DIPLOMARBEIT

Monitoring energietechnischer Referenzanlagen

Ludwig-Maximilians-Universität München
Fakultät: Informatik
Studiengang: Informatik (Diplom)

Autor: Christian Müller
Betreuer: Christian Hilgers, ZAE Bayern
Prüfer: Prof. François Bry

Ausgabedatum: 29.06.2012
Abgabedatum: 21.12.2012

Erklärung

Hiermit erkläre ich gemäß § 31 Abs. 7 der Rahmenprüfungsordnung, dass ich die vorliegende Arbeit mit dem Titel „Monitoring energietechnischer Referenzanlagen" selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, den 21. Dezember 2012

_____

Unterschrift

Danksagung

Zunächst gilt mein Dank Prof. Dr. François Bry, der sich bereit erklärt hat, diese Diplomarbeit an einem externen Institut zu betreuen. Er hatte während der gesamten Zeit immer ein offenes Ohr für mich und ermunterte mich stets bei meiner Arbeit.

Ganz besonders möchte ich mich bei Christian Hilgers bedanken, der mich am ZAE Bayern über seine Betreuung hinaus mit technischer Hilfe und richtungsweisenden Tipps unterstützt hat. Insbesondere bedanke ich mich für das geduldige Lesen und die vielen konstruktiven Kommentare, die zur Qualität dieser Diplomarbeit enorm beigetragen haben. Weiterer Dank gilt Martin Helm, der mir als Betreuer der Referenzanlage hilfreich im Aufdecken von Fehlern und Erklären von technischem Sachverhalt zur Seite gestanden hat.

Einen ganz besonderen Dank richte ich an meine Eltern, die mich über die gesamte Zeit meines Studiums unterstützt haben und immer voll hinter meinem Tun und Wirken stehen.

Am herzlichsten bedanken möchte ich mich bei meiner eigenen, kleinen Familie, allen voran bei meiner Frau Alexandra, die mir den Rücken freigehalten hat, wo sie nur konnte, bei meiner Tochter Lena, die ich während dieser Zeit hoffentlich nicht zu sehr vernachlässigt habe und bei meinem neugeborenen Sohn Toni, der sich wirklich (extra für mich?) mit nächtlichem Geschrei zurückgehalten hat.

Abstract

Im Rahmen dieser Diplomarbeit wird ein Monitoring energietechnischer Referenzanlagen entwickelt. Dabei werden verschieden Fehlerklassen und Monitoring Methoden betrachtet. Nach der Auswahl einer geeigneten Referenzanlage wird besonderer Wert auf die Softwareanalyse gelegt. Unterschiedliche Business Rule Management Systeme werden auf ihre Tauglichkeit zur Umsetzung eines regelbasierten Monitorings untersucht. Das ausgewählte BRMS JBoss Drools wird für die Implementierung verwendet und in die bestehende Softwarearchitektur eingebunden. Theoretische Betrachtungen der Wissensverarbeitung und des Rete-Algorithmus vertiefen das Verständnis zur Regeldefinition. Durch Wissensextraktion aus Referenzkurven werden spezifische Regel-Parameter bestimmt. Abschluss der Arbeit stellen Ansätze zur Erweiterung dar, um mittels Simulation und neuronaler Netze das Monitoring Konzept für energietechnische Referenzanlagen weiter voranzutreiben.

Within this diploma thesis, a monitoring for an energy technology reference system is developed. Different error classes and monitoring methods are considered. After the selection of a suitable reference system, special emphasis is placed on the software analysis. Different business rule management systems are examined for their suitability to implement a rule-based monitoring. The selected JBoss Drools BRMS is used for implementation and is integrated into the existing software architecture. Theoretical considerations of knowledge engineering and the Rete algorithm deepen the understanding of rule definition. By knowledge extraction from reference curves specific rule parameters are determined. Completion of the thesis is the enhancement approach to use simulation and neural networks to further the monitoring concept of energy technology reference systems.

Inhaltsverzeichnis

Erklärung I

Danksagung II

Abstract III

Inhaltsverzeichnis IV

1 Einleitung 1

1.1 Motivation 1

1.2 Beispiele 2

1.3 Zielsetzung 2

2 Stand der Technik 3

2.1 Historie 3

2.2 Expertensysteme 4

2.3 Intelligente Systeme 6

2.4 Einsatz in der Anlagentechnik 7

3 Exkurs: Analyse der Referenzanlage 8

3.1 Aufbau der Referenzanlage 9

3.2 Perspektiven für die Anlagenüberwachung 11

4 Fehlerklassen 11

4.1 Sensorfehler 11

4.2 Steuer- bzw. Regelfehler 14

4.3 Komponentenfehler 15

5 Monitoring Methoden 15

5.1 Regelbasiert 15

5.2 Simulation und Schätzung 16

5.3 Neuronale Netze 17

6 Softwareanalyse für das regelbasierte Monitoring 17

6.1 Vorabüberlegungen zur Umsetzung 17

6.2 Sondierung von Business-Rule-Management-Systemen 19
6.2.1 d3web 19
6.2.2 SPIRIT 20
6.2.3 OpenRules 21
6.2.4 IBM WebSphere Operational Decision Management 21
6.2.5 JBoss Drools 22
6.3 Theorie: Der Rete-Algorithmus 23
6.4 Beurteilung und Auswahl des BRMS 28
7 Implementierung des regelbasierten Monitoring 30
7.1 Einbindung der Messdatenbank in die Wissensbasis 30
7.2 Regeldefinition 33
7.2.1 Regelsyntax 33
7.2.2 Beispielregeln 35
7.2.3 Übergabe der Regeln in die Wissensbasis 40
7.3 Allgemeine Detektion von Sensorfehlern 41
7.3.1 Parameterbestimmung 42
7.3.2 Parameterwissen 47
8 Ansätze zur Erweiterung 49
8.1 Simulationsdaten und Langzeiteffekte 49
8.2 Möglichkeiten durch neuronale Netze 50
9 Fazit und Ausblick 52
Abbildungsverzeichnis VI
Abkürzungsverzeichnis VIII
Literaturverzeichnis IX

1 Einleitung

1.1 Motivation

Die Abteilung „Technik für Energiesysteme und Erneuerbare Energien” des Bayerischen Zentrums für Angewandte Energieforschung e.V. (ZAE Bayern) in Garching forscht im Bereich der Energiespeicherung sowie Energieoptimierung von Gebäuden und Prozessen. So ist das ZAE Bayern beispielsweise auf dem Gebiet der Absorptionskältemaschinen mit Latentwärmespeichertechnik, die zum Heizen und Kühlen von Gebäuden verwendet werden, sowie der thermischen Nutzung von Solarenergie tätig. In diesem Rahmen arbeitet das ZAE Bayern an der Entwicklung von innovativen Komponenten und an der Planung energietechnischer Referenzanlagen.

Diese Anlagen verlangen nach einer sorgfältigen Überwachung des Gesamtsystems. Experten und Fachanwender übernehmen bislang den Großteil der Monitoring Aufgaben selbst, und beherbergen somit das spezifische Fachwissen über die Referenzanlage und ihrer Fehlervorkommen. Wechselt der Betreuer der Anlage beispielsweise durch Arbeitsplatzwechsel, so muss sichergestellt sein, dass der neue Mitarbeiter sämtliches Fachwissen über die Messeinrichtung erhält. Dazu gehört auch die Kenntnis über Störquellen und deren Behebung.

Es kann nicht gewährleistet werden, dass der Experte zu jeder Zeit über alle Vorgänge in der Anlage Bescheid weiß. Die nachsorgende Betreuung einer installierten und geprüften Messanlage ist kein full-time Job, sondern wird häufig beiläufig betrieben. Das Erreichen von Projektzielen kann gefährdet werden, wenn zum Beispiel aufgrund eines spät erkannten Fehlers Messdaten nicht aufgezeichnet wurden, die aber zu einer Auswertung oder Bilanzierung nötig gewesen wären oder die Anlage fehlerhaft lief.

Das Monitoring durch eine Softwarelösung kann in vielen Aspekten unterstützend für Fachanwender und den Betrieb sein. Dem Experten wird mit Hinweisen geholfen, dass der Fehler frühzeitig erkannt wird und gegebenenfalls Gegenmaßnahmen eingeleitet werden können. Der Betrieb profitiert davon, dass Teile des Transferwissens bei Messanlagenübergabe nicht verloren gehen und weniger Zeit aufgewendet werden muss einen neuen Mitarbeiter anzulernen.

1.2 Beispiele

Das ZAE Bayern entwickelt und betreut energietechnische Referenzanlagen im Bereich der erneuerbaren Energien. Mit teilweise über 3000 Messpunkten stellen diese Systeme eine Herausforderung an die Überwachung dar. Diese Messanlagen sind meist einzigartig auf ihrem Gebiet und befinden sich auf dem aktuellen Stand der Technik im Bereich der Energiesysteme. Die Anlagenkomponenten werden teilweise von Hand am Standort in Garching gefertigt und dann in Pilotanlagen eingesetzt. Der komplexe Aufbau der Komponenten selbst und die Verknüpfung der Komponenten zusammen in einem noch komplexeren Energiesystem, erfordern, durch Installation von Messapparaturen, die Analyse und Bewertung über die Vorgänge im System.
Bei einer so hohen Dichte an Messpunkten und Prototypen, treten mitunter unweigerlich Störfälle und Messtechnik- und Anlagenfehler auf. Hinzu kommt, dass die energietechnischen Referenzanlagen aufgrund ihres Prototypenstatus nicht nur bekannte Betriebszustände bedienen müssen. Damit sie wissenschaftlich begleitet und erforscht werden können, müssen die Anlagen teilweise in Grenzbereichen betrieben werden, für die sie normalerweise nicht konzipiert sind. Dies eröffnet ein weiteres Feld für mögliche Störursachen, die es zu bewältigen gilt.
Zudem betreibt das ZAE Bayern auch mobile Messstationen, die der natürlichen Witterung ausgesetzt sind, welche dann, besonders im Winter und Sommer zum Teil extremen Temperaturen ausgesetzt sind. Somit ist es nicht verwunderlich, wenn hier und da Fehler in der Messtechnik auftreten, die aufgrund von Frost oder Überhitzung entstanden sind. Außerdem sind solche mobilen Messstationen häufig auf Baustellen zugegen, auf denen es beispielsweise nicht selten zu Stromausfällen oder Kabeldurchtrennungen kommt.
All diese Beispiele zusammen genommen, deuten auf ein hohes Fehlerpotential hin, welche ein Monitoring unabdingbar für die Erkennung von Störungen machen.

1.3 Zielsetzung

Im Rahmen dieser Diplomarbeit soll untersucht werden, wie das Monitoring energietechnischer Referenzanlagen umgesetzt werden kann. Neben ausgiebiger Einarbeitung in die Thematik soll sich ein Überblick verschafft werden, worauf es bei der Überwachung von komplexen Messeinrichtungen ankommt, welche Fehler auftreten können, und wie man sie detektieren kann. Dazu ist es notwendig sich intensiv mit

Fehlerklassen zu beschäftigen, die auch in der Realität auftreten können. Weiterhin ist es notwendig sich mit der Wissensverarbeitung auseinanderzusetzen, damit anwendbare Methoden für ein Monitoring gefunden werden können.
Es soll eine Referenzanlage bestimmt werden, die sowohl eine Vielzahl von Monitoring Aspekten miteinander vereint, als auch den Spielraum bietet, möglichst viele Fehlerklassen zu verifizieren und praktisch nachzuweisen.
Der eigentlichen Implementierung soll eine Softwareanalyse vorausgehen, mit dem Ziel, die aktuell auf dem Markt erhältliche Software für die Tauglichkeit in einem Monitoring Kontext zu ermitteln. Auf Basis dieser Untersuchungen soll eine Software ausgewählt werden, die sich aufgrund ihrer Eigenschaften für das Monitoring eignet. Die Software soll in die Softwarearchitektur des ZAE Bayern integriert werden um eine bestmögliche Vernetzung mit den Messdaten zu gewährleisten und die Adaptierbarkeit auf andere Referenzanlagen zu ermöglichen.
Für die ermittelten Fehlerklassen sollen Regeln für das regelbasierte Monitoring entworfen und implementiert werden, die sich anhand der Messdaten verifizieren lassen können. Dadurch soll eine Grundlage geschaffen werden, die es ermöglicht, weitere Monitoring Aufgaben bereitzuzustellen. Durch die Kenntnis weiterer Monitoring Methoden, soll der regelbasierte Ansatz weiter vertieft und verbessert werden. Zum Abschluss der Arbeit kann dazu übergegangen werden, weitere Ansätze zu verfolgen und sie für die Tauglichkeit in Bezug auf das Monitoring energietechnischer Referenzanlagen hin zu überprüfen.

2 Stand der Technik

In diesem Abschnitt wird erläutert, welche Möglichkeiten und Anwendungsgebiete der Wissensverarbeitung es gibt und welche Rolle der Wissensverarbeitung heutzutage in der Informatik zugetragen wird. Angelehnt an die Einführungskapitel des Buches „Wissensverarbeitung“ [Boe07] wird ein anschaulicher Einblick in die Thematik der Künstlichen Intelligenz gegeben. Zudem wird der Einsatz in der Anlagentechnik und deren Tätigkeitsfeld untersucht.

2.1 Historie

Der Begriff „Künstliche Intelligenz“ (kurz KI) wurde erstmals in den 60er-Jahren des 20. Jahrhunderts aufgeworfen, als versucht wurde einen allgemeinen Problemlöser zu entwickeln, der u.a. Rätselaufgaben lösen sollte. Der Fokus lag dabei auf der

Entwicklung effizienter Suchstrategien. 10 Jahre später stellte sich heraus, dass ein allgemeiner Ansatz nicht zielführend ist, und man konzentrierte sich darauf, bereichsspezifische Lösungen zu finden. Es entstanden die ersten Expertensysteme, die regelbasiert waren und zum Teil bereits unsicheres Wissen behandeln konnten. Weitere 10 Jahre später wurde erkannt, dass sich die bis dahin entwickelten Systeme meist nur im Expertenwissen unterschieden, aber gleiche Wissensrepräsentation und Problemlösungsmethoden enthielten. Man begann gezielt Expertensystemschalen (Steuerungssystem eines Expertensystems) für bestimmte Problemlösungstypen zu entwickeln, die dann in hybrider Form mit verschiedenen Wissensrepräsentationsformalismen aufwarten konnten. Der Erfolg von reinen Expertensystemen blieb jedoch aus und man versucht heutzutage nicht mehr menschliche Experten komplett ersetzen zu wollen. Diese Erkenntnis fördert die sogenannten „Intelligenten Systeme“, welche sich als Integration der Wissensverarbeitung in herkömmliche Software oder technische Systeme versteht. Die Zukunft der „Künstlichen Intelligenz“ ist heute am Beispiel „Semantic Web“ zu sehen. Hierbei versucht man das in Webseiten vom Menschen gespeicherte Wissen für Computer verständlich und interpretierbar zu machen, damit es automatisch weiterverarbeitet werden kann.

2.2 Expertensysteme

Expertensysteme sind wissensbasierte Systeme, die neben einer Wissensbasis und der Problemlösungskomponente über weitere Komponenten zur Benutzerinteraktion verfügen. Die Architektur von Expertensystemen ist in der folgenden Grafik (Abbildung 1) veranschaulicht:

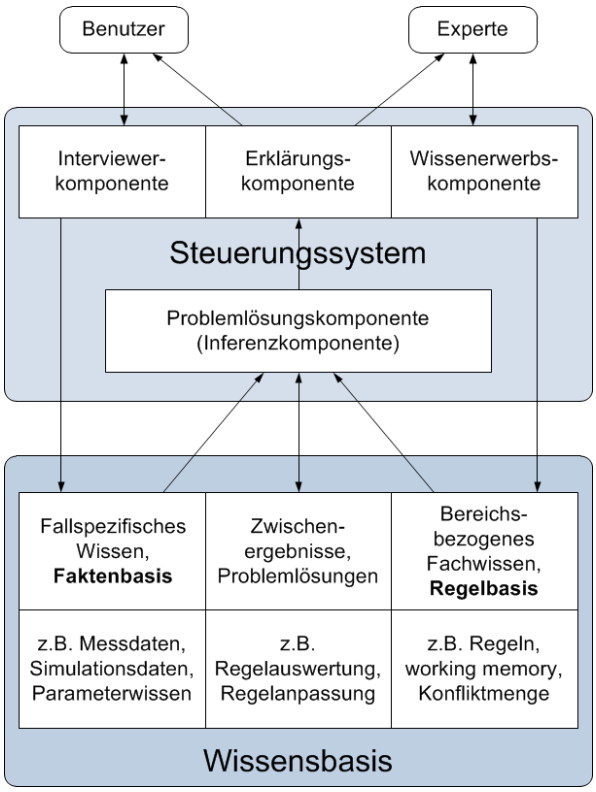

Abbildung 1: Architektur von Expertensystemen [Boe071]

Die zwei Hauptbestandteile sind das Steuerungssystem und die Wissensbasis. Innerhalb des Steuerungssystems ist die Problemlösungskomponente dafür zuständig, das Expertenwissen zu interpretieren, eine Problemlösung zu generieren und diese an die Erklärungskomponente weiterzuleiten. Die Interviewerkomponente ist für den Dialog mit dem Benutzer und das Einlesen von Messdaten verantwortlich, und erzeugt so das fallspezifische Wissen. Die Erklärungskomponente macht dem Menschen die Problemlösung verständlich, indem sie die Vorgehensweise des Systems transparent macht und eine Begründung für den vorgeschlagenen Lösungsweg liefert. Die Wissenserwerbskomponente stellt die Schnittstelle zwischen dem Experten und dem bereichsbezogenem Fachwissen dar und sorgt dafür, dass das Fachwissen aufgebaut, gewartet und erweitert wird.

Die Wissensbasis beinhaltet das bereichsbezogene Wissen, welches vom Experten, meist in Form von Regeln, beigesteuert wird. Das fallspezifische Wissen liefert der Benutzer selbst, welches sich auf den konkreten Anwendungsfall bezieht und meist in Form von Fakten hinterlegt ist. Die Zwischenergebnisse und Problemlösungen ergeben sich aus dem wechselseitigen Austausch mit der Inferenzkomponente.

Der Vorteil dieses zweistufigen Modells besteht darin, die sogenannte Expertensystemschale, also das Steuerungssystem mit unterschiedlichen Wissensbasen koppeln

zu können, da sich die Expertensystemanwendungen oft nur im Anwendungsbereich, aber nicht in den verwendeten Problemlösungsmethoden unterscheiden.
Expertensysteme werden heute als nützliche Werkzeuge zur Unterstützung von Fachleuten angesehen und können ihre Stärke da ausspielen, wo Menschen aufgrund der Komplexität der Probleme überfordert sind.

2.3 Intelligente Systeme

Im Gegensatz zu reinen KI-Systemen, wie den Expertensystemen, sind intelligente Systeme heutzutage von viel größerer Bedeutung. Der Ansatz, die Wissensverarbeitung und Nachbildung menschlicher Intelligenz in Software und Anlagen zu integrieren, und dabei nach außen hin unsichtbar zu sein, trägt wesentlich zum Erfolg dieser Systeme bei. Die Leistungsfähigkeit solcher Systeme hängt von den Komponenten und deren Zusammenspiel ab. Oft wird dabei nur das Produkt (PKW, Flugzeug) oder die Anlage (Kernkraftwerk, Stromnetz) als solches wahrgenommen und der hohe Softwarenteil übersehen, sowie die Bedeutung der Informatik unterschätzt. Die Integration der Wissensverarbeitung in Gegenstände unserer Umwelt bezeichnet man als „Umgebungsintelligenz“. Ein intelligentes System zeichnet sich dadurch aus, dass es mehrere Prinzipien der Wissensverarbeitung nutzt, um komplexe und dynamische Umgebungen beherrschbar zu machen.
Ein wichtiger Bestandteil ist dabei die Wissensrepräsentation, welche zur Aufgabe hat, das Wissen über den jeweiligen Anwendungsbereich darzustellen. Dazu gehört auch das Wissen über den aktuellen Zustand eines Systems, sowie die Repräsentation von Wissen über Möglichkeiten einen zukünftigen Zustand zu erreichen. Des Weiteren tragen effiziente Suchverfahren grundlegend zur Problemlösungsfindung in intelligenten Systemen bei. Sind die Anforderungen komplexer, gestatten es Constraintgraphen sich zielgerichteter in großen Suchbäumen zu bewegen. Mit der Aussagen- und Prädikatenlogik lassen sich Fakten, Regeln und Anfragen bearbeiten und beweisen, was insbesondere bei regelbasierten Systemen, wie den Expertensystemen zum Tragen kommt. Ist das Wissen, welches das intelligente System zur Verfügung hat unvollständig, kann man Annahmen über das fehlende Wissen treffen, z.B. eine Simulation als Referenzwissen heranziehen. Hierbei ist es wichtig, gegebenenfalls die Annahmen zu revidieren und die Wissensbasis dementsprechend zu aktualisieren. Einen Schritt weiter geht die Verarbeitung von unsicheren und vagen Daten, wie z.B. die Fuzzy-Logik, da von intelligenten Systemen erwartet wird, Entscheidun-

gen trotz unsicherer Informationen treffen zu können. Außerdem können Wissensrepräsentationen nicht nur manuell, sondern auch durch Methoden des maschinellen Lernens erstellt werden, z.B. über künstliche, neuronale Netze. Zeitliche Zusammenhänge, welche zum Teil bekannt sind bzw. abgeleitet werden sollen, sowie die Überprüfung auf Inkonsistenz von zeitlichem Wissen, spielen in vielen Anwendungen eine wichtige Rolle. Zu guter Letzt gehört die reaktive Handlungsplanung zu den Kernaufgaben autonomer mobiler Systeme, damit ein System adäquat auf eintretende Ereignisse reagieren kann.

Die Möglichkeiten intelligente Systeme zu erschaffen sind vielseitig. Es hängt vom speziellen Anwendungsfall ab, welche Prinzipien der Wissensverarbeitung man in die Anlagensoftware integriert, damit der Vorteil, nämlich die bestmögliche Erkennung von Fehlern, und die Unterstützung des Benutzers, zum Tragen kommt.

2.4 Einsatz in der Anlagentechnik

In der Literatur finden sich einige Beispiele für den Einsatz von intelligenten Systemen zur Anlagenüberwachung. Häufig liegt der Fokus des Monitorings dabei auf der Ausfallsicherung der Anlage.

Im Zuge der Energiewende werden beispielsweise riesige Windradparks zum Teil offshore als auch auf dem Festland installiert. Damit diese erneuerbaren Energiequellen zukünftig die konventionelle Energiegewinnung ablösen können, muss gewährleistet sein, dass die Windkraftanlagen technisch ausfallsicher sind. Die Veröffentlichung „Condition Monitoring Systeme für Windenergieanlagen - Einsatz und Zertifizierung“ [Muu05] beschreibt die Möglichkeiten eines Condition Monitoring Systems (kurz CMS) zur Zustandsüberwachung und zustandsorientierten Instandhaltung. Mithilfe von Sensoren werden die Windräder auf ihren technischen Zustand hin überprüft und gegebenenfalls rechtzeitig vor einem Totalausfall zur Wartung vorgeschlagen. Um die Anlage wirtschaftlich betreiben zu können, muss darauf geachtet werden, dass die Instandhaltung nicht zu früh angesetzt wird, um die Wartungskosten gering zu halten. Ein CMS hilft dabei die Balance aus Ausfallsicherheit und Kostenminimierung zu finden.

Ein weiteres Beispiel für ein Überwachungs- und Instandhaltungssystem ist das das Human Interface Supervision System (kurz HISS) der Firma Siemens AG. Das erstmals auf einer Erdgasförderanlage installierte System überwacht seine Umgebung mittels verschiedener Sensoren. Es besteht aus einem Bildverarbeitungssystem mit

Kameras, einer Audio Sensorik mit Mikrofonen und einer elektronischen Nase, die Gasleckagen detektieren kann. „Die Sensorik kommuniziert mit einer Auswerteeinheit, die mittels Fuzzy-Logik eine Bewertung der Signale anhand hinterlegter Szenarien vornimmt und bei Abweichungen vom normalen Anlagenbetrieb eine Meldung an den Systemoperator absetzt. Zudem kann das System Meldungen für Wartungsdienste erzeugen. HISS trägt zur Sicherheit der Anlage bei, erhöht die Verfügbarkeit und kann zu enormen Einsparungen bei der Instandhaltung beitragen. Übertragbar ist das System auf andere Bereiche wie die chemische Industrie, Petrochemie oder die Lebensmittelverarbeitung.“ [dre]

Neue Energiequellen, zum Beispiel Photovoltaik oder Windkraft, wirken sich durch die Fluktuation der Energiegewinnung auch auf die Art der Verteilung in den Stromnetzen aus. Sogenannte Smart Grids sollen in Zukunft dezentralisiert die Architektur heutiger zentraler Stromverteilsysteme ablösen. Die damit hereingehenden Schwankungen im Stromnetz müssen erkannt und präventiv ausgeglichen werden. Diesen Gesichtspunkten widmet sich die Veröffentlichung „Data Quality Assurance and Performance Measurement of Data Mining for Preventive Maintenance of Power Grid” [Leo11]. Für das Beispiel des Stromnetzes von New York City, welches häufig, vor allem in jüngerer Vergangenheit mit Stromausfällen zu kämpfen hat, entstand ein Evaluations-Framework namens NOVA (Neutral Online Visualization-aided Autonomic evaluation framework). Es evaluiert die Qualität der Messdaten mit den Ergebnissen des Data Minings und des maschinellen Lernens zusammen mit der Verlässlichkeit der Verbesserungsstrategien von eingesetzten Expertensystemen untereinander. MartaRank und ODDS sind zwei online machine learning und data mining-based feeder ranking Systeme zur präventiven Wartung, welche durch NOVA evaluiert werden konnten und nachweislich zur Effizienzsteigerung und Ausfallsicherheit beitrugen, wenn sie in Stromnetzen eingesetzt wurden.

Diese Beispiele zeigen, wie notwendig das Thema Monitoring gerade auch im Bereich der erneuerbaren Energien ist und lassen vermuten, dass in Zukunft komplexe Überwachungsstrategien weiter an Bedeutung gewinnen werden.

3 Exkurs: Analyse der Referenzanlage

Für das Monitoring soll eine Referenzanlage dienen, die typisch für viele Messaufbauten des ZAE Bayern ist und nach Möglichkeit eine große Bandbreite von Kompo-

nenten besitzt, so dass man unterschiedliche Aspekte der Anlagenüberwachung betrachten kann. In Zusammenarbeit mit den Fachanwendern des ZAE Bayern wurde eine Art Ausschreibung verfasst, die nach Rückmeldung von den Experten ausgewertet wurde. Insbesondere lag der Fokus dabei auf der Aktualität und Aktivität der Anlage, sowie der Bereitschaft des betreuenden Experten, Hilfestellung im wissenschaftlich-technischen Bereich beizusteuern.
Die Auswahl fiel letztendlich auf eine Gebäudeversorgungsanlage, die vor Ort am Institutsgebäude des ZAE Bayern installiert ist. Das Versorgungskonzept wurde am ZAE Bayern entwickelt und trägt den Namen „Solares Heizen und Kühlen mit Absorptionskältemaschine und Latentwärmespeicher“ (kurz SolarHeatCool+PCM).

Die Vorteile gegenüber den anderen, in Frage gekommenen Referenzanlagen, bezogen auf das Monitoring, sind:

- die Anlage vereint durch ihren komplexen Aufbau viele unterschiedliche, hochaktuelle Komponenten
- die lokale Installation hält aktuelle und direkt nachprüfbare Messdaten bereit
- die Anlage wird durch Experten aktiv betreut und überwacht
- die Anlage wird produktiv eingesetzt und bedarf aufgrund der Wichtigkeit (Heiz- und Kühlversorgung einen Gebäudes) einer Ausfallsicherheit
- sie ist bereits seit 2007 in Betrieb und verfügt daher über detaillierte Kenntnisse von Störfällen, die sich mit den aufgezeichneten Messdaten nachweisen lassen

Nachfolgend wird kurz beschrieben, aus welchen Komponenten sich die Referenzanlage zusammensetzt und zu welchem Zweck sie dienen.

3.1 Aufbau der Referenzanlage[1]

Das ZAE Bayern betreibt die Pilotinstallation SolarHeatCool+PCM am Standort in Garching (Abbildung 2). Die mit der Firma SK Sonnenklima entwickelte Absorptionskälteanlage mit 10 kW Nennkälteleistung wurde 2007 in das Gebäudesystem integriert. Zur Bereitstellung der erforderlichen Antriebswärme von 14 kW dient eine 57 m² große Flachkollektoranlage in Südausrichtung auf dem Dach des Institutsgebäu-

[1] Auszug aus dem Projektbericht [MHe11]

des, die über den Hochtemperaturverteiler und zwei in Serie verschaltet Pufferspeicher, mit jeweils 1 m³ Inhalt, Warmwasser in die Absorptionskälteanlage einspeist. Das erzeugte Kaltwasser von 15 °C kühlt über den Niedertemperaturverteiler und großflächige Deckensegel ca. 600 m² Büro- und Laborfläche. Die entstehende Abwärme der Absorptionskälteanlage von 24 kW wird über einen Hauptverteilkreis an den Rückkühlkreis übergeben. Der Latentwärmespeicher unterstützt das trockene Rückkühlwerk in Zeiten mit hohen Außenlufttemperaturen (> 26,3 °C), da dann nur ein Teil der Abwärme der Kältemaschine direkt über den trockenen Rückkühler an die Umgebung abgeführt werden kann.

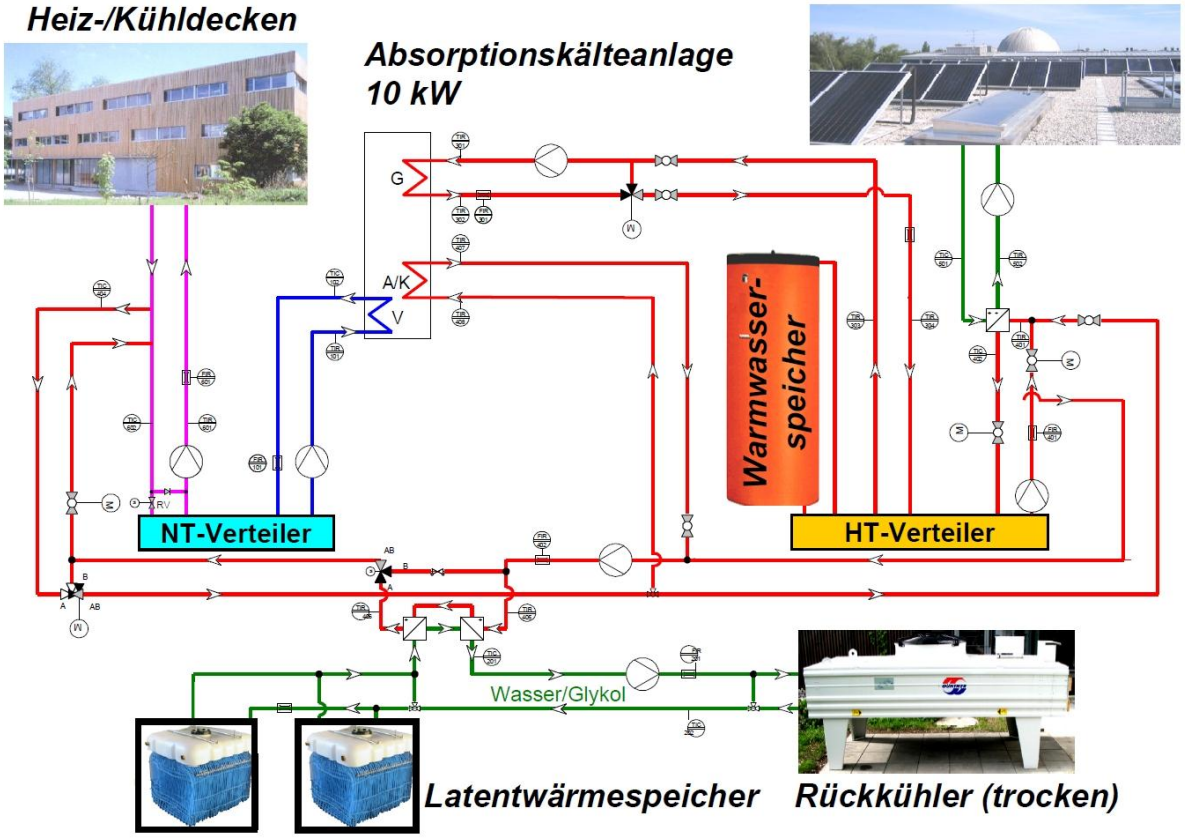

Abbildung 2: Hydraulische Verschaltung der Hauptkomponenten der SolarHeatCool+PCM Anlage

Im Winterbetrieb sind HT- und NT-Verteiler direkt miteinander verbunden. Zudem besteht die Möglichkeit über den Hauptverteilkreis Niedertemperaturwärme von der Solaranlage über eine Rücklaufanhebung direkt in das Heizungssystem einzuspeisen bzw. Überschüsse in den Latentwärmespeichern zu speichern.

3.2 Perspektiven für die Anlagenüberwachung

Die Visualisierung und Regelung der Anlage wurde mit National Instruments LabVIEW [lab] umgesetzt. Die Benutzeroberfläche dient dem Experten zur schnellen In Situ Kontrolle der Pilotanlage und gibt ihm die Möglichkeit bei Bedarf regelnd in den Anlagenprozess eingreifen zu können. Zudem werden die Rohdaten aller Messpunkte mit dem am ZAE Bayern entwickelten Messdatenverarbeitungsprogramm Medview2 in einer PostgreSQL-Datenbank [pos] gespeichert, die dann über den gesamten Zeitraum der Anlagenlaufzeit vorliegen.
Die Fachanwender führen ein Logbuch, in dem verzeichnet wird, ob Umbaumaßnahmen oder manuelle Eingriffe in die Regelung erforderlich waren. Außerdem werden Fehlfunktionen der Anlage, sowie Ausfälle an der Messtechnik mitprotokolliert. Mit Hilfe des Zeitstempels und der Messpunktbezeichnung können diese Vorfälle anhand der aufgezeichneten Messdaten nachvollzogen werden. Aufgrund der Komplexität der Anlage durch das Zusammenspiel der vielen unterschiedlichen Komponenten und der verschiedenen Betriebsmodi (im Sommer kühlen, im Winter heizen), ergeben sich natürlicher maßen Fehlerquellen, die es zu erkennen gilt. Da die Anlage seit geraumer Zeit in Betrieb ist, kann mit Hilfe des Logbuches und den zugehörigen Messdaten das Monitoring durch den Experten zurückverfolgt werden, welches dann als Vorlage für das maschinelle Monitoring per Software dienen soll.

4 Fehlerklassen

Fehlerquellen in komplexen Messanlagen gibt es viele. Nicht immer ist sofort ersichtlich woher der Fehler eigentlich rührt. Angelehnt an die Veröffentlichung „Sensor faults: Detection methods and prevalence in real-world datasets“ [Abh10], wird in diesem Abschnitt versucht, unterschiedliche Fehlertypen und Fehlerquellen zu beleuchten, um einen Überblick zu erhalten, welche Arten von Fehlern überhaupt in Messdaten vorkommen können.

4.1 Sensorfehler

Sensorfehler treten immer dann auf, wenn ein Messfühler inklusive Zuleitungen, ein Messgerät oder auch ein Messrechner einen Defekt aufweist. Dies kann unterschiedliche Auswirkungen haben.

Erleidet ein einzelner Sensor einen Totalausfall, so findet man in den Messdaten häufig einen Anstieg oder einen Abfall auf einen konstanten Wert, Zum Beispiel durch den Overflow Messwert des Messgeräts, der über die gesamte Zeit gehalten wird. Meist liegt dieser Wert weit außerhalb des üblichen Messbereichs des Fühlers. PT-100-Temperaturfühler messen beispielsweise den Widerstand des Fühlers über die Spannung bei eingespeistem, bekanntem Strom. Der Widerstand verändert sich mit der Temperaturänderung. Fällt ein Sensor aus, zum Beispiel weil das Fühlerkabel durchtrennt wurde, wird der maximale Widerstand gemessen, welcher außerhalb des gültigen Messbereichs liegt.

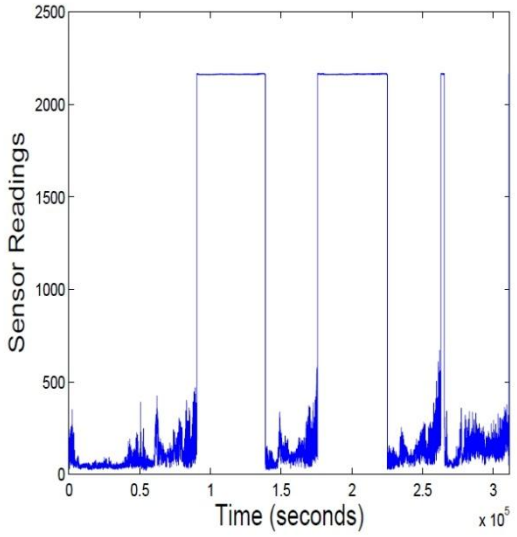

Abbildung 3: temporärer, konstanter Messfehler [Abh10]

Kabelbrüche, kalte Lötstellen oder Wackelkontakte haben in ihrer Ausprägung in den Messdaten einen ähnlichen Charakter wie Totalausfälle. Diese temporären Defekte können auch auf ein konstantes Werteniveau fallen oder steigen, gelangen aber nach einiger Zeit zurück auf ihr normales Niveau. Ist die Zeit während des Defekts sehr gering, spricht man von einem Ausreißer (Peak), der nicht auf einem konstanten Werteniveau bleibt, sondert sofort in seinen Ausgangszustand zurück springt.

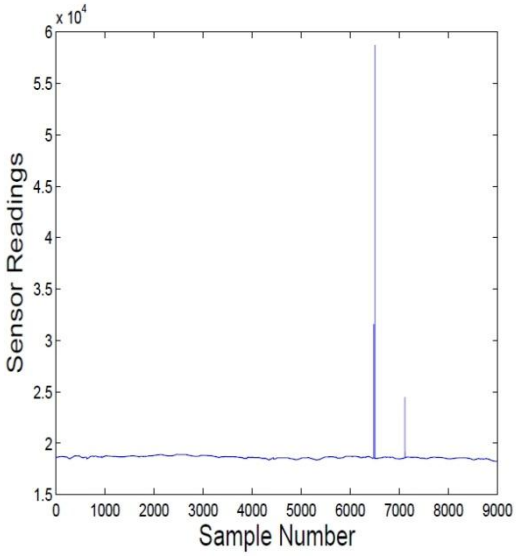

Abbildung 4: Messfehler-Peak [Abh10]

Schwieriger zu erkennen sind Messewerte, die aufgrund eines schleichenden Defekts von den üblichen Messergebnissen langsam wegdriften. Hier gibt es keine signifikanten Anzeichen für ein Fehlverhalten des Fühlers. Es zeigt sich kein sprunghaftes Verhalten und auch die Messwerte liegen weiterhin in einem gültigen Messbereich. Geht der Drift schnell von statten, wird erkennbar, dass die Messwerte zwar gültig sind, aber in einen unüblichen Bereich abdriften. Für langsame Abweichungen bedarf es zusätzlicher Informationen (zum Beispiel ein Referenzfühler, der die gleiche Messkurve aufzeichnet), die man gegeneinander in Beziehung setzen kann.

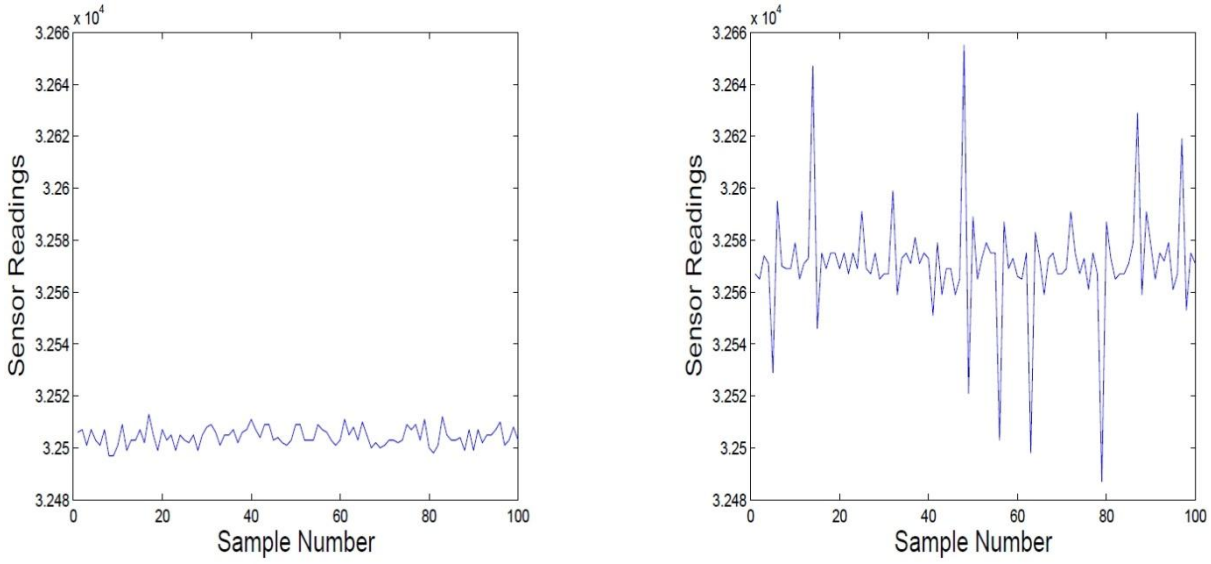

Abbildung 5: Unterschiedliche Ausprägungen von Rauschen [Abh10]

Ein weiterer, häufig auftretender Sensorfehler ist die Messwertüberlagerung mit einem Rauschen. Über einen längeren, oft anhaltenden Zeitraum bewegen sich die Messwerte zappelnd um den tatsächlich gemessenen Wert herum. Dabei kann es zu unterschiedlich starken Ausprägungen des Rauschens kommen. Sehr schwach auftretendes Rauschen, das auch als normal auftretende Messwertschwankungen gewertet werden kann, ist schwieriger zu erkennen als starkes Rauschen, das sich stark von der eigentlichen Messkurve abhebt.

4.2 Steuer- bzw. Regelfehler

Energietechnische Referenzanlagen bestehen aus mehreren Komponenten, die miteinander verschaltet sind und die je nach Betriebsmodus entsprechend gesteuert werden müssen. Dabei kann es, sei es durch Ausfall einer Regelkomponente oder Fehler in der Regelungsprogrammierung zu Störungen kommen, die sich in den Messdaten wiederspiegeln. Wenn beispielsweise im Sommer die Solaranlage mehr Heißwasser als benötigt bereitstellt, sollte die Temperatur im Pufferspeicher ansteigen. Misst man am Ausgang des Solarkreises eine hohe und ansteigende Temperatur, aber am Pufferspeichereingang bleibt die Temperatur konstant niedrig, ist davon auszugehen, dass hier ein Steuerungsfehler vorliegt und die Wärme beispielsweise direkt in den eigentlich abgeschalteten Heizkreis geleitet wird, weil zum Beispiel das entsprechende Ventil nicht geschaltet wurde. Das Detektieren solcher Regelfehler ist mit dem entsprechenden Fachwissen, sprich der genauen Abläufe der Steuerungsmechanismen realisierbar.

Dennoch können auch Fehlertypen, wie sie in den Sensorfehlern beschrieben sind innerhalb der Regelung auftreten. Ein falsch eingestelltes Mischer-Ventil beispielsweise, wird eine unübliche Abweichung der Messwerte hervorrufen. Ein falsch angesteuertes Druckventil kann durch schnelles Öffnen und Schließen einen Peak in den Messwerten hinterlassen. Eine genaue Abgrenzung zur Fehlerherkunft kann von vornherein nicht gewährleistet werden und Bedarf im Zweifel die Prüfung durch einen Mitarbeiter.

4.3 Komponentenfehler

Die einzelnen Komponenten eines Energiesystems sind teilweise selbst hoch komplex und können ebenso Fehler hervorrufen, die durch Ausfall oder Fehlfunktion der Komponente zu bergründen sind.
Da meist mehrere Messpunkte an einer Anlagenkomponente angebracht sind, ist es ratsam alle Messwerte der Komponente mit in die Fehlererkennung einzubeziehen. Fällt eine Komponente aus, werden alle Sensoren mehr oder weniger gleichzeitig abnormale Werte anzeigen. Funktioniert nur ein Teilbereich der Komponente nicht richtig, kann man anhand der Messdaten von örtlich beieinander liegenden Messpunkten die Fehlerquelle lokalisieren.
Aber auch hier können bekannte Fehlertypen wie Drifts auftreten, die dann nicht zweifelsfrei einer Fehlerquelle zugeordnet werden können. Zum Beispiel altert die Absorberschicht einer Solaranlage über die Jahre hinweg und ihr Absorptionsgrad verschlechtert sich. Dadurch wird weniger Solarenergie aufgenommen, was sich in abdriftenden Temperatur-Messwerten niederschlägt.

5 Monitoring Methoden

Um die vorgestellten Fehlerklassen zu detektieren, können unterschiedliche Monitoring Methoden angewendet werden. Je nach Art des Fehlertyps eignet sich die eine oder andere Methode besser und die Methoden können zum Teil auch miteinander kombiniert werden.

5.1 Regelbasiert

Beim regelbasierten Monitoring werden Regeln der Form WENN *Bedingung(en) erfüllt* DANN *Fehler erkannt* ausgewertet. Es gibt rückwärts- und vorwärtsverkettende Regelsysteme. Rückwärtsverkettende Systeme gehen von einer Hypothese aus, zum Beispiel *Solaranlage defekt* und versuchen dann, durch Erfüllung der Bedingungen diese Hypothese zu bestätigen. Bei vielen Regeln werden dementsprechend viele Hypothesen angenommen, die das System versucht zu lösen. Vorwärtsverkettende Systeme arbeiten in die andere Richtung. Sie gehen nicht davon aus, dass eine Hypothese bereits eingetreten ist, sondern überprüfen die Bedingungen der Regel. Sobald sich eine Bedingung als falsch ausgewertet wird, kann die Hypothese nicht bestätigt werden und die übrigen Bedingungen der Regel werden gar nicht erst

überprüft. Die meisten Expertensysteme arbeiten mit Vorwärtsverkettung, da dieses Vorgehen effizienter auf die Realität abbildet. Man geht davon aus, dass eine Messanlage fehlerfrei läuft und überwacht die Messdaten auf mögliche Fehler. Würde man rückwärtsverkettend agieren, geht das System davon aus, dass ständig Fehler vorliegen und ist die ganze Zeit damit beschäftigt, alle Fehlerhypothesen zu überprüfen.
Regelbasiertes Monitoring bietet sich immer dann an, wenn man sehr genau weiß, welche Messdaten zu erwarten sind oder man zusätzliche Informationen hat, die einen Messverlauf wahrscheinlich machen (siehe 5.2). Ein Temperaturfühler beispielsweise, der die Umgebungstemperatur an einem Standort in Deutschland misst, bewegt sich in der Regel in einem Temperaturbereich von -20°C und 35°C oder maximal von -40°C und +50°C (siehe [tem]). Die Regel dazu könnte lauten: WENN *T<-40* ODER *T>50* DANN *Fühler defekt*.

5.2 Simulation und Schätzung

Viele Referenzanlagen werden, bevor sie umgesetzt werden, simuliert. Dies ist nötig um vorab Aussagen treffen zu können, ob die Anlage ihren angestrebten Zweck erfüllen wird. Wurde die Anlage gebaut, kann man nicht nur die Messdaten nutzen, die Simulation zu kalibrieren, sondern auch die kalibrierte Simulation dazu nutzen, die Messdaten zu verifizieren. Für das Monitoring können die Simulationsdaten als Wissensquelle dienen, die einen Messverlauf als fehlerhaft erkennen kann. Dabei können die Simulationsergebnisse als Faktenwissen für regelbasiertes Monitoring genutzt werden, oder als Trainingswissen für neuronale Netze herangezogen werden.
Kann man den Verlauf einer Messkurve aufgrund vorangegangener Messungen verlässlich vorausschauend abschätzen, so kann auch diese Information genutzt werden, den zukünftigen Messverlauf zu überwachen. Auch hier hat man wieder die Möglichkeit das Wissen regelbasiert zu verarbeiten, oder einem neuronalen Netz zu Trainingszwecken zur Verfügung zu stellen.
Sind die Simulations- bzw. Schätzergebnisse belastbar, stellen sie eine gute Monitoring Methode dar. Gerade wenn Messungen immer den gleichen Verlauf nehmen, sind zum Beispiel Driftbewegungen detektierbar, die ohne dieses Wissen nur sehr schwer zu erkennen sind.

5.3 Neuronale Netze

Neuronale Netze oder genauer künstliche neuronale Netze verfolgen in der Informatik den Ansatz, das biologische Vorbild der Nervenzellen zu adaptieren und für die Wissensverarbeitung nutzbar zu machen. Besondere Eigenschaft eines neuronalen Netzes ist die Fähigkeit des maschinellen Lernens. Man kann ein neuronales Netz beispielsweise mit Simulationsdaten trainieren. Wird dieses Netz auf Messdaten angewendet, die einen ähnlich zu erwartenden Verlauf haben, erkennt das neuronale Netz diese Messdaten als Lösung an. Andernfalls würde eine signifikante Abweichung erkannt und die Messdaten als nicht plausible Lösung detektiert. Je besser das neuronale Netz trainiert wird, sprich je größer sein Erfahrungsschatz ist, desto robuster ist es gegenüber unwesentlichen Abweichungen der Messdaten.
Nachteil eines neuronalen Netzes ist, dass man nicht weiß, was es weiß. Man kann von außen betrachtet nicht vorhersagen, wie ein neuronales Netz sich entscheiden wird. Deshalb eignet es sich auch nicht für alle Arten von Fehlerklassen und umso wichtiger ist es, dass man beim Monitoring mit neuronalen Netzen gutes Trainingswissen als Voraussetzung hat.
In der Praxis kann das Fehlen von Trainingswissen zu einem Problem werden, da beispielsweise bei einer Klassifizierung fehlerfreie und fehlerhafte Messdaten als Trainingsdaten benötigt werden.

6 Softwareanalyse für das regelbasierte Monitoring

Für das Monitoring der Referenzanlage wird ein zunächst regelbasiertes System [reg] entwickelt. Die drei Hauptbestandteile dieses Systems sind die Faktenbasis, die Regelbasis und die Inferenzkomponente. Die Faktenbasis enthält die aufgezeichneten Messwerte, sowie beispielsweise analytisch ermittelte Kennzahlen, die die Referenzanlage spezifisch charakterisieren. Die Regelbasis umfasst die anzuwendenden Regeln, welche von der Inferenzkomponente passend ausgewählt und ausgewertet werden.

6.1 Vorabüberlegungen zur Umsetzung

Bei der Umsetzung des regelbasierten Monitorings soll ein Business-Rule-Management-System [ges] (kurz BRMS) zum Einsatz kommen. Dies hat mehrere Vorteile. Die Regelbasis wird separat vom übrigen Quellcode der Messdatenerfas-

sung in einem Business-Rule-Repository [bus1] bereitgestellt und per Schnittstelle eingebunden. Wird die Messdatenerfassung aktualisiert oder verändert, bleiben die Regeln davon unberührt. Die Regeln werden je nach BRMS in einer eigenen, vorwärtsverkettenden Regelsprache in der Form „WENN *Prämisse* DANN *Konklusion*“ verfasst. Damit erhält man einen guten und schnellen Überblick, worauf die Regeln abzielen. Zudem entwickelt man die einzelnen Regeln in einem abgeschlossenen System, sodass die entstandene Regelbasis auch auf eine andere Referenzanlage übertragen werden kann. Stellt das BRMS außerdem eine Verwaltungsoberfläche bereit, kann der Endanwender später selbst Regeln definieren oder anpassen. Die Inferenzkomponente eines BRMS nennt sich Business-Rule-Engine [bus] (kurz BRE) und hat die Aufgabe mittels geeignetem Algorithmus die Regeln effizient zu verarbeiten und auszuwerten.

Nachteile eines BRMS sind die Einarbeitung in die zum Teil sehr speziellen Regelsprachen, welche die Lesbarkeit fördern, aber die gewohnte Programmierung erschweren. Dies kann man bei manchen BRMS umgehen, da sie die Unterstützung für selbst definierte oder gängige Programmiersprachen mitbringen. Zudem steigt die Komplexität des gesamten Softwaresystems mit einer zusätzlichen Komponente und man muss sich um die Wartung und Pflege der BRMS-Komponente kümmern.

Im Allgemeinen sollte man sich gut überlegen, wann ein BRMS Sinn macht. Handelt es sich um eine Individuallösung oder ein kleines, überschaubares Softwareprojekt mit wenigen Regeln, fährt man ohne BRMS vermutlich besser. Der Aufwand das BRMS einzubinden und die Regeln zu definieren, ohne von den Vorteilen des BRMS später zu profitieren, rechtfertigt sich hier nicht.

In dem hier diskutierten Fall macht ein BRMS durchaus Sinn. Die Messdatenerfassung und Visualisierung, welche am ZAE Bayern entwickelt wurde und eingesetzt wird, ist in vielen Pilotanlagen in Benutzung und wird auch weiterhin vorangetrieben und verwendet. Ein zum Monitoring verwendetes BRMS kann so für eine breite Basis von Messanlagen zur Geltung kommen und das Regelwissen wächst mit den Einsatzgebieten. Da die Weiterentwicklung der Messdatenerfassung und der vermehrte Einsatz gewährleistet sind, lohnt es sich, das regelbasierte System auf eine solide Softwarebasis in Form eines BRMS zu setzen. Die Möglichkeit, dass die Anwender bzw. Experten ihrer Messanlage zukünftig auch eigenverantwortlich Regeln zur Überwachung beibringen können und sich der personelle Zeitaufwand der Sys-

temadministratoren dadurch dezimiert, macht den Einsatz eines BRMS nur noch attraktiver.

6.2 Sondierung von Business-Rule-Management-Systemen

In diesem Abschnitt wird ein Überblick über Business-Rule-Management-Systeme gegeben, welche für das regelbasierte Monitoring eingesetzt werden können. Dabei liegt der Fokus auf der Integrierbarkeit in das bestehende Messdatenerfassungssystem, sowie der komfortablen Verwaltung der Regelparadigmen und der effizienten Auswertung gegenüber der Faktenbasis. Die Kandidaten wurden durch eine Internet-Recherche ermittelt, wonach sie möglichst auch in Veröffentlichungen explizit erwähnt worden sind. Hauptquelle war die Übersichtsseite von Open Source Rule Engines für Java [ope]. Das am geeignetsten beurteilte BRMS wurde anschließend als Grundlage für die Implementierung verwendet. Die Auswahl der Systeme erhebt keinesfalls Anspruch auf Vollständigkeit, vielmehr wurde Wert darauf gelegt, möglichst verschiedene Arten von Wissensverarbeitungsystemen zu betrachten, um das Thema von verschiedenen Blickwinkeln zu beleuchten und die Variationsmöglichkeiten darzustellen.

6.2.1 d3web

d3web [d3w] ist weniger ein Business-Rule-Management-System als ein in Java geschriebenes Tool, welches die Generierung von Wissensbasen ermöglicht. Hierfür gibt es die grafische Entwicklungsumgebung, mit der sich die Regeln und Fakten erstellen und verwalten lassen. Hat man eine Wissensbasis erzeugt, kann diese per Weboberfläche den Benutzern zugänglich gemacht werden und über einen Frage-Antwort-Dialog ausgewertet werden. Anwendung findet d3web hauptsächlich in Fällen, bei denen es um beratende Problemlösungsstrategien geht, wie zum Beispiel im medizinischen Sektor zur Vorabdiagnose. d3web wurde am Lehrstuhl für Informatik IV „Künstliche Intelligenz und Angewandte Informatik“ der Julius-Maximilians-Universität Würzburg entwickelt und ist seit 2007 ein OpenSource-Projekt [d3w1]. Eine Übersicht mit bereits realisierten Anwendungen findet man auf der Projektseite [dem] des Lehrstuhls. Die nicht allzu gut strukturierte und überschaubare Dokumentation für Entwickler, die die erzeugte Wissensbasis nicht über den Webdialog benutzen wollen, zeigt wie man d3web in ein bestehendes Java-Projekt einbinden und weiterverarbeiten kann. Die Community um d3web kann man als nicht existent be-

zeichnen. Es gibt nahezu keine Diskussionsbeiträge oder Aktualisierungen des Quellcodes seitens der Community, lediglich eine kleine interne Entwicklergruppe des Projekts veröffentlich von Zeit zu Zeit Updates. So verwundert es auch nicht, dass es außer Studienarbeiten kaum produktiv umgesetzte Expertensysteme auf Grundlage von d3web gibt.

6.2.2 SPIRIT

Ein Vertreter von Expertensystemen, die auch mit unsicherem Wissen umgehen können ist SPIRIT (**S**ymmetrical **P**robabilistic **I**ntensional **R**easoning in **I**nference Networks in **T**ransition). Die Kurzbeschreibung auf der Homepage von SPIRIT [spi] stellt das Projekt vor: „Häufigkeitsverteilungen, die auf statistischen Erhebungen beruhen, können gleichermaßen berücksichtigt werden wie heuristische Regeln; Wissen des Experten wird ebenso integriert, wie subjektive Einschätzungen seitens des Benutzers einfließen. SPIRIT ist in der Lage, selbständig aus eingegebenen Daten und Beispielen zu lernen und gehört damit zu den aktiv lernenden Systemen. Durch einen induktiven Lernprozess und aufgrund von Experteninformationen werden auch indirekte (d.h. nicht direkt beobachtete) Zusammenhänge durch transitives Schließen erfasst. Hervorzuheben ist weiterhin die Fähigkeit des Systems, logische Schlussfolgerungen (im Sinne der erweiterten Prädikatenlogik) aus der gemeinsamen Wahrscheinlichkeitsverteilung abzuleiten.“

SPIRIT wurde am Forschungsbereich „Operations Research“ der Fakultät für Wirtschaftswissenschaft der FernUniversität Hagen entwickelt und steht dort zum Download unter der GNU Public License zum Download bereit. Für das in Java geschriebene Expertensystem gibt es eine Dokumentation für Entwickler, die die Bibliothek in ihr eigenes Projekt integrieren wollen. Jedoch umreißt diese nur die grundlegendsten Funktionalitäten und geht nicht weiter ins Detail. Für die integrierte Entwicklungsumgebung, mit der sich probabilistische Wissensbasen erzeugen lassen, fehlt leider jegliche Dokumentation. Hier muss man sich mit Beispielwissensbasen selbst ein Bild machen. Es existiert ein Entwicklernetzwerk, welches man bei konkreten Anwender- oder Programmierfragen ansprechen kann - eine gewachsene Community, die sich mit dem Projekt intensiv auseinandersetzt, gibt es aber nicht. In der Praxis wurde SPIRIT bislang nicht sehr häufig eingesetzt. Als Beispiele werden Anwendungen im Finanz- und Medizinsektor angeführt. Workshops und Publikationen zu dem

Thema sind zuletzt 2010 zu finden. Seitdem findet man keine aktuellen Veröffentlichungen, die auf eine aktive Weiterentwicklung des Projekts schließen lassen.

6.2.3 OpenRules

OpenRules [ope1] beschreibt sich selbst als Business Decision Management System (kurz BDMS). 2003 noch geplant als klassisches BRMS, legte man zukünftig mehr Wert auf die Involvierung der Fachexperten, die selbst, ohne die Vorarbeit und Unterstützung eines Programmierers, Entscheidungstabellen anlegen und auswerten sollen. Die Idee dahinter ist, dass letztendlich diejenigen, die mit den Prozessen am meisten vertraut sind, selbst die Handhabe über den Entscheidungsprozess erhalten. Somit dient ein gängiges Tabellenkalkulationsprogramm wie Microsoft Excel oder OpenOffice Calc als Regeleditor, mit denen die Anwender mit einer intuitiv verständlichen Syntax Entscheidungstabellen schreiben können. Die IT-Abteilung eines Unternehmens muss sich nur um die Integration des in Java geschriebenen BDM Systems in die bestehende Software-Infrastruktur kümmern. Hierfür gibt es ein Eclipse Plugin, so dass Eclipse als Entwicklungsumgebung direkt unterstützt wird.
OpenRules fährt eine zweistufige Lizensierungspolitik. Zum einen frei erhältlich mit einer GNU General Public License für reine Open Source Projekte, zum anderen gibt es non-GPL Lizenzen für kommerzielle Anwendungen und solche deren Quellcode nicht veröffentlicht werden kann. Eine Einzelplatzlizenz fängt bei 3200$ an. Die Dokumentation ist sehr umfangreich und detailliert. Zudem gibt es eine gute Schnellstart-Anleitung, die den Einstieg in OpenRules in anschaulicher Weise erleichtert. Neben kommerziellen Support- und Trainingsangeboten gibt es eine Google Diskussionsgruppe in der Fragen gestellt und beantwortet werden. Die Einträge halten sich jedoch in Grenzen, so dass man nicht von einer sehr ausgeprägten und aktiven Community sprechen kann. Dennoch wird das Produkt von Herstellerseite stetig weiterentwickelt und ca. vierteljährlich mit Updates versorgt, so dass man davon ausgehen kann, langfristig mit OpenRules arbeiten zu können.

6.2.4 IBM WebSphere Operational Decision Management

Früher und noch immer weitläufig bekannt als IBM WebSphere ILOG JRules stellt IBM WebSphere Operational Decision Management (kurz WODM) [ibm1] einen der bekanntesten und verbreitetsten Vertreter der rein kommerziellen B2B-BRM Systeme dar. Neben FICO Blaze Advisor und JBoss Enterprise BRMS (kommerzieller Ableger

von JBoss Drools) ist WODM einer der Marktführer im professionellen Geschäftsumfeld, wenn es um den Einsatz eines BRMS geht. Mit Lizenzkosten von 12000$ pro Einzelplatzlizenz (typische Anwendung um die 50000$) ist das Produkt für Unternehmen geeignet, welche aufgrund ihrer Größe mit enorm vielen Mechanismen im Verwaltungs- oder Produktionsbereich umgehen müssen. WODM kann Herstellungs- und Geschäftsabläufe optimieren, sowie Kunden- oder Finanzdaten auswerten. Die Anwendungsbereiche sind vielfältig, da das BRMS auf jedes Teilgebiet individuell abgestimmt werden kann. Auf dem Produktdatenblatt [ibm] wirbt man wie folgt für das Produkt: „Das vollständigste BRMS. JRules BRMS erfüllt die verschiedenen Anforderungen der Geschäftsbereiche, IT-Entwickler und des IT-Betriebes, um einfach und erfolgreich die Entscheidungslogik, die die Unternehmenssysteme steuert, zu verwalten. Es beschleunigt Markteinführungen und senkt die Gesamtbetriebskosten (Total cost of ownership, TCO) durch vier wichtige Funktionsmerkmale:

- Gemeinsame Regelverwaltung für Fachanwender-Teams
- Umfangreiche Anwendungsentwicklung und Implementierung für IT-Teams
- Überlegene Entscheidungssteuerung für Fach- und IT-Anwender
- Leistungsfähige, skalierbare und steuerbare Ausführung von Regeln

Die JRules BRMS Produktreihe bietet Werkzeuge und Umgebungen, die sich an die Arbeitsweise von Unternehmen anpassen und liefert eine umfangreiche Unterstützung für verschiedene Plattformen, die innerhalb der IT Architektur eines Unternehmens eingesetzt werden.“

Neben einer guten Eclipse-Integration, sowie der Möglichkeit Regeln per Plug-In in Microsoft Office zu bearbeiten, gibt es auch eine Weboberfläche zum Definieren und Verwalten von Regeln. So können je nach Fach- oder IT-Wissen Anwender mit den Arbeiten an der Regelbasis betraut werden. Neben dem kommerziellen Herstellersupport gibt es auf den Seiten von IBM diverse Foren, auf denen Entwickler Fragen stellen und Erfahrungen austauschen können.

6.2.5 JBoss Drools

JBoss Drools [dro] ist ein vollwertiges Open Source BRM-System, welches unter einer Apache 2.0 Lizenz veröffentlicht ist. Vom Umfang unterscheidet sich Drools nicht von seinem kommerziellen Pendant JBoss Enterprise BRMS. Lediglich auf den Herstellersupport muss verzichtet werden. Es bietet neben einer sehr guten Eclipse Integration mit eigenem Workbench und Rule-Flow-Editor, ein Webinterface zum

Verwalten der Regeln und kann auch mit Entscheidungstabellen aus Tabellenkalkulationsprogrammen umgehen. Das in Java geschriebene BRMS stellt Schnittstellen zu gängigen Java-Frameworks wie Spring oder JBoss Seam bereit, so dass die Einbindung von Drools in bestehende Softwareprojekte erleichtert wird. Regeln werden in der Drools Rule Language (kurz DRL) verfasst und auf Objekte der Faktenbasis mittels JavaBeans [jav1] zugegriffen. Es besteht weiterhin die Möglichkeit, Regeln in eine beliebige, natürliche Sprache zu übersetzen und diese Definitionen im Domain Specific Language (kurz DSL) Editor zu hinterlegen. Dies ermöglicht es, Fachanwendern leichteren Zugang zum Bearbeiten der Regeln zu geben.
Die Dokumentation ist vorbildlich und umfangreich, der Einstieg wird mit Beispielapplikationen und Tutorials vereinfacht. Drools ist weit verbreitet und es gibt zahlreiche Veröffentlichungen zu umgesetzten Projekten. JBoss stellt selbst eine Community-Plattform mit einem Wiki, User Forum, Mailing Liste, Chat und Blog bereit, die allesamt intensiv genutzt werden. Auch auf Nebenschauplätzen, wie Codebeispielseiten âla stackoverflow.com finden sich zahlreiche Einträge rund um das BRM-System. Updates von finalen Versionen werden kontinuierlich in etwa halbjährlichen Abstand herausgebracht. Währenddessen kann auch auf Beta- und RC-Versionen zurückgegriffen werden. Die letzte stabile Version 5.5.0 wurde am 13. November 2012 veröffentlicht. Diese Tatsache, zusammen mit der sehr aktiven Community, macht das BRMS interessant für produktive, langfristige Einsätze, wenn man nicht auf kommerziellen Support angewiesen ist und die Hilfe der Entwicklergemeinschaft ausreichend ist.

6.3 Theorie: Der Rete-Algorithmus

Bereits 1979 wurde der Rete-Algorithmus [ret] von Charles Forgy im Rahmen seiner Doktorarbeit und der Entwicklung des Expertensystems OPS5 an der Carnegie Mellon University entwickelt. Neben der Veröffentlichung durch die Doktorarbeit wird der Algorithmus erstmals 1974 in seinem Arbeitspapier erwähnt, später, 1982 in einem Artikel des Journals „Artificial Intelligence“ ausgiebig erläutert [For82]. Bis heute stellt Rete (lateinisch für „Netz“) als Pattern Matching Algorithmus die Basis für viele bekannte und verbreitete Business Rule Management Systeme dar (u.a. die vorgestellten BRMS Jboss Drools und WODM). Wurde er auch im Laufe der Zeit weiterentwickelt und verbessert, hat sich am Grundprinzip nichts geändert. Lediglich die Laufzeit wurde optimiert, so dass heutige Implementierungen etwa 500-mal so schnell sind

wie das Original [ret1]. Bis heute gibt es neben dem Rete-Algorithmus keine ernsthaften Alternativen, da sie entweder zu speziell auf ein Problem zugeschnitten sind, oder keine merklichen Verbesserungen gegenüber den weiterentwickelten und ausgereiften Rete-Versionen hervorbringen.

Anlehnend an einem Auszug einer Vorlesung der Wirtschaftsinformatik [ret2] wird die Funktionsweise des Algorithmus erklärt. Der Rete-Algorithmus verfolgt das Ziel, für eine Menge von Mustern, sprich Regelbedingungen, und eine sich ändernde Objektmenge, zum Beispiel Messdaten, auf effiziente Weise die Muster zu bestimmen, die durch die aktuelle Objektmenge erfüllt werden. Der Vorteil des Rete-Algorithmus gegenüber herkömmlichen Pattern Matching Algorithmen ist die Generierung und Aktualisierung einer baumähnlichen Datenstruktur zum Abarbeiten der Regeln. Der Rete-Algorithmus erzeugt aus einer Menge von Regelprämissen ein Entscheidungsnetzwerk in Form eines Datenflussgraphen. Dies ist zwar speicherintensiver als das sequentielle Auswerten beim Pattern Matching, aber auch um ein vielfaches schneller, da der Baum immer im optimalen Abfragezustand ist. Teilweise identisch genutzte Regelprämissen werden nur einmal in der Datenstruktur mitgeführt und müssen so auch nur einmal für alle Vorkommen in der Faktenbasis ausgewertet werden. Ändern sich während der Abarbeitung der Regeln Inhalte der Faktenbasis, so wird der Entscheidungsbaum automatisch dahingehend überprüft, ob diese Änderungen auf andere Regeln Einfluss nehmen. Nichterfüllbare Regeln verschwinden aus der Datenstruktur, mögliche Regeln kommen neu hinzu. Jede Anfrage an die Regelmaschine läuft so auf dem aktuellsten Stand aller Fakten. Nachfolgend (Abbildung 6) ist schematisch dargestellt, wie der Rete-Algorithmus gleiche Anfangstücke der Regelprämissen so miteinander verschmilzt, dass keine Auswertung doppelt erfolgen muss.

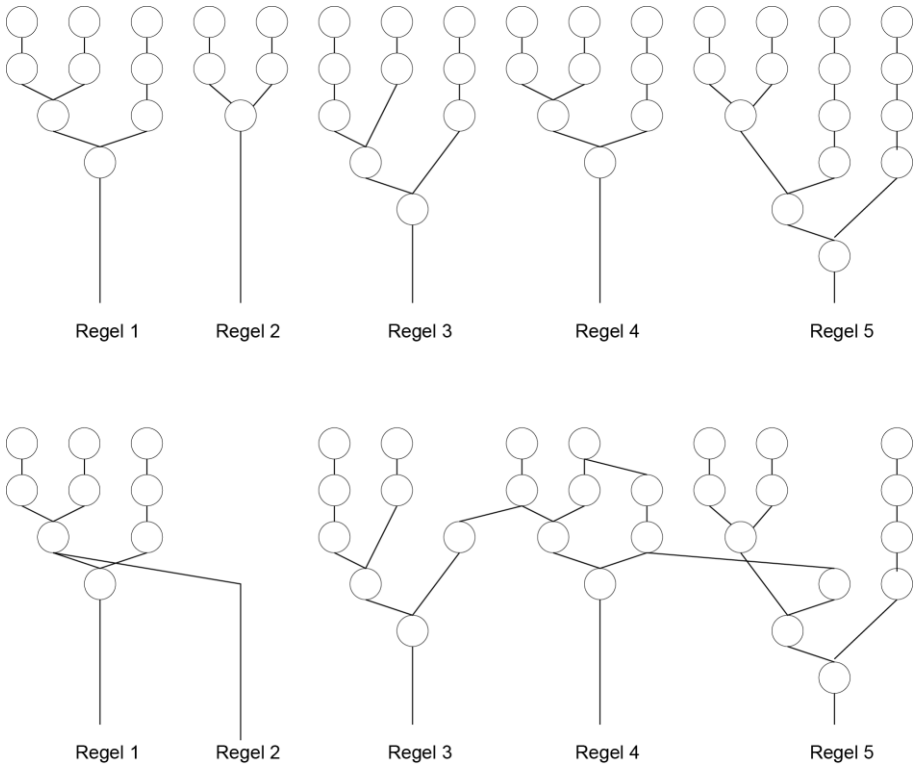

Abbildung 6: Verschmelzung von identischen Regelprämissen in einem Rete-Netzwerk [ret2]

Die Nomenklatur des Rete-Algorithmus bezeichnet die Regelprämissen als „left hand side“ (kurz LHS), die Konklusionen werden „right hand side“ (kurz RHS) genannt. Die Faktenbasis heißt, wie bei vielen BRM-Systemen auch, „working memory“. Das erzeugte Entscheidungsnetzwerk besteht aus zwei Arten von Knoten. Zum einen aus α-Knoten, die Selektionsbedingungen darstellen, welche sich auf Fakten aus dem working memory beziehen, zum anderen aus β-Knoten, die Bedingungen repräsentieren, die über Junktoren und Variablenbindungen verknüpft sind. Im Folgenden wird auf ein fiktives Beispiel eingegangen, das die Arbeitsweise des Rete-Algorithmus verdeutlicht.

Die Regel sei wie folgt in Pseudocode gegeben:

```
WHEN                                    # Praemisse: LHS
    x.Temperatur > 30                   # alpha-Knoten: A(x)
AND                                     # beta-Knoten: A(x) ^ B(x,y)
    x.Temperatur < y.Temperatur         # alpha-Knoten: B(x,y)
AND                                     # beta-Knoten: B(x,y) ^ C(y)
    y.Temperatur > 40                   # alpha-Knoten: C(y)
THEN                                    # Konklusion: RHS
    Klimanlage.Switch = 1
```

Abbildung 7: Fiktive Regel in Pseudocode zur Veranschaulichung der Rete-Knoten

Umgangssprachlich würde man die Regel so ausformulieren: Wenn die Temperatur am Messpunkt x höher als 30 Grad Celsius ist, und die Temperatur am Messpunkt y höher ist als am Messpunkt x, und die Temperatur am Messpunkt y höher als 40 Grad Celsius ist, dann schalte die Klimaanlage ein. Dabei können die Variablen x und y mit beliebigen Messpunkten belegt werden.
Instanziiert man nun zum Beispiel die Messpunkte mit den Bezeichnern x=1 und y=2 (z.B. Temperaturfühler an Messpunkt 1 und Temperaturfühler an Messpunkt 2) ergibt sich folgender Datenflussgraph (Abbildung 8) in einem Rete-Entscheidungsnetzwerk:

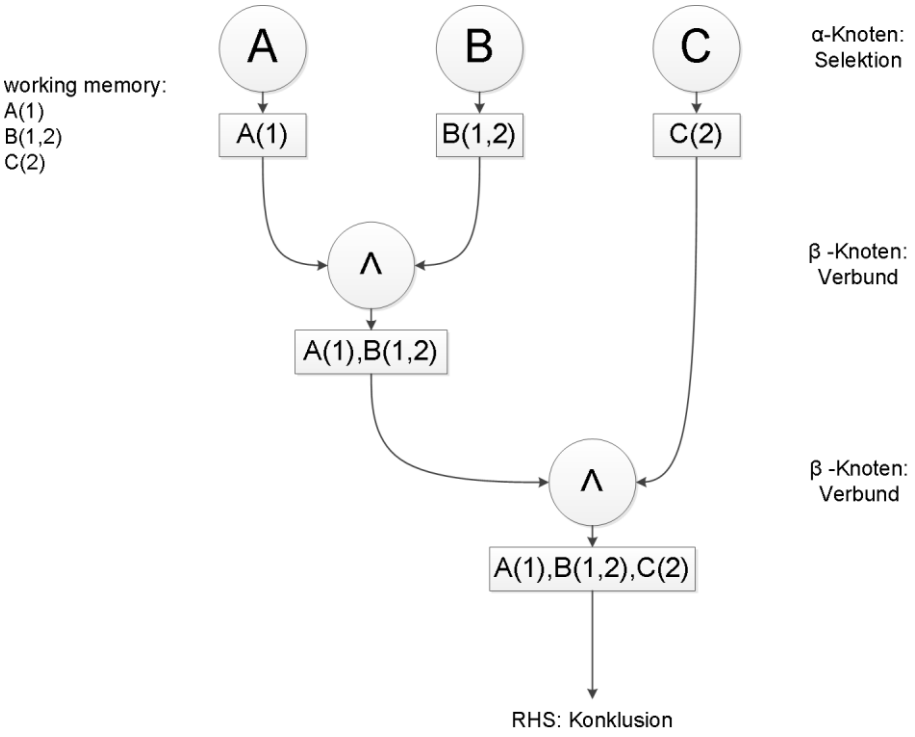

Abbildung 8: Beispiel eines Rete-Datenflussgraphen nach der Instanziierung

Zu den jeweiligen Knoten werden die Ergebnisse gespeichert. Zu einem α-Knoten die Objekte, welche die Selektionsbedingung erfüllen, zu einem β-Knoten die Tupelkombinationen, welche die Verbundbedingung erfüllen. Kommt der Verbund nicht durch eine gemeinsame Variable zustande, so liefert der β-Knoten das Kreuzprodukt aus den Eingangstupelmengen als Ergebnis. Der letzte Knoten einer Regel enthält als Ergebnis die Menge der Regelinstanzen, die diese Regel erfüllen. Die Menge aller letzten Knoten in einem Rete-Netzwerk nennt man Konfliktmenge. Nach dem initialen Durchlauf genügt es nun Überprüfungen allein gegenüber der Konfliktmenge zu anzustellen.
Werden Änderungen in der Faktenbasis vorgenommen, so werden diese durch das Netzwerk des Datenflussgraphen propagiert.

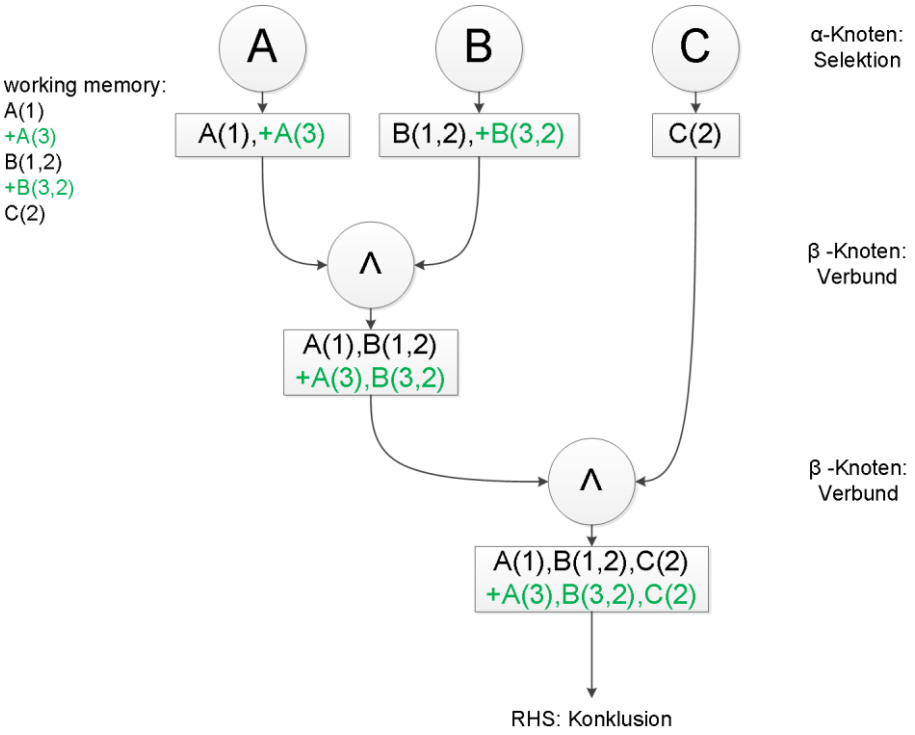

Abbildung 9: Einfügen zwei neuer Tupel in das working memory

Wird in das working memory ein Tupel eingefügt, so wandert dieses als sogenanntes Plus-Tupel (+) durch das Rete-Entscheidungsnetz und die Knotenspeicher werden dementsprechend aktualisiert (Abbildung 9). Analoges gilt für die sogenannten Minus-Tupel (-) (Abbildung 10)

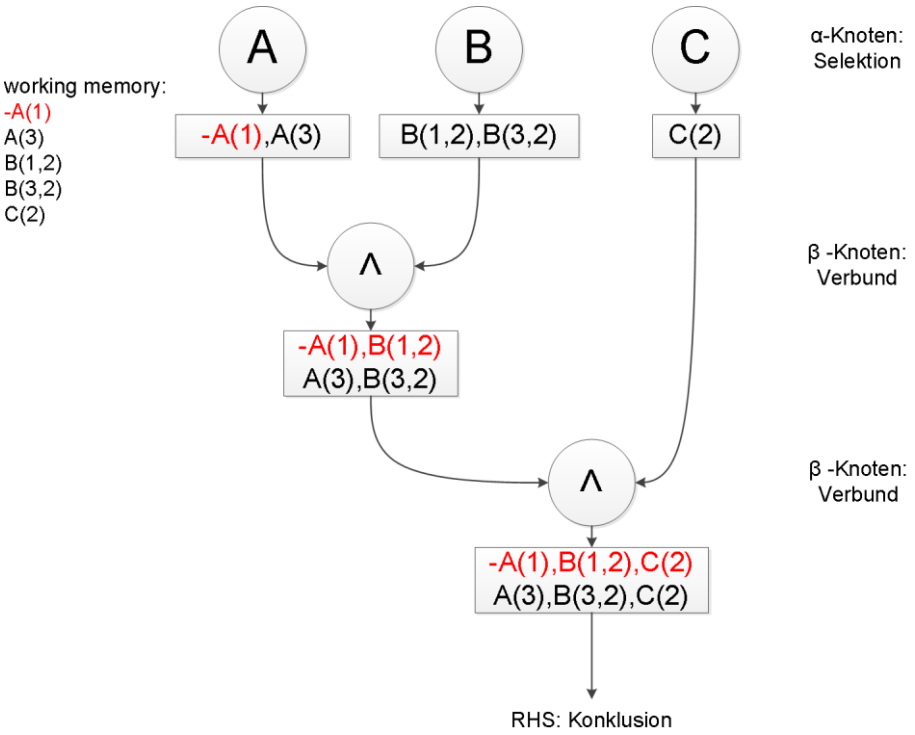

Abbildung 10: Entfernen eines Tupels aus dem working memory

Wird durch die Änderungen auch das Ergebnis des letzten Knotens einer Regelprämisse verändert, so wird auch die Konfliktmenge entsprechend aktualisiert.
Ist die Funktionsweise des Algorithmus klar, kann man diese Erkenntnisse nutzen, um den Auswertungsablauf weiter zu optimieren. Es ist zu empfehlen, die Regeln so spezifisch wie möglich zu verfassen, um die Zwischenergebnisse an den Knoten zu verkleinern. Außerdem sollte man beachten, dass Verbindungen an β-Knoten, welche nicht über eine Variable, sondern über das Kreuzprodukt zustande kommen, die Knotenergebnisse sehr groß werden lassen können. Hierbei kann man versuchen, durch Umstrukturierung der Knoteneingänge, Fakten mit vielen Ergebnistupeln zu trennen.

6.4 Beurteilung und Auswahl des BRMS

Für die Umsetzung dieser Arbeit wurde das BRM-System JBoss Drools ausgewählt. Zusammen mit dem kommerziellen IBM WebSphere Operational Decicion Management System hat es den größten Funktionsumfang im Vergleich zu den übrigen Anwärtern. Da es bei dieser Arbeit auch um die Abschätzung geht, ob ein solches BRMS für das Monitoring von energietechnischen Referenzanlagen geeignet ist, ist der Kostenfaktor einer kommerziellen Software zu berücksichtigen. Davon abgese-

hen, dass für das kommerzielle Produkt erhebliche Kosten aufzuwenden sind, steht JBoss Drools seinem Pendant WODM in Sachen Aktualität, Verbreitungsgrad, Funktionsumfang und Dokumentation in nichts nach. Der gute Eindruck der Open Source Community um JBoss Drools, lässt erwarten, dass man hier auch ohne den kommerziellen Support schnell und kompetent Problemhilfestellungen bekommt. Diese Faktoren heben JBoss Drools somit deutlich von den anderen betrachteten Systemen ab.

Aufgrund sehr begrenzter Einsatzmöglichkeiten, schlechter Dokumentation und wenig umgesetzter Projekte kann d3web für diesen Einsatzzweck nicht überzeugen. Es spielt seine Stärken im Bereich Online-Befragesysteme aus und sollte auch hauptsächlich für diesen Einsatzzweck verwendet werden.

Der probabilistische Ansatz des SPIRIT-Expertensystems bietet sich für Anwendungen an, bei denen man sich nicht genau im Klaren ist, welche Vorgänge sich abspielen und man auf jeden Fall eine wahrscheinliche Abschätzung als Ergebnis erzielen möchte. Für die Arbeit ist der Ansatz ungeeignet, da es sich beim Monitoring energietechnischer Referenzanlagen größtenteils um deterministische Fehlerquellen handelt, welche es zu überwachen gilt und die der Experte kennt und benennen kann. Man könnte die Regeln natürlich mit 100% Wahrscheinlichkeit ansetzen, hätte dann aber den Grundgedanken von SPIRIT verfehlt. Weiterhin lassen die Dokumentation sowie die Beispielanwendungen zu wünschen übrig, so dass sich der Einstieg nicht so leicht bewerkstelligen lässt wie bei JBoss Drools.

OpenRules zeichnet sich durch die sehr gute Integration in Tabellenkalkulationsprogramme aus, was vor allem dann zum Tragen kommt, wenn die Wissensbasis auch in Tabellenform vorliegt und die Anwender stärker in den Entscheidungsprozess eingebunden werden sollen. Die leicht zu erlernende Syntax und die vertraute Entwicklungsumgebung einer Tabellenkalkulation erleichtert es dem Anwender, selbst Entscheidungstabellen zu verwalten. Beim Monitoring energietechnischer Referenzanlagen am ZAE Bayern liegen die Messdaten in einer Datenbank vor, und sollen nach Möglichkeit direkt aus dieser, ohne Import in eine Tabellenkalkulation, verarbeitet werden. Die Einbindung des Experten kann bei JBoss Drools durch die Übersetzung der Regeln im DSL-Editor und die Verwaltung per Webapplikation erfolgen.

Alles in allem überzeugt JBoss Drools mit dem besten Gesamteindruck der hier vorgestellten BRM-Systeme, und wird daher für das angestrebte Projekt herangezogen.

7 Implementierung des regelbasierten Monitoring

Am ZAE Bayern ist die Messdatenerfassungs- und Visualisierungssoftware Medview2 (ZAE Zentralsystem) entwickelt worden, welche es ermöglicht, Messdaten von Projektanlagen zu visualisieren und auszuwerten. Die Messdaten selbst werden in einer PostgreSQL-Datenbank [pos] bereitgehalten.

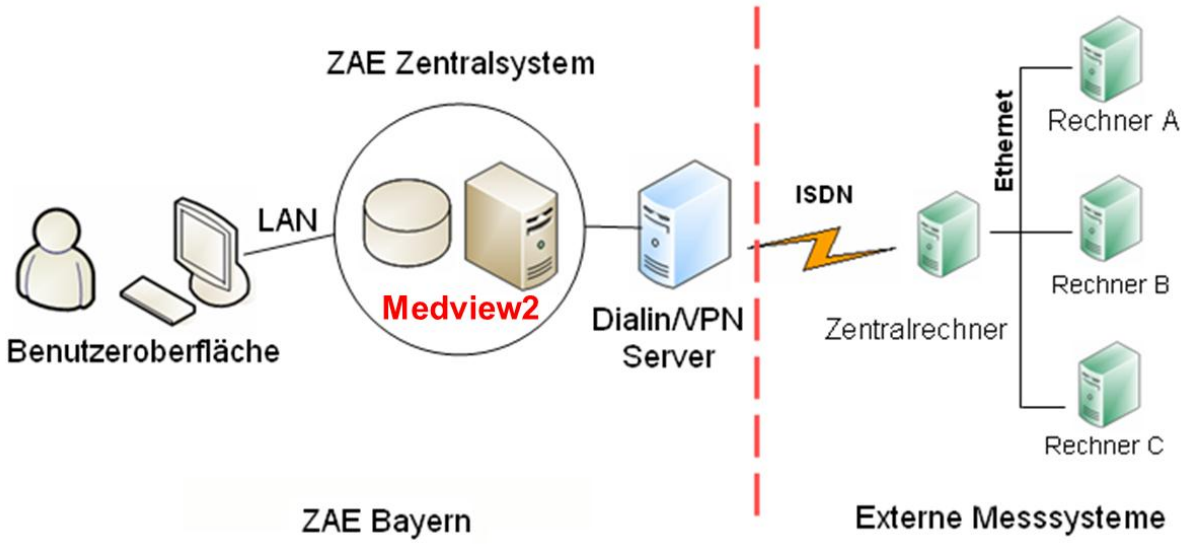

Abbildung 11: Schema des bestehenden Systems

Die erfassten Messdaten werden von zum Teil externen Rechnern an das Zentralsystem (Medview2) am ZAE Bayern per HTTPS Upload in die Datenbank übertragen (Abbildung 11). Das Backend von Medview2 ist technisch als Webservice (Apache Axis2 [apa]) in Java, sowie für die Persistenz mittels einer PostgreSQL-Datenbank realisiert, so dass man leicht mit verschiedenen Programmen, zum Beispiel über die JDBC-Schnittstelle [jav] darauf zugreifen kann. Dadurch sind mit geringem Aufwand Anfragen über eine Java-Applikation möglich, die Messdaten aus der Datenbank liefern. Somit ist die technische Voraussetzung gegeben, Messwerte anzufordern, die dann als Fakten in die Wissensbasis eingepflegt werden können.

7.1 Einbindung der Messdatenbank in die Wissensbasis

Nachdem die quelloffene Java-Entwicklungsumgebung Eclipse [ecl] für die Benutzung mit JBoss Drools vorbereitet wurde, gilt es zunächst die Fakten in Form von Messdaten in die Wissensbasis einzupflegen. Wenn dieser Prozess abgeschlossen

ist, kann die Regelmaschine die erstellten Regeln gegenüber der Faktenbasis verifizieren.

Um SQL-Anfragen aus einer Java-Applikation an die PostgreSQL-Datenbank stellen zu können, muss vorab eine Kommunikationsverbindung zur Datenbank aufgebaut werden. In diesem Fall wird dies über die JDBC-Schnittstelle mit passendem PostgreSQL-Treiber realisiert.

```
DBintoKB(   "foyer.solarheatcool",
            "jdbc:postgresql://192.168.3.10/SolarHeatCoolPCM",
            "***",
            "***" );

static void DBintoKB(   String dbTbl,
                        String dbUrl,
                        String dbUsr,
                        String dbPwd ) {
    Connection cn = null;
    try {
        cn = DriverManager.getConnection(dbUrl,dbUsr,dbPwd);

// hier ist die Verbindung hergestellt,
// und es koennen Anfragen an die Datenbank gestellt werden.

        } catch( Exception ex ) {
            System.out.println( ex );
        } finally {
            try {
                if( cn != null ) cn.close();
            } catch( Exception ex ) {
                /*nothing to do*/
            }
    }
}
```

Abbildung 12: Kommunikationsverbindung per JDBC zur PostgreSQL-Datenbank aufbauen

Die Parameter für den Konstruktor sind die Bezeichnung der Datenbanktabelle, welche die Messwerte enthält, die URL zum Datenbankserver mit vorangestellter JDBC-Treiberspezifikation, sowie Benutzername und Passwort für den Zugriff auf die Datenbank. Für das Monitoring genügt zunächst nur der Lesezugriff auf die Datenbank, da im vor-produktiven Einsatz noch nichts in die Datenbank zurückgeschrieben werden soll.

Ist die Verbindung hergestellt, können Abfragen an die Datenbank gestellt werden. Dies kann über einen implementierten Prozeduraufruf der JDBC-Schnittstelle erfolgen, bei dem ein ResultSet-Objekt [res] zurückgegeben wird, welches das Ergebnis des SQL-Statements beinhaltet. Diese Methode ist nützlich, um schnell einen Überblick zu erhalten, was die Anfrage zurückgeliefert hat. Man durchläuft das ResultSet-Objekt und gibt die Einzelelemente zum Beispiel in der Konsole aus.

Für das Monitoring sollen die Messdaten jedoch in die Wissensbasis als Faktenwissen übertragen werden. Dazu könnte man das ResultSet-Objekt durchlaufen und die einzelnen Elemente zu einem JavaBean-Konstrukt zusammenbauen. Ein eleganterer Ansatz wird durch die Java-Klassenbibliothek DbUtils [dbu] möglich. Sie hält Klassen und Methoden bereit, um automatisch ein JavaBean-Objekt aus einem ResultSet-Objekt zu generieren. DbUtils ist eine freie Klassenbibliothek (steht unter Apache License Version 2.0) des Apache Commons Projekts [apa1] und bietet Erweiterungen und Vereinfachungen für den Zugriff auf Datenbanken mittels JDBC. Um DbUtils nutzen zu können, genügt es die DbUtils Jar-Datei von der Projekthomepage herunterzuladen und in das Eclipse-Projekt einzubinden.

```
import org.apache.commons.dbutils.*;

public static List<Map<String,Object>> query(   Connection cn,
                                                 String query) {
    List<Map<String,Object>> result = null;
        try {
            QueryRunner qrun = new QueryRunner();
            result = (List<Map<String, Object>>) qrun.query(cn,
                                                            query,
                                                            new MapListHandler());
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            DbUtils.closeQuietly(cn);
        }
        return result;
    }

//...

// hier ist die Verbindung hergestellt,
// und es koennen Anfragen an die Datenbank gestellt werden.

List<Map<String,Object>> Anfrage = query(   cn,
                                            "SELECT * FROM foyer.solarheatcool " +
                                            "WHERE \"Time\" >= '2011-08-13 10:00:00' " +
                                            "AND \"Time\" <= '2011-08-13 24:00:00' " +
                                            "ORDER BY \"Time\"");

// ...
```

Abbildung 13: DbUtils ermöglicht das Mapping der SQL-Anfrage auf ein JavaBean-Objekt

Das JavaBean-Objekt enthält jetzt ein Mapping des Ergebnisses der SQL-Anfrage. Dieses kann nun in die Wissensbasis von JBoss Drools eingefügt werden und steht dann den Regeln als Faktenwissen gegenüber. Die Voraussetzungen, damit die Regeln sich gegenüber den Messdaten abgleichen können, sind somit gegeben. Nachfolgend wird die Regelspezifikation in Drools beschrieben und anhand von Beispielregeln für das Monitoring veranschaulicht.

7.2 Regeldefinition

Um einen Überblick über die Regeldefinition zu erhalten, wurde die Dokumentation [dro1] ausgiebig studiert. Die Regeln werden bei Drools in einer oder mehreren separaten Regeldateien hinterlegt. Im Grunde genommen sind das Textdateien, denen eine Dokumentstruktur zugrunde liegt, damit Drools die Informationen verarbeiten kann. Für die Dateiendung hat sich das Kürzel „.drl“ etabliert, welches sich direkt mit der Drools Regelsprache DRL (Drools Rule Language) identifiziert. Legt man die Regeldateien innerhalb von Eclipse mit eingebundener Drools Umgebung an, wird die Dokumentstruktur gleich mitgeliefert und es stehen Syntax-Hervorhebung und Autovervollständigung für die DRL zur Verfügung. In einer Regeldatei können mehrere Regeln, Anfragen und Funktionen hinterlegt sein. Zusätzlich können Quellenangaben in Form von Importdeklarationen, globalen Variablen und Attributen angegeben werden, die dann in den Regeln und Anfragen verwendet werden können. All diese Angaben, wie zum Beispiel auch die Angabe einer Package-Zugehörigkeit, sind optional. Im einfachsten Fall genügt eine einzige Regeldefinition, die zur Auswertung herangezogen werden kann.

7.2.1 Regelsyntax

Regeln in Drools werden immer nach einer bestimmten syntaktischen Vorgabe gemäß der DRL-Spezifikation definiert (Abbildung 14). Eine Regel beginnt mit dem reservierten Schlüsselwort *rule*, gefolgt von einem eindeutigen Bezeichner, der die Regel möglichst genau umschreibt. Danach können optionale Regelattribute angegeben werden, die zum Beispiel die Regelausführungsreihenfolge bestimmen. Es folgt das Schlüsselwort *when*, das die LHS (left hand side), also die Prämisse der Regel einleitet. Im LHS-Teil werden ein oder mehrere Prüfbedingungen deklariert, das Schlüsselwort *then* initiiert die RHS (right hand side), also die Konklusion der Regel. Ein *end* schließt die Regeldefinition ab.

```
rule "<name>"
    <attribute>*
when
    <conditional element>*
then
    <action>*
end
```

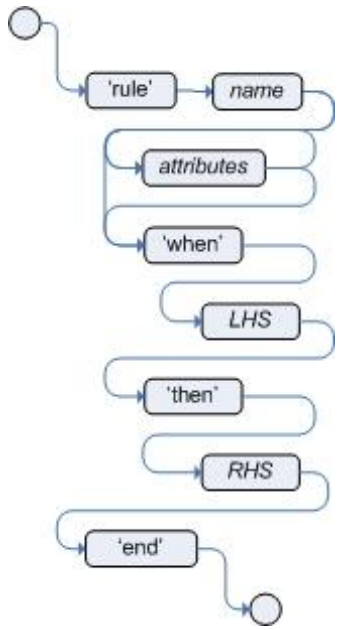

Abbildung 14: syntaktischer Aufbau einer Regel in DRL-Regelsprache [dro1]

Die Prämisse (LHS) kann keine oder beliebig viele Prüfbedingungen enthalten. Ist die Prämisse leer, wird sie automatisch als *true* ausgewertet und die Konklusion wird einmalig ausgeführt. Will man in Drools etwas überprüfen, geschieht das in Form von Muster-Vergleichen (pattern matching). Muster können, je nach Verallgemeinerung, auf alle Fakten der Wissensbasis (working memory) zutreffen. Definiert man beispielsweise das Muster *Object()*, so würde der Vergleich für jedes Element im working memory als *true* ausgewertet.

Abbildung 15: Syntaxdiagramm für eine Musterdefinition [dro1]

Die einfachste Form einer Musterdefinition ist somit eine Typüberprüfung ohne Einschränkungen, zum Beispiel *Klasse()*. Alle Objekte des Typs *Klasse()* werden damit erfasst. Innerhalb der Klammen können keine oder beliebig viele Zwangsbedingungen angegeben werden, um das Muster näher zu spezifizieren. Diese Elemente sind Java-Ausdrücke, die mittels booleschen Operatoren zu *true* oder *false* ausgewertet werden können. Mehrere Zwangsbedingungen werden mit logischen Junktoren (Konjunktion *&&*, Disjunktion || und schwache Konjunktion ,) miteinander verknüpft. Es

kann direkt auf JavaBean-Eigenschaften zugegriffen werden, so wird beispielsweise *Sensor(temp==30)* ausgewertet wie *Sensor(getTemp()==30)*.

```
$sensor: Sensor( ( active == true && temp > 30 ) || volt != 0 )
```

Abbildung 16: fiktive Musterdefinition zur Veranschaulichung der Syntaxmöglichkeiten

Muster (sowie Eigenschaften) können an eine Variable gebunden werden, so dass man sowohl tiefer in der Prämisse, als auch später in der Konklusion darauf referenzieren kann. Dies geschieht durch einen Variablenbezeichner gefolgt von einem Doppelpunkt. Der Konvention zufolge stellt man der Variablenbezeichnung ein Dollar-Zeichen *$* voran, so dass leichter ersichtlich ist, ob es sich beim Bezug darauf um eine Variablenbindung oder eine Zwangsbedingung handelt. Mehrere Muster werden miteinander logisch verknüpft, entweder implizit konjugiert, indem man sie einfach untereinander schreibt, oder aber explizit durch Schlüsselwörter wie *and*, *or*, *not*. In der Dokumentation [dro1] finden sich noch viele weitere Syntaxmöglichkeiten um Muster und Regelprämissen näher zu spezifizieren. An dieser Stelle wird aber nicht weiter darauf eingegangen, sondern bei Verwendung der Syntax in den Beispielregeln diese kurz erläutert.
In der Konklusion (RHS) kann beliebiger Java-Code stehen, der nach Erfüllung der Prämisse ausgeführt werden soll. Dabei kann man sich zum Beispiel auf gebundene Muster aus der Prämisse beziehen oder, falls nötig, neue Fakten zum working memory hinzufügen beziehungsweise bestehende Fakten aktualisieren.

7.2.2 Beispielregeln

In diesem Abschnitt werden Regeln vorgestellt, die exemplarisch für das Monitoring der SolarHeatCool+PCM Anlage gelten sollen. Die Funktionsweise der Beispielregeln ist auf die anderen implementierten Regeln übertragbar und sie unterscheiden sich beispielsweise nur in der Art des Sensortyps und der zugehörigen Grenzwerte. Die absoluten Grenzwerte sind durch physikalische Gesetzmäßigkeiten gegeben. Ein Temperaturfühler kann keine Werte unter -273,15 °C annehmen, während ein Absolut-Drucksensor keine Werte kleiner 0 haben kann. Das Regelprinzip ist für beide Prüfbedingungen jedoch gleich:

```
rule "Drucksensor Grenzwertueberschreitung"
        when
                // Variablenbindung der gesamten Messdaten
                $list : List()
                // Iteration und Pruefung des Grenzwerts
                $map : Map( $p_sensor : this["P_Umgebungsdruck"] < 0 ) from $list
        then
                // Ausgabe einer Meldung auf der Konsole
                System.out.println(    "Regel P_Umgebungsdruck < 0 - "
                                       + $map.get("Time") + ": " + $p_sensor);
end
```

Abbildung 17: Beispielregel Grenzwertüberschreitung

Damit über alle Messwerte des Zeitraums der SQL-Anfrage iteriert werden kann, kommt die *from*-Klausel der DRL Spezifikation zum Tragen. Zunächst wird das Muster List() an die Variable *$list* gebunden. Hier befindet sich nun der gesamte Messdatensatz in Form von Bezeichner-Werte-Paaren des Typs *Map()*. Danach wird über die Elemente von *$list* mittels der *from*-Klausel iteriert. Dabei werden Muster des Typs *Map(…)* gesucht und an die Variable *$map* gebunden. Das Schlüsselwort *this* referenziert auf das *Map()*-Objekt und findet durch die Angabe des Attributs *["P_Umgebungsdruck"]* alle Bezeichner-Werte-Paare des Umgebungsdrucksensors. Die Zwangsbedingung *this["P_Umgebungsdruck"] < 0* reduziert die Treffermenge auf die Paare, bei denen der Grenzwert des Sensors unterschritten wurde. Durch die Variablenbindung *$p_sensor* sind diese Elemente in Konklusion direkt ansprechbar. In der Konklusion informiert eine Konsolenausgabe darüber, von welcher Regel die Meldung stammt, und durch Angabe des Zeitstempels *($map.get("Time")*) wird der Fehler zeitlich exakt bestimmt.

Sind mehrere Messpunkte des gleichen Sensortyps in der Messanlage verbaut, bietet es sich an die gleiche Regel so zu modifizieren, dass sie für alle gleichartigen Messpunkte gelten kann. So muss nicht für jeden einzelnen Bezeichner eine sonst identische Regel angelegt werden und die Gefahr eines Tippfehlers oder des Vergessens eines Messpunktes wird vermindert. Jedoch beruht diese Lösung auf der Konvention, dass gleiche Sensortypen ein Ähnlichkeitsmerkmal in ihrem Bezeichner mitführen. Im folgenden Beispiel würden alle Drucksensoren überprüft, deren Bezeichner mit „P_“ beginnen (analoges Beispiel: „T_“ für Temperatursensoren).

```
// …

// Iteration und Pruefung des Grenzwerts
$map: Map(this[$bezeichner] < 0, $bezeichner: String(this matches "P_.*" )) from $list

// …
```

Abbildung 18: Beispielregel Modifikation durch Reguläre Ausdrücke

Hierbei wird die *matches*-Klausel der DRL-Spezifikation verwendet, die einen regulären Ausdruck benutzt, um alle Bezeichner zu finden, die mit „P_“ beginnen. Sind weitere Ähnlichkeitsmerkmale im Bezeichner vorhanden, so können darüber, beispielsweise durch die *not matches*-Klausel, bestimmte Drucksensoren bei der Regel unberücksichtigt bleiben.

Durch Gespräche mit den Experten der Anlage und Sichtung des Fehlerlogbuchs konnten noch weitere Fehlertypen in den aufgezeichneten Messdaten gefunden werden.

Durch einen Unfall, bei dem ein Lampengehäuse auf die Messtechnik gefallen ist, wurden zwei Temperatursensoren beschädigt (Abbildung 19). Dieser Fehler ist auch durch die Regel der Grenzwertüberschreitung detektierbar, aber die Tatsache, dass beide Temperatursensoren am gleichen Messpunkt messen, bietet eine weitere Möglichkeit anders gelagerte Fehler zu erkennen.

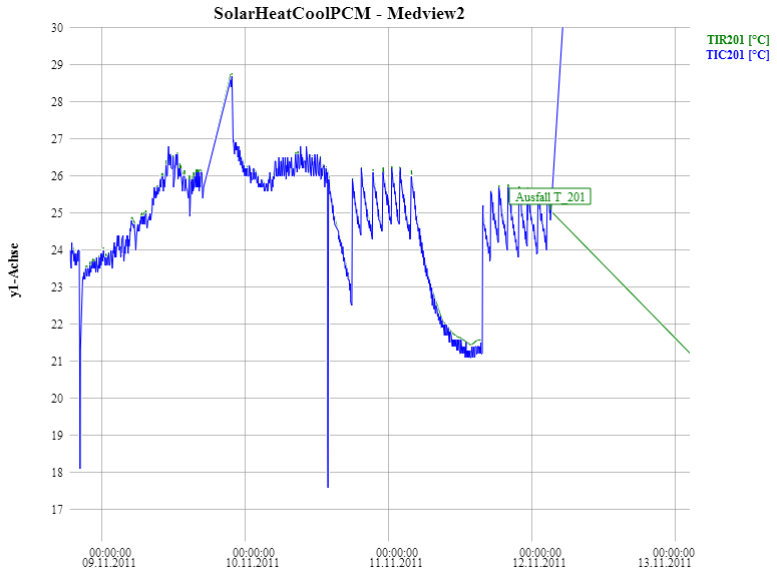

Abbildung 19: Ausfall von zwei Temperaturfühlern am gleichen Messpunkt

Es ist so möglich die beiden Sensoren in Beziehung zueinander zu setzen und eine Regel zu definieren, die Abweichungen voneinander erkennt. Dabei werden alle Fehlerklassen erkannt, die jeweils nur einen der beiden Messfühler betreffen.

```
import java.math.*;

rule "Temperaturfuehler Abweichungen"
        when
                // Variablenbindung der gesamten Messdaten
                $list : List()
                // Iteration und Pruefung auf Abweichungen voneinander
                $map : Map(Math.abs(this["TIR201"] - this["TIC201"]) > 1) from $list
        then
                // Ausgabe einer Meldung auf der Konsole
                System.out.println(    "Regel TIR201 != TIC201 : dT>1 - " +
                                       $map.get("Time") + ": " + $map.get("TIR201") +
                                       " != " + $map.get("TIC201"));
end
```

Abbildung 20: Beispielregel Abweichung der Temperatur am gleichen Messpunkt

In Drools ist es auch möglich Java Methoden zu benutzen. In der Beispielregel wird das *delta_T* der beiden Temperatursensoren bestimmt, indem mittels *Math.abs()* der absolute Betrag aus der Differenz der beiden Temperaturwerte gebildet wird. Wenn das *delta_T* zwischen den beiden Sensoren größer als 1 °C wird, wird die Prämisse wahr und die Konklusion wird ausgeführt. Im vorliegenden Real-Fall driften die beiden Temperaturfühler aufgrund ihrer unterschiedlichen Bauart nach dem Defekt voneinander weg und die Regel greift. Würden sie beide das gleiche Abdriftverhalten aufweisen, greift diese Regel nicht, jedoch würde die Grenzwertüberschreitung den Fehler erkennen. Viel interessanter ist diese Regel für Fehlerklassen, die ohne redundante Sensorausführung nur schwer zu erkennen sind. Driftet einer der beiden Fühler aufgrund eines Defekts nur minimal von den realen Werten ab, so würde der Vergleich mit dem anderen Referenzfühler den Drift erkennbar machen. Der Fühler einzeln betrachtet, würde in seinen Messwerten keine signifikanten Merkmale hinterlassen, die auf einen Drift hindeuten. Ebenso ein Wackelkontakt, ausgeprägt durch einen Peak, oder eine Überlagerung durch Rauschen sind beispielsweise durch die Abweichungsregel schnell und sicher detektierbar. Immer vorausgesetzt, das der Fehler nicht bei beiden Sensoren gleichzeitig auftritt. Generell ist zu empfehlen, systemkritische Fühler nach Möglichkeit redundant auszuführen. So ist gewährleistet, dass man bei Ausfall eines Sensors immer noch auf die Werte des Anderen zurückgreifen kann und durch die begünstigte Fehlererkennung, frühzeitig für den Ersatz

des defekten Fühlers sorgen kann In der Praxis jedoch, wird darauf aus Kostengründen oft verzichtet.

Ein weiteres Beispiel für eine zweifelsfreie Fehlererkennung sind Messungen am Betriebszustand der Anlage die physikalisch nicht möglich sind. Man kann nicht mehr Energie aus einem System gewinnen, als in das System hineingesteckt wurde. Der Energieerhaltungssatz [ene] besagt, speziell in Bezug auf den Energieaustausch offener Systeme: „Die Energie, die in ein System hineinfließt, minus der Energie, die es verlässt, ist die Änderung der Energie des Systems. Durch Betrachtung der Energieströme des Systems kann man auf Abläufe innerhalb des Systems schließen, auch wenn sie selbst nicht beobachtet werden können.“ Das und andere physikalische Gesetzmäßigkeiten kann man sich bei der Definition von Regeln zunutze machen.

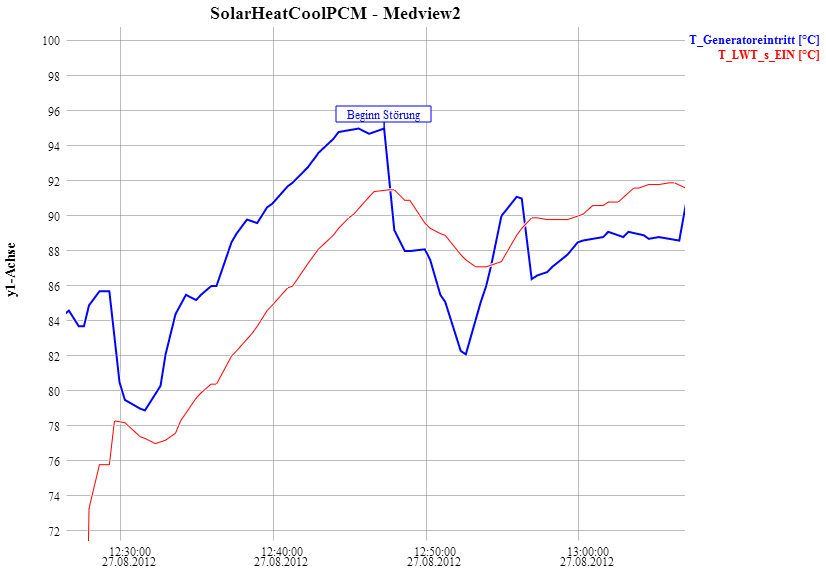

Abbildung 21: Störungserkennung durch Verletzung einer physikalischen Gesetzmäßigkeit

Der zugrundeliegende Real-Fall aus dem Logbuch beschreibt, dass in der Absorptionskälteanlage ein Temperatursensor am Wärmetauscheraustritt sekundär (*T_LWT_s_EIN*), höhere Werte als die Generatoreintrittstemperatur primär (*T_Generatoreintritt*) aufgezeichnet hat, was wiederum physikalisch nicht möglich ist. Die Ursache dabei war, dass einer der beiden Sensoren durch Defekt falsche Werte gemessen hat. Die folgende Beispielregel erkennt aufgrund einer solchen Gesetzmäßigkeit einen Betriebszustand der Anlage, der physikalisch nicht auftreten kann.

```
rule "T_LWT_s_EIN > T_Generatoreintritt"
	when
		// Variablenbindung der gesamten Messdaten
		$list : List()
		// Iteration und Pruefung auf physikalische Unstimmigkeit
		$map : Map(this["T_LWT_s_EIN"] > this["T_Generatoreintritt"]) from $list
	then
		// Ausgabe einer Meldung auf der Konsole
		System.out.println( "Regel T_LWT_s_EIN > T_Generatoreintritt - " +
				$map.get("Time") + ": " + $map.get("T_LWT_s_EIN") +
				" > " + $map.get("T_Generatoreintritt"));
end
```

Abbildung 22: Beispielregel Prüfung auf Widerspruch einer physikalischen Gesetzmäßigkeit

Die Regel überprüft, ob der Widerspruch, Temperatur am Messpunkt *T_LWT_s_EIN* ist größer als am Messpunkt *T_Generatoreintritt*, als wahr ausgewertet werden kann. Die Beispielregeln zeigen die grundsätzliche Vorgehensweise zum Definieren von Regeln in Drools und stehen exemplarisch für andere Regeln, die nach dem gleichen Schema aufgebaut sind. Hat man die Regeln in der DRL-Datei abgespeichert, so muss dieses Regelwissen noch der Wissensbasis zugeführt werden, damit Drools die Fakten gegenüber den Regeln auswerten kann.

7.2.3 Übergabe der Regeln in die Wissensbasis

Das Einpflegen der Regeln und Initialisieren der Wissensbasis übernimmt in Drools der Aufruf der Methode *readKnowledgeBase()*, die eine Wissensbasis zusammen mit den integrierten Regeln (hinterlegt in der Datei „rules.drl“) als Regelwissen zurückgibt.

```
private static KnowledgeBase readKnowledgeBase() throws Exception {
    KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
    kbuilder.add(ResourceFactory.newClassPathResource("rules.drl"), ResourceType.DRL);
    KnowledgeBuilderErrors errors = kbuilder.getErrors();
    if (errors.size() > 0) {
        for (KnowledgeBuilderError error: errors) {
            System.err.println(error);
        }
        throw new IllegalArgumentException("Could not parse knowledge.");
    }
    KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
    kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
    return kbase;
}
```

Abbildung 23: Wissensbasis mit integrierter Regelbasis generieren

Dabei werden auch die Regeln syntaktisch überprüft und gegebenenfalls Fehlermeldungen ausgegeben. Abschließend muss der Aufruf der Methode im Hauptpro-

gramm erfolgen, nachdem die Messdaten aus der Datenbank für die Wissensbasis vorbereitet wurden (Kapitel 7.1).

```
import org.drools.*;

//... siehe Kapitel 7.1

try {
    // Wissensbasis initialisieren
    KnowledgeBase kbase = readKnowledgeBase();
    StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();

    // JavaBean-Objekt der Wissensbasis uebergeben
    ksession.insert(Anfrage);

    // Regeln auswerten
    ksession.fireAllRules();

} catch (Throwable t) {
    t.printStackTrace();
}
```

Abbildung 24: Wissensbasis mit Regelbasis initialisieren und Messdaten übergeben

Durch *ksession.insert(Anfrage)* werden die Messdaten in die Wissensbasis eingepflegt. Danach werden per *ksession.fireAllRules()* alle eingelesenen Regeln gegenüber den Messdaten verifiziert.

7.3 Allgemeine Detektion von Sensorfehlern

Sollen Fehlerklassen wie Peaks oder Rauschen in Messdaten erkannt werden, bei denen man sich beispielsweise nicht auf redundante Referenzfühler beziehen kann, müssen spezifische Informationen über den Messkurvenverlauf gesammelt werden. Aus diesen Informationen kann man Schlüsse über die zukünftig zu erwartenden Messdaten ziehen. Peaks wie in Abbildung 4 können durch Ermittlung der Steigung [ste] zwischen zwei aufeinanderfolgenden Messzeitpunkten erkannt werden.

$$|m| = \left| \frac{Messwert_{t+1} - Messwert_t}{Zeitstempel_{t+1} - Zeitstempel_t} \right|$$

Um eine Aussage treffen zu können, ob die ermittelte Steigung einen Peak darstellt, muss zuvor ein allgemein gültiger Steigungsgrenzwert für die Messkurve ermittelt werden. Alle Steigungswerte die größer als das gesetzte Limit sind, werden als Ausreißer, beispielsweise aufgrund eines Wackelkontakts erkannt.

Eine Messdatenüberlagerung durch Rauschen wie in Abbildung 5 kann mithilfe der Standardabweichung [sta] über eine bestimmte Anzahl von Messwerten detektiert werden.

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Messwert_i - \bar{M})^2}$$

$S: Schätzfunktion\ der\ Standardabweichung$

$n: der\ Strichprobenumfang\ (Anzahl\ der\ Messwerte)$

$\bar{M}: arithmetisches\ Mittel\ aller\ Messwerte\ der\ Stichprobe$

Der errechnete Wert muss auch hier mit einem zuvor ermittelten Standardabweichungsgrenzwert für die Messkurve verglichen werden. Diesen erhält man, indem man die Standardabweichung eines fehlerfreien Messverlaufs ermittelt. Wird der Grenzwert während der Überprüfung der Messdaten überschritten, ist das ein Indiz für die Messdatenüberlagerung durch Rauschen.

Auch temporäre, konstante Messfehler wie in Abbildung 3 kann man mittels der Standardabweichung detektieren. Hierbei genügt es zu überprüfen, ob die errechnete Standardabweichung über die Messwertstichprobe den Wert 0 annimmt. Das bedeutet, dass sich während des untersuchten Zeitraums die Messwerte nicht verändert haben, also immer der gleiche Wert gemessen wurde. In der Messkurve ist das durch eine konstant horizontale Linie zu beobachten. Aber auch hier muss von Sensor zu Sensor unterschieden werden. Es gibt Messkurven, bei denen es durchaus vorkommt, dass sich zu bestimmten Zeiten horizontale Linien ausbilden. Mit genauer Kenntnis über den Messverlauf, kann man versuchen die Regel weiter zu spezifizieren, indem man sie nur für einen Zeit- oder Wertebereich geltend macht.

7.3.1 Parameterbestimmung

Die wenigsten Messkurven haben einen sehr geradlinigen und ruhigen Verlauf, der es erleichtern würde Fehler zu detektieren bzw. Grenzwerte zu finden, die über den gesamten Messbereich gültig sind. Um diesem Umstand entgegen zu wirken, wird eine Parameterbestimmung aus fehlerlosen Messdaten gewonnen, die dann zur Regelauswertung herangezogen werden kann. Am Beispiel eines Real-Falles aus dem Logbuch wird die Vorgehensweise veranschaulicht.

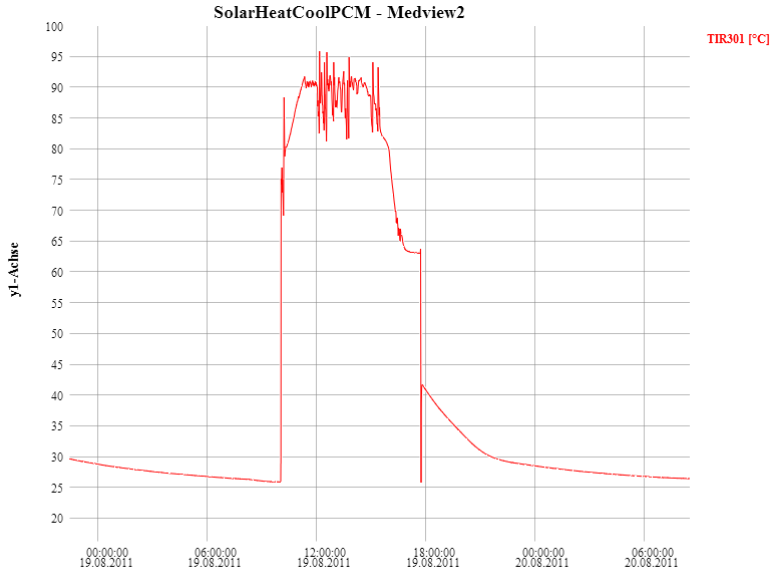

Abbildung 25: fehlerfreier, typischer Temperaturverlauf eines Sensors zur Parameterbestimmung

Abbildung 25 zeigt einen fehlerfreien und typischen Tagestemperaturverlauf des Temperaturfühlers *TIR301*. Anders als in Kapitel 7.2.2 (Abbildung 20) steht hier kein zweiter, redundanter Temperatursensor am gleichen Messpunkt zur Verfügung. Somit kann man nicht gegenüber den Messwerten eines anderen Referenzfühlers Vergleiche anstellen, die zur Detektion von Fehlern beitragen würden.

Weiterhin kann man beispielsweise keinen allgemeingültigen Grenzwert für die Standardabweichung über den gesamten Tagesmessdatensatz angeben. Bei hohen Temperaturen bildet sich eine fluktuative Messkurve ab, die quasi nicht von herkömmlichem Rauschen zu unterscheiden wäre. Auch ein Peak müsste sich in der Steigung deutlich von den natürlichen Temperaturanstiegen unterscheiden, um als solches erkannt zu werden. Lediglich die Ausprägung einer horizontalen Linie ist aus dem Referenzgraphen nicht zu erwarten, so dass die Nullregel der Standardabweichung hier bedenkenlos angewendet werden kann.

Im Logbuch der Referenzanlage konnte eine Störung gefunden werden, die mit Parameterbestimmung erkannt werden kann.

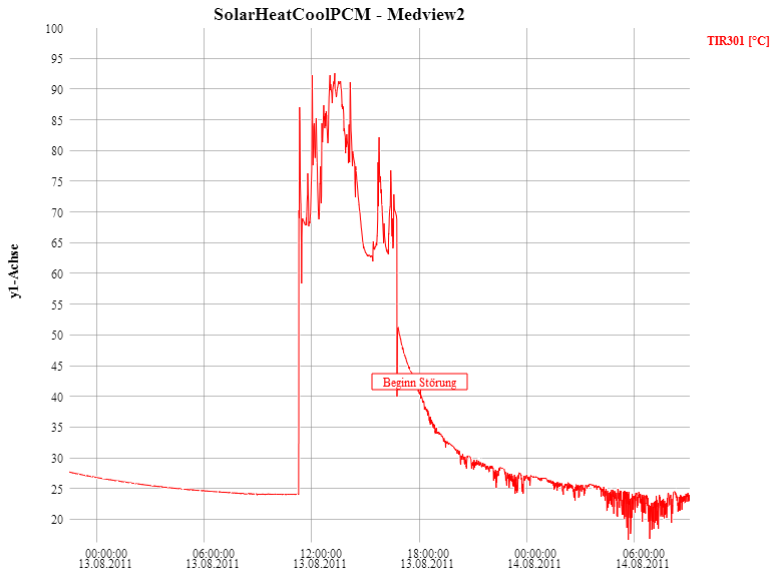

Abbildung 26: Sensorfehler Messdatenüberlagerung durch Rauschen

Abbildung 26 zeigt den Sensorverlauf des Temperaturfühlers *TIR301* und den Beginn der Störung. Im Vergleich zur fehlerfreien Messkurve aus Abbildung 25 zeigt sich ab Einsetzen des Fehlers deutlich, wie die Messdaten durch ein Rauschen überlagert sind.

In diesem Fall bietet es sich an, die Parametrisierung der Messkurve anhand von Temperaturniveaus zu partitionieren. Die Messwerte werden in Bereichen im Abstand von 10°C separat analysiert. So ergeben sich unterschiedliche Standardabweichungen für die verschiedenen Temperaturbereiche. Während der fehlerfreie Temperaturfühler für hohe Temperaturen eine große Standardabweichung aufweist, ist sie für niedrige Temperaturen sehr gering. Dieses Verhalten kann man sich zunutze machen, um das Rauschen im niedrigen Temperaturbereich zu erkennen, da mit dem Rauschen die Standardabweichung über die üblichen Werte ansteigt.

Zur Bestimmung der Standardabweichung wird die Berechnung für auflaufende Messwerte [sta1] herangezogen:

$$S = \sqrt{\frac{1}{n-1}\left[\left(\sum_{i=1}^{n} Messwert_i^2\right) - \frac{1}{n}\left(\sum_{i=1}^{n} Messwert_i\right)^2\right]}$$

Diese Berechnung lässt sich für jeden eintreffenden Messwert sofort aktualisieren, wenn man die Summe der Messwerte, sowie die Summe ihrer Quadrate mitführt und fortlaufend aktualisiert.
Umgesetzt in Java Code liest sich die Berechnung wie folgt:

```
// ...

Double StdAbw, xi_mw, xi_sum, xi_sum_2, xi;
int AnfrageSize, n;
Map<Double,Double> StdAbw_map = new TreeMap<Double,Double>();

AnfrageSize = test.size();
xi_sum = 0.0;
xi_sum_2 = 0.0;
n = 0;

for (int i = 0; i < AnfrageSize; i++) {
    n = n + 1;
    // Messwert des Temperaturfuehlers TIR301
    xi = (Double)Anfrage.get(i).get("TIR301");
    xi_sum = xi_sum + xi;
    xi_sum_2 = xi_sum_2 + xi * xi;
    // Berechnung der Standardabweichung
    StdAbw = Math.sqrt(((n * xi_sum_2) - (xi_sum * xi_sum)) / (n * (n - 1)));
    // Stichprobengroesse 100 Messwerte
    if (((i+1) % 100) == 0) {
        // Mittelwert der Messwerte
        xi_mw = xi_sum / 100;
        xi_sum = 0.0;
        xi_sum_2 = 0.0;
        n = 0;
        // Speichern des Paares (Mittelwert,Standardabweichung)
        StdAbw_map.put(xi_mw, StdAbw);
    }
}

// ...
```

Abbildung 27: Java Code zur Bestimmung der Standardabweichung

Die TreeMap *StdAbw_map* enthält nach Abarbeitung der SQL-Anfrage Wertepaare, bestehend aus dem Mittelwert der Temperaturen aus 100 Messungen, sowie die entsprechend berechnete Standardabweichung. Nun werden die Temperaturmittelwerte im Abstand von 10°C ihrem Temperaturbereich zugeordnet und ein gültiger Grenzwert für die Standardabweichung in diesem Bereich ermittelt.

```
// ...

Map<Integer,Double> StdAbw_part_map = new TreeMap<Integer,Double>();

for (int i = 0; i < 100; i += 10){
    Double StdAbw_mw = 0.0;
    Double StdAbw_sum = 0.0;
    int n = 0;
    for (Map.Entry<Double,Double> entry : StdAbw_map.entrySet()) {
        // key ist die Mittelwerttemperatur T
        Double key = entry.getKey();
        // value ist die Standardabweichung
        Double value = entry.getValue();
        // falls T im Bereich
        if (key >= i && key < i+10){
           n++;
           StdAbw_sum = StdAbw_sum + value;
        }
    }
    if (n != 0) {
        StdAbw_mw = StdAbw_sum / n;
    } else {
        StdAbw_mw = 0.0;
    }
    // Speichern des Paares (Bereichskennzahl,Standardabweichung)
    StdAbw_part_map.put(i, StdAbw_mw);
}

StdAbw_part_map.values().removeAll(Collections.singleton(0.0));

for (Map.Entry<Integer,Double> entry : StdAbw_part_map.entrySet()) {
    Integer key = entry.getKey();
    Double value = entry.getValue();
    System.out.println(key + "-" + " => " + value);
}

// ...
```

Abbildung 28: Java Code zur Partitionierung der Standardabweichung auf Temperaturbereiche

Die TreeMap *StdAbw_part_map* enthält nach Abarbeitung aller Temperaturbereiche eine Partitionierung der Standardabweichungen in Bezug auf ein Temperaturniveau. Die Ausgabe auf der Konsole gibt den Inhalt der Treemap wieder, hier als Beispiel die Werte der fehlerfreien Messkurve aus Abbildung 25:

```
20 => 0.10053069610710734
30 => 0.4669774394588071
40 => 2.158602418864549
60 => 2.722953515519635
70 => 11.468366069644356
80 => 2.3438188467047114
90 => 0.7712586490227332
```

Das bedeutet: Temperaturen größer/gleich 20°C und kleiner als 30°C haben in diesem Beispiel eine mittlere Standardabweichung von 0,10. Werte größer/gleich 30°C und kleiner als 40°C eine mittlere Standardabweichung von 0,47, usw. Für Messwerte unter 20°C, über 100°C und im Bereich 50°C bis 60°C konnte keine Standardabweichung aufgrund nicht vorhandener Messdaten ermittelt werden. Eine Prüfung gegenüber den Rohdaten der Beispielkurve bestätigt das.

Mit dem Werkzeug der Parameterbestimmung können nun Regeln verfasst werden, die sich dieses Wissen zunutze machen und dadurch schwer zu erkennende Fehler detektieren können.

7.3.2 Parameterwissen

Im vorangegangen Kapitel wurde beschrieben, wie Parametrisierung, als zusätzliche Information über einen Kurvenverlauf, dazu beitragen kann, Fehler zu finden. Dies gilt es nun verifizieren und Regeln zu verfassen, die dieses Wissen verwenden.
Zur Ermittlung der Referenzkurve, von der die Parameterbestimmung abgeleitet werden soll, ist genaue Anlagenkenntnis durch einen Experten von Nöten. Nur er kann eruieren, welcher Messdatenzeitraum sich für die Ermittlung der Kennzahlen eignet. Nach Rücksprache mit dem Experten, kann die in Abbildung 25 dargestellte Kurve als Referenzkurve gelten. Ihr gegenüber soll nun verifiziert werden, ob die Überlagerung durch Rauschen der Messkurve aus Abbildung 26 als Fehler erkannt werden kann. Zur Veranschaulichung werden die ermittelten Kennzahlen aus beiden Graphen in einer Tabelle gegenüber gestellt:

Partitionierung °C	fehlerfreier Graph б	einsetzendes Rauschen б	fehlerfreier Graph б * 2	Vortag 2 б	Vortag 1 б	einsetzendes Rauschen б	Folgetag 1 б	Folgetag 2 б
0								
10							3,18	
20	0,10	0,48	0,20			0,48	1,46	2,25
30	0,47	0,64	0,93	0,33	0,31	0,64	0,96	2,34
40	2,16	1,18	4,32	1,44	1,39	1,18	2,32	2,06
50		15,62				15,62		11,62
60	2,72	2,88	5,45			2,88		
70	11,47	5,72	22,94	4,68	3,69	5,72		3,46
80	2,34	2,98	4,69	1,29	1,69	2,98		2,89
90	0,77		1,54	0,33	0,30			
100								

Abbildung 29: Tabelle zur Veranschaulichung der parametrisierten Fehlererkennung

Dargestellt ist die Partitionierung der Temperaturbereiche von 0°C bis 100°C und die zugehörig ermittelten Standardabweichungen der Messkurven. Zunächst werden die Werte der Referenzkurve (fehlerfreier Graph б) mit dem Graphen aus Abbildung 26 (einsetzendes Rauschen б) miteinander verglichen. Hervorgehoben sind die Zellen, in denen die Werte der Standardabweichung größer sind als die der Referenzkurve und somit eine Messdatenüberlagerung durch Rauschen andeuten sollen. Es ist erkennbar, dass das Rauschen in den niedrigen Temperaturbereichen erfolgreich

erkannt wird, jedoch gibt es auch falsche Treffer in den oberen Temperaturregionen. Dies ist dem Umstand geschuldet, die Referenzkurve auch im fehlerfreien Zustand in den oberen Temperaturregionen rauschartiges Verhalten aufweist. Will man die falschen Diagnosen eliminieren, kann man dieses Wissen dazu benutzen die Regel zu modifizieren. Da man anhand der Werte der Referenzkurve weiß, dass die Standardabweichungen bei niedrigen Temperaturen für gewöhnlich sehr klein sind, kann der Grenzwert nach oben korrigiert werden. In der Tabelle wurden die Werte der Standardabweichungen der Referenzkurve verdoppelt (fehlerfreier Graph σ * 2) und wieder mit dem Fehlergraph verglichen. Jetzt fallen die falschen Diagnosen aus dem Raster, das Einsetzen des Rauschens wird aber weiterhin erkannt. Zusätzlich wurden noch die Folgetage 1 und 2 aufgetragen, bei denen das Rauschen ebenso aktiv war. Auch hier wird das Rauschen in den unteren Temperaturpartitionen zweifelsfrei erkannt. Zur Verifizierung sind beispielhaft noch die Vortage 1 und 2, bei denen der Fehler noch nicht auftritt, angegeben, um zu sehen, ob eventuell fälschlicher Weise ein Rauschen erkannt wird. Dies ist hier nicht der Fall.
Die Erkenntnisse haben die Parameterbestimmung als Hilfsmittel zur Fehlererkennung verifiziert. Dieses Wissen gilt es nun der Wissensbasis in Drools zuzuführen und eine Regel zu erstellen, die dadurch Fehler detektieren kann.
Die Parameterbestimmung der Referenzkurve erfolgt nur einmalig am Beginn des Hauptprogramms unter Anwendung des Java Codes aus Abbildung 27 und Abbildung 28. Die dadurch erhaltene TreeMap wird wie die Ergebnisse der Datenbankabfrage zur Wissensbasis hinzugefügt.

```
//...

// Parameterwissen der Wissensbasis uebergeben
ksession.insert(RefStdAbw_part_map);
ksession.insert(StdAbw_part_map);

//...
```

Abbildung 30: Hinzufügen des Parameterwissens zur Wissensbasis

Die Parameterbestimmung der zu überprüfenden Messdaten, wird analog zur Referenzkurve generiert und in Form einer Map der Wissensbasis zugänglich gemacht. Jetzt kann eine Regel dazu verwendet werden, Fehler durch Parameterwissen zu erkennen. Die Allgemeingültigkeit der Parameter der Referenzkurve sollte dabei an

ebenfalls gültigen Messdaten mittels der Regel getestet werden. Die Definition erfolgt dabei analog zu den Beispielregeln aus Kapitel 7.2.2.

```
rule "Messdatenueberlagerung durch Rauschen"
	when
		// Variablenbindung des Parameterwissens
		$treemap : TreeMap()
		$map : Map()
		// Pruefung auf Rauschen
		($treemap(Integer()) == $map(Integer())), ($treemap(Double()) < $map(Double()))
	then
		// Ausgabe einer Meldung auf der Konsole
		System.out.println("Messdatenueberlagerung durch Rauschen");
end
```

Abbildung 31: Regelbeispiel Messdatenüberlagerung durch Rauschen

Auf diese Weise kann auch anderes Parameterwissen, wie zum Beispiel Kenntnisse über temperaturabhängige Steigungskennzahlen oder Minimal- bzw. Maximalwerte einer Referenzkurve als zusätzliches Wissen zur Fehlererkennung eingebunden werden.

8 Ansätze zur Erweiterung

Mit dem BRM-System Drools wurde ein regelbasiertes Monitoring zur Fehlerüberprüfung implementiert, welches noch Erweiterung oder Anpassung ermöglicht. Diese sollen in diesem Kapitel untersucht werden.

8.1 Simulationsdaten und Langzeiteffekte

Mit dem regelbasierten Monitoring kann bereits ein Großteil der angesprochen Fehlerklassen detektiert werden. So stellt es sich nach wie vor als schwierig heraus, Langzeiteffekte wie Drifts zu erkennen, wenn keine redundanten Sensoren vorhanden sind, auf die man sich beziehen kann.

Abhilfe können hier Simulationsdaten liefern, die energietechnische Referenzanlagen durch Rechenmodelle abbilden. Häufig wird vor dem Bau einer Anlage das Potential durch Simulationen ermittelt. Diese Simulationen entscheiden oft über die Realisierung oder Nichtrealisierung eines Projekts. Deshalb sind bei realisierten Messanlagen meist Simulationsmodelle vorhanden, die zur Fehlerüberprüfung herangezogen werden können. Dabei muss der Experte vorab aber genauestens prüfen, ob sich die simulierten Daten mit den tatsächlich gemessenen Werten in Einklang bringen lassen. Dies wird im Rahmen einer Verifikation der Simulation durchgeführt. Dabei sollte

versucht werden, die Simulationsergebnisse den real auftretenden Messergebnissen anzugleichen. Nur dann sind die Simulationsmodelle als Faktenwissen zu gebrauchen.

Stehen verwendbare Simulationsdaten zur Verfügung, können die Daten in die Datenbank eingepflegt werden, wo sie dann zur Unterstützung des regelbasierten Monitoring abgerufen werden können. Bei sehr genauen und gut abbildenden Daten können die Simulationsergebnisse wie ein redundanter Sensor eingesetzt werden. Dabei sollte man das Delta zwischen Messwert und Simulationswert so wählen, dass die Fehlererkennung robust gegenüber falschen Treffern agiert. Durch dieses zusätzliche Wissen ist man nun in der Lage, Langzeiteffekte, wie das Abdriften einer Messkurve zu erkennen.

Außerdem können Simulationsdaten als Referenzdaten auch zur Parameterbestimmung verwendet werden. Wenn man beispielsweise noch keine passenden Messkurven für eine Parametrisierung aufzeichnen konnte, bieten sich Simulationskurven als Idealkurven gerade zu an. Oder man möchte die Anlage im laufenden Betrieb zu den Simulationsergebnissen hin optimieren. Dann kann das Monitoring mit Simulationsparameter helfen, den besten Betriebspunkt einer Anlage herauszufinden. Es genügt, solange verschiedene Einstellungen zu testen, bis das parametrisierte Monitoring keine Fehler mehr erkennt. Jedoch ist das in der Praxis meist nur für kleine Anlagen oder Komponenten möglich – komplexere Anlagen lassen sich nicht so exakt abbilden.

8.2 Möglichkeiten durch neuronale Netze

Neuronale Netze können durch adaptives, maschinelles Lernen von antrainiertem Wissen für weiteres Monitoring Konzept interessant sein. Ähnlich wie bei der Parameterbestimmung wird ein neuronales Netz mit mehreren Referenzkurven trainiert und kann dann bisher unbekannte Messkurven verifizieren. Vorteil hierbei ist, dass neuronale Netze im Gegensatz zur Parameterbestimmung, nicht ein starres Raster generieren, gegenüber dem geprüft wird, sondern durch das adaptive Lernen die Charakteristik der Messkurve komplett verinnerlicht wird.

Um zu überprüfen, ob neuronale Netze geeignet sind, Monitoring Aufgaben zu übernehmen, wurde in dieser Arbeit eine Bibliothek namens PyBrain [pyb] verwendet. PyBrain steht für „Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library” und ist eine Bibliothek zur Generierung neuronaler Netzwer-

ke. Dabei liegt der Fokus auf einer einfachen Handhabung und dem Ziel möglichst unkompliziert auch ohne großes Vorwissen die Möglichkeiten neuronaler Netze zu studieren. Durch die sehr gute Dokumentation und der Bereitstellung von Tutorials wird der Einstieg erleichtert und es lassen sich in relativ kurzer Einarbeitungszeit neuronale Netze erstellen.

Trainiert wurde das Beispielnetz anhand von nicht fehlerbehafteten Messwerten ca. 1 Woche vor Einsetzen eines Fehlers in den Messdaten. Dabei wurde überwacht gelernt, d.h. zu den Eingabemesswerten wurden auch die zugehörigen Ausgabemesswerte mitgeliefert. Dadurch lernt das neuronale Netz bei welchen Eingabemustern, welche Ausgabemuster zu erwarten sind. Diese Art des Lernens ist sehr zielgerichtet und macht es dem neuronalen Netz leichter zu Generalisieren. Nach der Trainingsphase wurden die fehlerhaften Messwerte dem neuronalen Netz übergeben und die Ergebnisse des Testlaufs betrachtet.

Abbildung 32: Ergebnis eines Testlaufs mit neuronalem Netz

Sehr gut zu erkennen ist, wie das neuronale Netz aufgrund der eingehenden Messdaten (In) prognostiziert, welche Werte die tatsächlichen Messdaten (Out) annehmen

sollten. Die zu erwartenden Messwerte (Prognose) sollten in etwa dem Verlauf der Eingabewerte abzüglich eines Offsets folgen. Da in diesem Fall, aufgrund eines Defekts, die tatsächlichen Messdaten (Out) aber eine sehr große Abweichung gegenüber der Prognose haben, stellen sie keine Lösung für das antrainierte neuronale Netz dar und der Fehler würde somit durch das neuronale Netz detektiert.
Die Überwachung von Messanlagen durch neuronale Netze stellt somit eine weitere Möglichkeit dar, durch adaptives Lernen Fehler zu erkennen. Auch hierbei können Simulationsdaten als Trainingsmuster dienlich sein, ein neuronales Netz zu generieren. Für unauffällige Fehler, die regelbasiert nicht oder kaum zu erfassen sind, kann ein entsprechend antrainiertes, neuronales Netz zum Monitoring Konzept beitragen.

9 Fazit und Ausblick

In dieser Arbeit wurde eine solide Basis für das Monitoring energietechnischer Referenzanlagen geschaffen. Die lehrreiche Auseinandersetzung mit verschiedenen Fehlerklassen und Monitoring Methoden hat dazu beigetragen, dass die Umsetzung leichter und zielgerichtet zu bewerkstelligen war. Durch die Befragung von Experten konnten neben der Auswahl einer Referenzanlage auch weitere Fehlerklassen und Informationsquellen zur Überwachung gefunden werden Die vorausgegangene Softwareanalyse und der Vergleich von Business Rule Management Systemen haben sich als sehr wertvoll für die weitere Vorgehensweise herausgestellt.
Mit JBoss Drools wurde ein Business Rule Management System gefunden, dass sich uneingeschränkt für das Monitoring energietechnischer Referenzanlagen einsetzen lässt. Die Ankopplung von JBoss Drools an die Messdatenbank des ZAE Bayern eröffnete alle Möglichkeiten des regelbasierten Monitorings. Es wurden Regeln verfasst, die einen Großteil der Fehlerklassen erkennen können und während des Anlagebetriebs validiert werden konnten. Darüber hinaus konnten durch die erweiterte Fehleranalyse Monitoring Methoden geschaffen werden, die über die einfache Messwertverifizierung hinausgehen. Die Frage, ob regelbasiertes Monitoring für den Einsatz in energietechnischen Referenzanlagen mithilfe eines Business Rule Management Systems zu bewerkstelligen ist, konnte positiv beantwortet werden.
Die Überlegungen und Ausführungen zur Erweiterung des Monitoring Konzepts trugen dazu bei, die Möglichkeiten des Monitorings durch Simulationsdaten und den Einsatz von neuronalen Netzen abzuschätzen. Die Auseinandersetzung mit der Ge-

schichte der Wissensverarbeitung und der Recherche über aktuelle, umgesetzte Monitoring Projekte lassen darauf schließen, dass in Zukunft die Thematik, vor allem in Bezug auf die Bereitstellung erneuerbarer Energien, noch weiter an Bedeutung gewinnen wird.

Ausblickend sollte zunächst der regelbasierte Ansatz weiter ausgebaut werden, in dem man beispielsweise Störungsmeldungen per E-Mail an die Fachanwender verschickt Die Einbeziehung einer weiteren Referenzanlage wäre ein nächster Schritt, um das Regelwissen gegenüber anderem Faktenwissen zu prüfen. Wird das System von den Benutzern wohlwollend angenommen, kann man versuchen mit den Werkzeugen von JBoss Drools die Fachanwender stärker in die Wartung und Regelerstellung mit einzubeziehen. Dafür kann man die Regelbausteine in natürliche Sprache übersetzen und das Webinterface für die Experten zugänglich machen.

Die vielversprechenden Ansätze zur Erweiterung des Monitoring Konzepts durch Simulationsdaten und neuronalen Netzen stellen eine weitere Herausforderung für zukünftige Implementierungen dar. Hierbei sollte man den Zeitverlauf bzw. das Zeitverhalten mit in die Betrachtung einbeziehen.

Mit Abschluss dieser Arbeit steht ein funktionierendes, regelbasiertes Monitoring für energietechnische Referenzanlagen bereit, das durch Erweiterung und Verbreitung zu einer vollständigen Monitoring Softwarelösung reifen kann.

Abbildungsverzeichnis

Abbildung 1: Architektur von Expertensystemen [Boe071] 5
Abbildung 2: Hydraulische Verschaltung der Hauptkomponenten der SolarHeatCool+PCM Anlage 10
Abbildung 3: temporärer, konstanter Messfehler [Abh10] 12
Abbildung 4: Messfehler-Peak [Abh10] 13
Abbildung 5: Unterschiedliche Ausprägungen von Rauschen [Abh10] 13
Abbildung 6: Verschmelzung von identischen Regelprämissen in einem Rete-Netzwerk [ret2] 25
Abbildung 7: Fiktive Regel in Pseudocode zur Veranschaulichung der Rete-Knoten 26
Abbildung 8: Beispiel eines Rete-Datenflussgraphen nach der Instanziierung 26
Abbildung 9: Einfügen zwei neuer Tupel in das working memory 27
Abbildung 10: Entfernen eines Tupels aus dem working memory 28
Abbildung 11: Schema des bestehenden Systems 30
Abbildung 12: Kommunikationsverbindung per JDBC zur PostgreSQL-Datenbank aufbauen 31
Abbildung 13: DbUtils ermöglicht das Mapping der SQL-Anfrage auf ein JavaBean-Objekt 32
Abbildung 14: syntaktischer Aufbau einer Regel in DRL-Regelsprache [dro1] 34
Abbildung 15: Syntaxdiagramm für eine Musterdefinition [dro1] 34
Abbildung 16: fiktive Musterdefinition zur Veranschaulichung der Syntaxmöglichkeiten 35
Abbildung 17: Beispielregel Grenzwertüberschreitung 36
Abbildung 18: Beispielregel Modifikation durch Reguläre Ausdrücke 37
Abbildung 19: Ausfall von zwei Temperaturfühlern am gleichen Messpunkt 37
Abbildung 20: Beispielregel Abweichung der Temperatur am gleichen Messpunkt 38
Abbildung 21: Störungserkennung durch Verletzung einer physikalischen Gesetzmäßigkeit 39
Abbildung 22: Beispielregel Prüfung auf Widerspruch einer physikalischen Gesetzmäßigkeit 40
Abbildung 23: Wissensbasis mit integrierter Regelbasis generieren 40

Abbildung 24: Wissensbasis mit Regelbasis initialisieren und Messdaten übergeben 41
Abbildung 25: fehlerfreier, typischer Temperaturverlauf eines Sensors zur Parameterbestimmung 43
Abbildung 26: Sensorfehler Messdatenüberlagerung durch Rauschen 44
Abbildung 27: Java Code zur Bestimmung der Standardabweichung..... 45
Abbildung 28: Java Code zur Partitionierung der Standardabweichung auf Temperaturbereiche 46
Abbildung 29: Tabelle zur Veranschaulichung der parametrisierten Fehlererkennung 47
Abbildung 30: Hinzufügen des Parameterwissens zur Wissensbasis 48
Abbildung 31: Regelbeispiel Messdatenüberlagerung durch Rauschen 49
Abbildung 32: Ergebnis eines Testlaufs mit neuronalem Netz 51

Abkürzungsverzeichnis

B

B2B ..... *Business-to-Business*
BDMS ..... *Business Decision Management System*
BRE ..... *Business-Rule-Engine*
BRMS ..... *Business-Rule-Management-System*

C

CMS ..... *Condition Monitoring Systems*

D

DRL ..... *Drools Rule Language*
DSL ..... *Domain Specific Language*

G

GPL ..... *GNU General Puplic License*

H

HISS ..... *Human Interface Supervision System*

I

IT ..... *Informationstechnik*

J

JDBC ..... *Java Database Connectivity*

K

KI ..... *Künstliche Intelligenz*

L

LHS ..... *left hand side*

R

RC ..... *Release Candidate*
RHS ..... *right hand side*

S

SolarHeatCool+PCM ..... *Solares Heizen und Kühlen mit Absorptionskältemaschine und Latentwärmespeicher*
SPIRIT ..... *Symmetrical Probabilistic Intensional Reasoning in Inference Networks in Transition*
SQL ..... *Structured Query Language*

W

WODM ..... *IBM WebSphere Operational Decision Management*

Literaturverzeichnis

[Abh10] Abhishek B. Sharma, L. G. (Juni 2010). Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Journal: Transactions on Sensor Networks (TOSN)*, (S. Volume 6 Issue 3, Article No. 23). New York, NY, USA.

[apa] *Apache Axis*. (26.11.2012). Von http://de.wikipedia.org/wiki/Apache_Axis abgerufen

[apa1] *Apache Commons*. (26.11.2012). Von http://de.wikipedia.org/wiki/Apache_Commons abgerufen

[Boe07] Boersch, I., Heinsohn, J., & Socher, R. (2007). *Wissensverarbeitung* (2. Ausg.). S. 6-16. München: Elsevier Spektrum Akademischer Verlag.

[Boe071] Boersch, I., Heinsohn, J., & Socher, R. (2007). *Wissensverarbeitung* (2. Ausg.). S. 8. München: Elsevier Spektrum Akademischer Verlag.

[bus] *Business-Rule-Engine*. (26.11.2012). Von http://de.wikipedia.org/wiki/Business-Rule-Engine abgerufen

[bus1] *Business-Rule-Repository*. (26.11.2012). Von http://de.wikipedia.org/wiki/Business-Rule-Repository abgerufen

[d3w] *d3web*. (26.11.2012). Von http://www.is.informatik.uni-wuerzburg.de/forschung/anwendungen/d3web/ abgerufen

[d3w1] *d3web*. (26.11.2012). Von http://d3web.sourceforge.net/ abgerufen

[dbu] *DbUtils*. (26.11.2012). Von http://commons.apache.org/dbutils/ abgerufen

[dem] *Demonstratoren von d3web*. (26.11.2012). Von http://www.is.informatik.uni-wuerzburg.de/forschung/anwendungen/d3web/anwendungen/ abgerufen

[dre] *Drei Sinne für die Sicherheit*. (26.11.2012). Von http://www.maschinewerkzeug.de/index.cfm?pid=1418&pk=54454&p=1 abgerufen

[dro] *Drools - The Business Logic integration Platform*. (26.11.2012). Von http://www.jboss.org/drools abgerufen

[dro1] *Drools Expert User Guide*. (26.11.2012). Von http://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/html_single/ abgerufen

[ecl] *Eclipse (IDE)*. (26.11.2012). Von http://de.wikipedia.org/wiki/Eclipse_(IDE) abgerufen

[ene] *Energieerhaltungssatz: Energieaustausch*. (26.11.2012). Von http://de.wikipedia.org/wiki/Energieerhaltungssatz#Energieaustausch abgerufen

[For82] Forgy, C. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence* (19), S. 17-37.

[ges] *Geschäftsregel-Managementsystem*. (26.11.2012). Von http://de.wikipedia.org/wiki/Geschäftsregel-Managementsystem abgerufen

[ibm] *IBM - BRMS JRules datasheet*. (26.11.2012). Von ftp://ftp.software.ibm.com/software/emea/de/websphere/BRMS_JRules_datasheet_7030_WSD14062-DEDE-00.pdf abgerufen

[ibm1] *IBM - WebSphere ILOG Business Rule Management Systeme*. (26.11.2012). Von http://www-01.ibm.com/software/de/websphere/business-rule-management/ abgerufen

[jav] *Java Database Connectivity*. (26.11.2012). Von http://de.wikipedia.org/wiki/Java_Database_Connectivity abgerufen

[jav1] *JavaBeans - Wikipedia*. (26.11.2012). Von http://de.wikipedia.org/wiki/JavaBeans abgerufen

[lab] *LabVIEW*. (26.11.2012). Von http://de.wikipedia.org/wiki/LabVIEW abgerufen

[Leo11] Leon Wu, G. K. (2011). Data Quality Assurance and Performance Measurement of Data Mining for Preventive Maintenance of Power Grid. *KDD4Service '11.* San Diego, California, USA.

[MHe11] M. Helm, K. H. (21.03.2011). *Solares Heizen und Kühlen mit Absorptionskälteanlage und Latentwärmespeicher.* Von http://idserver.fiz-karlsruhe.de/cgi-bin/ih?ID=14106.1.0.3564862543&ACTION=grep&PARAM=htmpdf&term=pdf;2011_0329605O.1.pdf abgerufen

[Muu05] Muuß, T. (2005). *Condition Monitoring Systeme für Windenergieanlagen - Einsatz und Zertifizierung.* Rostock : DGZfP-Jahrestagung 2.-4. Mai 2005.

[ope] *Open Source Rule Engines in Java*. (26.11.2012). Von http://java-source.net/open-source/rule-engines abgerufen

[ope1] *OpenRules*. (26.11.2012). Von http://openrules.com abgerufen

[pos] *PostgreSQL*. (26.11.2012). Von http://de.wikipedia.org/wiki/PostgreSQL abgerufen

[pyb] *Pybrain*. (26.11.2012). Von http://pybrain.org/ abgerufen

[reg] *Regelbasiertes System*. (26.11.2012). Von http://de.wikipedia.org/wiki/Regelbasiertes_System abgerufen

[res] *ResultSet*. (26.11.2012). Von http://docs.oracle.com/javase/6/docs/api/java/sql/ResultSet.html abgerufen

[ret1] *Rete algorithm - Rete-NT*. (26.11.2012). Von http://en.wikipedia.org/wiki/Rete_algorithm#Rete-NT abgerufen

[ret] *Rete algorithm*. (26.11.2012). Von http://en.wikipedia.org/wiki/Rete_algorithm abgerufen

[ret2] *RETE-Algorithmus.* (26.11.2012). Von http://www2.inf.fh-rhein-sieg.de/~pbecke2m/xps/regel2.pdf abgerufen

[spi] *SPIRIT - The next generation expert system*. (26.11.2012). Von http://xspirit.de/ abgerufen

[sta] *Standardabweichung*. (26.11.2012). Von http://de.wikipedia.org/wiki/Standardabweichung abgerufen

[sta1] *Standardabweichung: Berechnung für auflaufende Messwerte*. (26.11.2012). Von http://de.wikipedia.org/wiki/Standardabweichung#Berechnung_f.C3.BCr_auflaufende_Messwerte abgerufen

[ste] *Steigung*. (26.11.2012). Von http://de.wikipedia.org/wiki/Steigung abgerufen

[tem] *Temperaturextrema*. (26.11.2012). Von http://de.wikipedia.org/wiki/Temperaturextrema abgerufen