibt eine Änderung hinsichtlich physikalischer Sensorwerte. Diesbezüglich werden Methoden zur Generierung und Modellierung von relevanten Ereignissen für die Instandhaltung betrachtet. Im Folgenden wird das Vorgehen zur Erstellung von Ereignissen vorgestellt. Hierzu sind generell jene technischen Systeme, welche die Instandhaltung betreffen, hinsichtlich der verfügbaren Information – wie beispielsweise Sensorwerte, Grenzbereiche, Anforderungen und Statusmeldungen – zu analysieren, um die möglichen Ereignisse zu identifizieren. Die Analyse wird auf Basis der Datensätze des technischen Systems erfolgen. Auf Grundlage dieser Analyse werden die identifizierbaren Ereignisse klassifiziert und priorisiert. Für die Herleitung der Ereignisse aus Modellen werden Systemmodelle und Testmodelle herangezogen. Während die Systemmodelle strukturelle Restriktionen des Systems offenbaren, fokussieren sich Testmodelle gezielt auf das Sollverhalten. Beide Arten von Modellen beschreiben die Struktur und das Verhalten des Systems und werden zum Zwecke der Analyse für das technische System modelliert. So enthalten die Systemmodelle Informationen darüber, welche Komponenten es gibt und wie deren Interaktion abläuft. Testmodelle enthalten zusätzlich die Information darüber, welche Systemzustände normal, unerwartet oder kritisch sind. Sowohl Basisereignisse als auch temporale Ereignisse können aus solchen Modellen abgeleitet werden, die strukturelle und dynamische Eigenschaften formal repräsentieren können. Hierdurch werden Testmodelle als Informationsquelle für Ereignismodelle verwendet. Allgemein kann zwischen fünf verschiedene Hauptklassen von Ereignissen unterschieden werden:

i. Basisereignisse (Statusmeldungen oder Sensorwerte)
ii. Filterereignisse (auch mit der Anwendung von Schwellwertüberschreitungen)
iii. Räumliche oder logische Ereignisse (mit der Anwendung von logischen Operatoren wie AND, OR, NOT etc.)

iv. Temporale Aggregatereignisse (mit der Anwendung von Aggregatfunktionen wie AVERAGE, COUNT, MIN, MAX etc.)
v. Maschinelle Lernereignisse (mit der Anwendung von maschinellen Lernverfahren)

Die erste Klasse beschreibt jene Ereignisse, die von dem Adapter aus den Ereignisquellen für die Verarbeitung im CEP-CMS-System transformiert werden. Diese erste Ereignisklasse umfasst die sogenannten Basisereignisse.

Die Ereignisklasse „Filterereignisse“ umfasst die Ereignisse, die sich zum Beispiel durch die Überschreitung eines Schwellwerts identifizieren lassen.

Die Ereignisklasse „räumliche oder logische Ereignisse“ umfasst die Ereignisse, die sich über einen Raumbereich erstrecken und deren Detektion die Beteiligung und logische Komposition von mehreren Ereignissen erfordern.

Die Ereignisklasse „temporale Aggregatereignisse“ umfasst die Ereignisse, die einen zeitlichen Kontext erfordern. Solche temporalen Ereignisse lassen sich durch Aggregatfunktionen für einen Zeitraum (window) definieren.

Die letzte Ereignisklasse „maschinelle Lernereignisse“ steht für die Ereignisse, die aus der Verwendung von maschinellen Lernverfahren, wie von künstlichen neuronalen Netzen oder K-means, aus einem oder vielen Ereignissen resultieren.

Auf Basis dieser fünf Hauptklassen von Ereignissen können beliebig viele Arten von komplexen Ereignissen hergeleitet werden. In diesem Zusammenhang lässt sich jedes Ereignis in einem Ereignismodell für komplexe technische Systeme einer der Klassen zuordnen.

In der Abbildung 32 ist die Grundstruktur des zu entwickelnden Ereignismodells dargestellt. Für jeden physikalischen Sensor und für die Instandhaltung externer relevanter Prozessdaten wird ein Eingangsereignis zugeordnet und generiert. Beispielweise stellen die Sensorereignisse und Wartungsereignisse in der Abbildung 32 die Eingangsereignisse dar.

Diese Eingangsereignisse bilden die sogenannten Basisereignisse, die allein keine semantische Signifikanz für den Entscheidungsprozess besitzen. Aus den Basisereignissen können logische, aggregierte Filter oder maschinelle Ereignisse generiert werden. Die Generierung von logischen Ereignissen erfolgt durch die Zusammenführung von verschiedenen Ereignissen durch den Einsatz von logischen Operatoren (AND, OR, NOT etc.). Die temporalen aggregierten Ereignisse werden unter Einsatz von Aggregatfunktion (MIN, MAX, SUM, AVERAGE etc.) für ein bestimmtes Zeitfenster definiert. Die Filterereignisse werden durch Anwendung von Filterbedingungen generiert. Diese neu generierten Ereignisse können miteinander wieder aggregiert, gefiltert oder durch Einsatz von ML-Methoden zu maschinellen Ereignissen kombiniert werden. Durch diese multiple und mehrfache Aggregation, Filterung, Korrelation und Zusammenführung

der entstehenden Ereignisse erfolgt die Generierung von komplexen Ereignissen, die auf eine Störung oder eine Warnung hindeuten können.

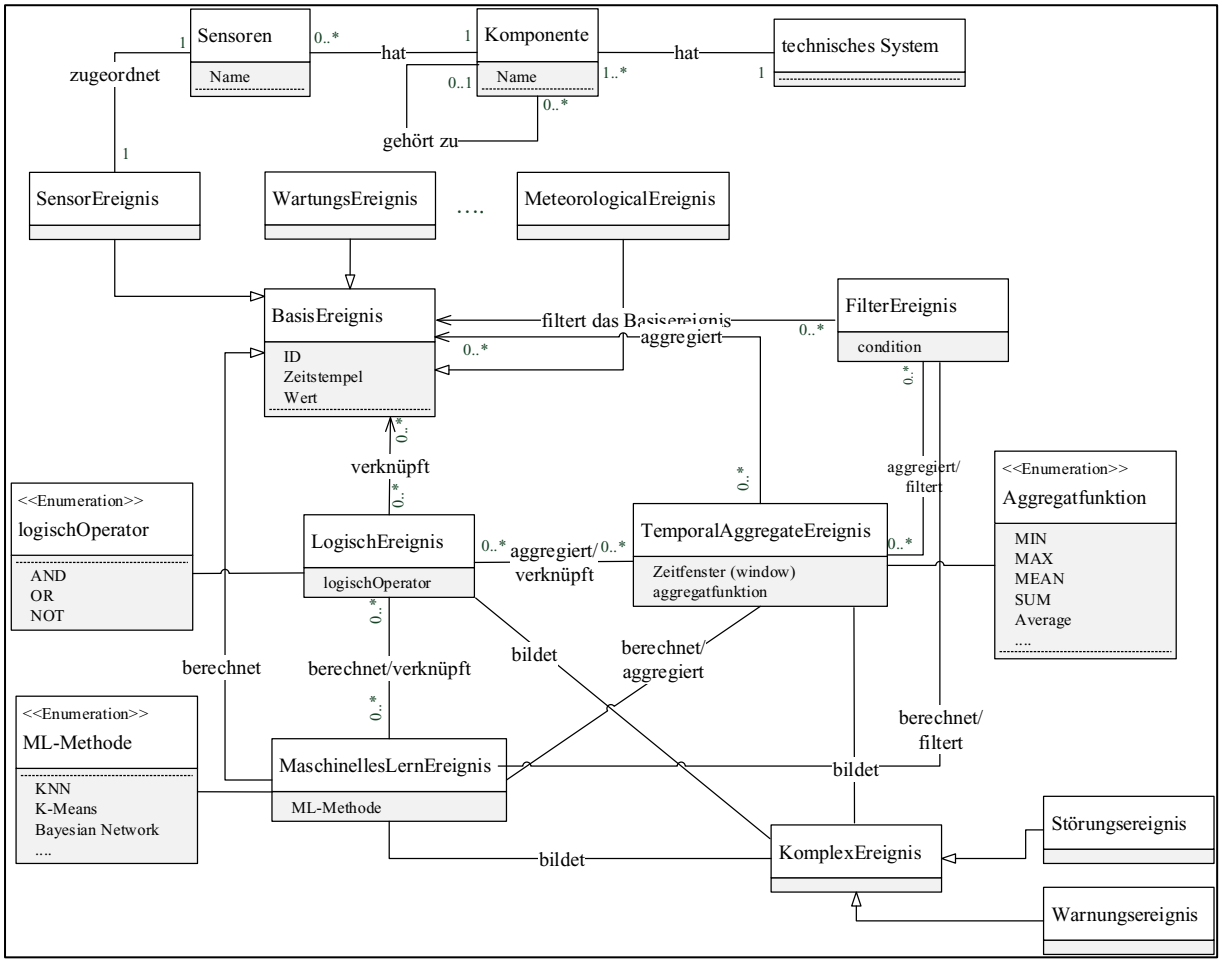

Abbildung 32: Struktur des Ereignismodells für das CEP-CM-System und dessen grundlegenden Abstraktionsebenen von Ereignissen

In Abbildung 33 ist ein einfaches Beispiel für ein Kühlsystem einer Maschine, bestehend aus einer Pumpe und einem Ventilator, dargestellt. Zur Verdeutlichung der Generierung des Ereignismodells wird hier dieses einfache Kühlsystem als Beispiel dienen.

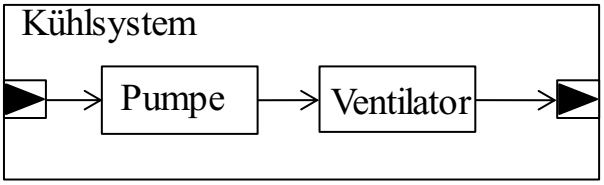

Abbildung 33: Beispiel eines einfachen Kühlsystems

Zur Überwachung des Kühlsystems werden neben drei Temperatursensoren auch ein Drucksensor und ein Sensor zum Messen der Wasserströmung verbaut (Ait Alla et al. 2016), (Lewandowski 2016). Ziel ist es, aus diesen Sensoren verschiedene Störungen an dem Kühlsystem zu detektieren. Diese Störungen sind entweder eine Leckage oder ein Pumpendefekt. Angenommen, es ergibt sich aus

dem Offline-Modul der Referenzarchitektur, dass die Störungen durch den Einsatz eines neuronalen Netzes, das als Input die durchschnittlichen Sensorwerte der letzten 10 Sekunden der verschiedenen Sensoren nimmt, in verschiedene Störungsklassen klassifiziert werden, dann sieht das Ereignismodell dieses Instandhaltungssystems wie in Abbildung 34 dargestellt aus.

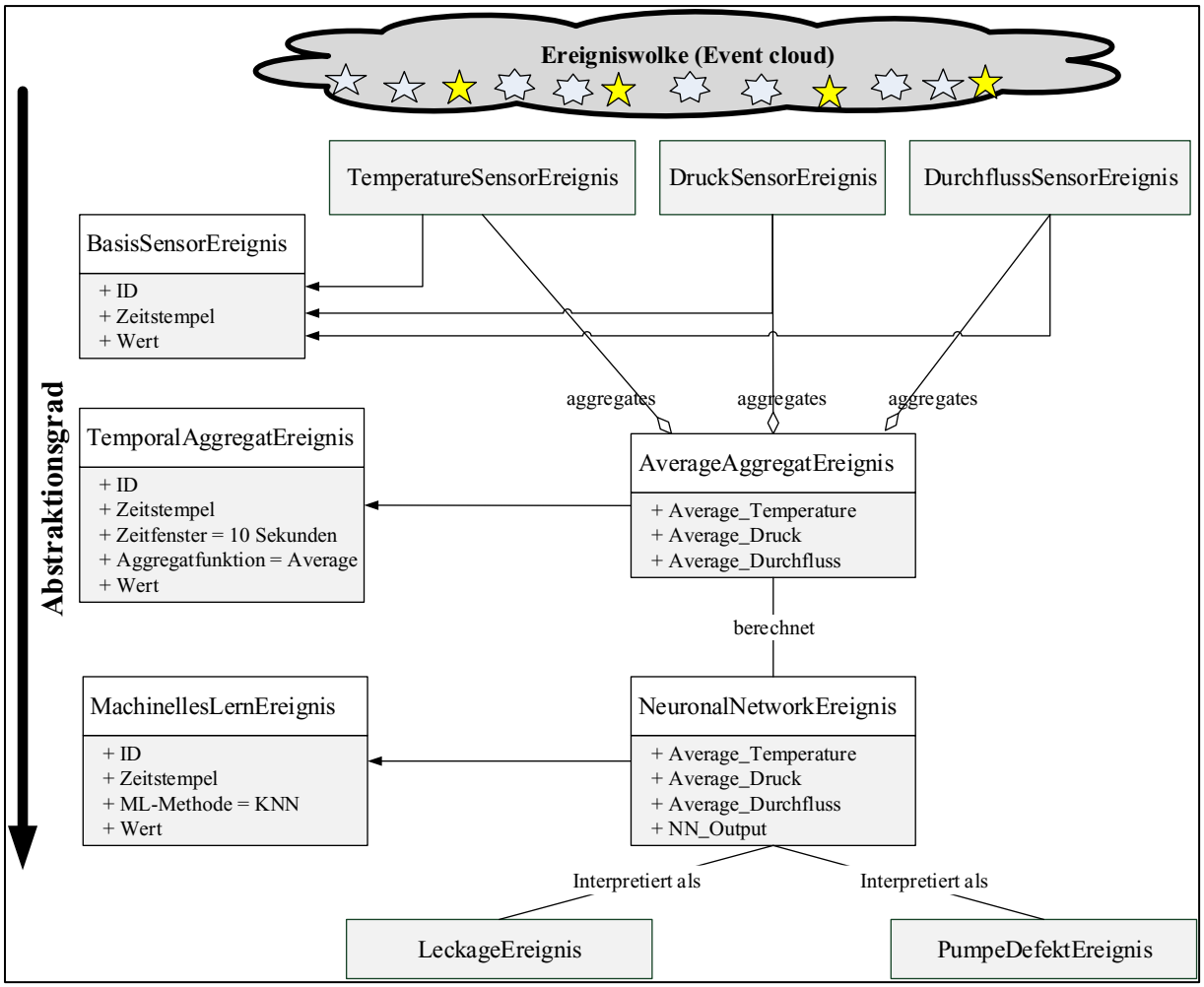

Abbildung 34: Beispiel eines UML-Ereignismodells für das Kühlsystem

Zu Beginn bilden alle Sensorereignisse, die durch den Ereignis-Adapter aus den Sensordaten erzeugt werden, die sogenannte Ereigniswolke. Aus diesem Ereignisstrom werden die Durchschnittswerte der in den letzten 10 Sekunden aufgetretenen Sensorereignisse berechnet. Sie bilden folglich einen neuen Ereignisstrom von zeitlich aggregierten Ereignissen (AverageAggregatEreignis). Für eine Störungserkennung wird ein neuronales Netz verwendet. Deswegen erfolgt die Berechnung des Netzoutputs daraufhin auch durch ein NeuronalNetworkEreignis, das die verschiedenen Durchschnittswerte der Sensoren als Input hat. Grundsätzlich wurde das neuronale Netz zuvor im Offline-Datenanalyse-Modul trainiert. Dementsprechend werden die Störungen aus dem neuronalen Netz abgeleitet und darauffolgend in LeckageEreignis oder PumpeDefektEreignis klassifiziert.

5.2. Konfiguration und Optimierung

Ein Zustandsüberwachungssystem erfordert ein kontinuierliches Training bzw. Lernen des Systemverhaltens, da sich aufgrund von natürlicher Alterung oder auch neu eingebauten Komponenten bestehende Wertebereiche verschieben können. Für den in dieser Arbeit verfolgten CEP-Ansatz bedeutet dies eine Anpassung sowohl der Ereignisse (Basis- und Komposite-Ereignisse) als auch der Ereignisregeln. Zudem kommt bei den Basisereignissen hinzu, dass die Schwellwertbereiche a priori erstellt wurden und gegebenenfalls in keiner Beziehung zu Störungsereignissen stehen.

Das hierarchisch aufgebaute CEP-Konzept ist darauf angewiesen, dass die Abstraktion der Ereignisse auf eine Weise erfolgt, bei der sich relevante Beziehungen zwischen einzelnen Ereignissen erkennen lassen bzw. diese durch die Abstraktion noch verstärkt werden. In diesem Sinne ist es Aufgabe der Systemkonfiguration bzw. der Justierung, die Ereignisverarbeitung so anzupassen, dass sich Störungsereignisse sicher detektieren lassen. Idealerweise können auch die Störungsursachen ermittelt werden, um entsprechende Instandhaltungsmaßnahmen einzuleiten. Die Justierung bzw. Parametrisierung umfasst dabei die automatische Regelextraktion für komplexe Ereignisse und eine Anpassung existierender Ereignisse. Des Weiteren wird im Rahmen der Diagnostik untersucht, inwieweit sich die Rückverfolgbarkeit von Ereignisketten zur Identifikation der Störungsursachen eignet. In diesem Zusammenhang wird eine Konfigurationsmethodik entwickelt, die die Ereignisregeln unter Einsatz eines maschinellen Lernmodells extrahiert (Abbildung 35). Des Weiteren wird das Ziel verfolgt, ein maschinelles Lernmodell (ML) zu entwickeln. Das ML-Modell kann als ein Ereigniskorrelater gesehen werden, der die kausalen Beziehungen zwischen den resultierenden Basisereignissen durch Korrelation auf der Basis von temporalen und logischen Bedingungen einfängt. Ausgehend von diesem trainierten Modell werden Ereignisregeln extrahiert, welche die Zusammenhänge zwischen Basisereignissen und Systemstörungen in einer verständlichen und formalen Präsentation ermöglichen. Hierzu soll ein Konzept entwickelt werden, mit dessen Implementierung symbolische und logische Beschreibungen als Ereignisregeln aus dem ML-Modell extrahiert werden. Die Abbildung 35 veranschaulicht die Struktur der Offline-Konfigurationsmethodik. Diese besteht aus zwei Teilen. Der erste Teil befasst sich mit der Entwicklung des ML-Modells, das für die Klassifikation bzw. die Detektion von Systemstörungen im Rahmen des CEP-CM-Systems verwendet wird. Anschließend befasst sich der zweite Teil mit der Herleitung von Ereignisregeln aus dem ML-Aufbaumodell. In diesem Zusammenhang können sowohl neue Ereignisregeln in der Ereignisregel-Datenbank (Event-Pattern-Datenbank) hinzugefügt als auch bestehende Regeln angepasst werden.

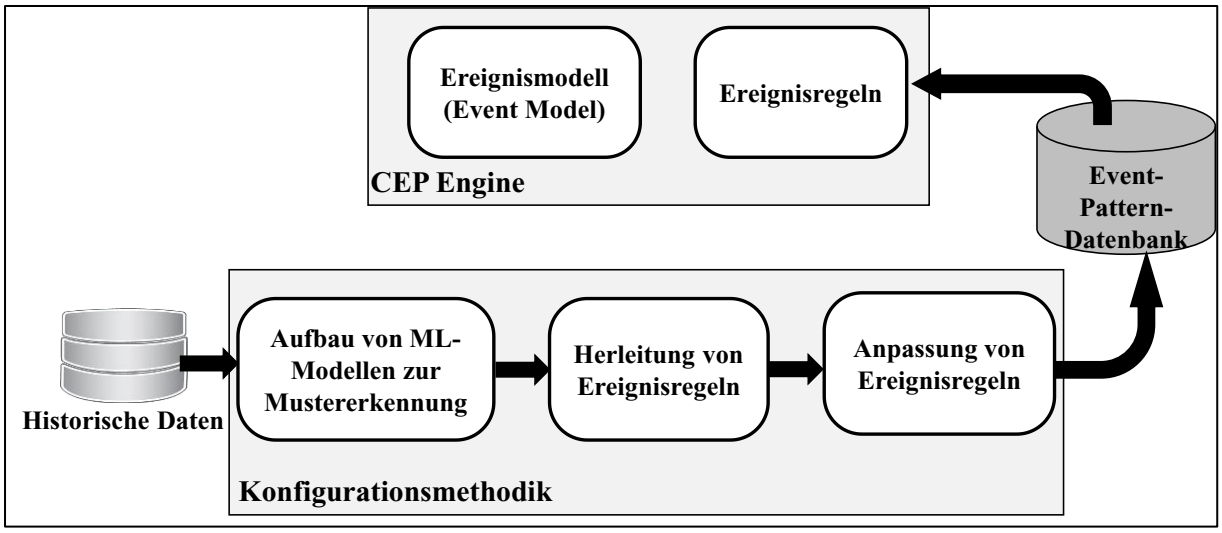

Abbildung 35: Grundprinzip der Konfigurationsmethodik

5.2.1. Konzeption der Offline-Datenanalyse – Aufbauphasen des ML-Modells

Der Aufbau eines Modells beginnt mit dem Erfassen und Sammeln der Daten. Als erster Schritt sollen diese Daten vorverarbeitet werden, da sie in der Regel unvollständig oder fehlerhaft und redundant sind. In diesem Zusammenhang kommen verschiedene Methoden zum Einsatz, beispielsweise „Imputation“, um die fehlenden Werte zu ergänzen, oder verschiedene Filtertechniken, wie z. B. ein gleitender Mittelwert, um die fehlerhaften Daten herauszufiltern. Anschließend werden die vorverarbeiteten Daten durch Verwendung von statistischen Methoden und graphischer Darstellung analysiert, um die Datenstrukturen, Zusammenhänge und Abhängigkeiten herauszufinden. Aus dieser Analyse können einige Berechnungen erfolgen, um die Daten für das Modell aufzubereiten. Dabei werden gegebenenfalls Aggregationsfunktionen berechnet oder statistische Methoden auf die Daten angewendet. Danach schließt sich die Bildung von Trainingsdaten an, die eine Zuordnung zwischen Modelleingaben bzw. -ausgaben bilden. Tabelle 3 zeigt die Struktur der Trainings- und Validierungsdaten. Der Modellaufbau befasst sich mit der Selektion des geeigneten Modells und der Modellparametrisierung. Dabei stellen die Qualität und Verfügbarkeit der Trainingsdaten ein wichtiges Kriterium für den guten Einsatz des Modells hinsichtlich der Klassifikation der Basisereignisse dar. Das Trainieren des Modells ist ein iterativer und in der Regel aufwendiger Prozess. Die Modellausgaben werden mit den Ausgaben der Trainingsdaten verglichen. Weichen die Modellausgaben stark von den Trainingsausgaben ab, müssen die Modellparameter angepasst werden, und das Training wird wieder neu gestartet. Ist die Trainingsphase abgeschlossen, erfolgt die Evaluation des Modells durch Validierungsdaten. Diese Daten des Modells sind neu und unbekannt und haben dieselbe Struktur wie die Trainingsdaten. Sind die Ergebnisse der Evaluation nicht gut, muss das Modell neu parametrisiert und wieder trainiert werden. Am Ende steht

ein Modell, das zur Klassifikation und Prognose von neuen Daten angewandt werden kann. Abbildung 36 veranschaulicht die Aufbauphasen der Offline-Datenanalyse zur Erstellung des ML-Modells.

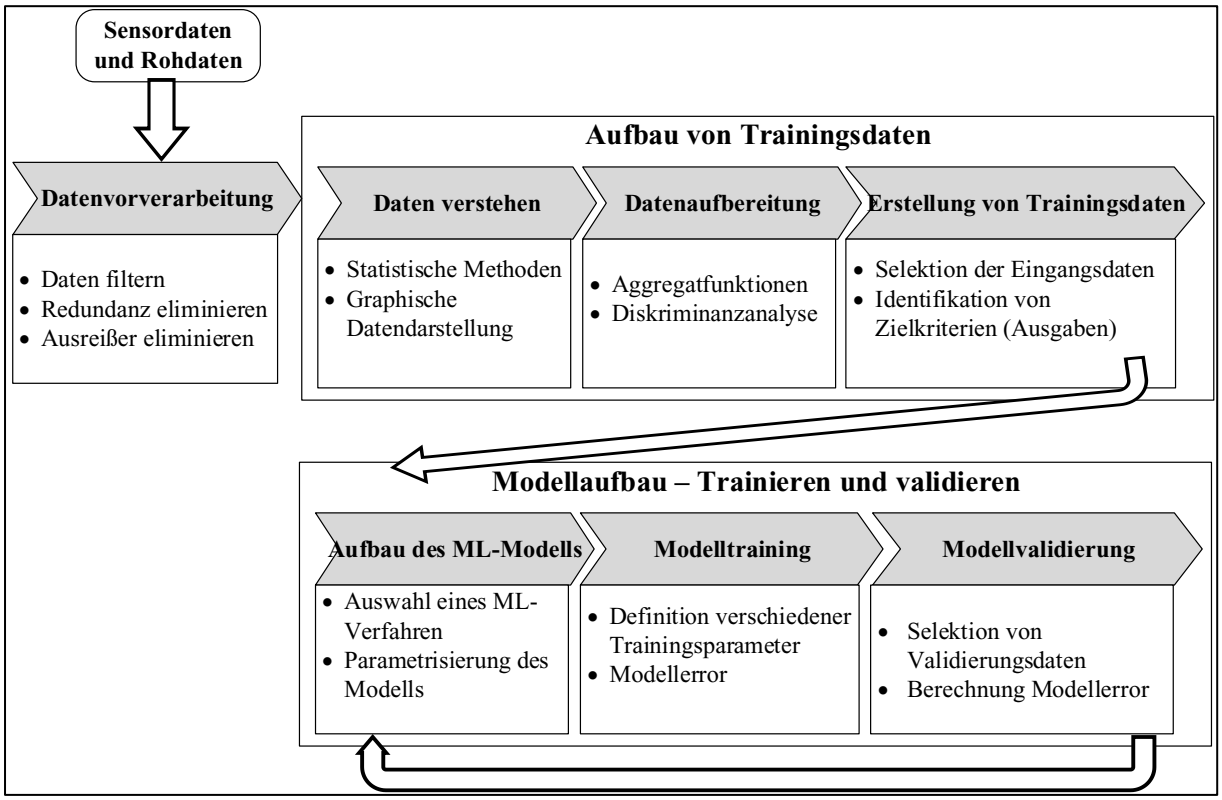

Abbildung 36: Aufbauphasen des ML-Modells

Das Modell wird danach immer kontinuierlich auf neue Situationen trainiert, da sich aufgrund von natürlicher Alterung oder auch neu eingebauten Komponenten bestehende Wertebereiche verschieben und neue Trainingsdaten nur in Realzeit generiert werden können. Diese stehen dann nicht von Beginn an für ein umfassendes Training des Modells zur Verfügung.

Tabelle 3: Struktur der Trainingsdaten und Validierungsdaten

	Modelleingabe				**Modellausgabe**			
1	I_{11}	I_{12}	….	I_{1n}	O_{11}	O_{12}	…	O_{1m}
2	I_{21}	I_{22}	….	I_{2n}	O_{21}	O_{22}	…	O_{2m}
3	I_{31}	I_{32}	….	I_{3n}	O_{31}	O_{32}	…	O_{3m}
i	I_{i1}	I_{i2}	….	I_{in}	O_{i1}	O_{i2}	…	O_{im}
N	I_{N1}	I_{N2}	….	I_{Nn}	O_{N1}	O_{N2}	…	O_{Nm}

Das zu entwickelnde ML-Modell soll idealerweise basierend auf möglichen, wenigen initialen Trainingsdaten, die normalerweise das normale Verhalten des Systems beschreiben, eine Generalisierung der Klassifizierung ermöglichen. Anschließend sollen die Ereignisregeln über ein Konzept ausgehend von dem Modell extrahiert werden. Die Regeln sollen an jede neue Klassifikation des Mo-

dells angepasst werden und darauf reagieren. Hierbei muss unterschieden werden, ob neue Ereignisregeln hinzugefügt oder vorhandene Regeln modifiziert werden.

5.2.2. Konzept der Regelextraktion aus der Offline-Datenanalyse

Das Konzept zur Regelextraktion aus den ML-Modellaufbauphasen ist in Abbildung 37 dargestellt. Ziel ist es, auf Basis von Ereignisanfragesprache die Ereignismuster auf einer hohen Abstraktionsebene deklarativ zu erkennen. Hierbei wird ein Konzept für die (teil-)automatisierte Generierung bzw. adaptive Anpassung von Ereignisregeln aus den verschiedenen Modellaufbauphasen entwickelt. Dabei gilt es, die Störungsereignisse mit Basisereignissen bzw. auch anderen komplexeren Ereignissen in Beziehung zu setzen. Sofern dies möglich ist, können Störungen über das Eintreten der in Beziehung stehenden Ereignisse detektiert werden. Diese Beziehungen werden durch einfache „Wenn-Dann-Regeln“ und SQL-ähnliche Abfragen ausgedrückt, die sich direkt durch Ereignisverarbeitungssprachen umsetzen lassen. Zur Aufbauphase des ML-Modells werden im ersten Schritt die Ereignisregeln aus der Datenvorverarbeitungsphase hergeleitet. Wie bereits erwähnt, werden hier die Eingangsdaten nach Redundanzen und Fehlern gefiltert. Diese Operationen werden basierend auf der ersten Definition des Ereignismodells in entsprechende Ereignisregeln transformiert, die gegebenfalls auch neue Filterereignisse erstellen können. Ein neuer Filterereignisstrom wird dementsprechend kontinuierlich generiert. Im zweiten Schritt werden die Ereignisregeln aus der Datenaufbereitungsphase extrahiert. Genau wie beim ersten Schritt werden hier alle durchgeführten Operationen aus dem Filterereignisstrom in Ereignisregeln transformiert, die neue Ereignisse auf einer höheren Abstraktionsstufe und einen neuen aggregierten Ereignisstrom generieren. Im Anschluss erfolgt die Herleitung der Ereignisregeln aus der Modellaufbauphase, die zur Erkennung von Störungsereignissen einen wichtigen Beitrag leistet. Durch die Anwendung des ML-Modells aus den Ereignissen des aggregierten Ereignisstroms entsteht der ML-Ereignisstrom, dessen Interpretation verschiedene Störungsereignisse des Systems erkennen kann.

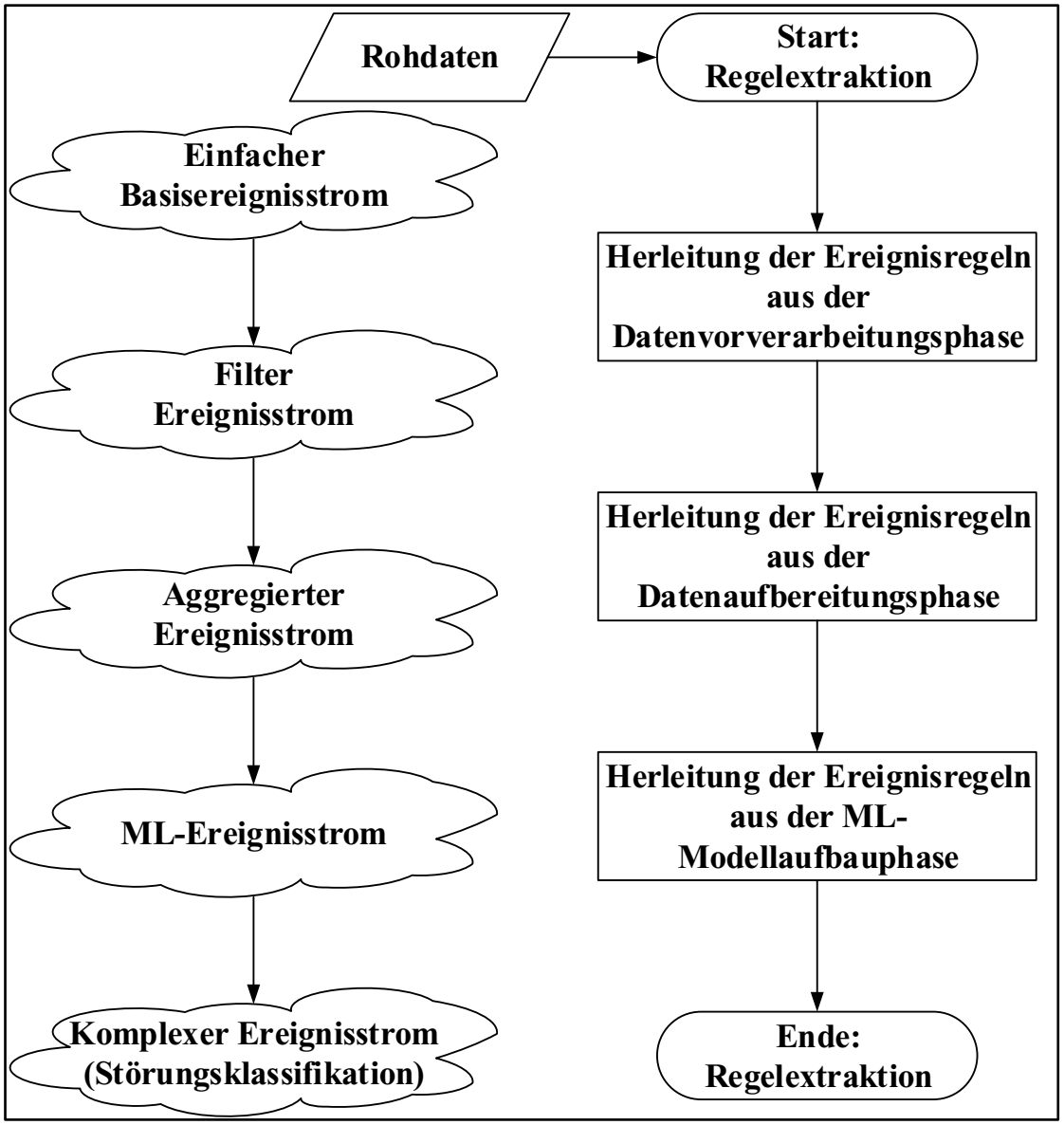

Abbildung 37: Konzept zur Extraktion von Ereignisregeln aus dem ML-Modell

5.2.2.1. Herleitung der Ereignisregeln aus der Vorverarbeitungsphase

Die Abbildung 38 stellt die Logik zur Extraktion der Ereignisregeln aus der Vorverarbeitungsphase dar. Die Ereignisregeln sind grundsätzlich auf Basis des entwickelten Ereignismodells und des extrahierten ML-Modells definiert. Basierend auf der Kategorisierung von Ereignissen in verschiedenen Klassen sind zunächst die Basisereignisse aus der Datenvorverarbeitungsphase zu definieren. Diese Basisereignisse stellen die unterste Abstraktionsebene der Ereignishierarchie dar.

Zuerst wird geprüft, ob die Eingangsdaten, welche durch den Ereignisadapter in einem einfachen Ereignisstrom transformiert wurden, einigen Filteroperationen unterzogen worden sind. Falls die Filterung der Eingangsdaten stattgefunden hat, werden diese Operationen dementsprechend in Ereignisregeln übermittelt.

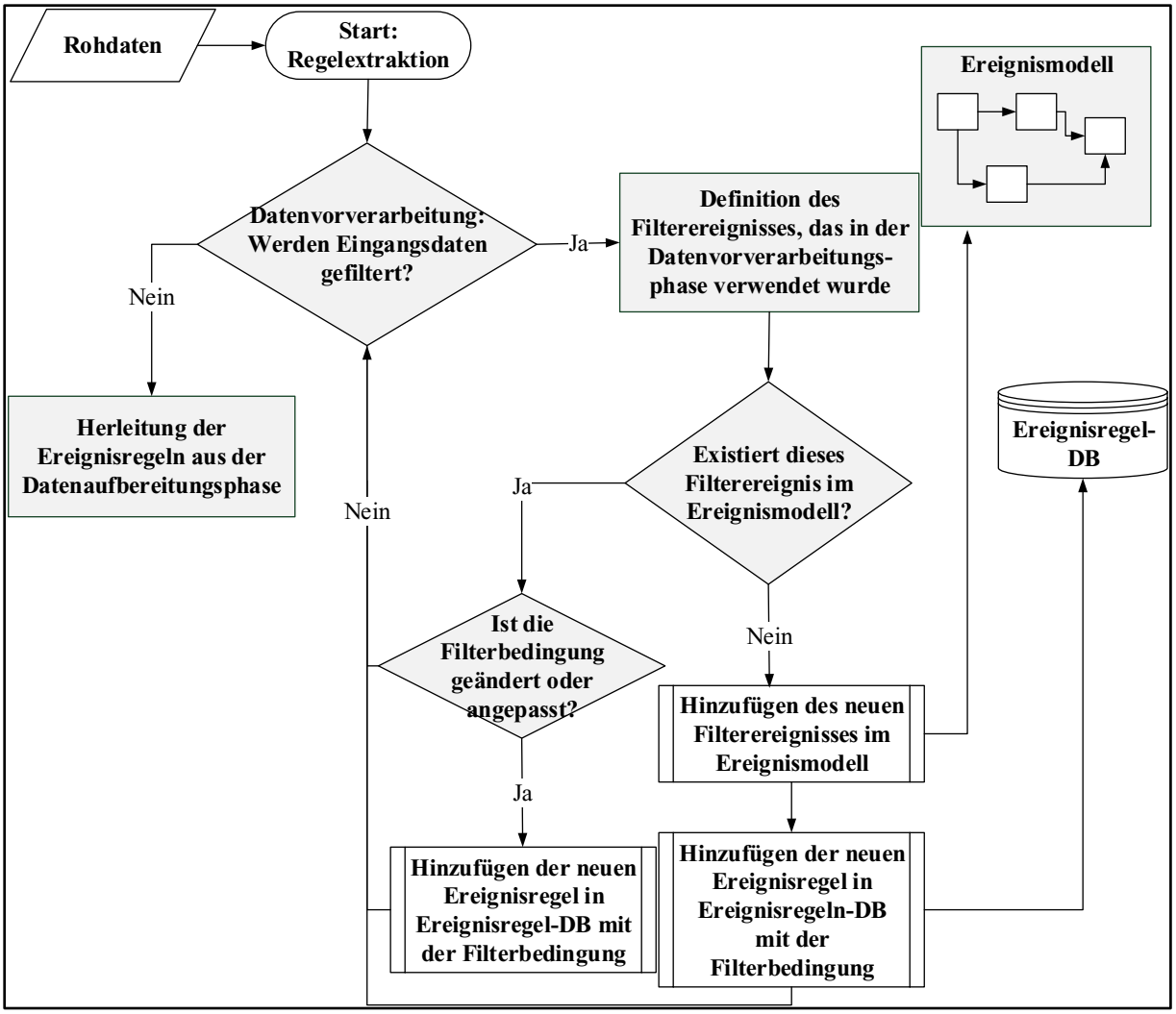

Abbildung 38: Regelextraktion aus der Datenvorverarbeitungsphase des ML-Modells

Durch diese Ereignisregeln entstehen neue Ereignisse auf einer höheren Abstraktionsstufe. Es handelt sich wiederum um einen neuen Ereignistyp, der nicht in dem Ereignismodell definiert wurde – es wird ein neuer Ereignistyp generiert und im Ereignismodell hinzugefügt. Folglich werden die entstehenden Ereignisregeln in der Ereignisregel-Datenbank hinzugefügt. Ist der Ereignistyp schon im Ereignismodell enthalten und die Ereignisregel in der Datenbank vorhanden, wird geprüft, ob die Regelbedingungen der Ereignisregel geändert wurden. Ist das der Fall, wird dies dementsprechend in der Datenbank angepasst. Dieser Vorgang wird solange wiederholt, bis alle in der Vorverarbeitungsphase durchgeführten Operationen behandelt wurden.

5.2.2.2. Herleitung der Ereignisregeln aus der Datenaufbereitungsphase

Die Logik zur Regelextraktion folgt, wie in Abbildung 39 veranschaulicht, derselben Logik, die auch bei der Regelextraktion aus der Datenvorverarbeitungsphase zugrunde gelegt wird. Auf Basis der Struktur des Ereignismodells sind die entstehenden Ereignisse aus der vorherigen Phase unter einer räumlichen Di-

mension gruppiert. Bei dieser Logik wird geprüft, ob Aggregatfunktionen aus den in der ersten Phase gefilterten Daten berechnet werden. Werden solche Operationen durchgeführt, ist damit zu rechnen, dass sowohl die räumliche als auch temporale Gruppierung der Daten in einem bestimmten Zeitfenster liegen.

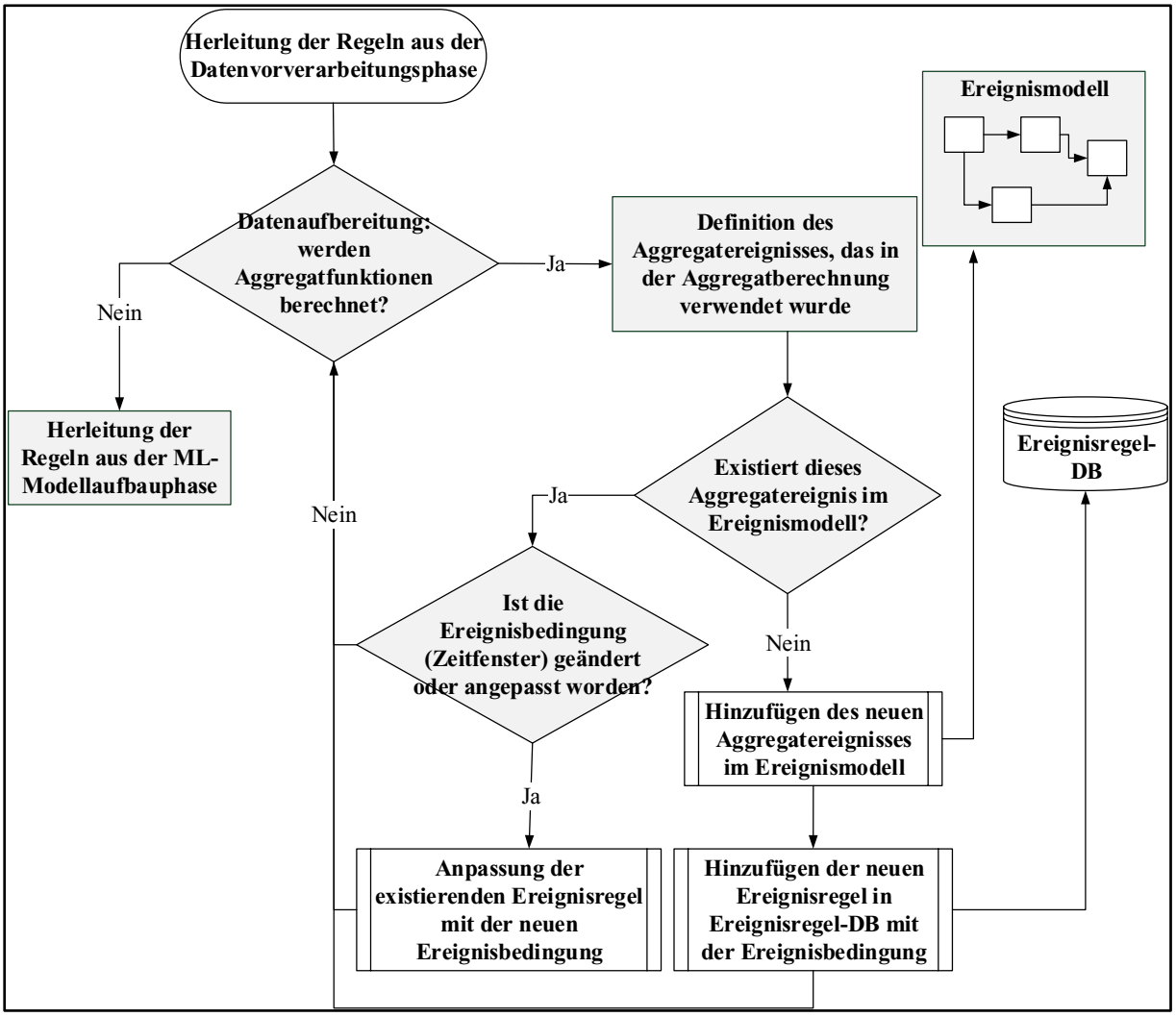

Abbildung 39: Regelextraktion aus der Datenaufbereitungsphase des ML-Modells

Die Operationen mit Aggregatfunktionen fassen diese Datengruppierung zu einem temporalen bzw. räumlichen Aggregatereignis zusammen. Wird dadurch ein neuer Ereignistyp generiert, wird das Aggregatereignis dementsprechend im Ereignismodell hinzugefügt. Die Ereignisregel wird weiterhin der Ereignisregel-DB hinzugefügt. Handelt es sich um eine Änderung einer bestehenden Ereignisregel, wird diese angepasst. Wie in der vorherigen Phase wird dieser Vorgang wiederholt, bis alle in der Aufbereitungsphase durchgeführten Operationen behandelt wurden.

5.2.2.3. Herleitung der Ereignisregeln aus der Aufbauphase des ML-Modells

Die grundlegende Logik ist der Logik der Regelextraktion aus der Datenvorverarbeitungsphase relativ ähnlich, wie in Abbildung 40 veranschaulicht. Zunächst

wird geprüft, ob ein ML-Modell in der Offline-Datenanalyse eingesetzt wurde. Der Aufbau des ML-Modells bedeutet, dass ein neues Ereignis von Typ „maschinelles Lernereignis“ im Ereignismodell hinzugefügt werden muss.

In diesem Zusammenhang wird zuerst geprüft, ob das ML-Ereignis bereits im Ereignismodell existiert. Ist dies nicht der Fall, werden das neu entstandene Ereignis dem Ereignismodell sowie die neue Ereignisregel hinzugefügt. Ansonsten wird geprüft, ob die Struktur des ML-Modells geändert wurde. Falls ja, wird die bereits existierende Ereignisregel dementsprechend angepasst.

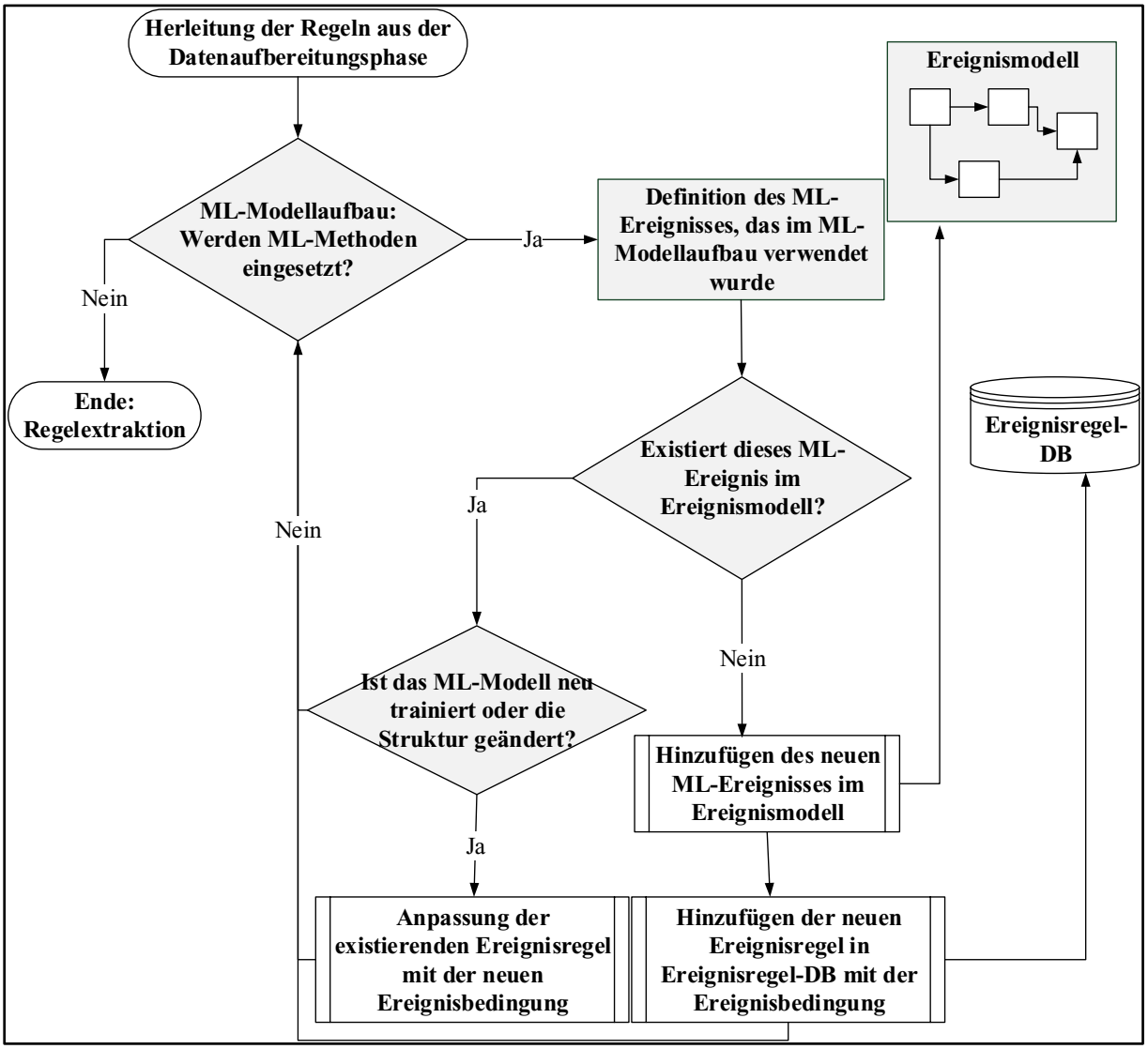

Abbildung 40: Regelextraktion aus der Aufbauphase des ML-Modells

5.2.3. Störungsbewertung und Ursachendiagnose

Ein Zustandsüberwachungssystem umfasst die Detektion von abnormalen Zuständen und die Identifikation von Ursachen solcher Ausfälle. Ausgehend von einer detektierten Funktionsstörung mittels CEP stellt sich die Frage nach der Schwere der Störung und der Diagnose möglicher Ursachen, um gegebenenfalls entsprechende Instandhaltungsmaßnahmen durchzuführen. Die Störung als sol-

che kann dabei entsprechenden Fehlerklassen zugeordnet werden, die von „toleranzabweichend“ bis hin zu „sicherheitskritisch“ reichen. Dementsprechend ist bei den Störungsmeldungen eine Feinjustierung vorzunehmen, die idealerweise die Dringlichkeit der durchzuführenden Instandhaltungsmaßnahme abbildet. Hierfür können zum Beispiel die absolute Anzahl an Störungsmeldungen oder die Kombination mit verschiedenen anderen Ereignissen als Indiz genutzt werden. In der Realität verursachen Totalausfälle großen Schaden, und es findet zum Beispiel eine präventive zyklische Instandhaltung statt, die dies verhindert, aber im Vergleich zur zustandsorientierten Instandhaltung weniger wirtschaftlich ist. Eine vorausgehende Ursachenanalyse ist wichtig, um die gestörte Komponente zu identifizieren und geeignete Instandhaltungsmaßnahmen in einer eingetretenen Situation einleiten zu können, da die Wartungsaufträge von den Ursachen der Störungen eingeleitet werden. CEP bietet neben seiner Echtzeitfähigkeit in der Ereignisverarbeitung die Möglichkeit, die Ursachen von Vorkommen komplexer Ereignisse anhand der Analyse von Zusammenhängen zwischen untergeordneten Ereignissen abzuleiten. Als Ergebnis dieser Ursachenanalyse kann deduktiv im Ereignismodell bzw. in den Ereignisregeln nach ausgelösten Ereignissen gesucht werden. Die deduktive Methode basiert auf dem Prinzip von Rückwärtsverkettung (engl. backward chaining). Über die für die Ereignisse hinterlegten Sensoren kann dann auch auf gestörte Komponenten geschlossen werden. Es wird dabei von dem komplexen Ereignis ausgegangen, das die Störung beschreibt, und es werden Ereignisregeln überprüft, die das komplexe Ereignis als Aktionsteil haben. Der Rückwärtsverkettungsmechanismus wird dann rekursiv herangezogen, um die Basisereignisse zu ermitteln, deren Vorkommen die Störung verursacht haben. Gegebenenfalls sind für bestimmte Störfälle zusätzlich spezifische Störungsursachen zu hinterlegen.

Als Nächstes wird überprüft, ob die Ursachen in der Realität stattgefunden haben. Allerdings ist es möglich, dass die Systemstörung durch falsche Werte des Sensors ausgelöst wurde, da der Sensor nicht mehr funktionstüchtig ist. Hierzu verfolgt die Überprüfung der Störungsursachen das Ziel, die Ereignisregeln zu validieren. Für die Validierung der Ereignisregeln wird zwischen zwei möglichen Fällen unterschieden:

- CEP erkennt eine Störung und die Sensoren sind funktionstüchtig.
- CEP erkennt eine Störung und die Sensoren sind nicht funktionstüchtig.

Im ersten Fall muss die Ereignisregel angepasst werden. Im zweiten Fall sollen die nicht funktionstüchtigen Sensoren ausgetauscht werden.

6. Konzeptumsetzung und Evaluation

Aufbauend auf dem im vorangegangenen Kapitel vorgestellten Referenzmodell widmet sich dieses Kapitel der Konzeptevaluation durch die praktische Umsetzung des zuvor entwickelten Modells. Die Evaluierung erfolgt basierend auf zwei Anwendungsfällen: einer prototypischen Erprobung eines Modells an einem Fuhrpark von Portalhubwagen (engl. van carrier) und bei der Vereisung an Windenergieanlagen. Beim ersten Anwendungsfall liegt der Fokus auf der Erkennung von Systemstörungen, während bei der zweiten Fallstudie die Detektion von Vereisungen an den Rotorblättern von Windenergieanlagen im Fokus steht. Der Grund für die Auswahl von zwei Implementierungsszenarien liegt in der Notwendigkeit, das entwickelte Referenzmodell an verschiedenen technischen Systemen zu testen, um die Übertragbarkeit und die technische Machbarkeit überprüfen zu können.

Beide Anwendungsfälle wurden mit der Siddhi CEP-Engine unter Verwendung des von WSO2 verwendeten Complex Event Processors (Version 4.4.0) durchgeführt. Es wurde ein 3.00 GHz AMD Phenom® II X4 945-Rechner mit 8.00 GB Arbeitsspeicher und dem Betriebssystem Windows 10® Pro 64-bit verwendet.

Die Teilziele des Kapitels umfassen daher die folgenden Punkte und werden im Folgenden detailliert beschrieben:

- Durchführung definierter Anwendungsfälle
- Festlegung der Kriterien zur Bewertung
- Evaluation der Testergebnisse

In jedem Abschnitt dieses Kapitels wird das Anwendungsszenario kurz vorgestellt. Anschließend wird die Anwendbarkeit des zuvor vorgestellten Referenzmodells (Kapitel 5) mit Verwendung von ML-Verfahren getestet. Schließlich werden die Ergebnisse dargestellt und diskutiert.

6.1. Durchführung des Evaluationsprozesses

In Abbildung 41 wird der Ablauf der Konzeptevaluation verdeutlicht. Im ersten Schritt werden Schnittstellen zu den Datenquellen implementiert. Es wird zwischen den Schnittstellen für die historischen und für die Livedaten unterschieden. In diesem Zusammenhang liegen die historischen Daten häufig in einer Datenbank oder als CSV-Datei vor, während die Livedaten beispielsweise über die REST-API (engl. representational state transfer – Application Programming Interface) an das CEP-System versendet werden. Das CEP-System sammelt dann kontinuierlich die Livedaten und speichert diese in der historischen Datenbank. Die historischen Daten werden für die Offline-Datenanalyse verwendet, während die Livedaten für die Konzeptevaluierung eingesetzt werden. Ausgehend von der System- und der Situationsanalyse wird das zu lösende Problem grund-

legend untersucht. Dementsprechend erfolgt anschließend die Identifikation von Sensoren und relevanten Daten, die für die Störungserkennung erforderlich sind. Dabei werden sowohl die nötigen Eingangsdaten als auch die erwarteten Störungen identifiziert und definiert. Des Weiteren werden die möglichen Störungsursachen definiert. Daraus erfolgt die Generierung einer ersten Version des Ereignismodells. Dieses Modell stellt die detaillierte Definition der Ereignisse dar, die im CEP-CM-System verarbeitet werden. Aus der Offline-Datenanalyse wird das ML-Modell aufgebaut. Auf dessen Grundlage werden, wie in Kapitel 5.2 beschrieben, das Ereignismodell und die Ereignisregeln erstellt, angepasst und erweitert. Dabei werden verschiedene Kennzahlen definiert. Für die Ergebnisevaluierung wird die Modellgenauigkeit (engl. accuracy) verwendet. Die Genauigkeit wird aus der Anzahl der korrekten und falschen Klassifikationen bzw. Vorhersagen berechnet. Deswegen werden die Größen True Positive (TP), False Positive (FP), True Negative (TN) und True Positive (TP) berechnet. Um die Echtzeit des Systems zu evaluieren, wird die Reaktionszeit des Systems berücksichtigt. Die Modellgenauigkeit wird wie folgt in Anlehnung an Mousheimish (2017) definiert:

$$accuracy = \frac{TP + TN}{TP + FP + TN + TP}$$

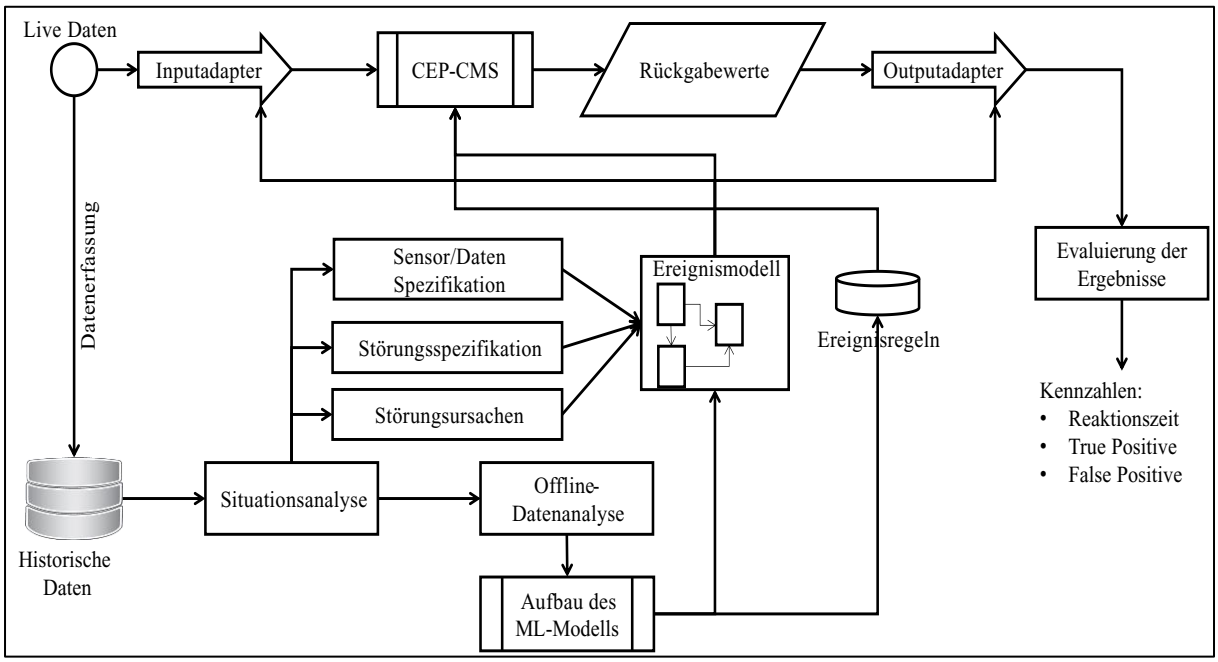

Abbildung 41: Gesamter Ablauf des Evaluationsprozesses

6.2. Szenario 1: Kühlkreislauf der elektrischen Radnabenmotoren eines Van Carriers

Beim Einsatz von Van Carriers (VC) werden hohe Anforderungen an die Geräte in Bezug auf Verfügbarkeit und Betriebssicherheit gestellt. Dies erfordert eine effektive Instandhaltung. Während die derzeitige Instandhaltung von VCs zyk-

lisch in festen Betriebsstundenintervallen erfolgt, wird unterdessen der Wechsel von der festen zyklischen Instandhaltungsstrategie zu einer zustandsorientierten Instandhaltungsstrategie verfolgt (Scholz-Reiter et al. 2009), (Lewandowski 2016), (Eberlein et al. 2018).

Die zustandsorientierte Instandhaltung von VCs erfordert die kontinuierliche Überwachung der betrachteten Geräte mit allen Komponenten. Daher sollen, ausgehend von den erfassten Sensordaten an wichtigen Komponenten, zuverlässige Bewertungen von Betriebsdaten und eine Datenanalyse in Echtzeit durchgeführt werden, um die Zielsetzung der echtzeitigen, zustandsorientierten Instandhaltung zu erfüllen.

Wie in Abbildung 41 veranschaulicht ist, erfolgt die Generierung von Ereignisregeln basierend auf der Offline-Datenanalyse und dem ML-Modell. Für die Entwicklung der ML-Modelle ist entsprechend die Verfügbarkeit aussagekräftiger Sensordaten erforderlich (Lewandowski 2016).

Da in der Regel immer noch eine zyklische Instandhaltung für VC durchgeführt wird, werden die meisten Störungen in den Daten nicht aufgezeichnet, wodurch eine Prognose über das Ausfallverhalten der VC nicht möglich ist. Um mit diesem Problem umzugehen, werden entsprechende Funktionsmodelle entwickelt, welche die Funktionen des realen Systems abbilden (Lewandowski 2016). In diesen Funktionsmodellen können somit gezielt Störungen eingeführt und simuliert werden. In diesem Kontext wird hier ein Funktionsmodell für den Kühlkreislauf elektrischer Radnabenmotoren eines VCs aufgebaut. Auf dessen Basis wird das entwickelte Referenzmodell evaluiert.

6.2.1. Beschreibung des Funktionsmodells

Die Abbildung 42 liefert einen Überblick über das aufgebaute Funktionsmodell des Kühlkreislaufs für elektrische Radnabenmotoren eines VCs. Das Funktionsmodell verwendet Wasser als Kühlmittel (im realen Fall Öl) und besteht aus einer Pumpe, einem Wärmetauscher mit Ventilator (Lüfter) und einer Motor-Kühler-Kombination (zur Simulation der vom Antrieb erzeugten Abwärme). Die Pumpe ist dafür verantwortlich, dass das Wasser durch das System zirkuliert, um die Wärme, die der laufende Motor erzeugt, abzuführen. Der Lüfter ist für die Wärmeabfuhr verantwortlich. Während das warme Kühlmittel durch den Kühler gepumpt wird, findet der Wärmetauscher mit dem Ventilator in der Kühlung des Kühlmittels Verwendung.

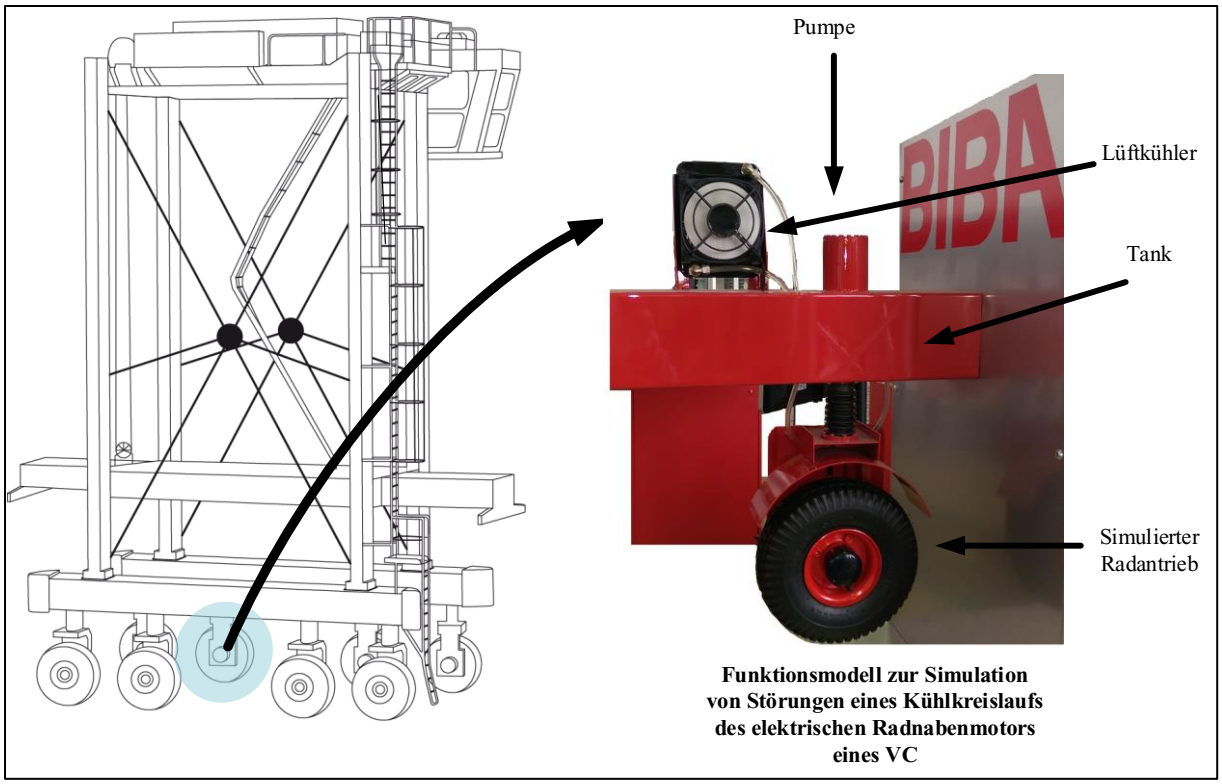

Abbildung 42: Schematischer Überblick über das Funktionsmodell

Mit dem Demonstrator können neben Leckagen an verschiedenen Stellen auch noch ein Druckverlust der Pumpe und eine Fehlfunktion des Ventilators simuliert werden. Lewandowki (2016) hat bereits ein ähnliches Funktionsmodell verwendet. Dabei befasst sich seine Arbeit mit der Entwicklung eines Integrationsmodells zur Zustandserfassung im Kontext der zustandsorientierten Instandhaltung von technischen Systemen. Lewandowski (2016) hat ein NN-Netz entwickelt, um eine Leckage zu detektieren und sein Integrationskonzept zu validieren. Sein Integrationsmodell basiert mehr auf einem herkömmlichen Datenbankmanagementsystem, weswegen keine echtzeitfähige Detektion der Störungen verfolgt wurde.

6.2.2. Situationsanalyse – Datenspezifikation

Zur Überwachung des Flüssigkeitskühlsystems sind fünf Sensoren im System verbaut (siehe Tabelle 4). Diese setzen sich aus folgenden Sensoren zusammen: einem Drucksensor, drei Temperatursensoren und einem Sensor zur Messung des Wasserdurchflusses. Es gibt keinen direkten Sensor für das Kühlmittelreservoir, weshalb mögliche Ausfälle (z. B. simulierte Leckagen) indirekt durch die oben genannten Sensoren ermittelt werden müssen (Lewandowski 2016), (Ait Alla et al. 2016). Durch ein Onboard-Unit-System werden die Daten aus den verbauten Sensoren kontinuierlich jede Sekunde erfasst und an einen Server gesendet, um das Verhalten des Systems zu bewerten. Diese Sensordaten werden von dem CEP-CM-System über den Inputadapter geladen und in das Format der CEP-Engine transformiert.

Tabelle 4: Spezifikation der verbauten Sensoren

Sensor	Minimalwert	Maximalwert	Einheit
Temperatur 1	0	40	°C
Temperatur 2	0	40	°C
Temperatur 3	0	60	°C
Druck	0	220	mbar
Durchfluss	0	10	l/min

Erwartete Störungen und deren mögliche Ursachen

Durch das Funktionsmodell werden verschiedene Störungen selbst eingeführt. Das Ziel ist es, diese Störungen und deren Ursachen in Echtzeit zu detektieren. Die simulierten Systemstörungen, die auftreten können, sind:

- Druckverlust durch Leckage oder Pumpendefekt
- unerwarteter Temperaturanstieg durch einen Ausfall des Lüfters

Ein Druckverlust kann verschiedene Gründe haben. Entweder kommt durch eine Leckage keine Kühlflüssigkeit mehr am Motor an, oder durch einen Pumpendefekt wird keine Kühlflüssigkeit mehr umgewälzt. Der unerwartete Temperaturanstieg kann auf einen Ausfall des Lüfters hindeuten, da die Kühlflüssigkeit im Wärmetauscher nicht mehr abgekühlt werden kann.

6.2.3. Erstellung der ersten Version des Ereignismodells

Die Situationsanalyse umfasst die Identifikation und die Definition der relevanten Daten, Störungen und deren Ursachen. Auf dieser Identifikation aufbauend, wird ein erstes Ereignismodell erstellt. Hier wird für jeden Sensortyp ein Basissensorereignis erstellt. In diesem Kontext werden ein ***Temperatursensorereignis***, ein ***Drucksensorereignis*** und ein ***Durchflusssensorereignis*** erstellt. Dabei werden für das Temperatursensorereignis drei Instanzen und für das Druck- und Durchflusssensorereignis eine Instanz initiiert. Aus diesen Basissensorereignisinstanzen wird ein zusammengeführtes ***Sensorereignis*** erstellt. Dieses Ereignis wird durch eine eindeutige Identifikationsbezeichnung *(EreignisID)* zur Unterscheidung des Ereignisses im System, den Auftrittszeitpunkt *(Zeitstempel)* und die Werte aller Sensoren repräsentiert.

Die Erkennung von Störungen erfolgt durch den Einsatz von statistischen Methoden und maschinellen Lernverfahren. Verschiedene Berechnungen werden an den Sensordaten und Aggregatfunktionen durchgeführt, aus welchen Indikationen auf die Störungen identifiziert werden.

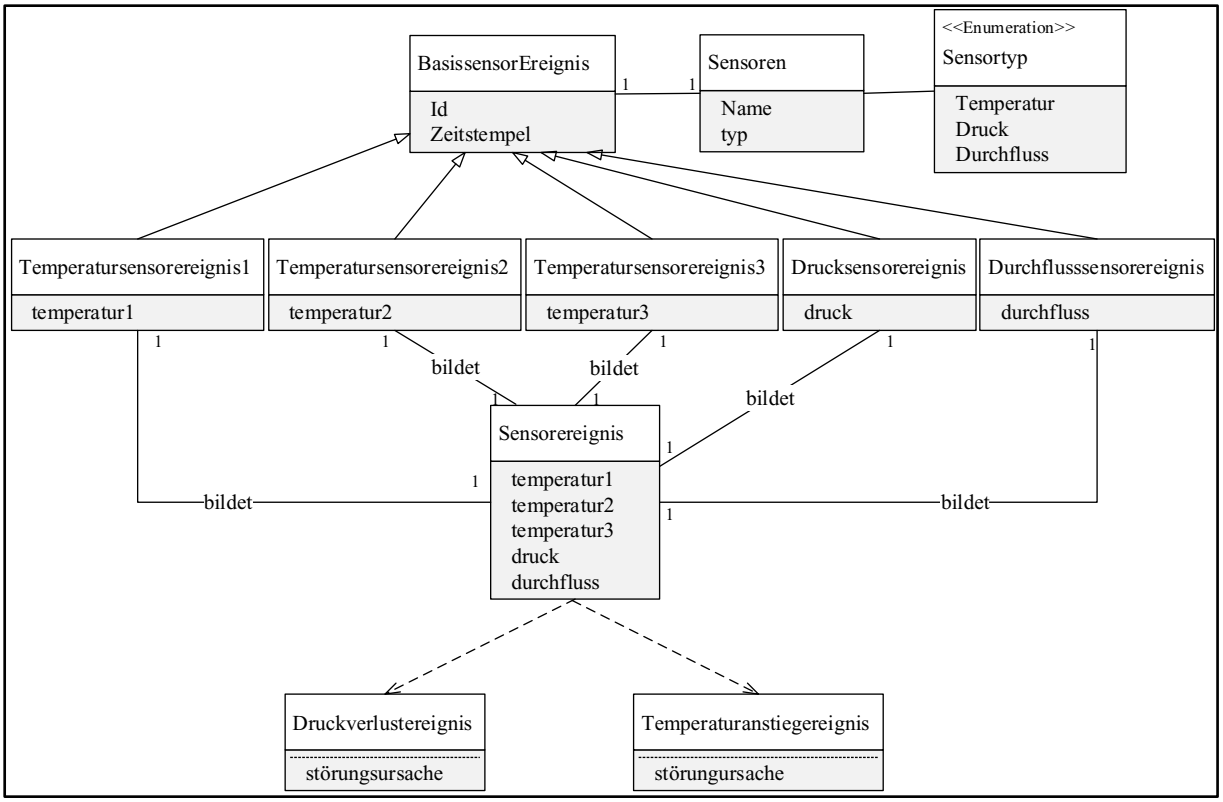

Abbildung 43: Aufbau der ersten Version des Ereignismodells

Aus der Störungsdefinition werden hierbei zwei Störungsereignisse „***Druckverlustereignis***“ und „***Temperaturanstiegsereignis***“ als komplexe Ereignisse definiert. Die Ursachen dieser Störungen können auf Leckage, Pumpendefekt oder Ausfall des Lüfters zurückgeführt werden (siehe Abbildung 43).

6.2.4. Konfigurationsmethodik zur Umsetzung des CEP-Systems für das erste Szenario

6.2.4.1. Offline-Datenanalyse / ML-Modellaufbau

Auf Basis der Offline-Datenanalyse wird nach den Zusammenhängen zwischen Sensordaten und den eingeführten simulierten Störungen gesucht. Dabei wird zuerst eine Datenvorverarbeitung durchgeführt, da einige Sensordaten fehlerhafte Informationen enthalten, die nicht Teil der analysierten Daten sein sollen.

Wie in Abbildung 44 zu sehen, enthalten die Sensordaten aufgrund der günstigen Sensoren, die im Funktionsmodell verbaut worden sind, ungültige Werte.

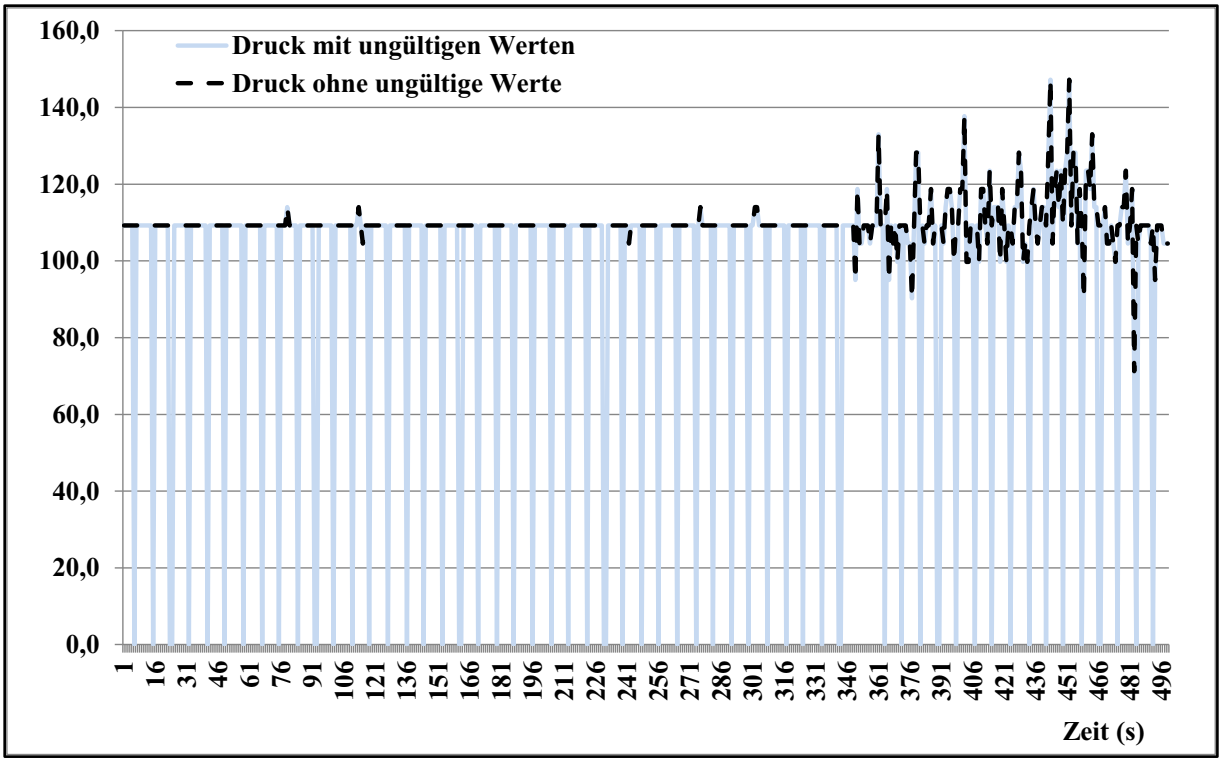

Abbildung 44: Beispiel von Druckwerten mit jeweils ungültigen Werten und ersetzten Werten

In dem Beispiel stellt ein Wert von 0 für alle Sensoren in demselben Zeitpunkt einen ungültigen Wert dar. Daher müssen die Sensorwerte zunächst vorverarbeitet und die ungültigen Werte gefiltert oder ersetzt werden, um die Daten erfolgreich handhaben zu können. Wenn die fehlenden bzw. ungültigen Werte nicht richtig behandelt werden, kann es zu einer fehlerhaften Schlussfolgerung über die Daten kommen. In der Vorverarbeitung können die ungültigen Daten entweder entfernt oder durch eine Datenimputation (lineare Interpolation oder Extrapolation) ersetzt werden. Eine andere Möglichkeit besteht darin, die fehlenden bzw. ungültigen Daten durch Durchschnittswerte zu ersetzen. Das Entfernen der ungültigen Daten hat den Nachteil, dass einige Dateninformationen im DM-Prozess nicht betrachtet werden. Das Ersetzen der ungültigen Daten kann aber auch zu einer Beeinflussung der Mittelwerte und Standardabweichung im DM-Prozess führen (Bramer 2013). In diesem Anwendungsfall beträgt der Anteil der ungültigen Sensordaten maximal 5 % aller Daten.

Angenommen, der Sensorwert S_t sei im Zeitstempel t ungültig. Dann wird dieser Wert durch den Mittelwert von S_{t-1} *und* S_{t-2} wie in der folgenden Formel ersetzt:

$$S_t = \frac{S_{t-1} + S_{t-2}}{2}$$

Diese neuen ausgefilterten Sensordaten stellen die Eingangsdaten für das ML-Modell dar. Neben diesen Eingangsdaten werden für alle Eingangsdaten der letzten fünf Sekunden folgende Aggregatfunktionen (Standardabweichung und die Steigung) berechnet. Zusammen mit den Eingangsdaten bilden die berechneten Aggregatfunktionen die Modellinputs.

Für diesen Anwendungsfall wurden 24 Druckverluste wegen Leckage, 20 Druckverluste wegen Pumpendefekt und 16 Temperaturanstiege aufgrund von Ausfällen des Lüfters im Demonstrator eingeführt und simuliert. Für das Modelltraining wurden 42 Trainingsdaten und für die Modellvalidierung 18 Validierungsdaten verwendet (siehe Tabelle 5). Die Simulation der Störungen fand an verschiedenen Tagen statt, um die Auswirkung einer vorherigen Störung auf die nächste eingeführte Störung zu vermeiden.

Tabelle 5: Gesamter Überblick über die Anzahl von Trainingsdaten bzw. Validierungsdaten

	Druckverluste wegen Leckage	**Druckverluste wegen Pumpendefekt**	**Temperaturanstieg wegen Ausfall des Lüfters**
Trainingsdaten	16	14	12
Validierungsdaten	8	6	4
Total	24	20	16

Da es sich hier um ein Klassifikationsproblem handelt und das Modell wenige Inputs hat, eignet sich für die Datenanalyse dieses Anwendungsfalls gut das Modell des Entscheidungsbaums. Entscheidungsbäume haben den Vorteil einer klaren Struktur, bei der direkt Entscheidungsregeln aus den Daten abstrahiert werden. Die Abbildung 45 zeigt das Ergebnis des Entscheidungsbauverfahrens.

Aus dem Entscheidungsbaum kann abgeleitet werden, dass ein Pumpendefekt durch einen starken Druckverlust in den letzten fünf Sekunden detektieren werden kann. Ein abweichender Druckabstieg mit leichtem Durchflussabstieg deutet auf einen Druckverlust wegen einer Leckage hin. Eine starke Temperaturdifferenz mit einem positiven Temperaturanstieg in den letzten fünf Sekunden des Temperatursensors T3 weist auf einen Ausfall des Lüfters hin.

Ein großer Vorteil des Entscheidungsbaumverfahrens besteht in der einfachen Interpretation der Ergebnisse durch Entscheidungsregeln. In diesem Kontext werden aus dem Entscheidungsbaum folgende Regeln abgeleitet.

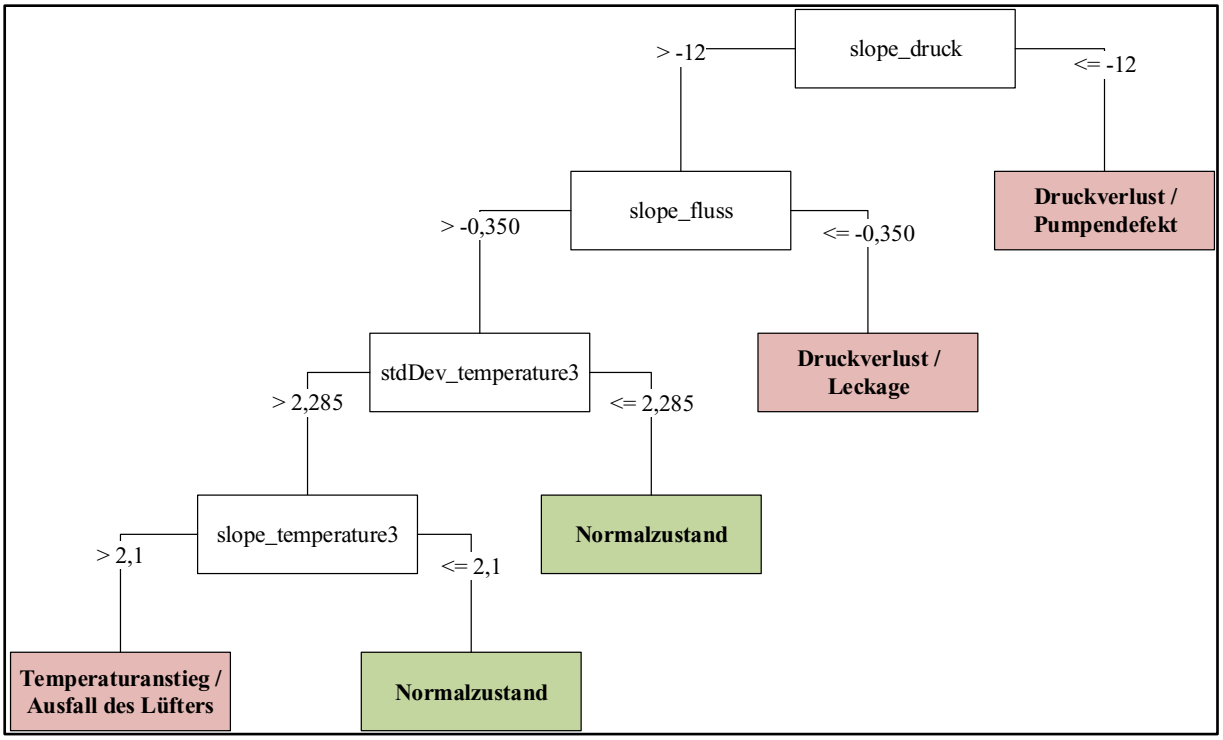

Abbildung 45: Struktur des Entscheidungsbaums zur Störungsklassifikation

If slope_druck > -12
| If slope_flow > -0.350
| | If stdDev_temperature3 > 2.285
| | | If slope_temperature3 > 2,1: **Temperaturanstieg** (Ausfall des Lüfters)
| | | Else If slope_temperature3 <= 2,1: **Normalzustand**
| | Else If stdDev_temperature3 <= 2.285: **Normalzustand**
| Else if slope_flow ≤ -0.350: **Druckverlust** (Leckage)
Else if slope_druck ≤ -12: **Druckverlust (**Pumpendefekt)

Tabelle 6: Validierungsergebnisse der Störungserkennung des Entscheidungsbaums

	Anzahl der Störungen	**Anzahl der korrekten Klassifikationen**	**Modellgenauigkeit (accuracy) (%)**
Druckverluste wegen Leckage	8	7	87,5
Druckverluste wegen Pumpendefekt	6	5	83,33
Temperaturanstieg wegen Ausfall des Lüfters	4	4	100
Total	18	16	88,9

Der Entscheidungsbaum weist eine gute Genauigkeit für die Störungserkennung auf (siehe Tabelle 6). Das Modell konnte alle Temperaturanstiege bedingt durch den Ausfall des Lüfters korrekt klassifizieren. Das Modell hat auch alle Druck-

verluste detektiert, aber die Störungsursache zweimal falsch klassifiziert. Einmal wurde eine Leckage-Störungsursache als Pumpendefekt-Störungsursache und ein anderes Mal eine Pumpendefekt-Störungsursache als Leckage-Störungsursache klassifiziert. Der Grund dafür könnte beispielsweise die Auswirkung eines Pumpendefekts auf eine nachfolgende Leckage sein, auch wenn darauf geachtet wurde, solche Auswirkungen zu vermeiden.

6.2.4.2. Regelextraktion aus der Offline-Datenanalyse und Erweiterung des Ereignismodells

Bevor mit der Generierung von Ereignisregeln begonnen wird, sind die verschiedenen notwendigen Ereignisströme aus dem initialen Ereignismodell zu definieren. Zunächst müssen die Datenströme (engl. streams) festgelegt werden, aus denen die zu analysierenden Daten entnommen werden sollen. Für diesen Anwendungsfall wird für jeden Sensor ein Inputstrom definiert. Als Beispiel legt die folgende Definition den Inputstrom für die Temperaturdaten aus dem Sensor „Temperatur 1“ fest. Es gibt zwei Attribute, die im Schema des Ereignismodells (Abbildung 43) aufgeführt sind, um den Wert der Temperatur als double und den Ereigniszeitstempel als Long-Wert zu erfassen. Den Parametern für den Strom entsprechen die Klassenattribute, definiert im Ereignismodell (Abbildung 43).

Define stream ***TemperaturInputStrom1****(timestamp long, temperature double);*

Die Definitionen der anderen Sensorinputströme sind der Definition des Inputstroms ***TemperaturInputStream1*** ähnlich.

Define stream ***TemperaturInputStrom2****(timestamp long, temperature double);*
Define stream ***TemperaturInputStrom3****(timestamp long, temperature double);*
Define stream ***DruckInputStrom*** *(timestamp long, druck double);*
Define stream ***DurchflussInputStrom*** *(timestamp long, durchfluss double);*

Wie im Ereignismodell (Abbildung 43) abgebildet ist, werden alle Sensorereignisdaten in einem einzigen Sensorereignis zusammengeführt. Der Sensorereignisstrom hat sechs Attribute, die Werte aller Sensoren und den Zeitstempel.

Define stream ***Sensorereignisstrom****(timestamp long, temperature1 double, temperature2 double, temperature3 double, druck double, durchfluss double);*

Der folgende Pseudocode definiert die Ereignisregel, wie die zu verarbeitenden Sensoreingabedaten aus den jeweiligen Sensorinputströmen übernommen werden. Die Ereignisregel überprüft, ob alle Ereigniswerte im selben Zeitpunkt eingetreten sind, und fügt die ankommenden Werte dem *Sensorereignisstrom* hinzu.

Pseudocode 1: Zusammenführung verschiedener Inputströme in einem Sensorereignisstrom.

Input: TemperaturInputStrom1, TemperaturInputStrom2, TemperaturInput-Strom3, DruckInputStrom, DurchflussInputStrom
Output: Sensorereignistrom

```
Für jedes Ereignis e1 in TemperaturInputStrom1 wiederhole
  Für jedes Ereignis e2 in TemperaturInputStrom2 wiederhole
    Für jedes Ereignis e3 in TemperaturInputStrom3 wiederhole
      Für jedes Ereignis e4 in DruckInputStrom wiederhole
        Für jedes Ereignis e5 in DurchflussInputStrom wiederhole
            Wenn e1.timestamp = e2.timestamp und
            e1.timestamp = e2.timestamp und
            e1.timestamp = e3.timestamp und
            e1.timestamp = e4.timestamp und
            e1.timestamp = e5.timestamp dann
                Select e1.timestamp, e1.temperature as temperature1,
                e2.temperature as temperature2, e3.temperature as
                temperature3, e4.druck as druck, e5.durchfluss as fluss
                Insert into Sensorereignisstrom
            Wenn_ende
        Wiederhole_ende
      Wiederhole_ende
    Wiederhole_ende
  Wiederhole_ende
Wiederhole_ende
```

6.2.4.3. Regelextraktion aus der Vorverarbeitungsphase

In diesem Abschnitt werden die Ereignisregeln durch den Einsatz des Konzeptes aus Kapitel 5.2.2 abgeleitet. Dementsprechend wird das Ereignismodell erweitert.

Aus der Vorverarbeitungsphase (Unterkapitel 5.2.2.1) lässt sich das Ereignismodell durch das Erstellen eines Ereignisses von Filterereignistyp „Filtersensorereignis“ erweitern. Dabei werden fehlerhafte Ereignisse aus dem Sensorereignisstrom korrigiert und herausgefiltert. Daher wird entsprechend ein Filtersensorereignisstrom mit denselben Attributen des Sensorereignisstroms wie folgt hinzugefügt:

*define stream **Filtersensorereignisstrom**(timestamp double ,temperature1 double, temperature2 double, temperature3 double, druck double, durchfluss double);*

Um die Sensorereignisse herauszufiltern und die Ergebnisse in den Filtersensorereignisstrom einzufügen, wird folgende Ereignisregel, im Folgenden als Pseudocode dargestellt, hinzugefügt:

Pseudocode 2: Bereinigung vom Sensorereignisstrom und Hinzufügen bereinigter Ereignisse im Filtersensorereignisstrom.

Input: Sensorereignisstrom
Output: Filtersensorereignisstrom

```
Für jede Ereignissequenz e1->e2->e3 vom Sensorereignisstrom wiederhole
    Wenn e3.temperature1 = 0 und
    e3.temperature1 = 0 und
    e3.temperature2 = 0 und
    e3.temperature3 = 0 und
    e3.druck = 0 und
    e3.durchfluss = 0 dann
        Select   e3.timestamp,
        (e1.temperature1 + e2.temperature1) / 2 as temperature1,
        (e1.temperature2 + e2.temperature2) / 2 as temperature2,
        (e1.temperature3 + e2.temperature3) / 2 as temperature3,
        (e1.druck + e2.druck) / 2 as druck,
        (e1.durchfluss + e2.durchfluss) / 2 as durchfluss
        Insert into Filtersensorereignisstrom
    Wenn_ende
Wiederhole_ende
```

6.2.4.4. Regelextraktion aus der Datenaufbereitungsphase

Aus der Datenaufbereitungsphase (Unterkapitels 5.2.2.2) wird das Ereignismodell durch das Erstellen eines aggregierten, temporalen Ereignistyps ergänzt. Der temporale Ereignistyp wird für die Berechnung der Aggregatfunktionen (Standardabweichung und Steigung (slope)) aus den Ereignissen des Filtersensorereignisstroms verwendet. Aus diesem Grund wird ein neues Ereignis „Aggregatsensorereignis“ im Ereignismodell hinzugefügt und dementsprechend ein neuer Aggregatsensorereignisstrom wie folgt definiert:

*Define stream **Aggregatsensorereignisstrom** (timestamp double , stdDev_temperature1 double, stdDev_temperature2 double, stdDev_temperature3 double, stdDev_druck double, stdDev_fluss double, slope_temperature1 double, slope_temperature2 double, slope_temperature3 double, slope_druck double, slope_durchfluss double);*

Der folgende Pseudocode berechnet die Aggregatfunktionen (Standardabweichung und slope) basierend auf der Teilmenge der Ereignisse, die während der letzten fünf Sekunden an den Aggregatsensorereignisstrom gesendet wurden.

Pseudocode 3: Berechnung von Aggregatfunktionen aus der Teilmenge der Ereignisse, die in die letzten fünf Sekunden gesendet wurden.

Input: Filtersensorereignisstrom
Output: Aggregatsensorereignisstrom
Erzeuge eine Queue der Länge fünf Sekunden
Für jedes Ereignis e1 in Filtersensorereignisstrom wiederhole
    Füge das Ereignis e1 im Queue
    Wenn die Queue voll ist dann
        Für jedes Attribut (temperature1, …, druck, durchfluss) wiederhole
            Berechne Standardabweichung des Attributs (temperature1, …, durchfluss) von allen Ereigniswerten aus dem Queue
            Berechne Steigungsfunktion des Attributs (temperature1, …, durchfluss) von allen Ereigniswertn aus dem Queue
            Lösche des letzten Ereignisses aus dem Queue
        Wiederhole_end
        select timestamp, stdDev(temperature1) as stdDev_temperature1,
        stdDev(temperature2) as stdDev_temperature2,
        stdDev(temperature3) as stdDev_temperature3,
        stdDev(druck) as stdDev_druck,
        stdDev(durchfluss) as stdDev_ durchfluss,
        slope(temperature1) as slope_temperature1,
        slope (temperature2) as slope _temperature2,
        slope (temperature3) as slope_temperature3,
        slope (druck) as slope_druck,
        slope (durchfluss) as slope_ durchfluss,
        insert into Aggregatsensorereignisstrom;
    Wenn_ende
Wiederhole_ende

Die Steigungsfunktion (slope) ist normalerweise nicht im Siddhi-Kern oder dessen Erweiterung bereitgestellt, deswegen ermöglichen es Skripte, Funktionen in Javascript zu schreiben und innerhalb von Siddhi-Abfragen auszuführen. Die durch Skripte definierten Funktionen können in Abfragen ähnlich wie bei jeder anderen eingebauten Funktion aufgerufen werden.

Die Steigungsfunktion für eine Liste von Punkten $P_i(x_i,y_i)$ der Länge n wird wie folgt definiert:

$$Slope(P_i) = \frac{\sum_{i=1}^{n}(x_i - \bar{X}) * (y_i - \bar{Y})}{\sum_{i=1}^{n}(x_i - \bar{X})^2}$$

$\bar{X}$ und $\bar{Y}$ sind hier die Mittelwerte von jeweils x_i und y_i.

6.2.4.5. Regelextraktion aus der Lernmodellaufbauphase

Für das Beispiel der Erkennung von verschiedenen Störungen im Funktionsmodell wurde mithilfe eines Entscheidungsbaums eine Klassifikation aus den Trai-

ningsdaten erlernt (Abbildung 45). Die Ergebnisse des Entscheidungsbaums können zu Entscheidungsregeln transformiert werden. Diese Entscheidungsregeln können direkt als Ereignisregeln im CEP transformiert und ausgedrückt werden. Aus diesem Grund eignen sich Ereignisse von Typ Filterereignis, welche die Entscheidungsregeln in der Filterbedingung festlegen. Dementsprechend werden zwei Ereignisströme (Temperaturanstiegsstrom und Druckverluststrom) erstellt. Diese neu erstellten Ereignisströme bilden die komplexen Ereignisse. Der Temperaturanstiegsstrom hat den Zeitstempel und die Störungsursache (Ausfall des Lüfters) als Attribute, während der Druckverluststrom den Zeitstempel und die Störungsursachen (Pumpendefekt oder Leckage) als Attribute hat.

*Define stream **Temperaturanstiegsstrom** (timestamp double, stoerungsursache string);*

Die folgenden Pseudocodes (4 und 5) leiten die komplexen Ereignisse (Temperaturanstiegsereignis und Druckverlustereignis) durch das Filtern von dem Aggregatsensorereignisstrom her.

Pseudocode 4: Herleitung von Temperaturanstiegsereignissen mit der Identifikation der Störungsursache (Ausfall des Lüfters).

```
Input: Aggregatsensorereignisstrom
Output: Temperaturanstiegsstrom
Für jedes Ereignis e1 in Aggregatsensorereignisstrom wiederhole
    Wenn e1.slope_druck > -12 und e1.slope_fluss > -0.35 und
        e1. stdDev_temperature3 > 2.285 und e1.slope_temperature3 > 2.1
        dann
        select e1.timestamp, "Temperaturanstieg wegen Ausfall des Lüf-
        ters" as stoerungsursache
        insert into Temperaturanstiegsstrom
    Wenn_ende
Wiederhole_ende
```

*Define stream **Druckverluststrom** (timestamp double, stoerungsursache string);*

Pseudocode 5: Herleitung von komplexen Druckverlustereignissen mit der Identifikation der Störungsursachen (Leckage oder Pumpendefekt).

```
Input: Aggregatsensorereignisstrom
Output: Druckverluststrom
Für jedes Ereignis e1 in Aggregatsensorereignisstrom wiederhole
    Wenn e1.slope_druck <= -12 dann
        select e1.timestamp, "Druckverlust wegen Pumpendefekt" as stoe-
        rungsursache
```

```
            insert into Druckverluststrom;
        Wenn_ende
        Wenn e1.slope_druck > -12 und e1.slope_fluss <= -0.35 dann
            select e1.timestamp, "Druckverlust wegen Leckage" as stoerungsur-
            sache
            insert into Druckverluststrom
        Wenn_ende
Wiederhole_ende
```

Der komplette Pseudocode zur Detektion von Störungen am Funktionsmodell basierend auf dem CEP-Ansatz kann dem Anhang B entnommen werden.

6.2.5. Implementierung und Evaluation des entwickelten CEP-CM-Systems

Die Abbildung 46 zeigt das resultierende Ereignismodell mit den entsprechenden Ereignisströmen.

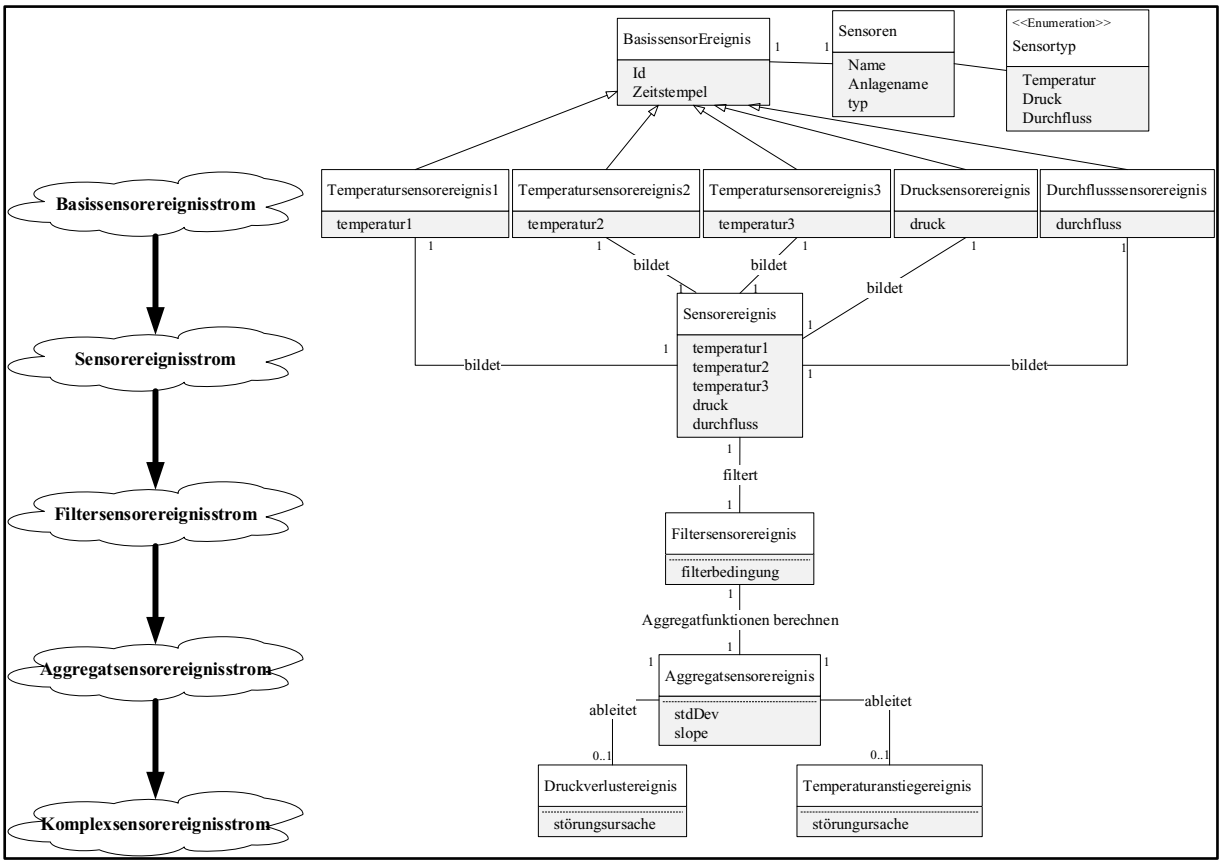

Abbildung 46: Ereignismodell und entsprechende Ereignisströme des betrachteten Funktionsmodells

Aus den Basisereignisströmen, die aus fünf Ereignisströmen der fünf Sensoren bestehen, wird der Sensorereignisstrom durch die Zusammenführung der Basisereignisströme generiert. Durch die Korrektur bzw. Filterung des Sensorereig-

nisstroms mithilfe von Ereignisbedingungen entsteht der Filterereignisstrom. Im nächsten Schritt werden die Aggregatfunktionen (Standardabweichung und Steigung) der letzten fünf Filterereignisse berechnet. Dadurch wird der Aggregatsensorereignisstrom generiert. Schließlich werden die Aggregatsensorereignisse durch Einsatz der Entscheidungsregeln aus dem Entscheidungsbaum herausgefiltert, um die komplexen Ereignisse (Druckverlustereignisse und Temperaturanstiegsereignisse) zu detektieren.

6.2.5.1. Beispiel der Detektion des Druckverlusts

Tabelle 7 und Tabelle 8 zeigen einen Ausschnitt der Sensordaten kurz vor einem Druckverlust aufgrund einer Leckage bzw. eines Pumpendefekts.

Tabelle 7: Ausschnitt aus den Trainingsdaten einer Leckage

Zeit-stempel	Druck	Temperatur1	Temperatur2	Temperatur3	Durchfluss	Zustand	StdDev_ temperature3	slope_ druck	slope_ temperature3	slope_ fluss
1236	187,2	26,0	26,4	49,5	4,0	Normal	0,0	0,0	0,0	-0,1
1237	187,2	26,0	26,4	49,5	4,0	Normal	0,0	-0,5	0,0	-0,1
1238	189,6	26,0	26,4	49,5	4,0	Normal	0,0	-0,7	0,0	0,0
1239	192,0	26,0	26,4	49,5	4,0	Normal	0,0	-0,2	0,0	0,1
1240	189,6	26,0	26,4	49,5	4,0	Normal	0,0	0,7	0,0	0,1
1241	175,2	26,0	26,4	49,5	2,6	Leckage	0,0	1,0	0,0	0,0
1242	160,8	26,0	26,4	49,5	1,2	Leckage	0,0	-2,4	0,0	-0,3
1243	163,2	26,0	26,4	49,5	0,5	Leckage	0,0	-7,4	0,0	-0,7
1244	163,2	26,0	26,4	49,5	0,5	Leckage	0,0	-8,6	0,0	-1,0
1245	165,6	26,0	26,3	49,5	1,2	Leckage	0,0	-6,5	0,0	-0,9
1246	160,8	26,0	26,3	49,5	0,5	Leckage	0,0	-1,7	0,0	-0,4
1247	165,6	26,0	26,1	49,5	0,5	Leckage	0,0	0,2	0,0	-0,1

Die Abbildung 47 veranschaulicht die Detektionszeit des durch eine Leckage bedingten Druckverlustes. Die Leckage wurde im Demonstrator zum Zeitpunkt 1240 eingeführt, und das CEP-CM-System detektierte die Leckage zum Zeitpunkt 1243, was bedeutet, dass der Detektionsprozess drei Sekunden gedauert hat.

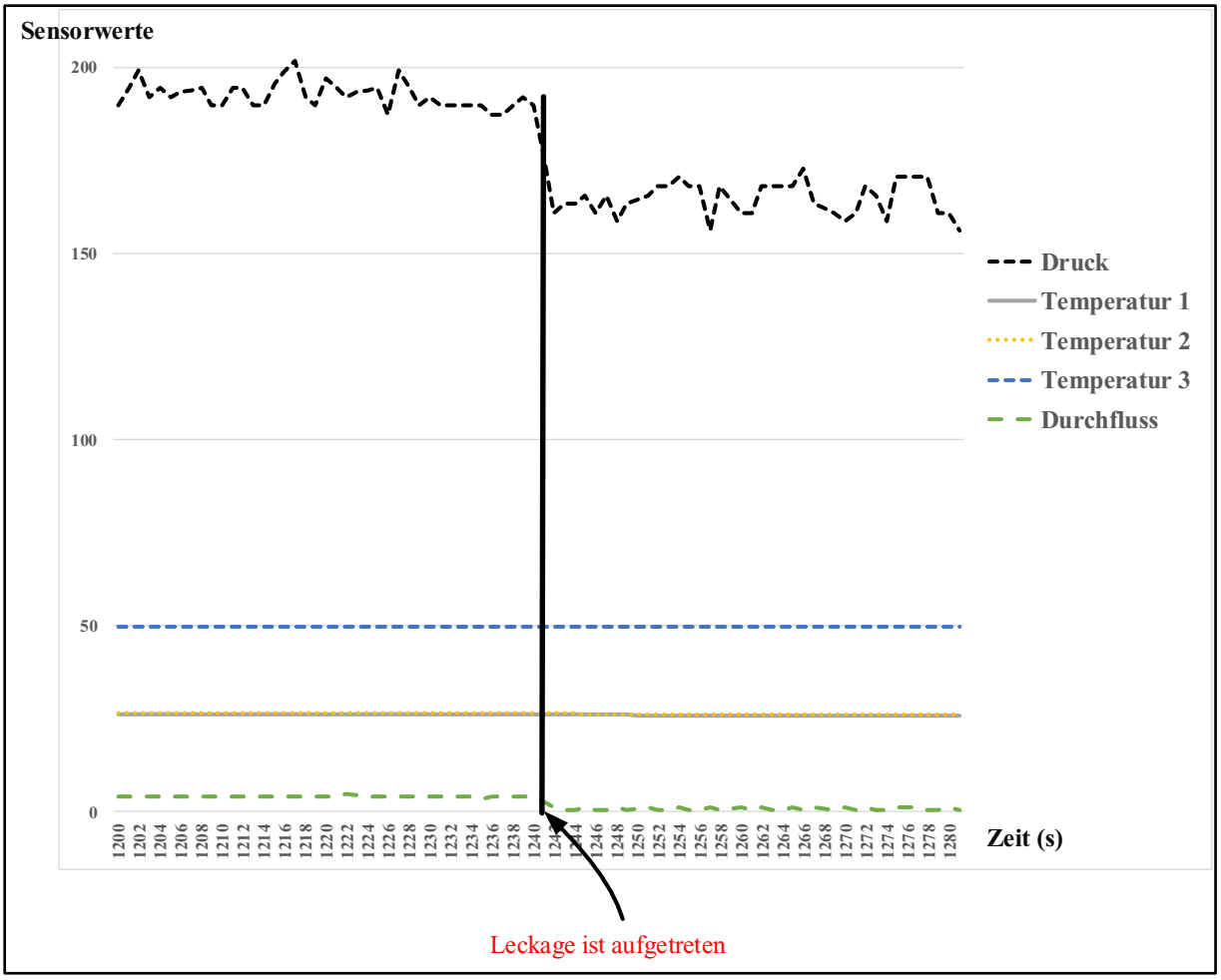

Abbildung 47: Entwicklung der Sensorwerte beim Auftreten einer Leckage

Tabelle 8: Sensordatenausschnitt eines Pumpendefekts

Zeit-stempel	Druck	Temperatur1	Temperatur2	Temperatur3	Durchfluss	Zustand	stdDev_ temperature3	slope_ Druck	slope_ T3	slope_ Flow
2836	192	26,0	26,4	50,4	4	Normal	0,0	0,7	0,0	0,1
2837	193,2	26,0	26,4	50,4	4	Normal	0,0	-0,2	0,0	0,0
2838	193,8	26,0	26,4	50,4	4	Normal	0,0	-1,7	0,0	-0,1
2839	194,4	26,0	26,4	50,4	4	Normal	0,0	-0,5	0,0	-0,1
2840	189,6	26,0	26,4	50,4	4	Normal	0,0	0,7	0,0	0,0
2841	163,2	26,0	26,4084	50,4	0,0	Pumpendefekt	0,0	-0,4	0	0,0
2842	146,4	26,0	26,4084	50,4	0,0	Pumpendefekt	0,0	-6,4	0	-0,8
2843	146,4	26,0	26,4	50,4	0,0	Pumpendefekt	0,0	-12,6	0,0	-1,2
2844	141,6	26,0	26,4	50,4	0,0	Pumpendefekt	0,0	-13,9	0,0	-1,2
2845	144,0	26,0	26,4	50,9	0,0	Pumpendefekt	0,0	-11,3	0,0	-0,8
2846	144,0	26,0	26,4084	50,9	0,0	Pumpendefekt	0,2	-4,3	0,1	0,0
2847	146,4	26,0	26,4084	50,9	0,0	Pumpendefekt	0,2	-0,7	0,1	0,0

Die Abbildung 48 veranschaulicht die Detektionszeit des Druckverlustes aufgrund eines Pumpendefekts. Ähnlich wie bei der Leckage wurde der Pumpendefekt zum Zeitpunkt 2840 in den Demonstrator eingeführt, und das CEP-CM-

System hat die Störung zum Zeitpunkt 2843 detektiert, was einer Reaktionszeit von 3 Sekunden entspricht.

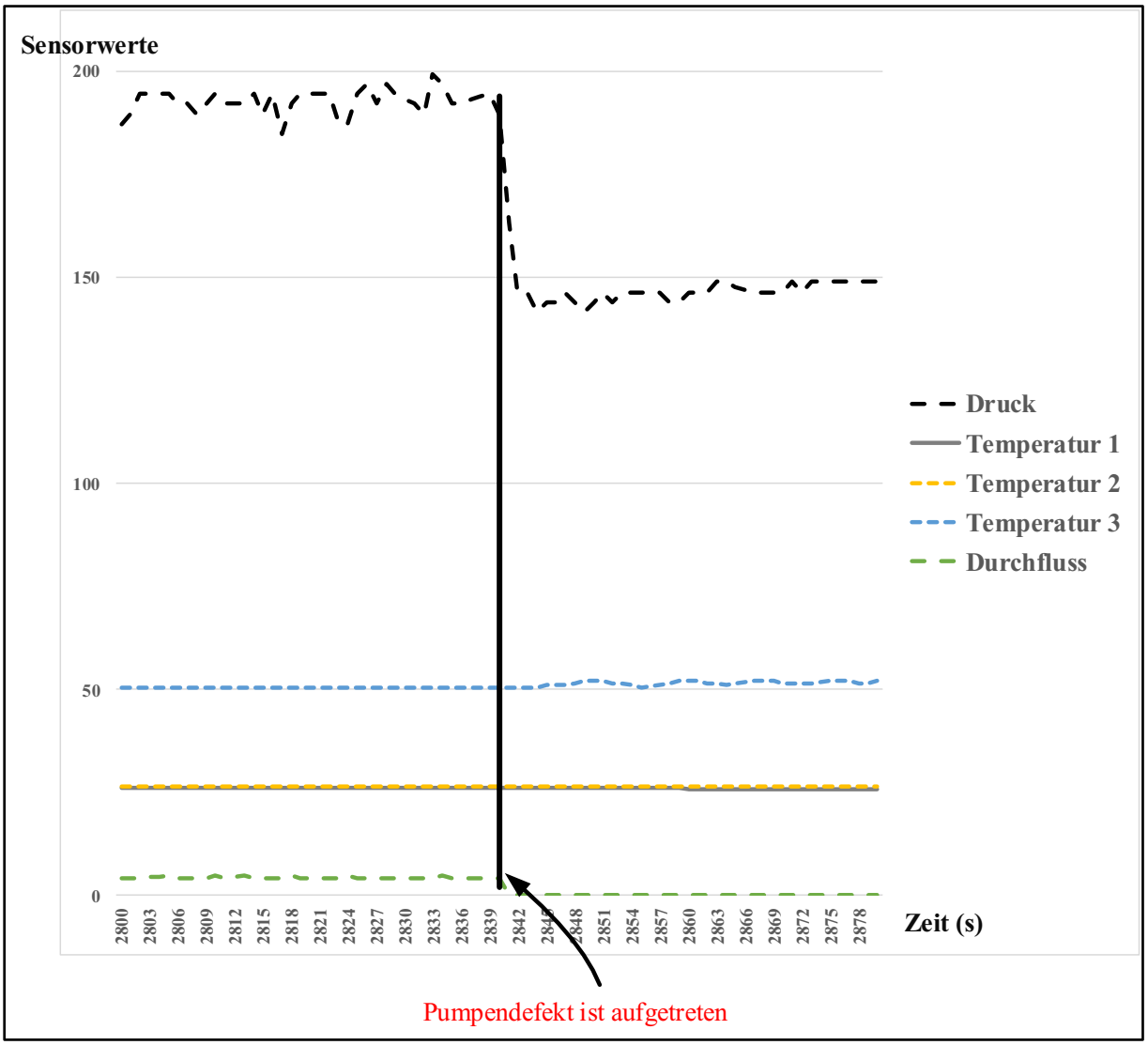

Abbildung 48: Verlauf der Sensorwerte beim Auftreten eines Pumpendefekts

Da es sich bei diesem ersten Anwendungsszenario um einen einfachen Anwendungsfall handelt, hat das CEP-System dieselbe Modellgenauigkeit wie der Entscheidungsbaum aus Abbildung 45 erreicht.

6.2.5.2. Leistungsvergleich zwischen CEP-CM-System und einem konventionellen CM-System

Die Reaktionszeit des entwickelten CEP-CM-Systems wird auf Basis eines Vergleichs mit einem herkömmlichen CM-Expertensystem evaluiert. Die Reaktionszeit ist die Zeit zwischen dem Datenempfang und der Datenauswertung des Systems. Das herkömmliche Expertensystem speichert die Sensordaten zuerst in einer MS-SQL-Datenbank und führt anschließend die Störungsdetektion mithilfe der Verwendung von Entscheidungsregeln aus dem Entscheidungsbaum

durch. Dabei wird die Reaktionszeit des herkömmlichen Systems in Zusammenhang mit der Datenbankgröße evaluiert. Die Szenarien werden auf Basis des gespeicherten Datenvolumens in der Datenbank definiert. Wenn jeder Sensor einen Wert pro Sekunde sendet, ergibt das für alle fünf Sensoren 432.000 Sensordaten pro Tag. Für die Szenarien mit mehreren Millionen Datensätzen werden zuerst zufällige Sensordaten generiert und in einer Datenbank gespeichert. Anschließend erfolgt das Senden von realen Sensordaten an das System, um die realen Störungen zu detektieren. Es werden insgesamt sechs Szenarien berücksichtigt (Tabelle 9). Die Reaktionszeit wird für jedes Szenario und für die acht Leckage bedingten Druckverluststörungen aus den Validierungsdaten von Tabelle 5 gemessen.

Tabelle 9: Definition der Evaluierungsszenarien

Szenario	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
Anzahl der gespeicherten Sensordaten (in Millionen)	1	3	4	6	28	55

Die Abbildung 49 veranschaulicht die durchschnittliche Reaktionszeit des CEP-CM-Systems und des konventionellen CM-Systems bei der Erkennung der Leckage als Ursache für den Druckverlust. Aus der Abbildung ist ersichtlich, dass die durchschnittliche Detektionsdauer des konventionellen Ansatzes mit dem in der Datenbank gespeicherten Datenvolumen zunimmt. Diese Detektionszeit bleibt bei dem CEP-Ansatz jedoch konstant. Das liegt daran, dass die Datenvorverarbeitung im konventionellen Ansatz zeitaufwendig ist und dementsprechend mehr Zeit für die Datenbankanfragen benötigt. Diese Datenbankanfragen entfallen im CEP-System, da die Daten direkt, ohne die Notwendigkeit von Datenbankanfragen, verarbeitet werden können.

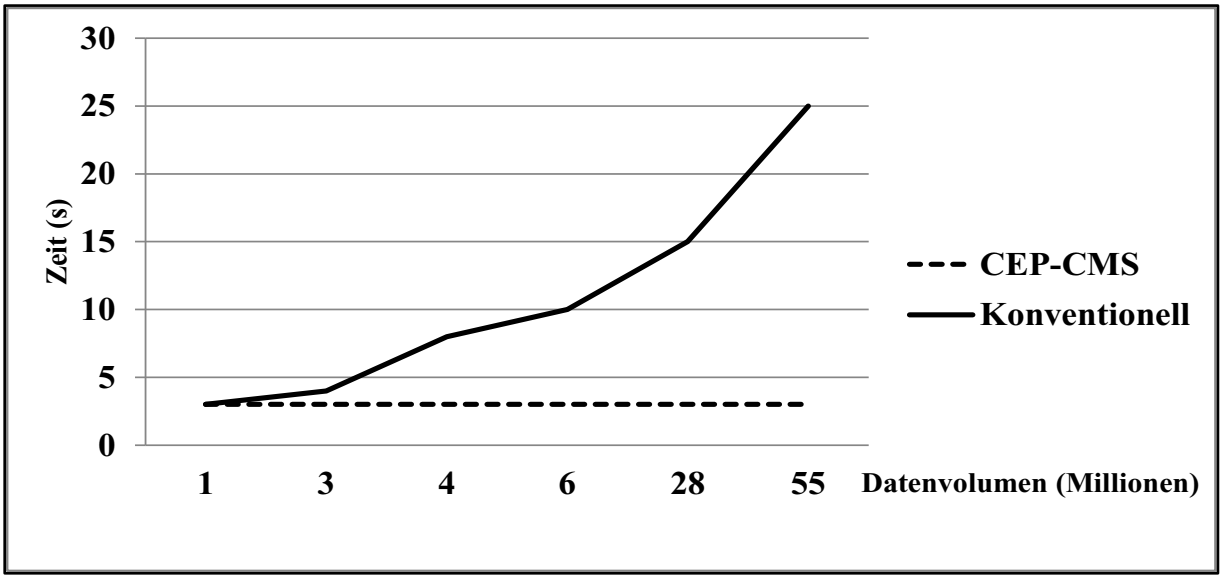

Abbildung 49: Vergleich der durchschnittlichen Reaktionszeit zwischen dem CEP- und dem konventionellen Ansatz.

6.3. Szenario 2: Vereisung der Rotorblätter von Windkraftanlagen

Der einsetzende Klimawandel hat das öffentliche Bewusstsein für den Austausch von fossilen Energieträgern durch erneuerbare Energien geweckt. Basierend auf den höheren Windgeschwindigkeiten und den konstanteren Windbedingungen wird dem Ausbau der Windenergieanlagen (WEA) ein hohes Potenzial beigemessen (Ait Alla et al. 2020). Darüber hinaus befinden sich viele WEA in kalten Regionen, die höhere Windgeschwindigkeiten und dementsprechend hohes Potential für die Energiegewinnung besitzen (Tabatabaei 2018). Jedoch reduziert die Eisbildung an Windenergieanlagen in kalten Regionen wie Nordeuropa oder Nordamerika nicht nur die Energieproduktion des Systems, sondern führt auch zu Ausfällen von Windenergieanlagen und stellt zudem ein Risiko für sich in der Nähe befindliche Fahrzeuge und Menschen dar.

6.3.1. Beschreibung des Vereisungsproblems von Windkraftanlagen

Da der Anteil der Windenergie an der Gesamtstromerzeugung weltweit zunimmt (Oelker et al. 2018), wächst auch die Nachfrage danach, den wirtschaftlichen Wirkungsgrad der Windturbine zu erhöhen und damit die Ausfallzeiten so weit wie möglich zu reduzieren. Eine wesentliche Ursache für Ausfallzeiten in kälteren Regionen wie Nordeuropa und Nordamerika ist die Vereisung der Rotorblätter der Windkraftanlage (Kreutz et al. 2019), (Botta et al. 1998). Das erhöhte Gewicht auf den Rotorblättern und deren veränderte Gewichtsverteilung während der Vereisung führen zu einer hohen Belastung der Gesamtstruktur, weswegen kein sicherer Betrieb der Turbine gewährleistet und somit eine Ausfallzeit erzwungen ist (siehe Abbildung 50). Dies führt zum Verlust von potenziell erzeugtem Strom (Hochart et al. 2008). Die Vereisung ist daher ein relevanter Faktor für die Wirtschaftlichkeit von Windenergieanlagen. Die Ausfallzeiten aufgrund von Vereisung können bei einigen Anlagen bis zu einem Monat pro Jahr betragen, was ein erhebliches Problem für den Anlagenbetrieb darstellt (Kreutz et al. 2019). Die rauen Wetterbedingungen führen zudem zu deutlichen Verzögerungen in den Instandhaltungsprozessen und tragen zu erheblich erhöhten Kosten bei der Wartung und Instandhaltung der Windenergieanlagen bei (Ait Alla et al. 2020).

Abbildung 50: Vereisung auf einem Rotorblatt © SPITZNER ENGINEERS GmbH[4]

Neben wirtschaftlichen Verlusten kann Eisabwurf von den Rotorblättern nicht nur die Rotorblätter und die Windturbine selbst treffen und beschädigen, sondern auch ein Risiko für sich in der Nähe befindliche Menschen, Fahrzeuge oder Tiere darstellen (Laakso et al. 2010). Das erklärt auch, warum die Windturbine nach der Eisbildung auf den Rotorblättern angehalten werden muss. In einigen Regionen legt das Gesetz die erforderlichen Sicherheitsmaßnahmen bei Eiswurf ausdrücklich fest, wie z. B. Mindestabstände zu nahe gelegenen Straßen und Gebäuden sowie Vorschriften für Vereisungs- und Enteisungssysteme an der Turbine selbst (Ministerium IL-NWF 2018). Um das Vereisungsproblem zu lösen, gibt es zwei Ansätze. Während der Enteisungsansatz (engl. de-icing) versucht, das Eis zu entfernen, nachdem es bereits auf den Rotorblättern gebildet wurde, versucht der Anti-Icing-Ansatz, das Auftreten des Eises zu antizipieren und vorbeugende Maßnahmen zu ergreifen, um die Bildung des Eises zu verhindern (Fakorede et al. 2016), (Parent et al. 2011). Deswegen leistet die Eiserkennung bzw. eine frühere Eisdetektion einen großen Beitrag zur Reduzierung von Vereisungsproblemen. In diesem Zusammenhang wird in dem vorliegenden Anwendungsszenario das Ziel verfolgt, ein Anti-Icing-System mithilfe des Einsatzes von CEP zu entwickeln, um die Eisbildung an Rotorblättern von Windenergieanlagen zu detektieren bzw. vorherzusagen. Hierbei wird ein echtzeitfähiges CM-System basierend auf dem CEP-Ansatz zur Erkennung von Vereisungsereignissen an Windenergieanlagen entwickelt. Das CEP-CM-System verwendet die his-

[4] https://spitzner-engineers.de/

torischen SCADA-Daten, Wetterdaten und Wartungsdaten mehrerer Windenergieanlagen als Systeminput, um die Vereisungsereignisse in Echtzeit frühzeitig zu detektieren.

6.3.2. Situationsanalyse – Datenspezifikation

Die in dieser Arbeit verwendeten Daten bestehen aus Daten von SCADA-Systemen verschiedener Windenergieanlagen, die über einen Zeitraum von vier Jahren aufgenommen worden sind. Die Daten werden in zwei Gruppen unterteilt: Statusdaten, welche die Eisalarminformationen enthalten, und Betriebsdaten, die verschiedene Parameter zu den Windenergieanlagen enthalten, wie z. B. die erzeugte Leistung oder die Außentemperatur. Die Eisalarme unterscheiden sich von Hersteller zu Hersteller. Bei den Betriebsdaten werden alle 10 Minuten neue Daten versendet und im SCADA-System erfasst. Eine Windkraftanlage, die den ganzen Tag in Betrieb ist, sendet daher (6 * 24 = 144) Datensätze pro Tag. Das ergibt für mehr als 1.000 WEAs über 144.000 Datensätze pro Tag und über 210 Millionen Datensätze in vier Jahren, die erfasst und verarbeitet werden. Die Anzahl der täglichen Datensätze kann erheblich größer sein, wenn auch die Anzahl der WEAs noch größer wird.

Die Anzahl der täglichen Datensätze scheint nicht sehr hoch zu sein, aber das CEP-CM-System ist für die Skalierung auf hundert oder tausend Windenergieanlagen ausgelegt, was bedeutet, dass ein beträchtliches Datenvolumen verarbeitet werden muss. Es gibt über 50 Parameter für die Betriebsdaten, aber nicht all diese Parameter sind für die prädiktive Eiserkennung relevant. Daher wurde eine Korrelationsanalyse durchgeführt, um die relevantesten Parameter zu identifizieren. Auf diese Weise wurde die Liste der Parameter auf sieben reduziert, da diese Variablen über 90 % der kumulierten Kovarianz darstellten (Abbildung 51). Die Windgeschwindigkeit, Gondeltemperatur, Generatortemperatur, Getriebetemperatur (engl. gearbox temperature), Gondelposition, Außentemperatur und Rotordrehzahl sind die wichtigsten Parameter des SCADA-Systems für die Eiserkennung. Da weniger Variablen verarbeitet werden müssen, reduziert sich der Zeitaufwand für die Erstellung der prädiktiven Modelle erheblich. Ein Eis-Alarm startet im SCADA-System immer mit einem ON-Status und endet mit einem OFF-Status. Die Zeit zwischen den beiden Status definiert die Eisalarmdauer.

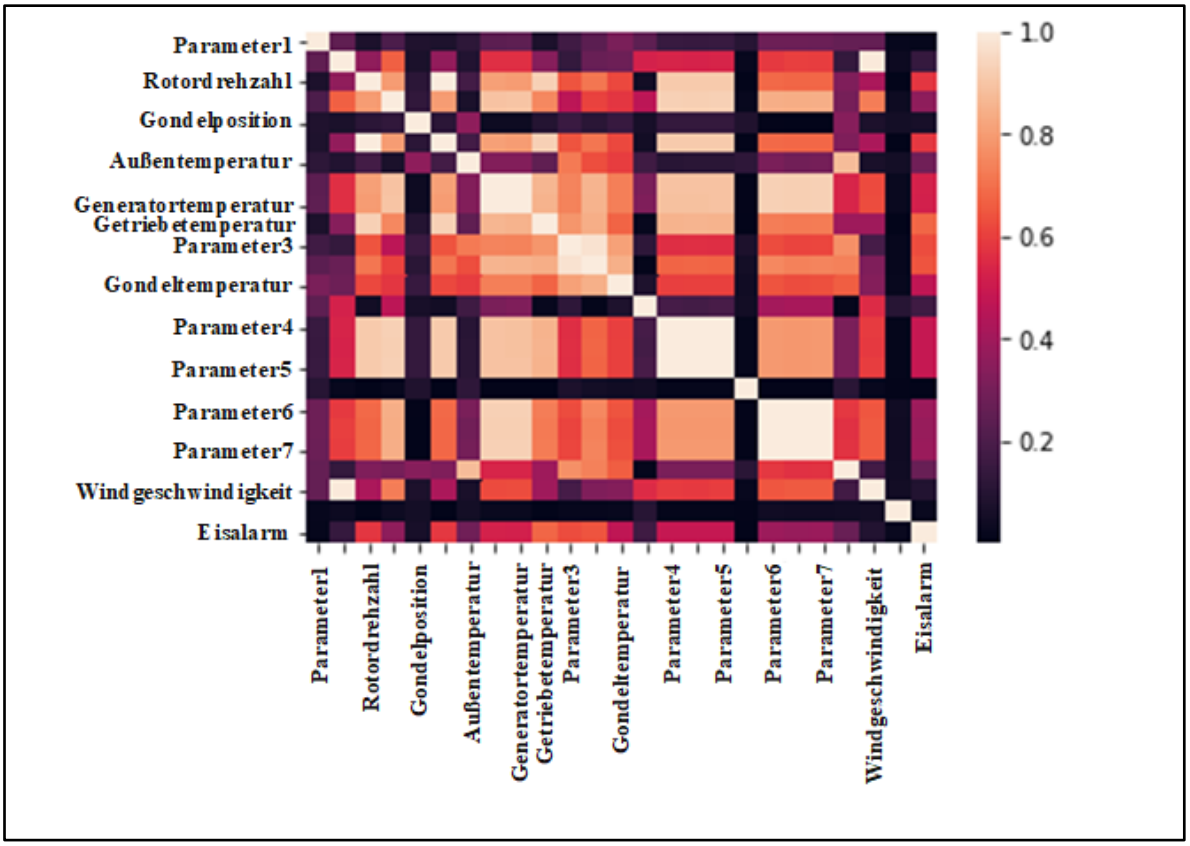

Abbildung 51: Ergebnisse der Korrelationsanalyse

Wie in Tabelle 10 ersichtlich, beinhalten die SCADA-Daten fehlerhafte Werte, die in einem ersten Schritt herausgefiltert und bereinigt werden müssen. Neben fehlerhaften Daten enthält das SCADA-System Eisalarme, bei denen die Außentemperatur während des Alarms im Schnitt 6 bis 7 Grad beträgt. Das weist voraussichtlich auf einen falschen Eisalarm hin, der in die Trainingsdaten nicht miteinbezogen werden soll. In diesem Kontext wurden graphisch die SCADA-Daten und die Eisalarmdauer bei falschen Alarmen untersucht. Anschließend wurde dann die Vermutung bestätigt, dass Eisalarme mit einer Dauer von weniger als fünf Minuten häufig falsche Alarme sind, welche in den Trainingsdaten nicht berücksichtigt werden sollen.

Abbildung 52 zeigt eine Verteilung der Vereisungsalarmdauern, die aus SCADA-Daten für mehrere Windparks erstellt wurde. Wie sich herausgestellt hat, können bei einigen Windturbinen sogar einzelne Vereisungsalarme bis zu einem Monat andauern. Dies ist in der Abbildung für eine isolierte Gruppe von Eisalarmzeiten zu sehen, deren Dauer ca. 650 Stunden (ca. ein Monat) beträgt. Abgesehen von der isolierten Gruppe betragen die längsten Alarme etwa 300 bis 400 Stunden, d. h. etwa einen halben Monat.

Tabelle 10: Gültigkeitsbereiche und tatsächliche Wertebereiche der ausgewählten Parameter

Parameter	**Min. Ist**	**Max. Ist**	**Mittelwert**	**Min. Soll**	**Max. Soll**	**Einheit**
Windgeschwindigkeit	-1,88	76,2	5,9	0	40	m/s
Gondeltemperatur	-10,9	17848	21,5	-50	100	°C
Generatortemperatur	-7,46	21018	39	-50	200	°C
Getriebetemperatur	-3,95	17849	59,97	-50	200	°C
Gondelposition	0	360	193,46	0	360	°
Rotordrehzahl	0	13,49	6,69	0	50	1/min
Außentemperatur	-15	17284	11,22	-50	100	°C

Diese große Bandbreite an Vereisungsalarmdauern, die von wenigen Stunden bis fast einem Monat reichen, liegt im Rahmen der Erwartungen, da die Daten aus verschiedenen Windparks mit unterschiedlichen Umgebungsbedingungen stammen.

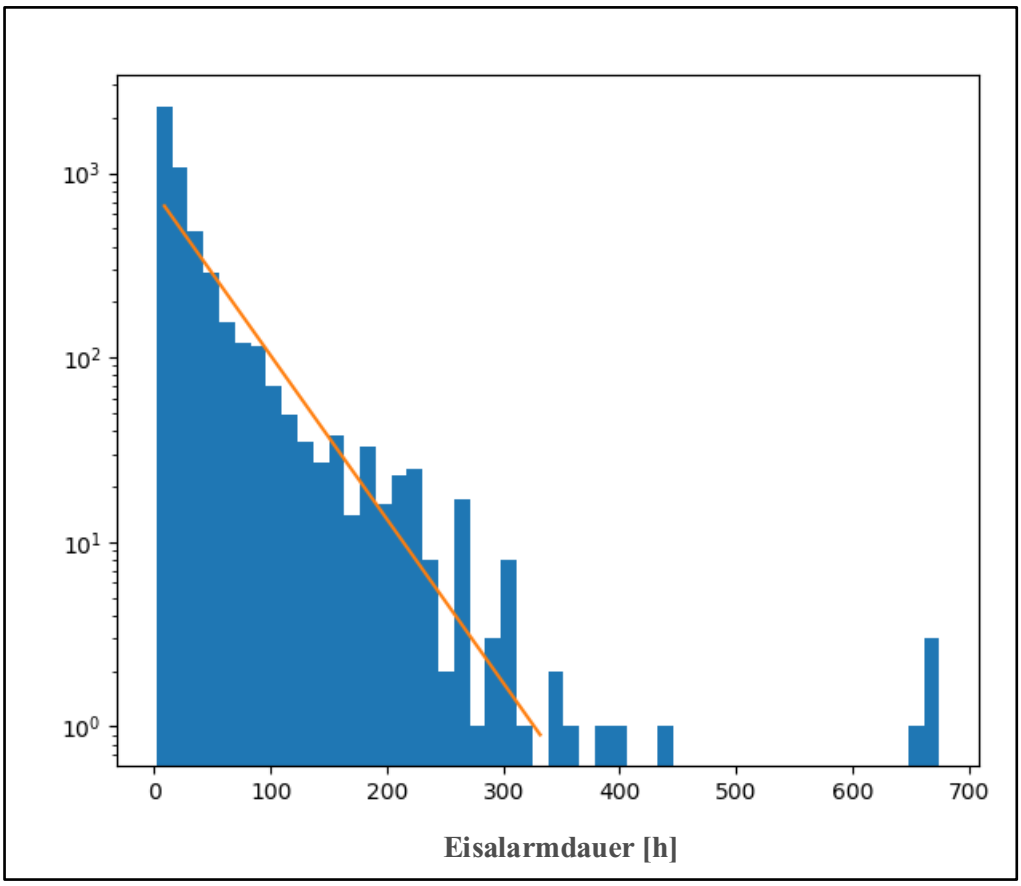

Abbildung 52: Histogramm der Vereisungsalarmzeiten aus SCADA-Daten mehrerer Windparks

Selbst innerhalb desselben Windparks variieren die Vereisungsalarmdauern teilweise stark, wie Abbildung 53 zeigt. Die Eisalarmdauerverteilungen von verschiedenen Windenergieanlagen desselben Windparks werden hier im Vergleich dargestellt und es ist zu sehen, dass die Eisalarmdauer bei einigen Windturbinen

auch unter gleichen klimatischen Bedingungen doppelt so lang sein kann wie bei anderen Windturbinen. Windenergieanlagen desselben Windparks können demnach ein unterschiedliches Vereisungsverhalten aufweisen. Die Ursache dafür könnte einfach in statistischen Schwankungen und Fluktuationen liegen, aber es besteht die Vermutung, dass das Alter und der Zustand der Windturbinenkomponenten wichtige Faktoren sind (Kreutz et al. 2019).

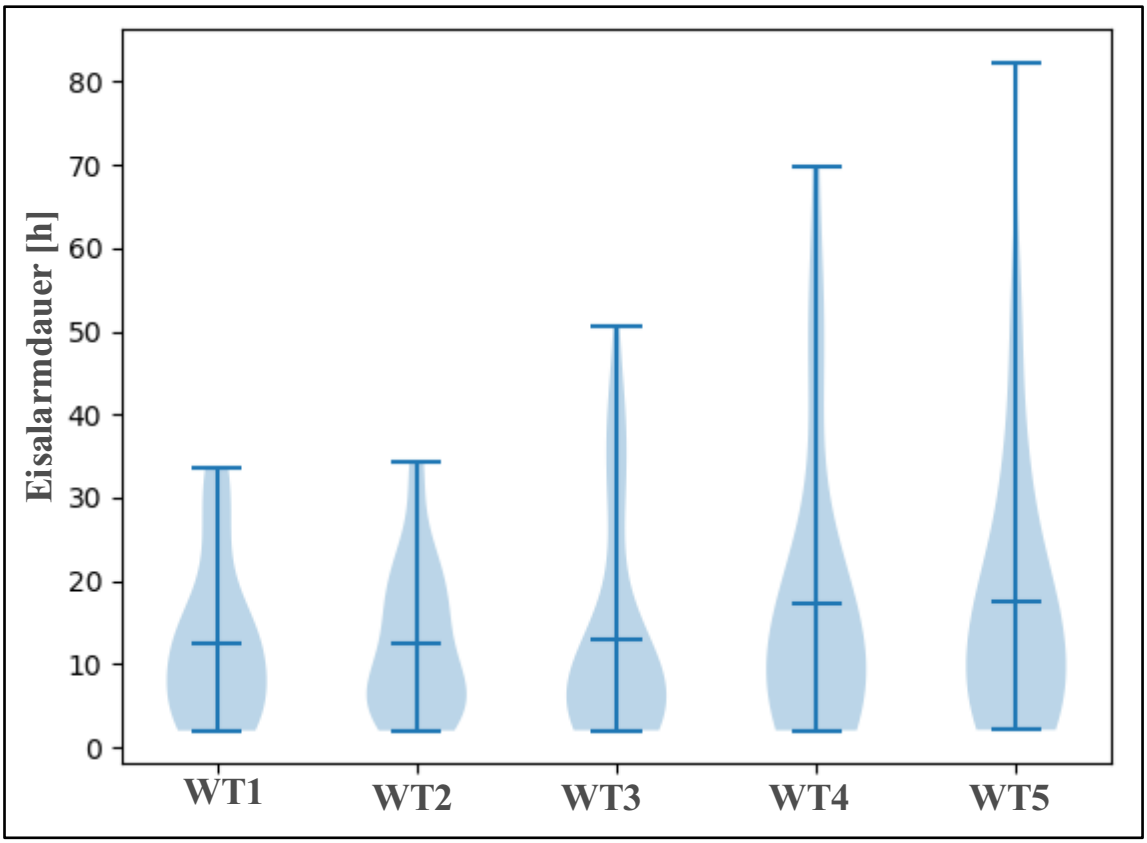

Abbildung 53: Variation der Eisalarmdauer innerhalb desselben Windparks

6.3.3. Erstellung einer ersten Version des Ereignismodells

Ausgehend von der System- und Situationsanalyse wurden die relevanten Parameter für die Eisalarmdetektion identifiziert. Darauf aufbauend wird ein erstes Ereignismodell erstellt. Dies umfasst die Integration bzw. Transformation der ausgewählten Parameterinputs als Basisereignisse. Anders als im ersten Anwendungsfall werden die Livedaten aus technischen Gründen nicht direkt von den Sensoren übertragen, da kein direkter Zugriff auf die Sensoren besteht. Deswegen werden die Livedaten direkt aus dem SCADA-System eingelesen. Die Filterung und Bereinigung von Redundanzen in den Daten bzw. Duplikaten entfällt, da die Daten im SCADA-System bereits ohne Duplikate vorliegen. Für den Modellaufbau werden daher die historischen Daten aus dem SCADA-System verwendet.

Für das Ereignismodell wird alle 10 Minuten ein Ereignis „***Parameterinputereignis***“ erstellt. Dieses Ereignis wird durch den Auftrittszeitpunkt *(Zeitstem-*

pel), die Werte aller sieben Parameter und den Anlagenamen repräsentiert. Die Definition des Parameterinputstroms im CEP-System lautet wie folgt:

Define stream ***parameterinputstrom****(timestamp long, windgeschwindigkeit double, gondeltemperatur double, generatortemperatur double, getriebetemperature double, gondelposition double, rotordrehzahl double, anlagename string);*

Da für die Außentemperatur die zweistündigen Vergangenheitswerte mit einstündigen Vorhersagewerten zu einem einzigen Ereignis „***aussentemperaturereignis***“ kombiniert werden, werden ein ***pastaussentemperaturereignis*** und ein ***prognoseaussentemperaturereignis*** erstellt.

Das ***pastaussentemperaturstrom*** gibt für einen Zeitpunkt in der Vergangenheit den Wert der Außentemperatur aus.

Define stream ***pastaussentemperaturstrom*** *(timestamp long, pastaussentemperatur double, anlagename string);*

Das ***prognoseaussentemperaturereignis*** stellt die Vorhersage der Außentemperatur für die nächste Stunde dar, daher besitzt das Ereignis neben dem Zeitstempel der Prognose sechs andere Attribute für die Vorhersage der Außentemperatur (eine Vorhersage für jeden 10-Minuten-Schritt in der Zukunft). Der ***prognoseaussentemperaturstrom*** ist wie folgt definiert:

Define stream ***prognoseaussentemperaturstrom*** *(timestamp long, aussentemperatur1 double, aussentemperatur2 double, aussentemperatur3 double, aussentemperatur4 double, aussentemperatur5 double, aussentemperatur6 double, anlagename string);*

Das ***aussentemperaturereignis*** bildet den Strom, der die Ergebnisse der Kombination vom ***pastaussentemperaturstrom*** und ***prognoseaussentemperaturstrom*** darstellt.

Define stream ***aussentemperaturstrom*** *(timestamp long, aussentemperatur double, anlagename string);*

Anschließend erfolgen die Berechnungen der arithmetischen Funktionen. Durch die Verwendung von maschinellen Lernverfahren werden die Vereisungssituationen vorhergesagt. Daher wird ein Vereisungsereignis als komplexes Ereignis definiert, das neben dem Zeitstempel noch den Namen der betroffenen Anlage enthält.

Define stream ***vereisungsstrom*** *(timestamp long, anlagename string);*

Zusammenfassend besteht das initiale Ereignismodell aus fünf Ereignissen. Im nächsten Abschnitt werden andere Ereignisse definiert, welche die verschiede-

nen Phasen der Offline-Datenanalyse und deren Verarbeitung zur Erkennung von komplexen Ereignissen (Vereisungsereignisse) darstellen.

6.3.4. Konfigurationsmethodik zur Umsetzung des CEP-Systems für das zweite Szenario

6.3.4.1. Offline-Datenanalyse / Aufbau des ML-Modells

Bevor mit der Entwicklung von ML-Modellen begonnen wird, müssen zunächst allgemeine Informationen über die Vereisung aus historischen Daten der SCADA-Systeme extrahiert werden. Die historischen Daten müssen ebenso von Ausreißern und falschen Vereisungssituationen bereinigt werden, da die Genauigkeiten der an Windenergieanlagen installierten Eiserkennungssensoren sehr unterschiedlich in der Qualität und bei weitem nicht perfekt sind (Parent et al. 2011), (Kreutz et al. 2019). In diesem Zusammenhang werden alle Vereisungsalarme mit einer Dauer von weniger als fünf Minuten aus den Trainingsdaten entfernt. Darüber hinaus werden auch jene Vereisungsalarme, die bei hohen Temperaturen auftraten, als falsche Alarme eingestuft und aus dem Datensatz herausgefiltert. Im nächsten Schritt wird ein datengesteuertes, überwachtes Lernmodell zur Vorhersage von Vereisungsalarmen auf der Grundlage historischer SCADA-Daten und vorhergesagter externer Außentemperatur in 100 Meter Höhe entwickelt. Die für das Modell verwendeten Daten wurden aus dem SCADA-System eines Windparks in Deutschland (bestehend aus mehr als zehn Windenergieanlagen) im 10-Minuten-Takt von August 2016 bis September 2018 gesammelt. In diesem Zeitraum sind verschiedene Vereisungsalarme von den installierten Vereisungssensoren in den Windenergieanlagen aufgezeichnet worden, die, wie bereits erwähnt, gefiltert wurden. Anschließend werden Trainingsdatensätze generiert. Danach erfolgt der Aufbau des Modells mit dem Ziel, die vorzeitige Identifizierung der Vereisungssituation für eine Stunde in der Zukunft vorherzusagen. Bezüglich der Generierung der Trainingsdaten werden Informationen aus zwei Zeitfenstern für die Generierung der Trainingsdaten verwendet: Das erste Zeitfenster umfasst 3 Stunden aus der Vergangenheit mit den in

Tabelle 11 aufgeführten Eingabeparametern.

Das zweite Zeitfenster erstreckt sich von den zwei zurückliegenden Stunden bis zu einer Stunde in die Zukunft und wird nur für den Außentemperaturparameter verwendet (siehe Abbildung 54). Die Außentemperaturwerte in der Zukunft für dieses Fenster sind Prognosewerte.

Tabelle 11: Ein- und Ausgabeparameter des Modells

Eingabeparameter (Zeitfenster der letzten 3 Stunden)	Windgeschwindigkeit Gondeltemperatur Generatortemperatur Getriebetemperatur Gondelposition Rotordrehzahl
Eingabeparameter (Zeitfenster aus den letzten 2 Stunden und zukünftigen 1 Stunde)	Außentemperatur (100 Meter Höhe)
Arithmetische Funktionen	Min Max Average Standardabweichung
Modellausgabe	Vereisungsereignis: Boolesch

Für jeden Eingabeparameter und beide Zeitfenster werden aus den 18 Werten des 3-Stunden-Fensters (3 Stunden × 6 Werte pro Stunde) durch Anwendung arithmetischen Funktionen (Max, Min, Average, Standardabweichung und Steigung) vier Werte erzeugt.

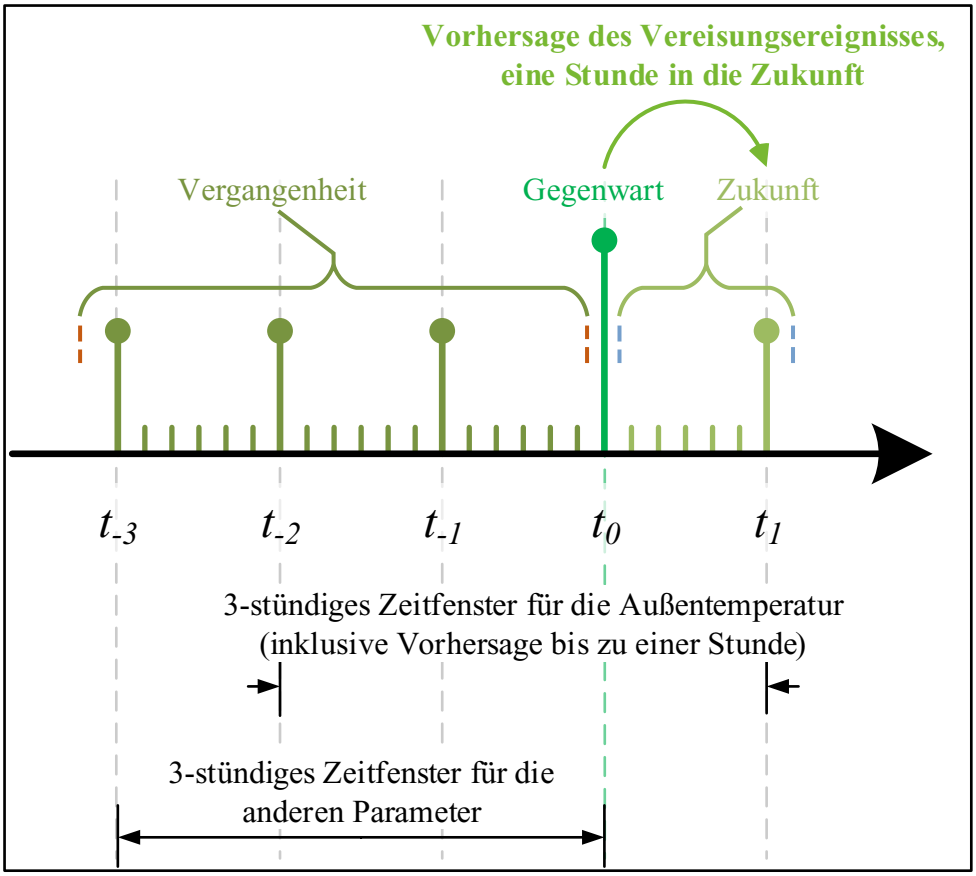

Abbildung 54: Skizze des Zeitfensters der verwendeten Parameter für das Modell

Auf diese Weise wurden für jeden 10-minütigen Zeitpunkt insgesamt 28 Werte (7 Eingabeparameter × 4 Funktionen) als Trainingsdatensatz und Input für das Modell generiert. Für jeden Trainingsdatensatz wird die Vereisungssituation für eine Stunde in die Zukunft als boolescher Wert zugeordnet, der angibt, ob eine Vereisung aufgetreten ist. Insgesamt wurden 950 Datensätze erfasst, wobei 710 Datensätze mit 400 Vereisungsereignissen und 310 No-Icing-Ereignisbespielen für das Modelltraining verwendet werden. Die restlichen 240 Datensätze mit 146 Vereisungsereignissen und 94 No-Icing-Ereignissen werden für die Modellvalidierung verwendet (siehe Tabelle 12). Im Anhang C befindet sich ein Beispiel der Trainingsdaten.

Tabelle 12: Gesamter Überblick über die Anzahl der Trainings- und Validierungsdaten

	Vereisung	**Keine Vereisung**	**Gesamte Anzahl**
Trainingsdaten	400	310	710
Validierungsdaten	146	94	240

Um dem Vereisungsproblem auf den Grund zu gehen, wurde ein neuronales Netzwerk entwickelt. Die Topologie des neuronalen Netzwerks ist wie folgt: Das NN besteht aus 28 Eingangsneuronen (7 Parameter × 4 arithmetische Funktionen), einer versteckten Schicht mit 20 Neuronen und 2 Ausgangsneuronen (ja, nein), welche die Vereisungssituation darstellen. Für die Aktivierungsfunktion wurde die Sigmoid-Funktion verwendet.

Für die Validierung des NNs wird abschließend die Genauigkeit der Modellausgaben, wie in Abschnitt 5.1.1 beschrieben, berechnet. Tabelle 13 veranschaulicht die Validierungsergebnisse des entwickelten NNs.

Tabelle 13: Validierungsergebnisse der Vereisungsvorhersage des NNs

	Anzahl der Beispiele	**Anzahl der korrekten vorhergesagten Ereignisse**	**Modellgenauigkeit (Accuracy) (%)**
Vereisungssituation	146	129 (TP)	88,38
Keine Vereisungssituation	94	72 (TN)	76,60
Total	240	201	83,75

Das entwickelte neuronale Netz weist eine gute Genauigkeit für die Vorhersage von Vereisungssituationen auf. Der Prozentsatz der korrekten Vorhersagen ist für die Vereisungsereignisse etwas höher als für die No-Icing-Ereignisse, aber beide Werte unterscheiden sich nicht wesentlich voneinander. Die Gründe hierfür können beispielsweise durch eine schlechte Filterung der Vereisungsereignisse während der Erhebung von Trainingsdaten oder durch die Ungenauigkeit der ange-

bauten Eissensoren an den Windenergieanlagen verursacht werden. Ein weiterer Grund könnte darin liegen, dass die Menge der Trainingsdaten nicht ausreichend ist und das neuronale Netz mehr Daten von mehreren Wintern für das Training benötigt. Darüber hinaus kann die Modellgenauigkeit bzw. Modellvorhersage durch Berücksichtigung weiterer Parameter wie Luftfeuchtigkeit oder Flüssigwassergehalt (engl. liquid water content), die für diese Arbeit nicht verfügbar waren, noch verbessert werden.

6.3.4.2. Regelextraktion aus der Datenaufbereitungsphase

Wie bereits erwähnt, werden die Daten aus dem SCADA-System zu Basisereignissen transformiert. Dabei werden diese Ereignisse mit der einstündigen Außentemperaturvorhersage in einen einzigen Ereignisstrom „Außentemperaturstrom“ zusammengeführt. Danach erfolgt die Berechnung der arithmetischen Funktionen. Das ergibt ein neues Ereignis des temporalen Aggregattyps, das den Aggregatstrom bildet. Der Aggregatstrom ist wie folgt definiert:

*define stream **aggregatstream**(timestamp long, min_wg double, max_wg double, avg_wg double, stdDev_wg double, min_gondel_T double, max_gondel_T double, avg_gondel_T double, stdDev_gondel_T double, min_genT double, max_genT double, avg_genT double, stdDev_genT double, min_getriebe_T double, max_getriebe_T double, avg_getriebe_T double, stdDev_getriebe_T double, min_gondel_P double, max_gondel_P double, avg_gondel_P double, stdDev_gondel_P double, min_rotordrehzahl double, max_rotordrehzahl double, avg_rotordrehzahl double, stdDev_rotordrehzahl double, min_aussenT double, max_aussenT double, avg_aussenT double, stdDev_aussenT double, anlagename string);*

Der folgende Pseudocode definiert die Ereignisregel, die das Vorgehen zur Generierung von Aggregatereignissen bzw. Aggregatstrom erläutert.

Pseudocode 6: Berechnung von arithmetischen Funktionen und Generierung von Aggregatereignissen und deren Einspeisung am Aggregatstrom.

Input: parameterinputstrom, aussentemperaturstrom
Output: aggregatstrom
Durchführung einer Verknüpfung (engl. join) aus den beiden Strömen über den
    Zeitstempel und Anlagename
    Erstelle ein 3-stündiges Zeitfenster (engl. sliding window)
    Für jedes Attribut A des Parameterinputstroms (Geschwindigkeit, …, Gondelposition, Rotordrehzahl) wiederhole
        Berechne Min, Max, Average und Standardabweichung des Attributs A der letzten Stunden
    Wiederhole_end

Für das Attribut aussentemperatur des Aussentemperaturstroms, Berechne Min, Max, Average und Standardabweichung der letzten Stunden
select timestamp, minimum (geschwindigkeit) as min_ *wg,* maximum (geschwindigkeit) as max_ *wg,* avg(windgeschwindigkeit) as avg_wg, stdDev(windgeschwindigkeit) as stdDev_wg,……
…… minimum(aussentemperatur) as min_aussenT, maximum(aussentemperatur) as max_aussenT, avg(aussentemperatur) as avg_aussenT, stdDev(aussentemperatur) as stdDev_aussenT
insert into aggregatstrom;

Ende

6.3.4.3. Regelextraktion aus der Lernmodellaufbauphase

Um die Vereisungssituationen vorherzusagen, wurde wie in Abschnitt 6.3.4.1 ein neuronales Netz entwickelt. Die in dieser Arbeit eingesetzte CEP-Engine unterstützt den Einsatz von bereits erstellten und trainierten maschinellen Lernmodellen wie neuronale Netze in der Formulierung der Ereignisregeln durch Tensorflow-Erweiterungen (engl. extension). Dabei sollen die ML-Modelle im Voraus erstellt und trainiert werden und dann im SaveModel-Format gespeichert werden. Die Regelsyntax für den Einsatz von Tensorflow-Erweiterungen für die Vorhersage ist wie folgt definiert:

from Aggregatstrom#tensorFlow:predict(Pfad zum Modell, Eingangsnamen im Modell, Ausgangsnamen im Modell, Attributwerte).

Die Eingangsnamen sind die Variablennamen, die bei der Erstellung des neuronalen Netzes verwendet werden. In diesem Beispiel sind die 'min_wg', 'max_wg', …. , 'avg_aussenT', 'stdDev_aussenT' die Variablennamen des in Tensorflow entwickelten NNs. Zur Vereinfachung wurden im CEP-System dieselben Variablenakronyme verwendet. Ähnlich wie bei den Eingangsnamen stellen die Ausgangsnamen die Variablennamen für das in Tensorflow entwickelte NN dar. Hierfür wird „vereisungvorhersage“ als Variablenname definiert. Schließlich sind die Attributwerte die Werte der Eingangsvariablen.

Wenn die Eingangs- und Ausgangsnamen aus mehreren Variablen bestehen, werden diese durch ein Komma getrennt und müssen aus primitiven Datentypen in Java bestehen.

Für diesen Anwendungsfall kommen die Modelleingaben aus dem Aggregatstrom. Die Modellausgabe stellt neben den Zeitstempel die Vereisungsvorhersage des Modells dar. Aus diesem Grund wird ein Ereignis von Typ maschinelles Lernereignis „***NNvorhersageereignis***“ mit dem Zeitstempel und der Vereisungsvorhersage als Attribute erstellt. Darauf aufbauend wird die Vereisungssituation hergeleitet, wenn die Vereisungsvorhersage True ist.

*Define stream **NNvorhersageStrom** (timestamp long, vereisungsvorhersage bool, anlagename string);*

*Define stream **Vereisungsstituation** (timestamp long, anlagename string);*

Der folgende Pseudocode definiert die Ereignisregel, die für die Vorhersage von Vereisungen und Herleitung von komplexen Ereignissen zuständig ist.

Pseudocode 7: Vorhersage von Vereisungen und Herleitung von komplexen Ereignissen.

```
Input: aggregatstrom
Output: vereisungstrom
Für jedes Ereignis e1 in aggregatstrom wiederhole
  Berechne die Modellausgabe aus den aggregierten Werten des Ereignisses e1
        from aggregatstream#tensorFlow:predict(Pfad zum Modell, 'min_wg',
        'max_wg', ….. , 'avg_aussenT', 'stdDev_aussenT',   'vereisungvorhersage',
        e1.min_wg, e1.max_wg, …., e1.avg_aussenT, e1.stdDev_aussenT)
        select e1.timestamp, vereisungvorhersage, e1.anlagename
        insert into NNvorhersagestrom;
Wiederhole_end
Für jedes Ereignis e1 in NNvorhersagestrom wiederhole
        Wenn e1.vereisungvorhersage == true dann
        select e1.timestamp, e1.anlagename
        insert into vereisungstrom
Wenn_end
Wiederhole_end
```

Abbildung 55 zeigt das daraus resultierende Ereignismodell.

Im Anhang D ist der gesamte Pseudocode zur Vereisungsvorhersage auf Basis des CEP-Ansatzes zu finden.

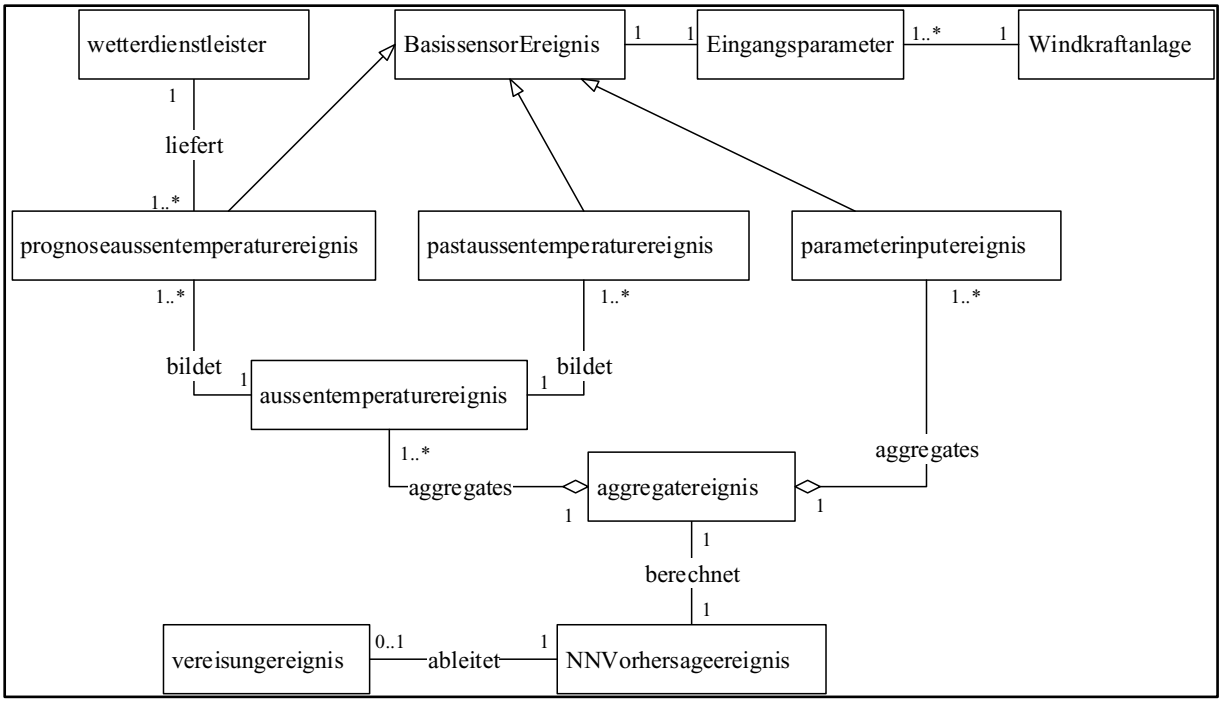

Abbildung 55: Ereignismodell für die Vereisungsvorhersage an Rotorblättern

6.3.5. Implementierung und Evaluation des entwickelten CEP-CM-Systems

Für die Implementierung und Umsetzung des CEP-CM-Systems für die Vereisungsvorhersage wurden Daten eines Windparks in Deutschland aus dem Winter 2017/2018 verwendet. In diesen Zeitraum fanden ca. 102 Vereisungen statt. Aufgrund von technischen Gründen war es nicht möglich, Livedaten direkt aus dem SCADA-System zu verwenden, deswegen wurde das Senden der Livedaten simuliert. Hierfür wurden die Daten für diesen Zeitraum aus dem SCADA-System in eine CSV-Datei extrahiert und das Senden der Daten aus dieser Datei als Livedaten simuliert. Da die Livedaten alle 10 Minuten versendet werden, würde eine reale Umsetzung bzw. Evaluierung des CEP-Systems für den betrachteten Zeitraum viele Monate dauern. Deswegen erfolgt das Versenden der Daten zum CEP-System pro Sekunde.

Für diesen Zeitraum konnte das CEP-CM-System 93 von 102 Vereisungen richtig vorhersagen, was 91 % aller Vereisungssituationen entspricht. Jedoch wurden auch für diesen Zeitraum zusätzlich 23 Vereisungsvorhersagen basierend auf falschen Klassifikationen (FN) ausgelöst. Dies ist vor allem auf das Fehlen von genügend Trainingsdaten und die Nichtberücksichtigung von mehreren externen Parametern wie Luftfeuchtigkeit oder Flüssigwassergehalt in den Trainingsdaten zurückzuführen.

Bezüglich der Reaktionszeit des CEP-CM-Systems wurde für den Vergleichszweck das Versenden der Livedaten für denselben Zeitraum mit demselben Intervall in das herkömmliche CM-System simuliert. Es wurden für beide Systeme die Reaktionszeiten für alle detektierten Vereisungen gemessen.

Eine Kopie der originalen historischen Datenbank wurde erstellt und für die Evaluierung verwendet, da das Speichern in der originalen Datenbank nicht erlaubt ist. Die Datenbank beinhaltet die Daten von mehreren Windparks, erhoben in einem Zeitraum von mehreren Jahren, was eine Datenbank von über 200 GB ergibt. Die MySQL-Datenbank ist auf einem leistungsfähigen Linux-Server mit einem AMD EPYC 7281 16-Core Processor und 256 GB Arbeitsspeicher installiert. Sofern ein neuer Datensatz empfangen wird, speichert das herkömmliche CM-System die Daten in der Datenbank und führt verschiedene Datenbankanfragen durch, um Sensordaten der letzten drei Stunden zu erhalten. Anschließend identifiziert das NN auf Basis der berechneten Aggregatfunktionen die Vereisungsvorhersage der nächsten Stunde. Die Abfrageoperationen aus der Datenbank sind zeitaufwändig und führen somit zu langen Reaktionszeiten, wie aus Tabelle 14 ersichtlich wird. Für die Messung der Reaktionszeiten der beiden Systeme wurden 200 Szenarien (102 Vereisungssituationen und 98 Nicht-Vereisungssituationen) ausgewählt.

Tabelle 14: Vergleich der Reaktionszeit bei der Vereisungsvorhersage

	Reaktionszeit		
	Min	**Max**	**Mittelwert**
CEP-CM-System	< 1 Sek.	< 2 Sek.	< 1 Sek.
Herkömmliches CM-System	31 Sek	55 Sek	48 Sek

Die Ergebnisse zeigen, dass die Reaktionszeiten bei dem herkömmlichen CM-System länger sind, was aber für diesen Anwendungsfall sehr akzeptabel ist, da es sich um einstündige Vereisungsvorhersagen handelt. Hierbei gilt zu beachten, dass in dieser Simulation ein leistungsfähiger Server eingesetzt wurde und dass nur das Speichern von Daten aus einem einzigen Windpark mit über zehn Anlagen simuliert wurde. Wenn diese Anzahl sich aber auf Tausende von Windenergieanlagen erhöht, wird erwartet, dass sich die Reaktionszeiten des herkömmlichen CM-Systems entsprechend erhöhen werden. Auf der anderen Seite ist die Reaktionszeit des CEP-CM-Systems konstant geblieben. Wenn das System jedoch die Daten aus mehreren Tausend Windturbinen verarbeiten soll, können durch die Skalierbarkeit des CEP-Ansatzes verschiedene CEP-Systeme auf günstigen Rechnern eingesetzt werden, um die Echtzeitfähigkeit des Systems unabhängig von der Anzahl der Windturbinen weiterhin zu gewährleisten.

6.4. Erweiterung und Anpassung des entwickelten CEP-CM-Systems

In den vorangegangenen Abschnitten des vorliegenden Kapitels wurde anhand zweier Beispiele die Evaluation des entwickelten CEP-Ansatzes durchgeführt. In diesem Abschnitt wird erklärt, wie das CEP-CM-System angepasst werden kann, wenn Änderungen in den Systemparametern vorgenommen oder neue

Sensoren hinzugefügt werden und dementsprechend bestehende Ereignisregeln nicht mehr ausgelöst werden können.

6.4.1. Struktur der Ereignisdatenbank

Wie schon im Kapitel 5 erwähnt, werden alle entstehenden Ereignisregeln in einer Datenbank gespeichert. Tabelle 15 zeigt ein Beispiel der Struktur der Datenbanktabelle der Ereignisregeln.

Tabelle 15: Struktur der Datenbanktabelle der Ereignisregeln

Name	**Ereignisregel**	**Inputstrom**	**Outputstrom**
Ereignis zur Filterung von	From temperaturstrom[temperatur > 40] Select timestamp, "high temperatur" as warning Insert into Meldungstrom	Temperaturstrom	Meldungstrom

Jede Ereignisregel hat einen Namen zur Aufgabenbeschreibung. Die Regel ist in Datenbank als Zeichenfolge und muss mit der Syntax von dem verwendeten CEP-Engine gespeichert. Der Grund hierfür ist, dass die in dieser Arbeit eingesetzte WSO2-Siddhi-CEP-Engine in ein Java-Projekt eingebettet werden kann. In diesem Fall können alle CEP-Ereignisregeln aus der Datenbank als Zeichenfolge ausgelesen und direkt in der Java-Anwendung ausgeführt werden. Die letzten zwei Spalten der Tabelle sind jeweils die Inputströme bzw. die Outputströme der Ereignisregel.

Neben der Ereignisregel-Datenbanktabelle wird eine Datenbanktabelle für die Definition der Ereignisströme erstellt (siehe Tabelle 16). Diese enthält die Information über die Ereignisse, die den Ereignisstrom bilden. Jeder definierte Ereignisstrom wird einem Datensatz in der DB zugeordnet.

Tabelle 16: Struktur der Datenbanktabelle der Ereignisströme

Name	Definition
Temperaturstrom	Define Temperaturstrom(timestamp double ,temperature double)

Hierbei gilt zu beachten, dass die Ereignisströme und Ereignisregeln in derselben Reihenfolge in der Datenbank gespeichert werden müssen, wie sie im CEP-System ausgeführt werden.

6.4.2. Anwendungsbeispiel für die Anpassung eines CEP-CM-Systems

Unter der Annahme, dass für das zweite Beispiel ein weiterer Parameter wie Luftfeuchtigkeit für die Vereisungsvorhersage berücksichtigt wird, wird der Datenbank ein neuer Ereignisstrom für eben diesen Parameter hinzugefügt. An-

schließend werden die arithmetischen Funktionen berechnet. Dementsprechend wird die Definition des Aggregatstroms mit den Attributen der arithmetischen Werte der Luftfeuchtigkeit in der Datenbank erweitert. Folglich wird die Ereignisregel aus dem Pseudocode 6 in der Datenbank angepasst. Schließlich soll das NN mit den neuen Parametern neu trainiert und validiert werden. Das neu erstellte NN wird im SaveModel-Format in dem vorgegebenen Ordner gespeichert. Die Ereignisregel aus dem Pseudocode 7 wird dementsprechend angepasst und ersetzt die alte Ereignisregel.

Wenn festgestellt wird, dass die Vereisungsvorhersagen ungenauer werden, wird das NN-Modell neu trainiert und validiert. Das neue, validierte NN soll im SaveModel-Format in dem vorgegebenen Ordner gespeichert werden. Nach dem Neustart des CEP-Systems kann automatisch das neue NN für die echtzeitfähige Vereisungsvorhersage verwendet werden.

Jede Änderung der Ereignisregeln bzw. der Ereignisströme in der DB hat keinen Einfluss auf den Quellcode des CEP-CM-Systems. Der Quellcode bleibt unverändert. Es muss lediglich das CEP-System neu gestartet werden, damit die Änderungen wirksam werden.

7. Fazit

In diesem Kapitel wird die in der Dissertation geleistete Arbeit zusammengefasst, die wichtigsten Forschungsergebnisse werden aufgezeigt und mögliche Themen für die zukünftige Forschung vorgeschlagen. Dementsprechend beginnt Abschnitt 7.1 mit einer Zusammenfassung der Dissertation und setzt sich mit den gewonnenen Ergebnissen und dem Beitrag dieser Forschung fort. Der Abschnitt 7.2 erklärt noch einmal die Antworten auf die Forschungsfragen. Abschließend werden in Abschnitt 7.3 die Grenzen der Forschung dargestellt und Vorschläge für zukünftige Arbeiten gemacht.

7.1. Zusammenfassung

Die Grundvoraussetzungen für die Umsetzung von CBM sind die Erarbeitung moderner IT-Technologien und die Entwicklung von Methoden zur Erfassung von Mess- und Betriebsdaten aus komplexen technischen Systemen sowie leistungsfähiger Zustandsüberwachungssysteme, die eine permanente Überwachung der erfassten Sensordaten ermöglichen. Mithilfe von Methoden des Data-Mining und maschineller Lernverfahren können die von Maschinen erfassten Daten verarbeitet, ausgewertet und daraus dann Informationen und Wissen über den Maschinezustand gewonnen werden. Allerdings stoßen die traditionellen CBM-Ansätze an ihre Grenzen, wenn es sich um die echtzeitfähige Verarbeitung von großen Datenmengen handelt. Deswegen sollen neue Ansätze entwickelt werden, welche eine echtzeitfähige Überwachung des Systems durch Verarbeitung großer Datenmengen ermöglichen. In diesem Zusammenhang bildet das CEP einen neuartigen Ansatz, der sich mit dem Problem auseinandersetzt, wie Ereignisse mehrerer und unterschiedlicher Quellen (Sensoren, Datenbanken, Logdateien etc.) korreliert und aggregiert werden können, um daraus in Echtzeit Schlussfolgerungen über physikalische Ereignisse, Aktivitäten oder Situationen ziehen zu können. Die Kernaufgabe von CEP besteht in der Verarbeitung von großen Mengen einströmender Daten mit einem hohen Maß an Heterogenität und der Entdeckung von Mustern in diesen in Form von komplexen Ereignissen. Insbesondere wurden Methoden für das Erkennen von Zusammenhängen, Abhängigkeiten und Korrelationen zwischen mehreren Ereignissen entwickelt.

In diesem Kontext war das Ziel dieser Arbeit die Entwicklung eines echtzeitfähigen, zustandsorientierten Instandhaltungskonzeptes durch die Integration von Methoden zur Verarbeitung komplexer Ereignisdaten. Dabei wurde untersucht, wie das CEP-Konzept auf das Zustandsüberwachungssystem übertragen werden kann, um komplexe technische Systeme besser instandhalten zu können. Für das Erreichen dieses Ziels wurde ein Gesamtkonzept, bestehend aus einem Referenzarchitekturmodell und einer Konfigurationsmethodik, entwickelt. Das Referenzarchitekturmodell definiert den systemischen Rahmen für die Übertragung der CEP auf CBM und die Konfigurationsmethodik zielt auf eine Systemkonfi-

guration durch automatische Anpassung der Ereignisregeln ab. Die Evaluierung des Gesamtkonzepts erfolgt in zwei Anwendungsfällen.

Das Ziel wurde anhand von zwei Forschungsfragen, dargestellt in sieben Kapiteln, untersucht.

In Kapitel 1 wird die Motivation, Problemstellung, Zielstellung, zu untersuchende Forschungsfragen und der Aufbau der Arbeit vorgestellt.

Kapitel 2 widmet sich den wissenschaftlichen Grundlagen der verschiedenen Instandhaltungsstrategien. Der Fokus wird dabei auf die zustandsorientierte Instandhaltung gelegt. In Kapitel 3 werden die Grundlagen des Data-Mining und des maschinellen Lernens erläutert. Vorgestellt wird ein allgemeiner Überblick über das Thema Data-Mining, ein kurzer Einblick in Data-Mining-Prozesse und Analytics und es gibt eine Einführung in verschiedene Methoden der maschinellen Lernverfahren, welche im Rahmen dieser Arbeit verwendet wurden. Als abschließendes Kapitel für die wissenschaftlichen Grundlagen befasst sich das Kapitel 4 mit der CEP-Grundlage.

Aufbauend auf dem Stand der Forschung werden in Kapitel 5 die Voraussetzungen und Anforderungen für die Übertragung des CEP-Ansatzes auf CBM erarbeitet. Hierfür wurde in Anlehnung an das Zustandsüberwachungssystem von OSA-CBM (siehe Abbildung 6) eine ereignisorientierte Referenzarchitektur entwickelt. Abgeleitet aus dieser Architektur wurden die wichtigsten Module identifiziert, welche im Rahmen einer ereignisgesteuerten Architektur in Betracht gezogen werden. Die Konfigurationsmethodik wurde als Voraussetzung für die Generierung und die Anpassung von Ereignisregeln, aufgrund von Systemänderungen, entwickelt. Dabei stellt die Offline-Datenanalyse unter Einsatz von maschinellen Lernmodellen den wesentlichen Bestandteil der Konfigurationsmethodik dar. Es zeigt sich, dass es durch den ML-Aufbauprozess möglich ist, das Ereignismodell und die Ereignisregeln zu generieren und anzupassen.

Im Kapitel 6 werden das zuvor im Kapitel 5 vorgestellte Referenzmodell und die Konfigurationsmethodik durch eine praktische Umsetzung evaluiert. Die Umsetzung und Evaluierung erfolgt basierend auf zwei Anwendungsfällen: einer prototypischen Erprobung eines Modells an einem Fuhrpark von Portalhubwagen und der Vereisung an den Rotorblättern von Windenergieanlagen. Dadurch ist es möglich, die Implementierung, die Übertragbarkeit des Konzeptes und die technische Machbarkeit nachzuweisen und darzulegen.

7.2. Beantwortung von Forschungsfragen

Mit der Motivation der Arbeit wurden zwei Hauptforschungsfragen aufgestellt. Im Folgenden werden diese Fragen beantwortet:

Forschungsfrage 1: *Wie muss das zustandsorientierte System im Hinblick auf seine Struktur entworfen werden, damit ein erfolgreicher Einsatz des CEP-Ansatzes realisiert werden kann?*

Durch die Beantwortung dieser Frage sollte das erste Ziel dieser Arbeit „Modellierung von ereignisorientierten Instandhaltungsprozessen im Hinblick auf die Realisierung eines Referenzmodells für ein ereignisorientiertes Instandhaltungssystem“ erreicht werden.

Jene Frage wird dann durch die Entwicklung eines Referenzarchitekturmodells beantwortet. Diese Arbeit bringt durch die Entwicklung eines Konzeptes zur Integration von CEP in ein CM-System die Grundvoraussetzungen einer echtzeitfähigen, zustandsorientierten Instandhaltung hervor. Aus der Betrachtung verschiedener etablierter CBM- und CEP-Architekturen ergibt sich die Identifikation der relevanten und wichtigen Module, die in ein funktionelles CEP-CM-System integriert werden müssen. In diesem Zusammenhang wurde auf die OSA-CBM-Spezifikation zurückgegriffen, um das Referenzmodell zur Integration des CEP-Ansatzes zu erarbeiten. Die Anforderungen, wie Skalierbarkeit, Echtzeitfähigkeit etc., wurden an dem CEP-CM-System ebenso erhoben. Die vier Module der Ereignisverarbeitungsschicht der CEP-CM-System-Architektur *Ereignisakquise*, *Ereignismanipulation*, *Zustandsüberwachung* und *Ereignisdiagnostik* unterstützen die Modulfunktionen der ersten vier Module der OSA-CBM-Architektur: „Datenerfassung, Datenverarbeitung, Zustandsüberwachung und Diagnostik“. In diesem Kontext wurden die Modulfunktionalitäten der OSA-CBM-Architektur in einer ereignisorientierten Weise in einen hierarchischen Modellaufbau eingebettet. Durch diese Einbettung der Module wurde die Struktur des zustandsorientierten Systems erarbeitet, damit ein erfolgreicher Einsatz des CEP-Ansatzes realisiert werden kann. Die Ereignisse werden somit in dieser Schichtenarchitektur in verschiedene Abstraktionsebenen definiert. In diesem Zusammenhang wurden die einfachen Ereignisse aus der Ereignisakquise generiert, die weiterhin durch die Ereignisregeln zu komplexen Ereignissen in der Ereignisdiagnostik abgeleitet wurden. Zur Interpretation von Ereignissen aus den verschiedenen Datenquellen wurde mittels eines Metamodells, das eine einheitliche Repräsentation der verschiedenen heterogenen Daten auf Datenebene ermöglicht, das Ereignismodell aufgebaut. Das Metamodell bietet eine wichtige Grundlage, um Wissen in Form von Ereignissen zur Interpretation des Systemzustands zu repräsentieren. Zur Bereitstellung dieser Ereignisse wurde der Ereignisadapter darüber hinaus für die Datentransformation in das interne Datenformat eingesetzt. Ein Großteil der auf dem Markt existierenden CEP-Software implementiert einen eigenen Ereignisadapter in der CEP-Engine, sodass eine Weiterentwicklung vom Ereignisadapter nicht erforderlich ist und die Datentransformation dadurch weniger Aufwand erfordert. Das Referenzmodell muss ebenso eine Plattform und Instrumente zur echtzeitfähigen Datenanalyse integrieren. Die Offline-Datenanalyse zur Extraktion der Ereignisregeln wurde basierend auf dem Batchverarbeitungskonzept entwickelt.

Als Ergebnis dieser Arbeit wurde ein Referenzarchitekturmodell entwickelt, das unter Anwendung des Ereignismodells eine Überführung der Daten aus verschiedenen heterogenen Quellen in eine explizite Ereignisrepräsentationsform unterstützt und somit die Rahmenbedingungen für eine erfolgreiche Übertragung des CEP-Ansatzes in eine CBM-Architektur definiert.

Forschungsfrage 2: *Wie muss das System konfiguriert werden, damit Sensordaten verschiedener Komponenten in Echtzeit zueinander in Beziehung gesetzt werden können, um globale Bedingungen richtig einschätzen zu können?*

Die zweite Frage ergibt sich aus dem zweiten Ziel dieser Arbeit: „Entwicklung eines adaptiven Ansatzes zur Generierung der Ereignisregeln mithilfe der Methoden der maschinellen Lernverfahren".

Diese Forschungsfrage wird anhand der Entwicklung einer Konfigurationsmethodik beantwortet. Ein großer Nachteil bei den aktuell bestehenden CEP-Ansätzen ist die Beschränkung der Ereignisabfragen auf vordefinierte Ereignisregeln, die zumeist manuell erstellt werden. Der Aufwand zur manuellen Anpassung von Ereignisregeln aufgrund von Systemänderungen ist jedoch noch hoch, was eines der Hauptargumente gegen die Nutzung des CEP-Ansatzes im CBM-Kontext darstellt. Um zukünftig eine breite CEP-Anwendung zur zustandsorientierten Zustandserkennung zu ermöglichen, ist die Entwicklung von Methoden zur automatischen Generierung von Ereignisregeln eine Hauptherausforderung für die Disziplin der echtzeitfähigen zustandsorientierten Instandhaltungssysteme. Deswegen wurde, als weiteres Ergebnis dieser Arbeit, eine Konfigurationsmethodik zur automatischen Generierung von Ereignisregeln entwickelt. Dabei muss das betrachtete technische System im ersten Schritt umfassend analysiert werden. Eine Analyse der aufgezeichneten Störungen hilft beim Aufbau eines ersten Ereignismodells, das die ersten Zusammenhänge zwischen Ereignissen abbildet. Die entwickelte Konfigurationsmethodik besteht aus zwei Teilen: 1) Entwicklung eines ML-Modells (Offline-Datenanalyse) und 2) Herleitung von Ereignisregeln aus dem ML-Modell. Aus der Offline-Datenanalyse resultieren unter der Anwendung verschiedener statistischer und maschineller Lernverfahren die Zustandsbewertung des betrachteten Systems und die Bildung des ML-Modells. Das ML-Modell kann dementsprechend aus den erhobenen Daten verschiedene Situationen detektieren, die den Rückschluss auf eine mögliche vorliegende Störung zulassen. Dabei stellt die Entwicklung von ML-Modellen eine sehr aufwändige Aufgabe dar. Darüber hinaus wurde im Rahmen dieser Dissertation ein Konzept entwickelt, das aus den verschiedenen Aufbauphasen des ML-Modells die Ereignisregeln generiert bzw. anpasst. Aus der Betrachtung der verschiedenen Phasen können die Ereignisse hierarchisch aufgebaut und strukturiert werden, wobei jedes Ereignis einer Aufbauphase zugeordnet ist und von der nächsten Aufbauphase unter Verwendung von abgeleiteten Ereignisregeln, die sich aus einer definierten Kombination von Ereignissen zusammensetzen, zu einem höhergestellten komplexen Ereignis generiert wird. Die Extraktion der Ereignisregeln wurde experimentell, basierend auf zwei Anwendungsfällen, als

Konzeptnachweis durchgeführt, um die Machbarkeit und die Leistungsfähigkeit des Konzepts zu demonstrieren. Dann wurde anhand dieser Anwendungsfälle das gesamte CEP-CM-System-Konzept exemplarisch durchgeführt. Es wurde nachgewiesen, dass die Übertragung des CEP-Ansatzes auf CBM durch das entwickelte Referenzmodell und die Konfigurationsmethodik maßgeblich unterstützt wird. Darüber hinaus wurde die Echtzeitfähigkeit des Systems basierend auf den Anwendungsfällen nachgewiesen. Im Rahmen dieser Arbeit konnte gezeigt werden, dass eine automatische Anpassung der Ereignisregeln durch den Ansatz der Konfigurationsmethodik (Offline-Datenanalyse) in einem Batchverarbeitungskonzept realisierbar ist. Dabei erfolgte die Konfigurationsmethodik zeitgesteuert einmal pro Tag und ersetzte die geänderten Ereignisregeln durch die neuen Regeln. Im Rahmen der Konzeptevaluierung konnte darüber hinaus gezeigt werden, dass die Implementierung des CEP-Ansatzes stark vom ML-Aufbauprozess abhängig ist.

Die mit dieser Arbeit angestrebten Zielsetzungen wurden somit erreicht, weswegen diese Arbeit einen maßgeblichen Forschungsbeitrag liefert.

7.3. Ausblick

Der Ausblick bezieht sich im Wesentlichen auf die Weiterentwicklung des CEP-CM-Systems. Daher wird im Folgenden auf einige Punkte eingegangen, die intensiver hätten behandelt werden können:

- Zur Entwicklung des gesamten CEP-CM-System wird zum einen eine Entwicklungsumgebung für die Offline-Datenanalyse und zum anderen eine andere Entwicklungsumgebung für die CEP-Engine eingesetzt. Dieser Fakt erschwert die Implementierung des CEP-CM-Systems, da die automatische Generierung von Ereignisregeln auf Basis der Offline-Datenanalyse und die Ereignisverarbeitung in der CEP-Engine erfolgt. Obwohl die bestehenden CEP-Engines ausreichend statistische und maschinelle Lernverfahren integrieren, können sie nicht alle ML-Methoden, wie z. B. neuronale Netze oder Support Vector Machine, anbieten. Deswegen erfolgt die Entwicklung von ML-Methoden in der Offline-Datenanalyse mit „Python“ als Programmiersprache in einer anderen Entwicklungsumgebung. Nach der Generierung des ML-Modells wird es, wie im Abschnitt 6.4.2 beschrieben, in einem vorgegebenen Ordner im SaveModel-Format hinterlegt. Danach kann es in der CEP-Engine in der Formulierung der Ereignisregeln verwendet werden. Für die CEP-Entwicklung werden je nach CEP-Engine verschiedene Programmiersprachen verwendet. Für diese Arbeit wurde Siddhi-CEP-Engine hinsichtlich der Ereignisverarbeitung verwendet. Diese Engine kann als eingebettete Java- und Python-Bibliothek ausgeführt werden. Es existieren andere Java-basierte Open-Source-CEP-Engines, wie das Java-basierte Softwareprodukt Esper oder die .NET-Variante NEsper. Diese unterschiedlichen Entwicklungsumgebungen machen die Implementierung des CEP-CM-Systems zu einer schwierigen Aufgabe. Bei der Implementierung der Offline-Datenanalyse und der CEP-

Engine in eine einzige Entwicklungsumgebung werden die Anpassungsaufwände minimiert. Auch wenn sich viele CEP-Engines bemühen, viele ML-Frameworks in deren Struktur zu integrieren, sind sie nicht dazu in der Lage, alle ML-Methoden anzubieten.

- Kontinuierliche Anpassung von Ereignisregeln bzw. Ereignismodell: Die bestehenden CEP-Ansätze sind unflexibel und erfordern einen hohen Konfigurationsaufwand. In dieser Arbeit wurde dementsprechend eine Konfigurationsmethodik entwickelt, um die automatische Generierung der Ereignisregeln zu ermöglichen. Für die Anpassung der Regeln wird immer noch auf eine Offline-Konfigurationsmethodik zurückgegriffen, was die Regelanpassung aufwendiger macht. Darüber hinaus stellt sich die Frage, wann eine Anpassung der Ereignisregeln durchgeführt werden soll. In dieser Arbeit wurde ein zeitorientierter Ansatz für die automatische Anpassung der Ereignisregeln verfolgt. Das bedeutet, dass nach einem festen Zeitintervall, zum Beispiel einmal pro Tag, die Ereignisregeln nach Plausibilität überprüft werden. Eine ereignisorientierte automatische Regelanpassung wurde nicht untersucht. Perspektivisch sollte die Konfigurationsmethodik weiterentwickelt werden, um die Ereignisregeln ereignisgesteuert anzupassen.
- Probabilistische Ereignisverarbeitung: In dieser Arbeit wurde die grundlegende Annahme getroffen, dass die Ereignisregeln deterministisch sind. In vielen ereignisgesteuerten Echtzeitanwendungen gilt diese Annahme jedoch nicht. Die probabilistischen Aspekte bzw. die unsicheren Ereigniseintritte sollen in der Generierung der Ereignisregeln mitberücksichtigt werden. In diesem Zusammenhang hat jede Ereignisregel eine Eintrittswahrscheinlichkeit, die die Unsicherheit der Auftritte des resultierenden Ereignisses darstellt. Ein Beispiel einer probabilistischen Ereignisregel wäre, dass ein resultierendes Ereignis bei Auslösung der Ereignisregel mit einer Wahrscheinlichkeit von 70 % abgeleitet wird. Das stochastische Verhalten von CEP-CM-Systemen sollte mithilfe der Wahrscheinlichkeitstheorie weiter untersucht werden. Dementsprechend sollte das CEP-CM-System weiterentwickelt werden, um die Unsicherheiten des CEP im CBM-Kontext einzuführen.
- Die Umsetzung und Evaluierung des Gesamtkonzepts erfolgte in dieser Arbeit zentral, und obwohl die Umsetzung eines verteilten Systems zwar technologisch möglich ist, wurde das in dieser Arbeit jedoch nicht weiterverfolgt. Verteilungssysteme steigern die Umsetzungskomplexität und es ist daher zu untersuchen, ob das entwickelte Referenzmodell durch den Einsatz des Verteilungskonzeptes umstrukturiert werden sollte, um die Integration des CEP in ein verteiltes CM-System zu ermöglichen.

Literaturverzeichnis

[Agrawal et al. 1994] Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).

[Ait Alla et al. 2016] Ait Alla, A.; Lütjen, M.; Lewandowski, M.; Freitag, M.; Thoben, K-D. (2016). Real-Time Fault detection for Advanced Maintenance of Sustainable Technical Systems. In: Teti, R. (eds.): Procedia CIRP. Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems, Elsevier, N.N, 2016, pp. 295-300.

[Ait Alla et al. 2020] Ait-Alla, A., Oelker, S., Lewandowski, M., & Freitag, M. (2020). Simulation of contrary maintenance strategies for offshore wind turbines. Journal of Simulation, 14(1), 76-82.

[Alpar et al. 2000] Alpar, P., & Niedereichholz, J. (2000). Data Mining im praktischen Einsatz–Verfahren und Anwendungsfälle für Marketing. Vertrieb, Controlling und Kundenunterstützung, Braunschweig/Wiesbaden.

[Alsyouf 2004] Alsyouf, I. (2004). Cost effective maintenance for competitve advantages (Doctoral dissertation, Växjö university press).

[Akbar et al. 2015] Akbar, A., Carrez, F., Moessner, K., & Zoha, A. (2015, December). Predicting complex events for pro-active IoT applications. In Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on (pp. 327-332). IEEE.

[Alias et al. 2016] Alias, C., Rawet, V. L., Neto, H. X. R., & Reymão, J. D. E. N. (2016). Investigating into the Prevalence of Complex Event Processing and Predictive Analytics in the Transportation and Logistics Sector: Initial findings from scientific literature. In MCIS (p. 2).

[Arab et al. 2013] Arab, A., Ismail, N., & Lee, L. S. (2013). Maintenance scheduling incorporating dynamics of production system and real-time information from workstations. Journal of Intelligent Manufacturing, 24(4), 695-705.

[Austerjost et al. 2016] Austerjost, M., Hegmanns, T., Kuhn, A., Fuß, B., Seibold, Z., Furmans, K., & Schieweck, S. (2016). 1. Beitrag einer nutzungsabhängigen Instandhaltung zur Planung von Leistungsverfügbarkeit. Logistics Journal, 2192(9084), 1.

[Babiceanu et al. 2015] Babiceanu, R.F. and Seker, R. (2015). “Service Orientation in Holonic and Multi-agent Manufacturing. ”Service Orientation in Holonic and Multi-Agent Manufacturing 594, 157–164.

[Backus et al. 2006] Backus, P., Janakiram, M., Mowzoon, S., Runger, C., & Bhargava, A. (2006). Factory cycle-time prediction with a data-mining approach. IEEE Transactions on Semiconductor Manufacturing, 19(2), 252-258.

[Bajaj et al. 2014] Bajaj, R. H., & Ramteke, P. (2014). Big data–the new era of data. International Journal of Computer Science and Information Technologies, 5(2), 1875-1885. ISSN. : 0975-9646.

[Basseville et al. 1996] Basseville, M., & Cordier, M. O. (1996). Surveillance et diagnostic de systèmes dynamiques: approches complémentaires du traitement de signal et de l'intelligence artificielle (Doctoral dissertation, INRIA).

[Bengtsson 2007] Bengtsson, M. (2007). On condition based maintenance and its implementation in industrial settings. 2007. Doktorarbeit. Mälardalen University. ISBN 978-91-85485-57-4.

[Benkedjouh et al. 2018] Benkedjouh, T., Zerhouni, N., & Rechak, S. (2018). Tool wear condition monitoring based on continuous wavelet transform and blind source separation. The International Journal of Advanced Manufacturing Technology, 97(9-12), 3311-3323.

[Beugin 2006] Beugin, J. (2006). Contribution à l'évaluation de la sécurité des systèmes complexes de transport guidé (Doctoral dissertation, Université de Valenciennes et du Hainaut-Cambresis).

[Bey-Temsamani et al. 2009] Bey-Temsamani, A., Engels, M., Motten, A., Vandenplas, S., &Ompusunggu, A. P. (2009). A practical approach to combine data mining and prognostics for improved predictive maintenance. The 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Paris.

[Bouvard et al. 2011] Bouvard, K., Artus, S., Bérenguer, C., & Cocquempot, V. (2011). Condition-based dynamic maintenance operations planning & grouping. Application to commercial heavy vehicles. Reliability Engineering & System Safety, 96(6), 601-610.

[Botta et al. 1998] Botta G, Cavaliere M, Holttinen H, 1998. Ice Accretion at Acqua Spruzza and Its Effects on Wind Turbine Operation and Loss of Energy Production. Ice Accretion at Acqua Spruzza and its Effects on Wind Turbine Operation and Loss of Energy Production, BOREAS IV.

[Bramer 2013] Bramer, M. (2013). Principles of data mining, second edition. Springer London Heidelberg New York Dordrecht: 2013. ISBN 978-1-4471-4883-8. DOI 10.1007/978-1-4471-4884-5

[Bruns et al. 2010] Bruns, R., & Dunkel, Jürgen. Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse. Springer-Verlag Berlin Heidelberg: 2010. ISBN 9783642024399.

[BSI, 1984] Glossar of maintenance terms in terotechnology. BS 3811. London: British Standard Institution (BSI).

[Buddhakulsomsiri et al. 2006] Buddhakulsomsiri, J., Siradeghyan, Y., Zakarian, A., & Li, X. (2006). Association rule generation algorithm for mining automotive warranty data. International Journal of Production Research, 44(14), 2749–2770. doi:10.1080/00207540600564633.

[Bülow et al. 2013] Bülow, S., Backmann, M., Herzberg, N., Hille, T., Meyer, A., Ulm, B.,Weske, M. (2013, August). Monitoring of business processes with complex event processing. In International Conference on Business Process Management (pp. 277-290). Springer, Cham.

[Butler 1996] Butler, K. L. (1996, January). An expert system based framework for an incipient failure detection and predictive maintenance system. In Proceedings of International Conference on Intelligent System Application to Power Systems (pp. 321-326). IEEE.

[Cabanillas et al. 2014] Cabanillas, C., Baumgrass, A., Mendling, J., Rogetzer, P., & Bellovoda, B. (2014). Towards the Enhancement of Business Process Monitoring for Complex Logistics Chains. In Business Process Management Workshops (pp. 305-317). Springer International Publishing.

[Camci 2009] Camci, F. (2009). System maintenance scheduling with prognostics information using genetic algorithm. IEEE Transactions on reliability, 58(3), 539-552.

[Candell et al. 2009] Candell, O., Karim, R., & Söderholm, P. (2009). eMaintenance—Information logistics for maintenance support. Robotics and Computer-Integrated Manufacturing, 25(6), 937-944.

[Canizo et al. 2017] Canizo, M., Onieva, E., Conde, A., Charramendieta, S., & Trujillo, S. (2017). Real-time predictive maintenance for wind turbines using Big Data frameworks. arXiv preprint arXiv:1709.07250.

[Chakravarthy et al. 2009] Chakravarthy, S., & Jiang, Q. (2009). Stream data processing: a quality of service perspective: modeling, scheduling, load shedding, and complex event processing (Vol. 36). Springer Science & Business Media.

[Chandola et al. 2009] V. Chandola, A. Banerjee and V. Kumar. Anomaly detection: a survey. In ACM Computing Surveys, 41(3), pages 15:1 -15:58, 2009.

[Chang et al. 2007] Chang, Q., Ni, J., Bandyopadhyay, P., Biller, S., & Xiao, G. (2007). Maintenance opportunity planning system. Journal of Manufacturing Science and Engineering, 129(3), 661-668.

[Chapman et al. 2000] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0 Step-by-step data mining guide.

[Chen et al. 2005] Chen, M. C., Huang, C. L., Chen, K. Y., & Wu, H. P. (2005). Aggregation of orders in distribution centers using data mining. Expert Systems with Applications, 28(3), 453-460.

[Choudhary et al. 2009] Choudhary, A. K., Harding, J. A., & Tiwari, M. K. (2009). Data mining in manufacturing: a review based on the kind of knowledge. Journal of Intelligent Manufacturing, 20(5), 501.

[Cleve et al. 2016] Cleve, J.; Lämmel, U.: Data Mining 2. Auflage. (2016) Walter de Gruyter GmbH, Berlin/Boston. ISBN. : 978-3-11-045675-2.

[Coenen 2011] Coenen, F. (2011). Data mining: past, present and future. The Knowledge Engineering Review, 26(1), 25-29.

[Correia et al. 2015] Correia, I., Fournier, F., & Skarbovsky, I. (2015, June). The uncertain case of credit card fraud detection. In Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems (pp. 181-192). ACM.

[Denkena et al. 2012] Denkena, B., Bluemel, P., Kroening, S., & Roebbing, J. (2012). Condition based maintenance planning of highly productive machine tools. Production Engineering, 6(3), 277-285.

[DIN 31051] DIN Deutsches Institut für Normung e.V. (Hrsg.): DIN 31051:2003-06. Grundlagen der Instandhaltung, Juni 2003.

[Dunkel 2009] Dunkel, J. (2009, March). On complex event processing for sensor networks. In Autonomous Decentralized Systems, 2009. ISADS'09. International Symposium on (pp. 1-6). IEEE.

[Dunkel et al. 2010] Dunkel, J., Fernández, A., Ortiz, R., & Ossowski, S. (2011). Event-driven architecture for decision support in traffic management systems. Expert Systems with Applications, 38(6), 6530-6539.

[Eberlein et al. 2018] Eberlein, S.; Oelker, S.; Schumacher, J.; Freitag, M. (2018). Automatisierung von Containerterminals: Konzept für den Aufbau einer Pilotanlage für automatisierte Straddle Carrier und emulationsbasierte Untersuchung der Skalierbarkeit. In: Industrie 4.0 Management, 34(2018)6, S. 25-29

[Eckert et al. 2009] Eckert, M., &Bry, F. (2009). Complex Event Processing (CEP).Informatik-Spektrum, 32(2), 163-167.

[Eickemeyer et al. 2014] Eickemeyer, S. C., Herde, F., Irudayaraj, P., & Nyhuis, P. (2014). Decision models for capacity planning in a regeneration environment. International Journal of Production Research, 52(23), 7007-7026.

[Esposito et al. 2015] Esposito, C., Ficco, M., Palmieri, F., & Castiglione, A. (2015). A knowledge-based platform for Big Data analytics based on publish/subscribe services and stream processing. Knowledge-Based Systems, 79, 3-17.

[Etzion et al. 2013] Etzion, O., & Adkins, J. M. (2013, June). Tutorial: why is event-driven thinking different from traditional thinking about computing?. In Proceedings of the 7th ACM international conference on Distributed event-based systems (pp. 269-270). ACM.

[Fakorede et al. 2016] Fakorede O, Feger Z, Ibrahim H, Ilinca A, Perron J, Masson C, 2016. Ice protection systems for wind turbines in cold climate: characteristics, comparisons and analysis. Renewable and Sustainable Energy Reviews 65, 662–675.

[Fayyad et al. 1996] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.: From Data Mining to Knowledge Discovery in Databases. In: AI Magazine 17 (1996) 3, S. 37-54.

[Ferreiro et al. 2012] Ferreiro, S., Arnaiz, A., Sierra, B., & Irigoien, I. (2012). Application of Bayesian networks in prognostics for a new Integrated Vehicle Health Management concept. Expert Systems with Applications, 39(7), 6402-6418.

[Flouris et al. 2017] Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M., & Mock, M. (2017). Issues in complex event processing: Status and prospects in the big data era. Journal of Systems and Software, 127, 217-236.

[Freitag et al. 2015a] Freitag, M.; Kück, M.; Ait Alla, A.; Lütjen, M. (2015). Potenziale von Data Science in Produktion und Logistik: Teil 1 - Eine Einführung in aktuelle Ansätze der Data Science. In: Industrie 4.0 Management, 31(2015)5, S. 22-26

[Freitag et al. 2015b] Freitag, M.; Kück, M.; Ait Alla, A.; Lütjen, M. (2015). Potenziale von Data Science in Produktion und Logistik: Teil 2 - Vorgehensweise zur Datenanalyse und Anwendungsbeispiele. In: Industrie 4.0 Management, 31(2015)6, S. 39-46

[Fülöp et al. 2010] Fülöp, L., Tolla,G., Racz,R., Panczel,J., &Gergely, T. (2010). Survey on Complex Event Processing and Predictive Analytics.Networks. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.2029&rep=rep1&type=pdf

[Fumeo et al. 2015] Fumeo, E., Oneto, L., & Anguita, D. (2015). Condition based maintenance in railway transportation systems based on big data streaming analysis. Procedia Computer Science, 53, 437-446.

[Gillani et al. 2017] Gillani, S., Kammoun, A., Singh, K., Subercaze, J., Gravier, C., Fayolle, J., & Laforest, F. (2017, November). Pi-CEP: Predictive Complex Event Processing using Range Queries over Historical Pattern Space. In Data Mining Workshops (ICDMW), 2017 IEEE International Conference on (pp. 1166-1171). IEEE.

[Goyal et al. 2015] Goyal, D., & Pabla, B. S. (2015). Condition based maintenance of machine tools—A review. CIRP Journal of Manufacturing Science and Technology, 10, 24-35.

[Greiner et al. 2006] Greiner, T., Düster, W., Pouatcha, F., von Ammon, R., Brandl, H. M., & Guschakowski, D. (2006, August). Business activity monitoring of norisbank taking the example of the application easycredit and the future adoption of complex event processing (CEP). In Proceedings of the 4th international symposium on Principles and practice of programming in Java (pp. 237-242). ACM.

[Han et al. 2006] Han, T., & Yang, B. S. (2006). Development of an e-maintenance system integrating advanced techniques. Computers in Industry, 57(6), 569-580.

[Harding et al. 2006] Harding, J. A., Shahbaz, M., & Kusiak, A. (2006). Data mining in manufacturing: a review. Journal of Manufacturing Science and Engineering, 128(4), 969-976.

[Hayashi et al. 2002] Hayashi, S., &Asakura, T. (2002).Study of machine fault diagnosis system using neural networks.Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290), 956-961. IEEE.

[Heng et al. 2009] Heng, A., Zhang, S., Tan, A. C., & Mathew, J. (2009). Rotating machinery prognostics: State of the art, challenges and opportunities. Mechanical systems and signal processing, 23(3), 724-739.

[Hochart et al. 2008] Hochart C, Fortin G, Perron J, Ilinca A, 2008. Wind turbine performance under icing conditions. Wind Energy. 11 (4), 319–333.

[Hodge et al. 2004] Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial intelligence review, 22(2), 85-126.

[ISKE 2009] Iske, F. (2009). 30 Jahre Entwicklung der Instandhaltung–von der ausfallorientierten Instandhaltung zum gemeinsamen TPM und RCM. In Betriebliche Instandhaltung (pp. 51-74). Springer, Berlin, Heidelberg.

[Jacobs et al. 2009] Jacobs, J., Junker, A., &Letmathe, P. (2009): Zustandsorientierte Maschinenzuordnungs- und Instandhaltungsplanung. Zeitschrift für Betriebswirtschaft, 79(11), 1259-1282.

[Jardine et al. 2006] Jardine, A. K. S., Lin, D., &Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mechanical Systems and Signal Processing, 20(7), 1483-1510.

[Kantz et al. 2004] Kantz, H.; Schreiber, T.: Nonlinear Time Series Analysis. Cambridge 2004.

[Khalid et al. 2014] Khalid, M.S: Fathi, Z. Mohamed, Integration of cyber-physical systems technology with augmented reality in the pre-construction stage, Technology, Informatics, Management, Engineering, and Environment (TIME-E), (2014) 151-156.

[Klemm et al. 2010] Klemm, P., Salinas, L., Oglodin, V. (2010). Konzept zur Steuerungsinternen vorbeugenden Maschinendiagnose. wt-online - Ausgabe 5-2010, S. 373-379.

[Koochaki 2009] Koochaki, J. (2009). Collaborative Learning in Condition Based Maintenance. Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, London, U.K.

[Kovács et al. 2011] Kovács, A., Erdos, G., Monostori, L., & Viharos, Z. J. (2011, August). Scheduling the maintenance of wind farms for minimizing production loss. In Proceedings of the 18th IFAC World Congress, Milano (Italy).

[Kreutz et al. 2019] Kreutz, M.; Ait Alla, A.; Varasteh, K.; Oelker, S.; Greulich, A.; Freitag, M.; Thoben, K. (2019). Machine learning-based icing prediction on wind turbines. In: Vrabic, R. (eds.): Procedia CIRP. 52nd CIRP Conference on Manufacturing Systems (CMS), Elsevier B.V, Amsterdam, NL, 2019, pp. 423 - 428

[Kriesel 2008] Kriesel, D.: Ein kleiner Überblick über Neuronale Netze, 2008, Information on http://www.dkriesel.com/ [30.03.2015]

[Kröning et al. 2013] Kröning, S., & Denkena, B. (2013). Dynamic scheduling of maintenance measures in complex production systems. CIRP Journal of Manufacturing Science and Technology, 6(4), 292-300.

[Ksystra et al. 2012] Ksystra, K., Triantafyllou, N., & Stefaneas, P. (2012, August). On the algebraic semantics of reactive rules. In International Workshop on Rules and Rule Markup Languages for the Semantic Web (pp. 136-150). Springer, Berlin, Heidelberg.

[Laakso et al. 2010] Laakso T, Baring-Gould I, Durstewitz M, Horbaty R, Lacroix A, Peltola E, Ronsten G, Tallhaug L, Wallenius T, 2010. State-of-The-art of wind energy in cold climates. English 152.

[Lebold et al. 2003] Lebold, M., Reichard, K., &Boylan, D. (2003).Utilizing dcom in an open system architecture framework for machinery monitoring and diagnostics.2003 IEEE Aerospace Conference Proceedings (Cat.No.03TH8652), 3, 3_1227-3_1236.Ieee.

[LeCun et al. 2015] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436.

[Leger et al. 1998] R. P. Leger, Wm. J. Garland & W. F. S. Poehlmanb: Fault detection and diagnosis using statistical control charts and artificial neural networks. Artificial Intelligence in Engineering, Vol 12(1), pp 35-47, 1998.

[Levrat et al. 2008] Levrat, E., Iung, B., & Crespo Marquez, A. (2008). E-maintenance: review and conceptual framework. Production Planning & Control, 19(4), 408-429.

[Lewandowski 2016] Lewandowski, M. (2016). Entwicklung eines Integrationsmodells zur Zustandserfassung und Optimierung der Instandhaltung komplexer technischer Systeme. Publisher: epubli ISBN: 3741863939.

[Li 2010] Li, M. (2010). Robust complex event pattern detection over streams (Doctoral dissertation, Worcester Polytechnic Institute).

[Li et al. 2014] Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., & Hampapur, A. (2014). Improving rail network velocity: A machine learning approach to predictive maintenance. Transportation Research Part C: Emerging Technologies, 45, 17-26.

[Lieber et al. 2013] Lieber, D., Stolpe, M., Konrad, B., Deuse, J., & Morik, K. (2013). Quality prediction in interlinked manufacturing processes based on supervised & unsupervised machine learning. Procedia CIRP, 7, 193-198.

[Luckham 2002] Luckham, D. C. The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Boston (2002). ISBN 978-0-201-72789-0.

[Mayer-Schönberger et al. 2013] Mayer-Schönberger, V., & Mayer-Schönberger, C. (2013). Big Data: die Revolution, die unser Leben verändern wird. Redline Wirtschaft. ISBN-10: 3868815066

[Mehdiyev et al. 2016] Mehdiyev, N., Krumeich, J., Werth, D., & Loos, P. (2016, January). Determination of event patterns for complex event processing using fuzzy unordered rule induction algorithm with multi-objective

evolutionary feature subset selection. In 2016 49th Hawaii International Conference on System Sciences (HICSS) (pp. 1719-1728). IEEE.

[Meister 2014] Meister, S. (2014). Erweiterung von Konzepten des complex event processings zur informationslogistischen Verarbeitung telemedizinischer Ereignisse zur Erlangung des Grades eines Doktors der Naturwissenschaften Der Technischen Universität Dortmund an der Fakultät für Informatik.

[Ministerium IL-NWF 2018] Ministerium des Innern des Landes Nordrhein-Westfalen, 2018. Erlass für die Planung und Genehmigung von Windenergieanlagen und Hinweise für die Zielsetzung und Anwendung: Windenergie-Erlass. Section 5.2.3.5. https://recht.nrw.de/lmi/owa/br_vbl_detail_text?anw_nr=7&vd_id=16977. Accessed 11 December 2018.

[Moubray 1997] Moubray, J. (1997) RCM – Die Hohe Schule der Zuverlässigkeit von Produkten und Systemen, Industrial Press, 2nd Edition, April 1997.

[Mousheimish 2017] Mousheimish, R. (2017). Combining the Internet of things, complex event processing, and time series classification for a proactive BPM. (Doctoral dissertation, Université de Versailles Saint-Quentin en Yvelines).

[Mouttham et al. 2009] Mouttham, A., Peyton, L., Eze, B., & Saddik, A. E. (2009). Event-driven data integration for personal health monitoring. Journal of Emerging Technologies in Web Intelligence, 1(2), 110-118.

[Mukherjee et al. 2010] Mukherjee, A., Diwan, P., Bhattacharjee, P., Mukherjee, D., Misra, P. (2010). Capital Market Surveillance using Stream Processing, 2nd International Conference on Computer Technology and Development (ICCTD 2010), 577-582.

[Muller et al. 2008] Muller, A., Marquez, A. C., & Iung, B. (2008). On the concept of e-maintenance: Review and current research. Reliability Engineering & System Safety, 93(8), 1165-1187.

[Nadj et al. 2016] Nadj, M., Jegadeesan, H., Maedche, A., Hoffmann, D., & Erdmann, P. (2016, June). A Situation Awareness Driven Design for predictive Maintenance Systems: the Case of Oil and gas Pipeline Operations. In ECIS.

[Nagappan et al. 2010] Nagappan, G., & Chellappan, C. (2010). Event Driven Architecture for Travel Time Reduction. International Journal of Computer Applications, 8(8), 16-20.

[Nicolai et al. 2008] Nicolai, R. P., & Dekker, R. (2008). Optimal maintenance of multi-component systems: a review (pp. 263-286). Springer London.

[Nielsen 2015] Nielsen, M. A. (2015). Neural networks and deep learning (Vol. 25). San Francisco, CA, USA: Determination press.

[Odenthal et al. 2014] Odenthal, H. J., Uebber, N., Schlüter, J., Löpke, Mike., Morik, K., Blom, H., & AG, S. S. (2014). Data-driven Prediction Model for the BOF process - application of data mining at AG der Dillinger Hüttenwerke, Germany. In MPT Metallurgical Plant and Technology 5:30-36.

[Oelker et al. 2018] Oelker S, Ait Alla A, Lütjen M, Lewandowski M, Freitag M, Thoben K-D, (2018). A simulation study of feeder-based installation concepts for offshore wind farms, in: Proceedings of the Twenty-eighth (2018) International Ocean and Polar Engineering Conference. Engineering

[Oelker et al. 2017] Oelker, S.; Ait Alla, A.; Lewandowski, M.; Freitag, M. (2017). Simulation konträrer Instandhaltungsstrategien für Offshore-Windenergieanlagen. In: Wenzel, S.; Peter, T. (Hrsg.): Simulation in Produktion und Logistik 2017. kassel university press, Kassel, 2017, S. 269-278

[Parent et al. 2011] Parent O, Ilinca A, 2011. Anti-icing and de-icing techniques for wind turbines: Critical review. Cold Regions Science and Technology 65 (1), 88–96.

[Patan 2008] Patan, K. (2008). Artificial neural networks for the modelling and fault diagnosis of technical processes. Springer.

[Petersen et al. 2016] Petersen, E., To, M. A., & Maag, S. (2016, November). An online learning based approach for CEP rule generation. In 2016 8th IEEE Latin-American Conference on Communications (LATINCOM) (pp. 1-6). IEEE.

[Pham et al. 2010] Pham, H. T., & Yang, B. S. (2010). Estimation and forecasting of machine health condition using ARMA/GARCH model. Mechanical Systems and Signal Processing, 24(2), 546-558.

[Quandt et al. 2017] Quandt, M.; Ait Alla, A.; Meyer, L.; Freitag, M. (2017). Success Factors for the Development of Augmented Reality-based Assistenace Systems for Maintenance Services. In: Schmitt, R.H.; Schuh, G. (Hrsg.): 7. WGP-Jahreskongress. Aachen, 5.-6. Oktober 2017, Apprimus Verlag, Aachen, 2017, S. 175-182.

[Reinhart et al. 2013] Reinhart, G.; Engelhardt, P.; Geiger, F.; Philipp, T. R.; Wahlster, W.; Zühlke, D.; Schlick, J.; Becker, T.; Löckelt, M.; Pirvu, B.; Stephan, P.; Hodek, S.; Scholz-Reiter, B.; Thoben, K.; Gorldt, C.; Hribernik, K. A.; Lappe, D.; Veigt, M. (2013). Cyber-Physische Produktionssysteme - Produktivitäts- und Flexibilitätssteigerung durch die Vernetzung intelligenter

Systeme in der Fabrik. wt-online - Ausgabe 2-2013 SONDERHEFT INDUSTRIE 4.0, S. 084-089.

[Rieker et al. 2015] Rieker, T., Zeiler, P., & Bertsche, B. (2015, January). Reliability analysis of a hybrid car drive system with ECSPN. In Reliability and Maintainability Symposium (RAMS), 2015 Annual (pp. 1-7). IEEE.

[Ryll 2008] Ryll, F. Gestaltung einer zustandorientierten Instandhaltungsstrategie auf der Grundlage einer erfahrungsbasierten Bewertung von Abnutzungsvorräten in technischen Anlagen. Magdeburg Fraunhofer Institut f. Fabrikbetrieb u. –automatisierung. Fraunhofer IRB Verlag, 2008, ISBN 976-3-8167-7658-1.

[Saimurugan et al. 2011] Saimurugan, M., Ramachandran, K. I., Sugumaran, V., & Sakthivel, N. R. (2011). Multi component fault diagnosis of rotational mechanical system based on decision tree and support vector machine. Expert Systems with Applications, 38(4), 3819-3826.

[Sammouri et al. 2013] Sammouri, W., Côme, E., Oukhellou, L., Aknin, P., & Fonlladosa, C. E. (2013, October). Floating train data systems for preventive maintenance: A data mining approach. In Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM) (pp. 1-7). IEEE.

[Schenk 2010] Schenk, M. (2010). Instandhaltung technischer Systeme: Methoden Und Werkzeuge Zur Gewährleistung Eines Sicheren Und Wirtschaftlichen Anlagenbetriebs. Berlin, Heidelberg: Springer Berlin Heidelberg 2010.

[Scholz-Reiter et al. 2009] Scholz-Reiter, B.; Lewandowski, M.; Dittmer, P.; Oelker, S. (2009). Neue Instandhaltungsstrategien in der Hafenlogistik - Ein Fallbeispiel für die zustandsorientierte Instandhaltung von Hafenumschlaggeräten. In: Industrie Management, 25(2009)5, S. 9-12

[Schutz et al. 2011] Schutz, J., Rezg, N., & Léger, J. B. (2011). Periodic and sequential preventive maintenance policies over a finite planning horizon with a dynamic failure law. Journal of Intelligent Manufacturing, 22(4), 523-532.

[Sha et al. 2005] Sha, D. Y., & Liu, C. H. (2005). Using data mining for due date assignment in a dynamic job shop environment. The International Journal of Advanced Manufacturing Technology, 25(11-12), 1164-1174.

[Sikorska et al. 2011] Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. Mechanical systems and signal processing, 25(5), 1803-1836.

[Şimşek et al. 2018] Şimşek, M. U., & Özdemir, S. (2018, September). CEP Rule Extraction From Unlabeled Data in IoT. In 2018 3rd International Con-

ference on Computer Science and Engineering (UBMK) (pp. 429-433). IEEE.

[Simeón et al. 2010] Simeón, E. A., Álvares, A. J., & Gudwin, R. R. (2010). An expert system for fault diagnostics in condition based maintenance. In ABCM Symp. Ser. Mechatron (Vol. 4, pp. 304-313).

[Sokha et al. 2013] Sokha Y; Karpjoo Jeong; Jonghyun Lee; Woojin Joe. (2013). A Complex Event Processing System Approach to Real Time Road Traffic Event Detection. In Journal of Convergence Information Technology. Oct2013, Vol. 8 Issue 15, p142-148.

[SSI Swedish Standards Institute. (2001). Maintenance Terminology. SS-EN 13306, 2001.

[Starr 1997] Starr, A. (1997). A Structured Approach to the Selection of Condition Based Maintenance. 5th International Conference on FACTORY 2000 – The Technology Exploitation Process, Cambridge, UK.

[Tabatabaei 2018] Tabatabaei, N. (2018). Impact of Icing on Wind Turbine Aerodynamics. Doctoral dissertation, 11 (2018). doi: 10:13140/RG:2:2:14843:57121.

[Tarkoma 2012] Tarkoma, S. (2012). Publish/subscribe systems: design and principles. John Wiley & Sons.

[Tishby et al. 2015] Tishby, N., & Zaslavsky, N. (2015, April). Deep learning and the information bottleneck principle. In 2015 IEEE Information Theory Workshop (ITW) (pp. 1-5). IEEE.

[Thoben et al. 2018] Thoben, K-D.; Ait Alla, A.; Franke, M.; Hribernik, K.; Lütjen, M.; Freitag, M. (2018). Real-time Predictive Maintenance Based on Complex Event Processing. In: Martin Zelm; Frank-Walter Jaekel; Guy Doumeingts; Martin Wollschlaeger (eds.): Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability. Wiley Online Library, 1(2018).

[Vaidehi et al. 2012] Vaidehi, V., Bhargavi, R., Ganapathy, Kirupa., Sweetlin Hemalatha, C. (2012). "Multi-sensor based in-home health monitoring using Complex Event Processing", proceedings of IEEE International Conference on Recent Trends in Information Technology (ICRTIT 2012), MIT, Anna University, pp.570-575.

[Walther 2011] Walther, M. 2011. Antriebsbasierte Zustandsdiagnose von Vorschubantrieben. ISW/IPA Forschung und Praxis 183. Jost-Jetter Verlag, Heimsheim, (Doctoral dissertation, Fraunhofer IPA).

[Wang et al. 2018] Wang, Y., Gao, H., & Chen, G. (2018). Predictive complex event processing based on evolving Bayesian networks. Pattern Recognition Letters, 105, 207-216.

[Wang 2002] Wang, H. (2002). A survey of maintenance policies of deteriorating systems. European journal of operational research, 139(3), 469-489.

[Wang et al. 2011] Wang, D., Rundensteiner, E. A., & Ellison III, R. T. (2011). Active complex event processing over event streams. Proceedings of the VLDB Endowment, 4(10), 634-645.

[Warnecke 2008] Warnecke, M. (2008). Optimierung der Instandhaltungsplanung mit genetischen Algorithmen unter Berücksichtigung des Speicherverhaltens bei der Verfügbarkeitsprognose von Kraftwerksanlagen. zur Erlangung des akademischen Grades eines Doktor-Ingenieurs.

[Wenzel et al. 2011] Wenzel, S., & Bandow, G. (2011). Condition monitoring in logistics–a new approach for maintenance. In ICEME 2011–2nd International Conference on Engineering and Meta-Engineering (pp. 27-30).

[Widder 2011] Widder, A. (2011). Complex Event Processing und maschinelle Lernverfahren. Inaugural-Dissertation zur Erlangung der Doktorwürde der Fakultät für Sprach-, Literatur- und Kulturwissenschaften der Universität Regensburg.

[Widodo et al. 2007] Widodo, A., & Yang, B. S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. Mechanical systems and signal processing, 21(6), 2560-2574.

[Windt et al. 2011] Windt, K., Knollmann, M., & Meyer, M. (2011). Anwendung von Data Mining Methoden zur Wissensgenerierung in der Logistik-Kritische Reflexion der Analysefähigkeit zur Termintreueverbesserung. Wissensarbeit-Zwischen strengen Prozessen und kreativem Spielraum, 223-249.

[Wirth et al. 2000] Wirth, R., & Hipp, J. (2000, April). CRISP-DM: Towards a standard process model for data mining. In Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining (pp. 29-39). Citeseer.

[Xu et al. 2005] Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3), 645–678. doi:10.1109/ TNN.2005.845141.

[Zemouri 2003] Zemouri, R. (2003). Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques: Application à la e-maintenance (Doctoral dissertation, Université de Franche-Comté).

[Zhou et al. 2012] Zhou, Q., Simmhan, Y., & Prasanna, V. (2012). SCEPter: Semantic complex event processing over end-to-end data flows. Technical Report, Computer Science Department, University of Southern California, 12-926.

Anhang

Anhang A – Zusammenfassende Darstellung von Methoden zur Prognose und Diagnose technischer Systeme in Anlehnung an Lewandowski (2016)

Methode	Data-Mining Aufgabe	Nutzung in				Anmerkungen	Exemplarische Literaturquellen
		Störungserkennung	Störungsklassifizierung	Ursachenanalyse	Zustandsprognose		
Künstliche neuronale Netze (auch selbstorganisierende KNN)	Segmentierung Klassifikation Regressionsanalyse Assoziationsanalyse	x	x	x	x	Vielseitige Anwendbarkeit in nahezu allen Bereichen; Benötigt eine große Datenmenge (Trainingsdaten); Erfordert Trainingsdaten über verschiedene Fehlerklassen, um sie zu klassifizieren (Supervisor-Lernmethodik); Blackbox, daher nicht immer ganz nachvollziehbar;	Patan 2008; Sikorska et al. 2011; Hayashi et al. 2002
K-Means	Segmentierung Klassifikation	x	x			Identifikation von unbekannten Zusammenhängen in den Daten; Geeignet, wenn keine A-priori-Wissen zur Verfügung steht	Lieber et al. 2013; Jardine et al. 2006
Support Vector Maschine (SVM)	Segmentierung Klassifikation	x	x	x	x	Genau wie KNN kann SVM in allen Bereichen angewendet werden; Im Gegensatz zu KNN nicht geeignet für unsupervised Learning Probleme	Odenthal et al. 2014; Widodo et al. 2007; Saimurugan et al. 2011

Methode	Data-Mining Aufgabe	Nutzung in				Anmerkungen	Exemplarische Literaturquellen
		Störungserkennung	Störungsklassifizierung	Ursachenanalyse	Zustandsprognose		
Entscheidungsbaum	Segmentierung Klassifikation Assoziationsanalyse	x	x	x		Einfach zu entwickeln und zu verstehen; nicht geeignet für Nicht-lineare Probleme	Li et al. 2014; Saimurugan et al. 2011
Autoregressive Moving Average (ARMA - ARIMA)	Regressionsanalyse				x	Einfache und schnelle Anwendung möglich; besser interpretierbar, aber im Allgemeinen weniger flexibel und genau.	Pham et al. 2010; Goyal et al. 2015
Fuzzy Logik	Klassifikation		x	x		basiert auf einem linguistischen Modell; Regelbasiert; die Datenunsicherheit (unpräzise oder unvollständig) ist in der Fuzzy-System betrachtet; längere Laufzeit für komplexe Systeme	Sikorska et al. 2011; Ryll 2008
Wavelet-Analyse / Fourier Transformation	Segmentierung	x				Geeignete Methode zur Analyse von Schwingungen; Transformation lässt Muster manuell erkennen; automatisierte Klassifizierung erfolgt nicht	Goyal et al. 2015; Benkedjouh et al. 2018
Bayes Verfahren	Assoziationsanalyse		x	x		Gut geeignet für Modellierung der Unsicherheit in komplexen technischen Systeme; Rechenintensiv (NP-hard); automatisierte Klassifizierung erfolgt nicht	Ferreiro et al. 2012

Methode	Data-Mining Aufgabe	Nutzung in				Anmerkungen	Exemplarische Literaturquellen
		Störungserkennung	Störungsklassifizierung	Ursachenanalyse	Zustandsprognose		
Expertensysteme	Segmentierung Klassifikation Assoziationsanalyse		x	x		Regebasiert und einfach zu verstehen; Schwierigkeiten beim Wissenserwerb; im Gegensatz zu Fuzzy Logik berücksichtigt keine Datenunsicherheiten	Butler 1996; Simeón et al. 2010

Anhang B – Quellecode zur Störungserkennung an dem Funktionsmodell

```
@App:name("KuehlkreislaufApp")

-- Benutzerdefinierte Funktion, die aus einer Menge von Daten eine Steigung
-- berechnet

define function slope[JavaScript] return double {

   var array = data;
   var cntpoint = array.length / 2;

   var avgx = 0;
   var avgy = 0;

   for(var i = 0; i < cntpoint; i++) {
      avgx = avgx + 1/cntpoint * data[i];
      avgy = avgy + 1/cntpoint * data[cntpoint+i];
   }

   var sumZaeler = 0;
   var sumTeiler = 0;
   for(var i = 0; i < cntpoint; i++) {
     sumZaeler = sumZaeler + (data[i] - avgx) * (data[cntpoint+i] - avgy);
     sumTeiler = sumTeiler + (data[i] - avgx) * (data[i] - avgx);
   }
   return sumZaeler / sumTeiler;

};

-- Definition von Inputströmen der fünf Sensoren
define stream TemperaturInputStream1 (timestamp double , temperature1 double);
define stream TemperaturInputStream2 (timestamp double , temperature2 double);
define stream TemperaturInputStream3 (timestamp double , temperature3 double);
define stream DruckInputStream(timestamp double , druck double);
define stream DurchflussInputStream(timestamp double , durchfluss double);

-- Definition der JOIN-Datenströmen, um die Inputströme
-- miteinander zusammenzuführen

define stream TemperaturJoinStream1_2 (timestamp double ,temperature1 double, tempera-
ture2 double);

define stream TemperaturJoinStream1_2_3 (timestamp double ,temperature1 double, tempera-
ture2 double ,temperature3 double);

define stream Druck_Durchfluss_JoinStream  (timestamp double ,druck double, durchfluss
double);

define stream Sensorereignisstrom(timestamp double ,temperature1 double, temperature2
double, temperature3 double, druck double, durchfluss double);
```

```
define stream Filtersensorereignisstrom(timestamp double ,temperature1 double, tempera-
ture2 double, temperature3 double, druck double, durchfluss double);

-- Definition der Aggregatströme

define stream AggregatSlopesensorereignisstrom(timestamp double ,slope_temperature1 dou-
ble, slope_temperature2 double, slope_temperature3 double, slope_druck double,
slope_durchfluss double);

define stream AggregatStdDevsensorereignisstrom(timestamp double ,stdDev_temperature1
double, stdDev_temperature2 double, stdDev_temperature3 double, stdDev_druck double,
stdDev_durchfluss double);

define stream Aggregatsensorereignisstrom(timestamp double ,stdDev_temperature1 double,
stdDev_temperature2 double, stdDev_temperature3 double, stdDev_druck double,
stdDev_durchfluss double, slope_temperature1 double, slope_temperature2 double,
slope_temperature3 double, slope_druck double, slope_durchfluss double);

-- Definition der komplexen Ereignisströme (Temperaturanstiegsstrom und
-- Druckverluststrom)

define stream Temperaturanstiegsstrom (timestamp double , stoerungsursache string,
stdDev_temperature3 double, slope_druck double, slope_fluss double, slope_temperature3
double);

define stream Druckverluststrom (timestamp double , stoerungsursache string, slope_druck
double, slope_fluss double);

-- Definition von Ereignisregeln zur Verknüpfung verschiedener Inputströme

from TemperaturInputStream1#window.length(2) as t1
  join TemperaturInputStream2#window.length(2) as t2
  on t1.timestamp == t2.timestamp
select t1.timestamp ,t1.temperature1  as temperature1, t2.temperature2 as temperature2
insert into TemperaturJoinStream1_2 ;

from TemperaturInputStream3#window.length(2) as t3   join TemperaturJoin-
Stream1_2#window.length(2) as t1_2   on t3.timestamp == t1_2.timestamp
select t3.timestamp ,t1_2.temperature1 as temperature1, t1_2.temperature2 as temperature2 ,
t3.temperature3  as temperature3
insert into TemperaturJoinStream1_2_3 ;

from DruckInputStream#window.length(2) as P
  join DurchflussInputStream#window.length(2) as F
  on P.timestamp == F.timestamp
select P.timestamp ,P.druck as druck, F.durchfluss as durchfluss
insert into Druck_Durchfluss_JoinStream  ;

from TemperaturJoinStream1_2_3#window.length(2) as t_1_2_3
  join Druck_Durchfluss_JoinStream #window.length(2) as P_F
```

```
  on t_1_2_3.timestamp == P_F.timestamp
select t_1_2_3.timestamp , t_1_2_3.temperature1 ,t_1_2_3.temperature2
,t_1_2_3.temperature3 ,P_F.druck as druck, P_F.durchfluss as durchfluss
insert into Sensorereignisstrom ;

-- Definition von Ereignisregeln zur Bereinigung der Ereignisse aus dem
-- Sensorereignisstrom

from every e1=Sensorereignisstrom, e2=Sensorereignisstrom,
   e3=Sensorereignisstrom[e3.temperature1 != 0 Or e3.temperature2 != 0 Or e3.temperature3
!=0 or e3.durchfluss != 0 or e3.druck != 0]
select e3.timestamp, e3.temperature1 as temperature1, e3.temperature2 as temperature2,
e3.temperature3 as temperature3, e3.druck as druck, e3.durchfluss as durchfluss
insert into Filtersensorereignisstrom;

from every e1=Sensorereignisstrom, e2=Sensorereignisstrom,
   e3=Sensorereignisstrom[e3.temperature1 ==0 and e3.temperature2 ==0 and
e3.temperature3 ==0 and e3.durchfluss ==0 and e3.druck ==0]
select e3.timestamp, (e1.temperature1 + e2.temperature1) / 2 as temperature1,
(e1.temperature2 + e2.temperature2) / 2 as temperature2, (e1.temperature3 + e2.temperature3)
/ 2 as temperature3, (e1.druck + e2.druck) / 2 as druck, (e1.durchfluss + e2.durchfluss) / 2 as
durchfluss
insert into Filtersensorereignisstrom;

-- Definition von Ereignisregeln zur Berechnung von Aggregatfunktionen

from every e1=Filtersensorereignisstrom, e2=Filtersensorereignisstrom,
e3=Filtersensorereignisstrom, e4=Filtersensorereignisstrom, e5=Filtersensorereignisstrom
select e5.timestamp, slope(e1.timestamp, e2.timestamp, e3.timestamp, e4.timestamp,
e5.timestamp, e1.temperature1,e2.temperature1,e3.temperature1,e4.temperature1,
      e5.temperature1) as slope_temperature1, slope(e1.timestamp, e2.timestamp,
e3.timestamp, e4.timestamp, e5.timestamp, e1.temperature2, e2.temperature2,
e3.temperature2, e4.temperature2, e5.temperature2) as slope_temperature2,
slope(e1.timestamp, e2.timestamp, e3.timestamp, e4.timestamp, e5.timestamp,
e1.temperature3, e2.temperature3, e3.temperature3, e4.temperature3,  e5.temperature3) as
slope_temperature3, slope(e1.timestamp, e2.timestamp, e3.timestamp, e4.timestamp,
e5.timestamp, e1.druck,e2.druck,e3.druck,e4.druck, e5.druck) as slope_druck,
 slope(e1.timestamp, e2.timestamp, e3.timestamp, e4.timestamp, e5.timestamp, e1.durchfluss,
e2.durchfluss, e3.durchfluss, e4.durchfluss, e5.durchfluss) as slope_durchfluss
insert into AggregatSlopesensorereignisstrom;

from Filtersensorereignisstrom#window.time(5 second)
select timestamp, stdDev(temperature1) as stdDev_temperature1, stdDev(temperature2) as
stdDev_temperature2, stdDev(temperature3) as stdDev_temperature3, stdDev(druck) as std-
Dev_druck, stdDev(durchfluss) as stdDev_durchfluss
insert into AggregatStdDevsensorereignisstrom;

from AggregatSlopesensorereignisstrom#window.length(2) as e1
  join AggregatStdDevsensorereignisstrom#window.length(2) as e2
  on e1.timestamp == e2.timestamp
```

```
select e1.timestamp, stdDev_temperature1,  stdDev_temperature2, stdDev_temperature3,
stdDev_druck, stdDev_durchfluss, slope_temperature1, slope_temperature2,
slope_temperature3, slope_druck, slope_durchfluss
insert into Aggregatsensorereignisstrom;

-- Definition von Ereignisregeln zur Herleitung von komplexen Ereignissen aus dem
--Aggregatsensorereignisstrom

from Aggregatsensorereignisstrom[slope_druck > -12 and slope_druck < 0 and
stdDev_temperature3 > 2.285 and slope_fluss > -0.35 and slope_temperature3 > 2.1]
select timestamp, "Temperaturanstieg wegen Ausfall des Lüfters" as stoerungsursache,
stdDev_temperature3, slope_druck, slope_fluss, slope_temperature3
insert into Temperaturanstiegsstrom ;

from Aggregatsensorereignisstrom[slope_druck <= -12]
select timestamp, "Druckverlust wegen Defekt der Pumpe" as stoerungsursache, slope_druck,
slope_fluss
insert into Druckverluststrom;

from Aggregatsensorereignisstrom[slope_druck > -12 and slope_druck < 0 and slope_fluss <=
-0.35]
select timestamp, "Druckverlust wegen Leckage" as stoerungsursache, slope_druck,
slope_fluss
insert into Druckverluststrom;
```

Anhang C – Exemplarischer Ausschnitt der Trainingsdaten für den Modellaufbau des zweiten Szenarios

Ja	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Vereisungsprognose	
-0,5	-0,7	0,3	-2,3	-1,6	-1,5	8,8	-3,4	-2,4	-3,8	-0,3	Min	Außentemperatur
0,0	-0,4	0,6	-1,6	-1,2	-0,5	9,3	-1,8	-0,5	-2,4	1,0	Max	
-0,3	-0,6	0,4	-1,9	-1,4	-1,1	9,0	-2,9	-1,4	-3,3	0,3	AVG	
0,0	0,0	0,0	0,1	0,0	0,1	0,0	0,2	0,3	0,1	0,2	VAR	
5,3	5,5	5,1	5,7	9,0	8,2	5,5	2,2	2,8	5,9	3,5	Min	Windgeschwindigkeit
6,9	8,0	6,8	8,4	14,7	13,8	7,8	5,7	4,8	10,2	4,8	Max	
6,1	6,6	5,9	6,9	11,2	10,5	6,4	4,0	4,0	7,8	4,0	AVG	
0,3	0,8	0,4	0,6	3,4	2,7	0,4	1,0	0,3	1,7	0,2	VAR	
19,4	26,0	33,8	30,9	3,8	8,8	40,6	15,7	26,1	36,3	14,4	Min	Generatortemperatur
22,6	31,6	45,0	39,4	4,2	10,8	45,1	19,6	32,4	60,5	29,2	Max	
20,8	29,3	37,2	33,8	4,0	9,8	42,8	17,8	30,2	50,8	23,5	AVG	
0,8	4,2	9,8	6,7	0,0	0,4	1,7	1,3	4,8	73,3	30,4	VAR	
36,0	61,3	73,9	73,8	9,3	15,9	70,3	35,5	59,1	73,2	56,1	Min	Getriebetemperatur
62,5	65,7	79,1	79,2	10,9	18,5	78,8	43,3	67,4	77,6	59,3	Max	
45,6	64,3	76,5	75,7	10,1	17,2	74,6	38,4	64,7	75,9	57,7	AVG	
69,7	1,2	2,4	3,4	0,3	0,7	4,4	6,4	6,5	2,3	1,0	VAR	
9,5	13,1	17,3	16,1	6,5	7,8	22,5	8,9	15,2	16,5	13,9	Min	Gondeltemperatur
13,0	14,4	18,2	20,0	7,6	8,8	26,9	10,2	17,6	29,4	15,4	Max	
10,8	13,6	17,7	17,4	7,1	8,3	25,3	9,5	16,9	20,7	14,6	AVG	
1,1	0,1	0,1	1,5	0,1	0,1	1,7	0,1	0,3	11,0	0,1	VAR	
46,0	46,0	293,0	223,0	24,0	24,0	240,0	167,0	100,0	97,0	118,0	Min	Gondelposition
59,0	52,0	299,0	232,0	24,0	48,0	251,0	247,0	121,0	105,0	166,0	Max	
52,7	48,9	293,8	227,3	24,0	25,3	244,6	229,6	109,6	101,3	147,1	AVG	
15,1	3,1	3,8	7,4	0,0	32,0	9,1	302,4	29,7	5,2	279,5	VAR	
0,0	3,3	8,7	9,4	0,0	0,0	10,3	0,0	0,0	11,4	7,6	Min	Rotordrehzahl
5,9	8,1	11,8	12,5	0,0	0,0	12,3	3,0	8,2	12,5	7,9	Max	
0,4	7,6	10,3	11,4	0,0	0,0	11,6	0,3	7,4	12,3	7,8	AVG	
46,0	46,0	293,0	223,0	0,0	0,0	0,4	0,5	3,6	0,1	0,0	VAR	

Anhang D – Quellecode zur Vereisungsvorhersage an den Rotorblattern der Onshore-Windenergieanalagen

```
@App:name("Vereisungsvorhersage")

-- Definition des Parameterinputstroms
define stream parameterinputstrom(timestamp long, windgeschwindigkeit double, gondeltem-
peratur double, generatortemperatur double, getriebetemperature double, gondelposition dou-
ble, rotordrehzahl double, anlageName string);

-- Definition des Pastaussentemperaturstroms
define stream pastaussentemperaturstrom(timestamp long, aussentemperatur double, anlage-
Name string);

-- Definition des Prognoseaussentemperaturstroms, wobei die aussentemperatur1 die
-- Prognose der Außentemperatur der nächsten 10 Minuten und aussentemperatur6 die
-- Prognose der Außentemperatur nach einer Stunde ist.
define stream prognoseaussentemperaturstrom(timestamp long, aussentemperatur1 double,
aussentemperatur2 double, aussentemperatur3 double, aussentemperatur4 double, aussentem-
peratur5 double, aussentemperatur6 double, anlageName string);

-- Definition des Außentemperaturstroms
define stream aussentemperaturstrom(timestamp long, aussentemperatur double, anlageName
string);

-- Definition des Aggregatstroms
define stream aggregatstrom(timestamp long, min_wg double, max_wg double, avg_wg dou-
ble, stdDev_wg double,  min_gondel_T double, max_gondel_T double, avg_gondel_T dou-
ble, stdDev_gondel_T double,   min_genT double, max_genT double, avg_genT double,
stdDev_genT double,   min_getriebe_T double, max_getriebe_T double, avg_getriebe_T dou-
ble, stdDev_getriebe_T double,  min_gondel_P double, max_gondel_P double, avg_gondel_P
double, stdDev_gondel_P double,  min_rotordrehzahl double, max_rotordrehzahl double,
avg_rotordrehzahl double, stdDev_rotordrehzahl double, min_aussenT double, max_aussenT
double, avg_aussenT double, stdDev_aussenT double, anlageName string);

-- Definition des NNvorhersagestroms
Define stream NNvorhersagestrom (timestamp long, vereisungvorhersage bool, anlageName
string);

-- Definition des Vereisungsstroms
Define stream vereisungsstrom (timestamp long, anlageName string, vereisungsmeldung
string);

-- Definition von Ereignisregeln zur Erstellung des Außentemperaturstroms aus dem 12
 --vergangenen Pastaußentemperaturen

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
```

```
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e1.timestamp, e1.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e2.timestamp, e2.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e3.timestamp, e3.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e4.timestamp, e4.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e5.timestamp, e5.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e6.timestamp, e7.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;
```

```
from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e8.timestamp, e8.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e9.timestamp, e9.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e10.timestamp, e10.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e11.timestamp, e11.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

from every e1=pastaussentemperaturstrom, e2=pastaussentemperaturstrom,
e3=pastaussentemperaturstrom, e4=pastaussentemperaturstrom,
e5=pastaussentemperaturstrom, e6=pastaussentemperaturstrom,
e7=pastaussentemperaturstrom, e8=pastaussentemperaturstrom,
e9=pastaussentemperaturstrom, e10=pastaussentemperaturstrom,
e11=pastaussentemperaturstrom, e12=pastaussentemperaturstrom
select e12.timestamp, e12.aussentemperatur, e1.anlageName
insert into aussentemperaturstrom;

-- Definition von Ereignisregeln zur Erstellung des Außentemperaturstroms aus dem
-- sechs zukünftigen prognoseaussentemperaturen

from prognoseaussentemperaturstrom
select timestamp, aussentemperatur1 as aussentemperatur, anlageName
insert into aussentemperaturstrom;
```

```
from prognoseaussentemperaturstrom
select timestamp, aussentemperatur2 as aussentemperatur, anlageName
insert into aussentemperaturstrom;

from prognoseaussentemperaturstrom
select timestamp, aussentemperatur3 as aussentemperatur, anlageName
insert into aussentemperaturstrom;

from prognoseaussentemperaturstrom
select timestamp, aussentemperatur4 as aussentemperatur, anlageName
insert into aussentemperaturstrom;

from prognoseaussentemperaturstrom
select timestamp, aussentemperatur5 as aussentemperatur, anlageName
insert into aussentemperaturstrom;

from prognoseaussentemperaturstrom
select timestamp, aussentemperatur6 as aussentemperatur, anlageName
insert into aussentemperaturstrom;

-- Definition der Ereignisregel zur Erstellung des Aggregatstroms aus den
-- beiden Ereignisströmen aussentemperaturstrom und parameterinputstrom für
-- ein 3-stündiges Zeitfenster

from aussentemperaturstrom#window.time(3 Hour) as e1 join parameterinput-
strom#window.time(3 Hour) as e2 on e1.timestamp == e2.timestamp
select e1.timestamp,   minimum(e2.windgeschwindigkeit) as min_wg, maxi-
mum(e2.windgeschwindigkeit) as max_wg, avg(e2.windgeschwindigkeit) as avg_wg, std-
Dev(e2.windgeschwindigkeit) as stdDev_wg,  minimum(e2.gondeltemperatur) as
min_gondel_T, maximum(e2.gondeltemperatur) as max_gondel_T, avg(e2.gondeltemperatur)
as avg_gondel_T, stdDev(e2.gondeltemperatur) as stdDev_gondel_T,     mini-
mum(e2.generatortemperatur) as min_genT, maximum(e2.generatortemperatur) as max_genT,
avg(e2.generatortemperatur) as avg_genT, stdDev(e2.generatortemperatur) as stdDev_genT,
minimum(e2.getriebetemperature) as min_getriebe_T, maximum(e2.getriebetemperature) as
max_getriebe_T, avg(e2.getriebetemperature) as avg_getriebe_T, std-
Dev(e2.getriebetemperature) as stdDev_getriebe_T,    minimum(e2.gondelposition) as
min_gondel_P, maximum(e2.gondelposition) as max_gondel_P, avg(e2.gondelposition) as
avg_gondel_P, stdDev(e2.gondelposition) as stdDev_gondel_P,  minimum(e2.rotordrehzahl)
as min_rotordrehzahl, maximum(e2.rotordrehzahl) as max_rotordrehzahl,
avg(e2.rotordrehzahl) as avg_rotordrehzahl, stdDev(e2.rotordrehzahl) as std-
Dev_rotordrehzahl,    minimum(e1.aussentemperatur) as min_aussenT, maxi-
mum(e1.aussentemperatur) as max_aussenT, avg(e1.aussentemperatur) as avg_aussenT, std-
Dev(e1.aussentemperatur) as stdDev_aussenT  , e1.anlageName
insert into aggregatstrom;

-- Definition der Ereignisregel zur Berechnung des ML-Modell-Outputs'

from aggregatstrom#tensorFlow:predict('C:/wso2sp-
4.4.0/samples/artifacts/TensorflowSample/Regression', 'inputPoint', 'outputPoint', min_wg,
max_wg, avg_wg, stdDev_wg,  min_gondel_T, max_gondel_T, avg_gondel_T, std-
```

```
Dev_gondel_T,   min_genT, max_genT, avg_genT, stdDev_genT, min_getriebe_T,
max_getriebe_T, avg_getriebe_T, stdDev_getriebe_T, min_gondel_P, max_gondel_P,
avg_gondel_P, stdDev_gondel_P, min_rotordrehzahl, max_rotordrehzahl, avg_rotordrehzahl,
stdDev_rotordrehzahl, min_aussenT, max_aussenT, avg_aussenT, stdDev_aussenT);
select timestamp, outputPoint1 as vereisung
insert into NNvorhersagestrom;

-- Definition der Ereignisregel zur Detektion von Vereisungen dem NNvorhersagestrom

from NNvorhersagestrom[vereisung == true]
select timestamp, anlageName, "Vereisungssituation detektiert" as vereisungsmeldung
insert into vereisungsstrom;
```