Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 41

Umsetzung eines Systems zur regelbasierten Analyse von Gesundheitsdaten

Johannes Ernst

Studiengang:	Informatik
Prüfer/in:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer/in:	Dipl.-Inf. Frank Steimle
Beginn am:	15. Juni 2015
Beendet am:	15. Dezember 2015
CR-Nummer:	C.2.4, H.2.8, H.3.1, H.3.4

Kurzfassung

Für die Enhancing Chronic Patients' Health Online (ECHO) Plattform wurde im Rahmen dieser Arbeit ein regelbasiertes Analysesystem für Gesundheitsdaten entwickelt. Das Analysesystem hat zum Ziel, die vorher im Quellcode der ECHO-Plattform integrierten Datenanalysen zu ersetzen und dem Anwender spezifische Analysen zu ermöglichen, die jederzeit erzeugt, geändert oder gelöscht werden können. Das Analysesystem soll der besseren Diagnose und Behandlung von chronisch obstruktiven Lungenerkrankungen dienen. Damit soll auch die Möglichkeit gegeben sein, über eine frühzeitige Erkennung von Exazerbationen die Lebensqualität der Patienten zu verbessern, indem der Patient rechtzeitig gewarnt werden kann. Die Früherkennung erfolgt durch die Verwendung von Data Mining-Verfahren. Zur Erstellung von regelbasierten Analysen verwendet das Analysesystem eine Analysatorvorlage, mit der Anwender die Regeln der Analyse bestimmen. Die Analysatorvorlage legt die Verwendung der Analyseoperatoren fest (wann die Analyse gestartet werden soll), und eine mögliche Aktion, die bei der Erfüllung der Analyseregeln ausgeführt wird. Das Analysesystem verwendet zur Ausführung der Analysatoren das Ausführungssystem Node-Red. Die Analysatorvorlage wird vom Analysesystem in einen Ablaufplan transformiert. Der Ablaufplan wird auf einem Ausführungssystem erzeugt und aktiviert. Mit der verwendeten Multi-Serverarchitektur kann das Analysesystem die Analysatoren auf unterschiedlichen Serversystemen aktivieren, um die Ausführungsgeschwindigkeit und Skalierung zu verbessern. Zur Kommunikation mit dem Analysesystem und dem Analysator wird das MQTT-Protokoll verwendet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Diese Arbeit hat zum Ziel, eine Regelsyntax und ein Analysesystem zu entwickeln, das für die regelbasierte Analyse von Gesundheitsdaten eingesetzt werden kann. Es wird in der Enhancing Chronic Patients' Health Online (ECHO) Plattform eingesetzt zur Performance-Steigerung bei der Durchführung von Datenanalysen seitens des ärztlichen Personals, um Diagnose, Behandlung und automatische Warnung von Patienten vor einer möglichen Verschlechterung des Krankheitsverlaufs zu verbessern. Die ECHO-Plattform entstand durch die Förderung des Bundesministeriums für Forschung und Bildung (BMBF) in einem deutsch-griechischen Forschungsprojekt, um die Gesundheit von Patienten mit chronisch obstruktiver Lungenerkrankung zu verbessern, indem die Patienten Onlinedienste verwenden. In diesem Kapitel wird der Hintergrund der ECHO-Plattform beleuchtet, es wird die Motivation erläutert, warum eine regelbasierte Analyse zur Unterstüzung beitragen kann, und die Aufgabenstellung der Arbeit beschrieben.

Hintergrund

Derzeit existiert ein System [Ste14] des Forschungsprojekts Enhancing Chronic Patients' Health Online zur Verwaltung und Bereitstellung von Gesundheitsdaten, das zur Erfassung von Patientendaten für Lungenerkrankungen über eine Online-Plattform bereitgestellt wird. Hierbei werden durch das ärztliche Personal bei den Kontrolluntersuchungen und durch tägliche Reports der Patienten selbst die Gesundheitsdaten erfasst, die an einer chronisch obstruktiven Lungenerkrankung (kurz COPD für Chronic Obstructive Pulmonary Disease) leiden, um den Krankheitsverlauf zu dokumentieren und den Arzt bei Diagnosen und der Verbesserung des Krankheitszustands zu unterstützen. Die Erkrankung COPD ist ein Sammelbegriff für eine Gruppe von Lungenkrankheiten, die den Befall des Lungengewebes zur Folge haben, wie die chronisch obstruktive Bronchitis in Kombination mit einem Lungenemphysem. Unter einer chronischen Bronchitis versteht man einen Husten mit Auswurf über zwei Folgejahre bei mehr als 3 Monaten des Auftretens pro Jahr. Ein Lungenemphysem ist eine Überblähung des Lungengewebes, die die Wandstruktur der Lungenbläschen zerstört. Bei COPD handelt es sich um eine chronische Erkrankung der Lunge, die zu Husten, Atemnot oder erhöhtem Auswurf führt. Die Weltgesundheitsorganisation (WHO) [Org] zählt weltweit mehr als 64 Millionen Menschen, die davon betroffen sind, was eine Verbreitungsquote von 14 Prozent der erwachsenen Menschen entspricht. COPD wird in der Regel durch Zigarettenrauchen oder das Passiv-Rauchen verursacht, auch oft bekannt als Raucherhusten oder Raucherlunge. Aber nicht nur Raucher sind davon betroffen, die Krankheit kann auch durch Umweltverschmutzung oder berufsbedingte Belastungen verursacht werden. [cop]
Die gesammelten Gesundheitsdaten der ECHO-Plattform könnten dazu verwendet werden, frühzeitig eine drastische Verschlimmerung der Krankheit zu erkennen, die als Exazerbation bezeichnet wird,

und um dann weitere Aktionen des Arztes einleiten zu können. [Ste14] Als Exazerbation versteht man die schubweise schnell auftretende Verschlimmerung der COPD-Symptome wie Husten oder Auswurf, die akut zunehmen und zu einer raschen Verschlechterung der Atmungsfunktion führen können. [cop]

Motivation

Das regelbasierte Analysesystem in der ECHO-Plattform soll dazu verwendet werden, dass ärztliches Personal eigene Analysen ohne Hilfe oder Änderungen am Programmcode erstellen kann, die auf personenspezifische Grenzwerte oder mögliche Kombinationen ausgewählter Werte reagieren und vom Anwender definierte Aktionen auslösen. Damit kann der Arzt selbst auf Basis einzelner Patienten oder des gesamten Patientenstamms regelbasierte Analysen in beliebigem Umfang anlegen, ohne programmiertechnische Kenntnisse oder eine komplexe Grammatik zu verstehen, die auf die vom Benutzer spezifizierten Werte hin Aktionen und Benachrichtigungen auslösen oder eigene Berechnungen durchführen. Das kann dem Patienten wie dem Arzt helfen, den aktuellen Stand der Erkrankung oder frühzeitige Verschlechterungen zu erkennen und den Patienten veranlassen, den Arzt zu besuchen. Damit werden die globalen und statischen Analysen abgelöst, die nur manuell geändert werden können und nur bestimmte Problemfälle abdecken, aber sich nicht beliebig und schnell modifizieren lassen und einen Eingriff des Programmierers erfordern.
Aktuell verfügt die ECHO-Plattform zur Datenerfassung und Verwaltung nur über statische Analysen, die fest einprogrammiert sind und nur feste Parameter der täglichen Reports prüfen, die auch nur verwendet werden, um die Patienten über fehlende Reports zu informieren oder Warnungen bei kritischen Werten zu versenden. Auch spezifische Intervall-Prüfung und Benachrichtigung lässt sich aktuell nicht umsetzen. Eine Änderung der Analysen ist durch das ärztliche Personal nicht möglich, sondern muss von einem Programmierer vorgenommen werden. Auch lassen sich durch das Personal keine gesonderten Regeln anlegen, um personenspezifisch die Daten der Patienten zu prüfen und auf besondere Eigenheiten des Krankheitsverlaufs einzugehen. Eine mögliche frühzeitige Erkennung einer Exazerbation lässt sich durch das bestehende System ebenfalls nicht implementieren.
Des Weiteren wäre es hilfreich, über die gesammelte Datenmenge anonymisiert eine frühzeitige Erkennung der Exazerbation zu ermöglichen, die per Data Mining Patient und Arzt über eine eventuell bevorstehende Verschlechterung informiert und rechtzeitig Unterstützung leistet, ohne dass bereits Symptome auftreten - sondern nur anhand der täglich abgegebenen Reports.

Aufgabenstellung

Das Ziel der Masterarbeit ist es, ein regelbasiertes Analysesystem zu entwickeln, das anhand einer einfachen Regelsyntax definiert und angesteuert wird. Dazu sollen erst existierende Systeme analysiert werden, welche Analysen von Gesundheitsdaten durchführen. Danach folgen der Vergleich, die Auswahl und die Tests verfügbarer Werkzeuge zur Datenanalyse, die zur Verwendung kommen könnten. Dann soll eine Definition einer Regelsyntax erfolgen, die zur Erstellung einfacher Datenanalysen dienen soll. Das Analysesystem soll später vom Anwender selbst verwendet werden, um Analysen zu deklarieren. Nach der Definition der Regelsyntax soll das Analysesystem mit der Regelsyntax

umgesetzt werden, was auch ein Konzept für den Datenschutz des Zugriffs auf die Gesundheitsdaten enthalten soll. Die Umsetzung soll als Webdienst erfolgen, der per REST API angeboten wird und das Management von Analysen übernehmen soll, welches auch den Zugriff auf die Ergebnisse der Analysen ermöglichen und bereitstellen soll. Durch anschließende Tests mit selbst generierten Testdaten soll die Funktion des Analysesystems geprüft werden. Die bereits im alten Analysesystem hinterlegten Datenanalysen sollen auf das neue regelbasierte Analysesystem migriert werden.

Des Weiteren soll im Analysesystem ebenfalls über die Regelsyntax prototypisch Data Mining zur Verfügung gestellt werden, das auf Basis der Regelsyntax initiiert werden kann. Hierbei sollen bestehende Data Mining-Werkzeuge auf Integration geprüft und eine mögliche Lösung präsentiert werden.

Gliederung

Nach folgender Gliederung ist die Arbeit aufgebaut:

Kapitel 2 beschreibt verwandte Arbeiten und Grundlagen.

Kapitel 3 stellt den Entwurf der regelbasierten Regelsyntax vor.

Kapitel 4 spezifiziert die Umsetzung des Analysesystems.

Kapitel 5 evaluiert die Performance und Skalierbarkeit.

Kapitel 6 fasst die Ergebnisse zusammen und gibt einen Ausblick.

2 Vergleichbare Arbeiten, Grundlagen und Analysetechniken

Das Kapitel befasst sich mit der Vorstellung vergleichbarer Arbeiten, die sich mit ähnlichen oder gleichen Themen auseinandersetzen. Anschließend werden Grundlagen und Methoden der Datenanalyse diskutiert. Es wird ein Einblick in regelbasierte Ausführungssysteme geliefert und eine Beschreibungssprache für eine Regelsyntax und ein Workflow Management System betrachtet. Abschließend folgt eine Einführung zum Thema Data Mining und verfügbare Analysetechniken.

2.1 Vergleichbare Arbeiten

Die Analyse von Datensätzen und im Besonderen auch Gesundheitsdaten kann über eine Vielzahl von Konzepten und Methoden erfolgen und hat ein großes Potenzial zur Unterstützung des Arztpersonals und die Früherkennung von Krankheiten. Wullianallur Raghupathi et al. beschreiben in [RR14] das Potenzial der Datenanalyse im medizinischen Umfeld und mögliche Methoden zur Umsetzung mit Big Data-Analysen. Durch das große Potenzial an Datenquellen, unterschiedlichsten Patienten, Datensätzen und deren Bedingung zur schnellen Verarbeitung für die Früherkennung ist dies für traditionelle Analysesysteme nur schwer umsetzbar. Aus diesem Grund werden auf Big Data spezialisierte Analysesysteme vorgeschlagen, die mit der Menge an Daten und Datenformen (strukturiert, unstrukturiert und teilstrukturiert) umgehen können. Sie eröffnen jedoch die Möglichkeit zur einfachen, schnellen und effizienten Bestimmung eines Krankheitsbefundes, Warnungen vor Krankheitsverschlechterung oder Risiken und Einsparungen im Gesundheitssystem. Als Analysemethode können statistische Verfahren für die klinische Auswertung von Patientendaten eingesetzt werden, die nicht auf einen konsistenten Fluss an Daten setzen. Weitere statische Analysen werden für Genom- und Profilanalysen empfohlen. Als weiteres Verfahren kann die Echtzeitanalyse von medizinischen Sensoren verwendet werden, die auf einem konstanten Fluss an Sensorwerten basiert. Bilder aus Röntgenaufnahmen oder 3D-Bilderzeugungsverfahren können in Bildanalysewerkzeugen verarbeitet werden.

Die vom Autor angewendete Methodik wird in vier Schritten klassifiziert: (i) Schritt 1 besteht darin, zu ermitteln, ob die Voraussetzung für eine Datenanalyse gegeben ist. (ii) Schritt 2 dient dazu, zu erfassen, welche Problemstellungen die Analysen prüfen sollen. (iii) In Schritt 3 wird die Architektur geschaffen. (iiii) Im letzten Schritt wird die Architektur umgesetzt, eine Evaluation und Validierung der geschaffenen Analysestruktur durchgeführt und mit Tests geprüft.

Die konzeptuelle Architektur für die Analysen besteht aus einem Framework, das die im Big-Data-Umfeld üblichen verteilten Strukturen einsetzt. Die darin enthaltenen Verarbeitungsknoten verfügen nur über den eigenen Teil der Daten. Es sieht Datenquellen von Rohquellen vor, die über die nächste

Stufe der Transformationsstufe in Middleware, Extract Transform Load oder Data Warehouse-Techniken transformiert werden, um sie für die Analyseverfahren aufzubereiten. Die transformierten Daten werden über bekannte Big Data Platform-Werkzeuge wie Hadoop oder MapReduce verarbeitet und die Ergebnisse in Reports, OLAP oder Tabellen bereitgestellt.

Die Echtzeitanalyse von Gesundheitsdaten und Messwerten wurde in den letzten 20 Jahren nur sehr wenig betrachtet und oft nur auf Displays dargestellt. Deshalb schlagen Marion Blount et al. in [BEE+10] eine flexible Analyseplattform namens Artemis Analytic System vor, die in Echtzeit anhand von Datenströmen medizinische Analysen durchführt, um Komplikationen frühzeitig und automatisiert zu erkennen. Sie bietet auch die Möglichkeit der automatischen Erkennung von Beziehungen zwischen physiologischen Datenströmen und latenten Erkrankungen sowie die laufende Verfeinerung von Analysen, indem fortwährend Grenzwerte und Analyseregeln angepasst werden. Dazu werden laufende Datenströme von medizinischen Sensoren vor Ort erfasst, über ein Transportprotokoll (IP) übertragen und zentral verwaltet in das System eingespeist. Das eignet sich deshalb besonders für größere klinische Einrichtungen. Für weitere Analysen wie Data Mining werden die Datenströme aufgezeichnet und persistent gespeichert. Die Analysen des ärztlichen Personals laufen in Echtzeit. Die Analysen bestehen aus Operatorknoten, welche aus Complex Event Processing (CEP) abgeleitet sind. Die Knoten empfangen einen oder mehrere Ströme, verarbeiten diese und erzeugen einen oder mehrere Ausgangsströme. Die Operatorknoten, die einen Teil der Analyse repräsentieren, werden mit der Programmiersprache Stream Processing Application Declarative Language (SPADE) für Stream Computing Systems beschrieben. SPADE bietet eine höhere Abstraktionsebene für Streaming-Systeme und hat hohe Ähnlichkeit mit StreamSQL für einen effizienten Einsatz und bietet die Möglichkeit für Erweiterungen in typischen Programmiersprachen. Die in SPADE beschriebenen Analysen werden in gerichtete Graphen kompiliert, die einem Datenflussgraphen gleichen. Eine Analyse enthält dabei einen oder mehrere SPADE-Operatoren, wovon wiederum mehrere Analysen auf derselben Engine betrieben werden. Die eingehenden Datenströme werden hierfür dupliziert und an alle Analyse Flows weitergeleitet. Bei Erreichen der festgelegten Regeln können vom System Benachrichtigungen und andere Aktionen ausgeführt werden, die den Patienten oder Arzt warnen. Somit ist es möglich, eine Echtzeitanalyse verschiedenster Patienten zentral zu betreuen, um Symptome und Krankheiten frühzeitig zu erkennen und zu bekämpfen.

Die Analyse von Datensätzen und im Besonderen Gesundheitsdaten kann über eine Vielzahl von Konzepten und Methoden erfolgen. In [Ste14] von Frank Steimle werden statische Analysen für die Gesundheitsdaten per Wenn-Dann-Regeln im Backend beschrieben und verwendet, um frühzeitige Exazerbationen zu erkennen und möglicherweise zu behandeln. Dabei wird global über die statischen Analyseregeln der Zustand aller Patienten geprüft. Die Regeln in der Analysesoftware sind dabei hardkodiert - als eine Kombination aus SQL-Prozeduren und Verarbeitungslogik in Javascript. Für die Analysen sind zwei Regeln vorhanden, eine für die Prüfung der täglichen Reports, wenn bestimmte Felder des Reports mit Ja oder Nein bestätigt werden. Ebenfalls wird über eine zeitbasierende Regel geprüft, ob der Patient seinen täglichen Report ausgefüllt hat. Beide Analysen werden über statische SQL Statements implementiert, die per dynamischer Einbettung im NodeJs Framework verarbeitet werden. Sie lösen bei der Ausführung eine Benachrichtigung aus, die an den Anwender verschickt wird.

A Situation Recognition Service based on Modeling and Executing Situation Templates (SitRS) von Pascal Hirmer et al. befasst sich in [HWS+15] mit der Analyse von Daten zur Situationserfassung

im Bereich Internet of Things. Es treten eine große Anzahl an unterschiedlichen Sensoren in Aktion, die erhebliche Mengen an Sensordaten liefern, die verarbeitet werden müssen, um Situationen aus diesen Daten erkennen zu können. Dazu wird eine Analyse der Raw-Werte der Sensoren durchgeführt, indem Low-Level-Daten abstrahiert und aggregiert werden, die sich einfacher verarbeiten lassen als eine größere Menge an reinen Sensordaten. In diesem Fall werden Situationen als XML Templates beschrieben, die ein Modell bilden. In der Vorlage werden die überwachten Sensoren und Bedingungen deklariert, die für die Situation vonnöten sind, und der Typ der erfassten Situation. Die Deklaration erfolgt anhand von Situation-Aggregation-Bäumen als gerichteter Graph in einer Baumstruktur. Die überwachten Sensoren werden in den unteren Blättern des Baums beschrieben. Die gelieferten Daten der Sensoren werden über die darüber liegenden Bedingungsknoten gefiltert. Die Bedingungsknoten liefern boolesche Werte aus dem Vergleich. Anhand der Template-Struktur werden mit boolesche Operatoren die Situationsbedingungen beschrieben, die je nach Rückgabe der booleschen Werte der Bedingungsoperatoren und der booleschen Verknüpfungen den Situationseintritt als erkannt oder nicht erkannt zurückliefern.
Die Analyse wird in der Open Source Workflow Engine Node-Red ausgeführt. Dazu wird das Modell in einen als Datenflussgraph beschriebenen Workflow transformiert, welcher in dem Extensible Markup Language (XML) Template beschrieben wurde. Diese Ausführung des Workflows in der Flow Engine liefert die booleschen Werte wahr oder falsch zurück, was den Eintritt oder Nicht-Eintritt einer Situation abbildet. Die Sensorwerte werden dazu in zeitabhängigen Intervallen per Representational State Transfer (REST)-Anfrage an die Flow Engine geliefert, die an Hand des Flows die Situationserkennung durchführt, bis das Situations-Template wieder gelöscht wird.

Ein Arbeitsablauf (Workflow) beschreibt Arbeitsschritte in einem klar definierten Ablauf, welcher Inputs konsumiert, verarbeitet, transformiert und Ergebnisse ausgibt. Workflows werden oft für Analysen in der Bioinformatik eingesetzt. Burkhard Linke et al. beschreiben in [LGG11] ein Workflow-System namens Conveyor zur Durchführung von Analysen der Annotation der Gene medizinischer Daten in der Bioinformatik. Conveyor beschreibt Analysen in Workflows als gerichtete Graphen, bestehend aus Knoten, die Daten einlesen, Knoten für die Berechnungen und für die Ausgabe der Resultate. Alle Knoten werden als Objekte erzeugt. Datentypen für die Analyse werden in Typ-Klassen beschrieben, die von einem Interface abgeleitet werden. Die Objekte bilden den Typ von dem Datensatz ab, der für die Analyse verwendet wird. Alle Datensätze müssen als Objektdarstellung deklariert werden. Funktionsbausteine, die selbst implementiert werden, sind vom Benutzer spezifisch programmiert und leiten ebenfalls von einem Interface ab. Beides kann flexibel per Hochsprache programmiert werden. Selbst geschriebene Funktionsbausteine werden als XML-Repräsentation abgespeichert und validiert, um die Funktionsweise sicherzustellen.
Das Implementieren der Analyseregeln erfolgt über ein grafisches Anwendungsprogramm, indem per Drag and Drop die Analysebausteine in ihren Arbeitsablauf geschoben und Verbindungen dazwischen gezogen werden. Durch Hintereinanderreihen von Arbeitsschritten, Schleifen und parallelen Verbindungen lassen sich damit komplexe Analysemethoden beschreiben. Durch Plugins werden die Regeln mit weiteren Funktionalitäten erweitert und zum Abruf von externen Daten verwendet, die für die Analyse notwendig sind, jedoch nicht als Datentyp beschrieben sind. Fertige Analyseregeln werden von der Anwendung auf die Processing Units übertragen, die Conveyor's Processing Engine enthalten und somit dem Client/Server-Prinzip folgen. Ausführende Knoten führen die Berechnungen der Analysen durch.

Zur Erfassung und Erkennung des Krankheitsverlauf verwendet [SMFGJ15] von Daniel Sanchez-Morillo et al. einen Onlinedienst, den die Patienten verwenden, um tägliche Berichte zu erstellen. Die Berichte enthalten Daten über die aktuellen Zustand des Patienten. Das Analysesystem soll die Lebensqualität von COPD-Patienten verbessern, indem die Exazerbation vorhergesagt wird. Die erstellten Berichte und Behandlungsdaten aller Patienten werden verwendet, um über einen Clustering-Algorithmus mit Entscheidungsbaum vorauszusagen, wann eine Exazerbation auftreten kann. Die Vorhersage erfolgt dabei pro Patient, indem die enthaltenen Daten des täglichen Berichts zur Analyse herangezogen werden. Bei der Testdurchführung von einer Gruppe von 16 Patienten konnten 31 von 33 Exazerbationen vorgesagt werden, was zur Verbesserung der Lebensqualität führte. Die Abweichung des vorhergesagten Zeitraums lag bei 4,5 ± 2,1 Tagen.

2.2 Datenanalyse

Bei der Datenanalyse oder generell bei der Analyse wird aus vorliegenden Informationen neues Wissen generiert. Betrachtet werden können Einzeldaten oder eine Vielzahl von Datensätzen. Die dazu angewendeten Verfahren - meist mathematischer oder statistischer Art - transformieren, sortieren, strukturieren oder kombinieren Daten, welche dann als Ergebnislieferanten für die vorher in der Datenanalyse gestellten Probleme dienen können. [Cle08]
Die Datenanalyse besteht aus fünf Phasen, die als wiederkehrender Kreislauf ausgeführt werden können. Die **Zieldefinition**, also die Beschreibung der Problemstellung, ist die erste Phase und definiert das Problem, welches durch die Datenanalyse gelöst werden soll. In der zweiten Phase, der **Datenbeschaffung**, werden für die Analyse Daten aus den laufenden Prozessen aufgezeichnet. Ohne verfügbare Daten ist eine Analyse und eine Anwendung der Methoden nicht möglich. Nachdem Daten erfolgreich gespeichert wurden, werden in der **Datenaufbereitung** die gesammelten Daten auf Vollständigkeit überprüft und per Plausibilitätsprüfung falsche Daten entfernt oder korrigiert. Eine Transformation ist erforderlich, wenn die Datenanalyse ein bestimmtes Datenmodell oder die Verknüpfung von Daten aus unterschiedlichen Quellen benötigt. Sind die Daten aufbereitet, startet die nächste Phase. Die **Datenauswertung** ist die eigentlich Anwendung der Analysetechnik - durch Verwendung der Algorithmen an den aufbereiteten Datensätze. Die Techniken liefern neue Erkenntnisse oder neue Datensätze als Ergebnisse, die in der letzten Phase der **Ergebnisinterpretation** zur Lösung des Problems angewendet werden. Ergebnisse aus der letzten Phase können wieder als Basis für neue Analysen angewendet werden, um den Kreislauf zu schließen und die Analyse zu verfeinern.

Datenanalyse beschreibt die Herangehensweise, von methodischen Verfahren an vorliegenden Datensätzen Erkenntnisse zu erhalten. Sie erlaubt auch die Suche nach Strukturen für weitere Analysen oder das Erstellen und die Prüfung von Hypothesen. Meist kommen bei der Analyse von Daten statistische und mathematische Methoden zum Einsatz. [Kam] Die Analysemethoden kann man in vier Gruppen einteilen:

Inferenzielle Datenanalyse ist ein Teil der Statistik und verwendet statische Verfahren. Hierbei werden einzelne Datensätze zufällig (probenartig) gewählt und deren Merkmale mathematisch analysiert und auf die Gesamtheit der Datensätze abstrahiert. Man schließt aus einer kleinen Menge an Datensätzen auf alle anderen. Es ist Teil der Stochastik der Mathematik. Eine Methode der inferenziellen Datenanalyse wäre die Intervallabschätzung von Parametern.

Deskriptive Datenanalyse ist das Finden von empirische Merkmalen in Datensätzen und deren Darstellung und Ordnung nach den gefunden Merkmalen. Das Merkmal beschreibt hierbei eine unterscheidbare Eigenschaft von verschiedenen Beobachtungen eines Merkmalträgers in quantitativer oder qualitativer Weise. Als Merkmal kann man etwa Datentypen, Datenlänge oder anderen Eigenschaften wählen. Die Merkmale werden verdichtet, um wesentliche Eigenschaften darstellen zu können. Als Darstellungsform der Daten in der deskriptive Datenanalyse können Diagramme, Tabellen und Parameter gewählt werden - sie hat somit nur beschreibenden Charakter.

Kontextbasierte Datenanalyse betrachtet die linguistische Ebene, bei der der Kontext von Datensätze anhand von verbalen Beschreibungen zur Einteilung in Äquivalenzklassen als Klassifikationsmerkmal verwendet wird. Dazu werden mit verbalen Schlüsselwörtern die Äquivalenzklassen beschrieben. Die Äquivalenzklassen besitzen Merkmale und definieren deren Kontext. Die Kombination aus Merkmalen und verbalen Schlüsselwörtern beschreibt die Mengen der Äquivalenzklassen. Wäre ein Merkmal der Äquivalenzklasse „Geschwindigkeit“, wird das Merkmal durch die verbalen Schlüsselwörter „niedrig“ oder „hoch“ klassifiziert.

Explorative Datenanalyse oder kurz EDA ist ein Teilgebiet der Statistik, bei dem Zusammenhänge zwischen Datensätzen untersucht werden, von denen nur wenige oder keine Informationen über die Beziehungen vorhanden sind, sondern hauptsächlich Informationen über den Datensatz selbst. Mit der EDA sollen Strukturen oder Zusammenhänge zwischen den Daten entdeckt werden - durch die Analyse aller vorhanden Datensätze, wobei mögliche Verbindungen oder Strukturzusammenhänge vorher oder auf ersten Blick noch nicht absehbar sind. Ebenfalls kann mit der EDA eine Plausibilitätsüberprüfung der vorhandenen Datensätze durchgeführt werden, indem große und kleine Werte validiert werden, ob sie überhaupt möglich sind. Dieses Verfahren erlaubt auch die Definition von Hypothesen auf dem stochastischen Modell. Es ist jedoch bei einigen Verfahren erforderlich, zu testen, ob alle Datensätze den Anforderungen entsprechen. So benötigen einige Methoden eine Normalverteilung von Datenwerten oder gleiche Varianzen. Diese Verfahren kommen meist im Data Mining-Umfeld oder in der Marktforschung zum Einsatz.

2.3 Regelmaschine

Eine Regelmaschine (englisch: Rule Engine) wird dazu verwendet, Analyse- oder Geschäftsregeln zu verwalten, speichern und anzuwenden. Dazu kann die Regelmaschine als Framework oder Bibliothek in bestehende Applikationen wie Business Logic Server, Rule-Processing Server oder Datenhaltungssysteme integriert werden. Das bietet den Vorteil, Regeln zur Laufzeit zu verändern, auszutauschen oder zu löschen, ohne dass Programmierkenntnisse erforderlich sind oder eine Änderung am Quellcode erfolgen muss. Dies kann auch über eine grafische Oberfläche stattfinden. Die Regelmaschine besteht dabei aus der Speicherungsplattform der Regeln, der Verwaltungsoberfläche und der Processing Engine zur Anwendung der Regeln. Die Regeln werden dabei entweder in einem Interpreter interpretiert oder kompiliert und ausgeführt - ähnlich der Ausführung von Programmsprachen. Bei der Ausführung der Regeln werden nacheinander die Bedingungen überprüft. Dazu werden die festgelegten Datenfelder mit dem Vergleichswert per Bedingung geprüft. Trifft eine Bedingung zu,

Listing 2.1 Beschreibt eine Regel zur Rabatt-Berechnung in RuleML

```
<Implies>
      <then>
            <Atom>
                  <Rel>Rabatt</Rel>
                  <Var>Kunde</Var>
                  <Var>Artikel</Var>
                  <Data>25.0 Prozent</Data>
            </Atom>
      </then>
      <if>
            <And>
                  <Atom>
                        <Rel>Exklusiv</Rel>
                        <Var>Kunde</Var>
                  </Atom>
                  <Atom>
                        <Rel>Standard</Rel>
                        <Var>Artikel</Var>
                  </Atom>
            </And>
      </if>
</Implies>
```

wird die dafür deklarierte Aktion ausgeführt. [NRD06]
Die Regeldefinition spezifiziert das Layout, Bestandteile und Eigenschaften, welche für die Analyse oder Entscheidung verwendet werden. Die Deklaration kann per Wenn-Dann-Regeln erfolgen. Wenn ein bestimmter Wert oder Eigenschaften erreicht werden, wird die Dann-Eigenschaft ausgeführt. Eine andere Art zur Definition von Regeln sind Entscheidungstabellen, bei denen die Bedingungen und Aktionen als Regelvorschrift des Tupels (Bedingung und Aktion) in einer Tabelle vorliegen. Das Datenformat der Regeln ist je nach Hersteller unterschiedlich klassifiziert.
Es steht als standardisierte Regeldefinition die Rule Markup Language (RuleML) zur Verfügung. Sie folgt dem XML-Standard zur Beschreibung der Regeln und erlaubt die Validierung gegen XML-Schema-Definitionen. Die RuleML wurde komplett im Sprachumfang reduziert, um nur Regeln zu beschreiben. [BPS10] Die Regel in Listing 2.1 zeigt ein Beispiel als RuleML-Beschreibung. Sie besteht aus zwei Wenn-Dann-Regeln in if-then Tags im unteren Teil (Child) der Regel. Sprachlich lässt sich die Regel folgend ausdrücken: Rabatt für einen Kunden entspricht bei einem Kauf 25% , wenn der Kunde den Status „Exklusiv“ und das Produkt den Status „Standard“ hat.

Der „<Implies>“-Tag deklariert die Bedingung als Vergleich, also entspricht der angegebene Wert dem im Feld enthaltenen. Die zwei Regeln sind per booleschem Operator „AND“ verbunden. Dazu besteht der Wenn-Teil aus einem Tupel von „Rel“, das den vergleichenden Wert beschreibt, und im „Var“-Teil das zu vergleichende Feld. Zusätzlich befindet sich im oberen Bereich (erster Kindknoten) die Aktion, wenn die Bedingung zutrifft. In diesem Beispiel werden 25% als Produkt-Discount ausgegeben sowie der Produkt- und Kundenstatus, welcher im Aktionsbereich festgelegt wird.
In Vergleichen lassen sich diese Regeln auch als SQL-Statement deklarieren. In Listing 2.2 ist die Regel in Listing 2.1 per RuleML als SQL-Statement ausgedrückt, die beide dasselbe Ergebnis zur Folge haben. Somit folgt RuleML der Vorgehensweise von relationalen Datenbanken, welche zwar in der

Listing 2.2 Beschreibt die RuleML-Regel als SQL-Statement

```
SELECT Kunde, Artikel, 25.0 AS Rabatt
FROM Atom
WHERE Kunde = 'Exklusiv' AND Artikel = 'Standard'
```

Syntax anders definiert werden, jedoch dem selben Schema der Deklaration der Bedingungen folgt, wie in der „WHERE“-Bedingung des SQL-Statements beschrieben. Ebenso lässt sich über die Auswahl der Felder mit Hinzufügen eines weiteren Felds der Dann-Teil emulieren.

Jess ist eine Regelmaschine auf Java-Basis von Ernest Friedman-Hill von den Sandia National Laboratories in Livermore, USA, beschrieben in [Hil03], indem per Editor Regeln definiert werden und in der Regelmaschine ausgeführt werden. Früher wurde diese Art von Werkzeugen Expertensysteme genannt, heute Business Rule Management Systeme, die das Regeln eines Systems mittels der Anwendung von Regeln durch ein wissensbasiertes System durchführen.
Die Regeln werden dabei in Jess oder in Jess Markup Language (JessML) beschrieben. Jess lautet die ebenfalls in der Regelmaschine verwendete Beschreibungssprache der Regeln und ist aus der Lisp-Sprache abgeleitet. Die Beschreibungssprache ist nicht deklarativ und ist auf Grund der Lisp-Ähnlichkeit unterschiedlich strukturiert und enthält eine andere Symbolsyntax als das Java Framework von Jess. Sie bietet jedoch ebenfalls alle bekannten Programmiersprachen-Funktionen wie Variablen, Listen, Funktion, Verknüpfungs- und Vergleichoperatoren und Kontrollflusskonstrukte wie for- und while-Schleifen. Es ist ebenfalls erlaubt, aus der Jess-Syntax über ein Interface erzeugte Java-Methoden aufzurufen. Aus dieser Syntax lassen sich dann beliebige Regeln konstruieren. Außerdem steht JessML zur Verfügung, die auf Basis einer XML-Syntax eine Beschreibung der Regeln in deklarativer Form ermöglicht - ohne die Verwendung der Sprache Jess, die intern von der Regelmaschine nicht verstanden wird. Der interne JessML Parser konstruiert aus der JessML-Datei die äquivalente Jess-Beschreibung, um die Regeln in der Regelmaschine zu verwenden.

Eine JessML beschriebene Regel besteht, wie in Listing 2.3 zu sehen, aus dem „Name“-Element, das die Regel benennt, einem „lhs“-Element, welches die Regel beschreibt, und einem „rhs“-Element, welches die Aktion beschreibt, die ausgeführt werden soll. In dem in Listing 2.3 dargestellten Beispiel existiert im „rhs“-Element die Aktion zur Ausgabe eines Textes mit dem String „Fehler“ auf die Konsole per „printout“-Kommando, welches mit den Steuerungssymbolen „t“ für die Variable und „crlf“ für eine neue Zeile darstellt werden kann. Dazu werden String- und Symbol-Element in „Value“-Elemente gruppiert. Alle „Value“-Elemente in einem „Funcall“-Element ergeben eine Aktion. Das „lhs“-Element besteht optional aus einem „Group“-Element für die Verknüpfung mehrerer Regeln mit AND, OR und NOT, das im Element „Name“ innerhalb des „Group“-Elements beschrieben wird. Ebenfalls ist das Element „Pattern“ im „Group“-Element enthalten, das die eigentlichen Regeln enthält und beschreibt.

Durch die deklarative Programmierung per Regeldefinition wird beschrieben, was erreicht werden soll. Die Regelmaschine berechnet selbst, wann dies zu trifft und wann welche Regel ausgeführt wird. Dies geschieht, indem Fakten in den Regeln als Bedingungen beschrieben werden: Falls diese eintreffen, wird die Aktion ausgeführt. In Listing 2.4 ist eine einfache Regel abgebildet, die äquivalent zur Regel in Listing 2.3 ist, die prüft, ob ein Fehler existiert. Ist das der Fall, dann wird die Aktion in der zweiten Zeile ausgeführt, die eine Fehlerausgabe produziert. Dazu wird in der ersten Zeile die

Listing 2.3 JessML-Beschreibung einer Fehlerverarbeitung mit Ausgabe des Fehlers auf die Konsole.

```
<?xml version='1.0' encoding='US-ASCII'?>
<rulebase xmlns='http://www.jessrules.com/JessML/1.0'>
      <rule>
            <name>
                  MAIN::Fehlerregel
            </name>
            <lhs>
                  <group>
                        <name>
                              and
                        </name>
                        <pattern>
                              <name>
                                    error-is-present
                              </name>
                        </pattern>
                  </group>
            </lhs>
            <rhs>
                  <funcall>
                        <name>
                              printout
                        </name>
                              <value><type>SYMBOL</type>t</value>
                              <value><type>STRING</type>Fehler</value>
                              <value><type>SYMBOL</type>crlf</value>
                  </funcall>
            </rhs>
      </rule>
</rulebase>
```

Listing 2.4 Jess-Beschreibung einer Fehlerverarbeitung mit Ausgabe des Fehlers auf die Konsole.

```
(defrule Fehlerregel
      (error-is-present))
(printout t "Fehler" crlf))
```

Regel „Fehlerregel“ festgelegt und in Zeile 2 die Bedingung definiert. Wird die Bedingung "Wahr", wird die in der Zeile 3 vorhandene Aktion ausgeführt, die über die Printausgabe den Text „Fehler“ ausgibt.

Die Regelmaschine wird nur die Entscheidung der Ausführung der Regel treffen, wenn der Fakt der Bedingung gegeben ist. Das ist der Fall, wenn „error-is-present“ bei einem wirklichen Fehler auf „Wahr“ gesetzt wird. Dann tritt der Fakt der Bedingung in Kraft, die Regel und die Aktion wird ausgeführt. Regeln, die diesen Fakt nicht in einer Bedingung haben, werden in diesem Fall nicht ausgeführt, weil die Grundbedingung, also das Vorhandensein des Fakts in der Bedingung schon nicht gegeben ist.

2.4 Workflow Management System

Ein **Workflow** (WF) unterstützt einen **Geschäftsprozess** und ist die Folge einzelner Arbeitspakete des Geschäftsprozesses, die in einer logischen Folge stehen. Der Prozess wird von einem Ereignis, das einen Geschäftsprozess auslöst, gestartet und folgt seiner logischen Verknüpfung in immer gleichen oder ähnlichen Schritten, wobei die Aktivitäten abgearbeitet werden. Die einzelnen Arbeitspakete werden von ein- und ausgehenden Datenoperationen beeinflusst. Am Ende des Prozesses steht das Ergebnis des Ablaufs fest, welches auch eine Aktivität erzeugen kann und/oder einen weiteren Prozess startet. Der Geschäftsprozess soll das Unternehmensziel unterstützen oder erreichen und der Wertschöpfungskette beitragen, was wiederum von einem WF unterstützt werden kann. [Lam13]

WFs lassen sich, wie in [Mü06] beschrieben, in unterschiedlichen Varianten modellieren. (i) Die erste Variante wäre die Modellierung per Petrinetze, die von Carl Adam Petri entwickelt wurden - zur Darstellung der Abläufe von Prozessen. Die Darstellung erlaubt die Beschreibung des Daten- und Kontrollflusses. Damit lassen sich Ereignisse oder Aktivitäten sowie die verwendeten Bedienungen für den Übergang beschreiben. (ii) Alternativ lassen sich WFs als ereignisgesteuerte Prozessketten (EPK) beschreiben. Die EPK ist eine Abwandlung der Petrinetze, welche die theoretische Basis darstellten. EPK ist eine grafische Modellierungssprache, in der Aktionen/Funktionen in einer Kette eingefügt und durch gerichtete Verknüpfungen verbunden werden. Eine Funktion erzeugt beim Ablauf ein Ereignis. Ereignisse einer Funktion stellen für die folgende Funktion der Prozesskette das Startereignis dar. Bei der Modellierung werden der Kontrollfluss der Prozesskette definiert und die jeweiligen Funktionen beschrieben. Ebenfalls lassen sich durch logische Operatoren alternative oder parallele Wege sowie Schleifen und die Zusammenfügung mehrere Wege darstellen. (iii) Die dritte Darstellungsform von WFs ist die Unified Modeling Language (UML), entwickelt von der Object Management Group (OMG). UML wird hauptsächlich für die Modellierung von objektorientierten Anwendungen verwendet, um Klassendiagramme, Anwendungsfälle, Zustandsdiagramme, Aktivitäten und weitere Modelle zu erstellen. UML wird in der International Organization for Standardization (ISO) standardisiert. [OMG05] Ein in UML erstellter WF besteht dabei aus einem Startknoten, einem Aktivitätsdiagramm als gerichteten Graph mit Prozessaktivitäten, die verbunden sind mit Verknüpfungen, welche die Bedingung für den Übergang spezifizieren, und den Endknoten, die das Ende des WFs aufzeigen.

WFs werden in einem Workflow Management System (WfMS) modelliert, verwaltet, getestet und ausgeführt. Dazu folgt es den in WF beschrieben Prozessschritten nach dem Daten- und Kontrollfluss, kommuniziert mit externen Systemen und prüft beim Übergang in den nächsten Schritt die erforderlichen Bedingungen. Erzeugte Ausgaben und Aktionen werden ebenfalls initiiert. Bei Eintritt des Startereignisses erzeugt das WfMS aus einem modellierten und aktivierten WF eine laufende Instanz und folgt den im Modell deklarierten Arbeitsschritten. Am Ende wird die laufende Instanz wieder vernichtet und bei neu wiederholtem Ereignis neu erzeugt.

Das WfMS wird in der Regel als Informationstechnik (IT)-Werkzeug mit grafischer Oberfläche zur Verfügung gestellt und verwendet 5 Komponenten. [Mü06] Es besteht aus den folgenden Komponenten:

Modellierungskomponente entwickelt auf grafische Art einen WF, indem Aktivitäten, Kommunikation und die Übergänge per Transitionen beschrieben werden. Dazu wird von der Model-

lierungskomponente sichergestellt, dass der entwickelte WF von der Steuerungskomponente (Workflow Engine) ausgeführt und interpretiert werden kann. Sie unterstützt ebenfalls den Designer des Workflows mit der Modellierung und Abbildung des Geschäftsprozesses.

Steuerungskomponente , auch Workflow Engine (WE) genannt, ist der Hauptbestandteil des WfMS. Die Ausführung der modellierten WFs erfolgt durch die WE. Dazu führt die WE alle in den WF beschriebenen Funktionen/Aktivitäten aus und interpretiert die darin enthaltene Logik. Wird ein WF gestartet, wird von dem modulierten WF eine Instanz erzeugt. Startet der WF mehrmals, werden immer unabhängige Instanzen des Flows erzeugt.

Überwachungskomponente wird verwendet, um die auf der WE laufenden WFs zu überwachen und zu protokollieren. Dazu können die Daten in instanzbezogener Weise gesammelt werden, die pro laufender Instanz anfallen Mit diesen Daten lassen sich die unterschiedlichen Instanzen vergleichen. Alternativ lassen sich in aggregierter Weise Daten sammeln, also über alle erzeugten Instanzen eines WF. Damit lässt sich die Effektivität des Prozesses prüfen.

Schnittstellenkomponente ist ein sehr wichtiger Teil des WfMS. Die Komponente vermittelt zwischen den Schnittstellen. Sie verarbeitet alle eingehenden und ausgehenden Daten, die zwischen einem WF und anderen externen Komponenten oder Systemen erzeugt werden. Dazu transformiert sie über die entsprechenden Adapter die Daten in ein interpretierbares Format der zu empfangenden Schnittstelle.

Simulationskomponente wird verwendet, um modulierte WFs zu testen und mögliche Schwachstellen oder Design-Fehler zu entdecken.

2.5 Data Mining

Data Mining wendet statistische Verfahren aus der explorativen Datenanalyse zur Datenanalyse an, die große Mengen an Daten verarbeitet, ohne vorher ein Ziel dafür definiert zu haben bzw. genau die Verbindung zu deklarieren, die man sucht. Es wird somit verwendet, um neue Verbindungen, Probleme oder Charakterisierungen aus einer Menge an Daten zu erhalten. Bei einer klassischen Datenanalyse (Analyse ohne Verwendung von Data Mining-Verfahren) ist vorher klar definiert, was analysiert wird und was die möglichen Ergebnisarten sein können. Das ist bei Data Mining nicht der Fall, es extrahiert Erkenntnisse oder Muster aus einer Datenmenge, die bis jetzt unbekannt war oder vorher nicht definiert wurde. Je mehr Datensätze für die Analyse zur Verfügung stehen, umso mehr Beziehungen zwischen den Unterschieden oder Gemeinsamkeiten der Datensätze können gezogen werden. Dies kann zu ganz neuen Schlüssen führen, die bei der Betrachtung einzelner oder weniger Datensätze nicht gefunden werden könnten.
Bekanntes Einsatzszenario von Data Mining ist die Erkennung von Betrug im Finanzsektor. Dazu werden alle älteren abgeschlossenen Transaktionen mit aktiven verglichen, um daraus abzuleiten, ob es sich bei der Transaktion um einen Betrug handelt. Dieser Prozess kommt ebenfalls per Data Mining in der Versicherungsbranche oder der Verkaufssparte vor, wo bestehende Verträge und Verkäufe analysiert werden, inwiefern der Neukunde ein längeres Vertragsverhältnis führen möchte oder eine schnelle Kündigung in Betracht zieht. Diese Art von Datenanalyse lässt sich über normale Analysen wie SQL-Abfragen nicht lösen, die nur einzelne Datensätze betrachten. [Agg15]

2.5.1 Prozessschritte

Der Data Mining-Prozess besteht nach [Agg15] aus einer Reihe unterschiedlicher Arbeitsschritte, welche die typische Arbeitsweise eines Data Mining-Ablaufs beschreiben:

Datenerfassung ist erforderlich zur Sammlung von Datensätzen für eine persistente Speicherung in einer Datenbank für die spätere Datenanalyse. Dies kann erfordern, dass aus externen Quellen Daten wie Sensornetzwerke oder Security-Logs erfasst werden. Dieser Schritt ist für ein gut funktionierendes Data Mining besonders wichtig. Je mehr und genauer die Datenerfassung ist und umso mehr Unterschiede der Datensatze in diesem Schritt vorhanden sind, desto besser kann die spätere Analyse Ergebnisse liefern. Aus nur wenigen Datensätze mit gleichen Datenwerte lassen sich weniger Rückschlüsse erfassen.

Datenauswahl und Aufbereitung werden auf die Datensätze angewendet. Ist ein größerer Datenbestand vorhanden, müssen die Datensätze vor der Analyse ausgewählt und bereinigt werden. In diesem Schritt ist es wichtig, bereits den Data Mining-Algorithmus zu wählen, der zum Einsatz kommen soll. Denn er hat Einfluss auf das zu verwendende Datenmodell und könnte eine Transformation der Daten erfordern.
Stammen die Daten aus unterschiedlichen Quellen, ist oft kein einheitliches Datenmodell gegeben. Dazu liegen oft die in der Datenbank gespeicherten Informationen nicht in der korrekten Form vor oder beinhalten Datenwerte, bei denen eine Verwendung nicht gegeben ist. Beinhalten die Datensätze etwa größere Mengen an Freitext, welcher nicht kontextbasierend analysiert wird, lässt sich diese Datenfeld entfernen. Oft wird auch eine Anpassung von einem Datumsfeld verlangt, welches statt dem Datum die Anzahl an Tagen eines Bezugspunkts enthält oder das Auflösen multidimensionaler Arrays. Die Art des Data Mining-Algorithmus bestimmt die Art der Analyse. Abhängig vom Data Mining-Algorithmus kann es erforderlich sein, dass Datensätze angepasst oder transformiert werden müssen. Die Analyse erfolgt nur auf den aufbereiteten und ausgewählten Daten. Zur Erhöhung der Performance wird die Analyse meist parallel ausgeführt.

Datenanalyse erfolgt erst in diesem Schritt des Data Mining-Prozesses. Alle vorherigen Schritte dienten nur der Aufbereitung der Daten für die Analysealgorithmen. In diesem Schritt werden die aufbereiteten Daten eingelesen und der vorher gewählte Algorithmus zur Analyse ausgeführt. Ein oder mehrere Durchläufe können erforderlich sein, einige Algorithmen müssen erst eine Klassifikation oder Einordnung vornehmen - verlangen somit eine Aufbauphase. Dazu durchläuft der Algorithmus zuerst alle Datensätze ohne den zu analysierenden Satz. Nach Fertigstellung wird der Algorithmus angewendet, um einen bestimmten Datensatz zu analysieren.

Ergebnisinterpretation ist der letzte Schritt im Prozess. Nach Durchlaufen der Analyse müssen die gefundenen Rückschlüsse interpretiert werden. Je nach Auswahl des Algorithmus können dies genaue Einzelwerte sein, etwa die Einordnung in eine Gruppe.

2.5.2 Verfahren

Data Mining verwendet unterschiedliche Algorithmen für die Lösung einzelner Probleme. Grob gegliedert sind sie in Betrachtungsprobleme zur Erkennung von gemeinsamen Eigenschaften der Datensätze - dazu wird die Clusteranalyse, Ausreißererkennung oder Assoziationsanalyse eingesetzt. Die weitere Kategorie der Data Mining-Analyse ist die Abgabe von Prognosen auf Basis vorher bekannter Datensätze, die mit Regressionsanalyse oder der Klassifikation gelöst wird.

Betrachtungsprobleme

Bei Betrachtungsproblemen werden alle Datensätze auf einmal betrachtet, wie in [Agg15] beschrieben, ohne dass im Vorlauf Wissen durch Übungsdaten extrahiert und daraus Schlüsse oder Folgerungen gezogen werden, die an neue Datensätze angewendet werden. Es soll aus bestehendem Wissen durch das Betrachten der Eigenschaften von Objekten untereinander vorher unbekannte Schlussfolgerungen erzielen. Folgende Data Mining-Analysen beschrieben in [Agg15] sind Betrachtungsprobleme:

Clusteranalyse wird verwendet, um Gemeinsamkeiten und Ähnlichkeiten zwischen unterschiedlichen Datensätzen zu erkennen, die sich einer Gruppe zuweisen lassen. Das Ziel ist, die Datensätze einzelnen neuen Gruppen zuzuordnen. Die Gruppenordnung ist dabei vorher nicht bekannt und wird erst bei der Analyse ermittelt und den einzelnen Datensätze zugeordnet. Somit ist es ein uninformiertes Verfahren, welches kein Gruppenwissen voraussetzt. Ein Cluster wird als eine Gruppe von Datensätze bezeichnet, die sich bei einzelnen Datenwerte ähneln und somit einer Gruppe zuweisen lassen. Bei der Clusteranalyse können unterschiedliche Verfahren eingesetzt werden - je nachdem, welches Problem gelöst werden soll.
Bei dem hierarchischen Clusterverfahren werden die Objekte mit geringer Distanz zueinander einer Gruppe zugeordnet, es wird also die Distanz zwischen den Objekten im Raum betrachtet. Dazu werden die Objekte mit geringer Varianz, also Werte im ähnlichen Wertebereich, gruppiert. Die Distanz stellt den Unterschied zwischen den Datenwerten dar, also ähneln sich die Werte der Objekte in einem Cluster sehr. Mit den Clustern wird eine Hierarchie gebildet. Das bedeutet: Der oberste Cluster enthält alle Objekte, die untersten Cluster enthalten nur ein einziges Objekt. Der Wertebereich wird damit von oben nach unten immer weiter in einzelne Bereiche, also Cluster, aufgeteilt und verfeinert.
Das partitionierende Clusterverfahren gibt eine feste Anzahl an Clustern vor, die den Wertraum teilen und das Clusterzentrum festlegen. Eine Fehlerfunktion wird definiert für die Verschiebung des Zentrums, welche das zu lösende Problem deklariert. Das Zentrum wird nun iterativ im Laufe der Analyse so lange verschoben, bis die Fehlerfunktion einen minimalen Wert erhält. Während des Verschiebens können die Objekte im Raum das Cluster wechseln.
Das dichtebasierte Verfahren wird verwendet, um Objekte in einem k-dimensionalen Raum zu analysieren, die nahe beieinander liegen und eine Verdichtung von Objekten in einem begrenzten Raum repräsentieren. Ein Cluster enthält Objekte, die nahe beieinander liegen und von einem Raum ohne hohe Objektdichte umschlossen werden.
Das kombinierte Verfahren wird oft nach der Bestimmung der Anzahl von Cluster angewendet, um das Resultat der Analyse zu verbessern. Die Verbesserung kann dabei erreicht werden, indem nach der Zuordnung in Cluster die Objekte im Cluster nochmals separat betrachtet

werden, um zusätzliche Informationen zu erhalten. Die Clusteranalyse liefert somit immer eine Anzahl an Gruppen zurück, die Objekte selben oder ähnlichen Typs enthalten.

Ausreißererkennung analysiert Datenwerte der Objekte, inwieweit sie in größerem Umfang vom Normalwert der restlichen Datensätze abweichen, zu vorherigen Werten inkonsistent sind oder Abnormitäten aufweisen. Gefundene Datensätze können anschließend verworfen werden. Alternativ können Datensätze mit Abnormität, die die Analysen verfälschen können, weiter verifiziert werden, um den Grund der Abnormität zu ermitteln. Diese Datensätze können dann wertvolle Informationen über die Gründe der Abweichungen liefern. Als Ergebnis werden einzelne Objekte zurückgeliefert, die größere Datenabweichungen enthalten.

Assoziationsanalyse erzeugt Schlussfolgerungen aus den Datensätzen durch die Häufung von Zusammenhängen zwischen Werten oder das Bestehen von festen Regeln. Damit lässt sich eine Korrelation zwischen unterschiedlichen Attributen ermitteln. Das bedeutet, dass aus der Häufung von Zusammenhängen, die in einer Menge von Datensätzen vorhanden sind, neue Schlussfolgerungen gezogen werden können. Daraus lassen sich dann Schlussfolgerungen ableiten, wie dass Elemente η und θ aus der Menge X die Elemente α und β aus der Menge Y zur Folge haben. Ein Anwendungsgebiet wäre die Warenkorbanalyse, welche ein abstraktes Modell der Daten als Ergebnis liefert.

Prognosen

Eine Prognose ist die Vorschau der Zustände oder Ereignisse in der Zukunft. Bei Data Mining-Verfahren zur Erstellung einer Prognose werden zuerst die gesamten Datensätze verwendet, um eine Struktur zu ermitteln, die dann genutzt wird, um eine Prognose über einen neuen Datensatz zu erstellen. Das kann eine Einordnung in eine Kategorie, mögliche Werte oder andere Aussagen sein. Die in [Agg15] beschriebenen Data Mining-Analysen sind Prognosen:

Regressionsanalyse hat das Ziel, Verbindungen zwischen abhängigen und unabhängigen Datenwerten zu finden, indem statistische Verfahren angewendet werden. Somit lässt sich eine Prognose eines Datenwertes anhand eines Vorhersagemodells durch funktionelle Zusammenhänge erstellen, das die Regressionsanalyse vorher durch die vorhandenen Datensätze erzeugt hat. Sie ermöglicht auch die Abschätzung fehlender Werte eines Datensatzes. Dies erlaubt es, die redundanten Informationen zu entfernen und die Komplexität zu reduzieren, was ein abstraktes Modell der Daten liefert.

Klassifikation ordnet Objekte zur Mustererkennung in Klassen ein. Jede Klasse erhält eine Klassenmarkierung. Sie behandelt ähnliche Probleme wie die Clusteranalyse, die auch eine Einordnung in Gruppen vornimmt, wobei die Klassifikationsanalyse auch neuere Objekte ohne Markierung in die Gruppen einteilen kann. Das Einordnen eines Datensatzes in eine Klasse nennt man Klassifizierung. Die Klassifikation wird ermöglicht, indem eine Menge an Datensätzen verwendet wird, um ein Klassifikationsmodell aufzubauen. Bei diesen Datensätzen ist bereits eine Einordnung und somit eine Markierung vorhanden, die als Übungsdaten bezeichnet sind. Die Einordnung ist abhängig von den Datenattributen der verwendeten Objekte aus den Übungsdaten, die ableiten, wie ein Datensatz gestaltet sein muss, um eine bestimmte Klasseneinteilung zu erhalten.

Der Aufbau des Modells kann zum Beispiel als Baumstruktur erfolgen und wird Entscheidungsbaum genannt, es sind jedoch auch andere Methoden vorhanden. Dazu wird der erste Datensatz verwendet, das erste Attribut im Objekt ausgelesen und als Knoten im Baum angelegt. Beim ersten Attribut ist dies die Wurzel. Das zweite Attribut wird ausgelesen und an das vorherige angehängt. Das wird so lange durchgeführt, bis alle Attribute abgearbeitet wurden und nur noch die Klassifikationsmarkierung vorhanden ist. Dieses Label wird als Blatt an den Baum angehängt. Der nächste Datensatz folgt nun derselben Strategie, nur dass bei jedem Knoten geprüft wird, ob dieser Werte oder die Kategorie bereits vorhanden sind. Sollte das der Fall sein, wird das nächste Attribut geprüft. Ist eine von beiden vorhanden, so wird vom Knoten aus ein neuer Knoten angelegt. Zum Schluss wird an den letzten Knoten, an dem sich der Datensatz befindet, die Markierung hinzugefügt. Ist sie bereits vorhanden, hat der davor erstellte Datensatz die gleiche Klassifikation und ebenso die gleiche Markierung.
Das Modell bestimmt die Klassifikation, welche bei neuen Datensätzen angewendet wird, die noch keine Klassifizierung haben. Dazu folgt sie dem Aufbau des Modells und landet nach dem Durchlaufen des Baums anhand der Attribute bei einem Blatt, das die Markierung darstellt. Diese Markierung repräsentiert dann die Klassifikation des Datensatzes.

3 Entwurf

In diesem Kapitel wird die Architektur des Analysesystems beschrieben, die verwendete Regelsyntax zur Beschreibung der Analyseregeln und der Aufbau der Analysatorvorlage für die Beschreibung einer Datenanalyse. Abschließend wird die Erweiterbarkeit der Architektur betrachtet, der Zugriffsschutz und die Data Mining-Analyse.

3.1 Architektur

Das Analysesystem besteht, wie in Abbildung 3.1 dargestellt, aus dem Analyse Management System. Das Analyse Management System verwaltet das Hinzufügen, Aktualisieren und Löschen von Analysen. Zur Beschreibung der Analysen wird eine Analysatorvorlage verwendet. Das Analyse Management System verwendet die Analysatorvorlage zur Interpretation der Konfiguration einer Analyse. Die Analysatorvorlage beschreibt in maschinenlesbarer Form die Eigenschaft der Analyse. Der Anwender versendet die Analysatorvorlage an das Analyse Management System zur Erzeugung einer Analyse. Die Analysatorvorlage wird in einer Regelsyntax beschrieben. Die Regelsyntax wird verwendet, um die in der Analyse enthaltenen Regeln zu beschreiben. Sie ist einfach gestaltet. Die Analysatorvorlage des Anwenders für eine Analyse besteht dabei aus Regeln, welche die Analyseeigenschaften beschreiben, das Event, welches die Analyse startet, und die Aktion, die bei Zutreffen der Analyse ausgelöst werden soll und anwenderspezifisch konfiguriert werden kann.

Im Analyse Management System sind ein oder mehrere Ausführungssysteme enthalten, die Ablaufpläne abarbeiten können. Ein Ablaufplan ist ein ausführbares Programm des Ausführungssystems. Die zweite Komponente des Analyse Management System ist der Parser zur Interpretation der Analysatorvorlage in einen Ablaufplan. Der Parser selbst wird als Ablaufplan im Ausführungssystem abgearbeitet. Der vom Parser erzeugte Ablaufplan einer Analysatorvorlage nennt sich Analysator und repräsentiert das Programm einer Datenanalyse. Der Analysator ist ein im Ablaufplan beschriebenes Programm, das Daten analysiert und vom Ausführungssysteme ausgeführt wird. Die dritte Komponente sind Adapter (Datenadapter, Aktionsadapter und Eventadapter) zur Kommunikation mit externen Systemen für Daten, Aktionen und Events. Die Adapter transformieren dabei die vom externen System kommenden oder zum externen System gesendeten Daten in eine interpretierbare Form des jeweils einzulesenden Systems. Der Datenadapter transformiert eingehende Daten in eine für den Analysator interpretierbare Form. Ebenso kann der Datenadapter das Datenmodell der Daten ermitteln. Der Aktionsadapter transformiert die auszuführenden Aktionen in ein für das aktionsausführende System lesbare Form. Der Eventadapter konvertiert eingehende Events in das für das Ausführungssystem interpretierbare Event. Zur weiteren Speicherung der Analysatoren - unabhängig vom Analysator - kann der Analysator in einem externen Datenhaltungssystem gespeichert werden.

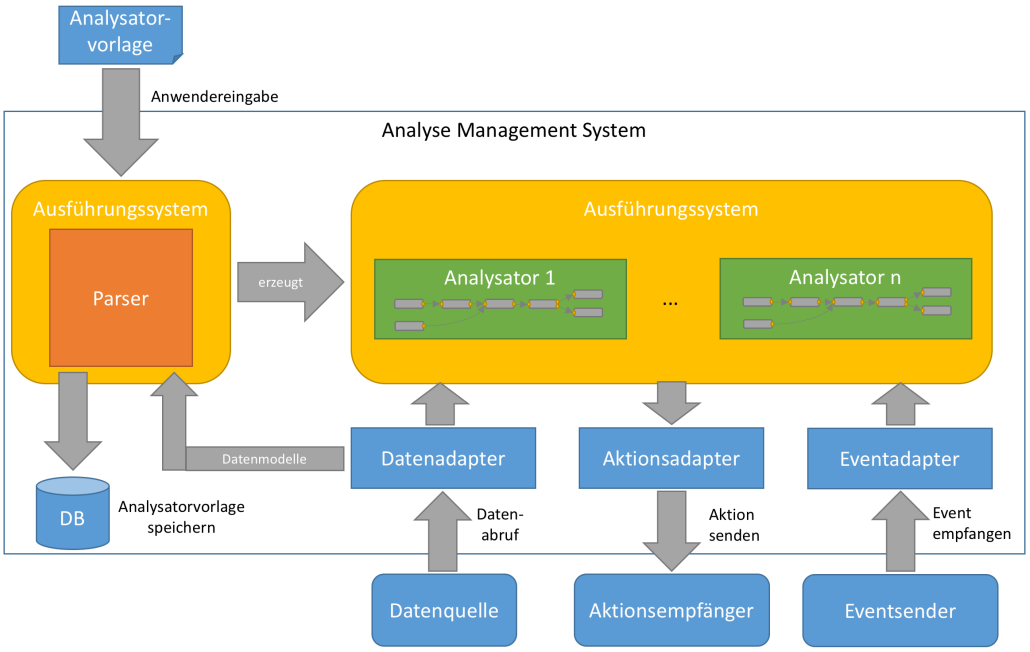

Abbildung 3.1: Darstellung der Analysearchitektur.

Das Analyse Management System soll - in eine bestehende Plattform für Gesundheitsdaten integriert - die bestehende Lösung für Analysen komplett ersetzen und erweitern. Diese Analysearchitektur wird dazu als Bestandteil in die ECHO-Plattform integriert. Die Abbildung 3.2 zeigt die ECHO-Plattform als Architekturmodell mit neuem Analysesystem. Die ECHO-Plattform besteht dabei aus der Anwendungsebene und dem Gesundheitsdatenmanagement. Über die Anwendungsebene kann der Arzt oder Patient auf die Plattform zugreifen. Dazu kann der Patient Anfragen an die Plattform stellen oder er erhält Benachrichtigungen. Die Mediziner können ebenfalls Anfragen stellen, um Daten abzurufen oder zu erstellen. Alle Anfragen werden an das Gesundheitsdatenmanagement weitergeleitet. Das Gesundheitsdatenmanagement stellt für Anfragen eine Health Application Programming Interface (API) mit REST bereit, besitzt das Modul Orchestrierung zur Steuerung und bietet Gesundheitsdienste für die Benutzer und interne Module an zum Zugriff auf die Gesundheitsdaten. Die Gesundheitsdaten werden im Gesundheitsdatenmanagement in einer Datenbank gespeichert. Das Analysemodul führt alle Datenanalysen der ECHO-Plattform durch und greift dazu auf die Gesundheitsdienste zu sowie auf die Datenbank. Das Modul Management und Provisionierung stellt die Umgebung der Plattform bereit. Die komplette Umgebung wird auf einer Serverumgebung betrieben.

Für die Integration wird das komplett rot eingefärbte Modul (Analysen) durch das neue Analysesystem ersetzt, das bereits in der Abbildung geändert wurde. Die rot umkreisten und gepunkteten Module erhalten Änderungen zur Integration von event-basierender Steuerung der Analysen und Anpassung im Datenbanklayout der Anwendung zur persistenten Speicherung von Analysen und Zugriff auf die Gesundheitsdaten für interne Analysen im Data Mining-Bereich. Für den Datenzugriff verwendet die Analysearchitektur die bereits vorhandene REST API, die für andere Anwendungen bereits existiert. Zur Authentifizierung mit der ECHO-Plattform werden ebenfalls vorhandene Lösun-

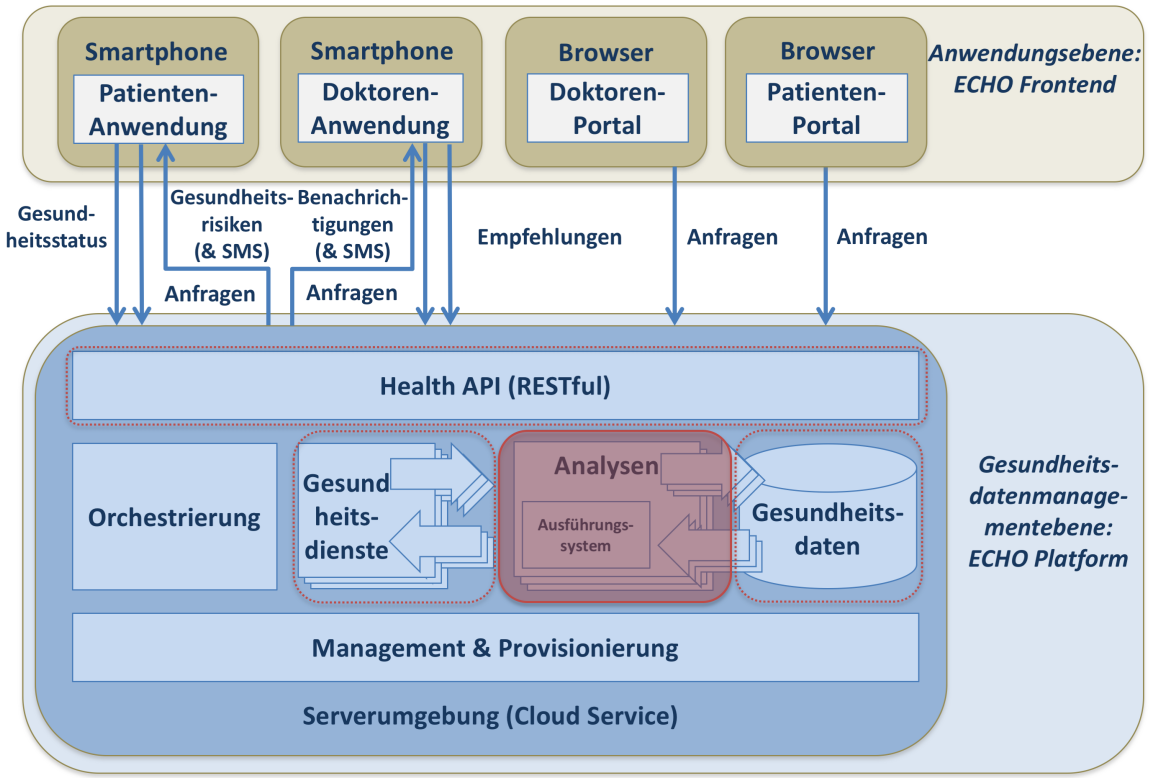

Abbildung 3.2: Darstellung der Änderungen und Integration in die bestehende ECHO-Plattform. [BKK+14]

gen verwendet. Der Data Mining-Zugriff erfolgt direkt über die Datenbank. Die Health API wurde um die REST-Schnittstellen zur Interaktion mit dem Analysesystem erweitert.

Im Zuge der Arbeit wurden die Regelsyntax für die Analysen selbst, die Analysatorvorlage für die Beschreibung von Analyseregeln, Aktion und Events, der Parser für die Ablaufflussübersetzung im Ausführungssystem, das Datenbankmodell für die persistente Speicherung der Analysen und die Adapter für die Transformation der Daten, den Empfang für Events und die Interaktion mit externen Systemen für die Auslösung von Aktionen entworfen und integriert. Ebenfalls wurden für die Integration in die ECHO-Plattform Änderungen im vorhandenen System vorgenommen.

3.2 Analysatorvorlage

Die Analysatorvorlage wird im Analyse Management System verwendet, um eine Analyse zu beschreiben, die später auf dem Ausführungssystem als Ablaufplan erstellt werden soll. Eine Analysatorvorlage erzeugt einen Analysator als Ablaufplan, wenn die Analysatorvorlage an das Analyse Management System gesendet wird. Damit die Analysatorvorlage maschinell gelesen werden kann, wird sie in

einem Container verpackt, der über eine Webschnittstelle oder einen anderen im Adapter verfügbaren Modus übertragen und später vom Parser ausgewertet werden kann. Dazu wird JavaScript Object Notation (JSON) verwendet. Zum Container-Format mehr im Implementierungskapitel in Kapitel 4. Das Container-Format ist eine Kapsel, um die Analysatorvorlage vom Anwendersystem zum Analyse Management System zu übertragen. Zur Erstellung der Analysatorvorlage können von der ECHO-Plattform die bereitgestellten Operatoren, Datenfelder, Events und Aktion abgerufen werden, was eine automatische und vereinfachte Erzeugung von Analysatorvorlagen ermöglicht, ohne dass die Anwender alle Optionen kennen müssen oder die Struktur selbst. Eine grafische Erzeugung der Vorlage ist damit möglich.

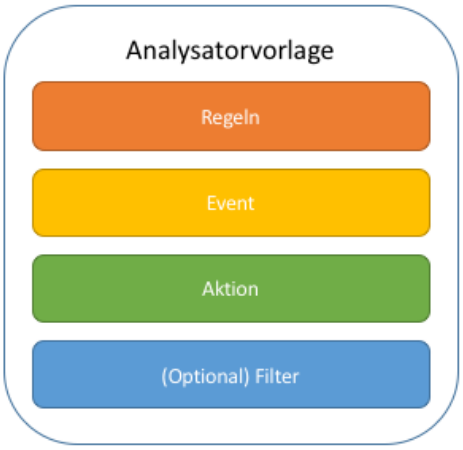

Abbildung 3.3: Beschreibung der Struktur der Analysatorvorlage.

Die Analysatorvorlage, die in Abbildung 3.3 dargestellt ist, besteht aus den Bestandteilen Regeln, Event, Aktion und optional einem Filter. Die **Regeln** werden in der Regelsyntax deklariert und beschreiben die Eigenschaft der Analyse, die abgebildet werden soll. Die Regelsyntax beschreibt dabei die Ausführungsweise der Analyse, unabhängig von Event oder Aktion und wird nur in dem Feld Regeln der Analysatorvorlage verwendet. Die Regeln werden in einer Liste abgelegt und können eine beliebige Anzahl enthalten. Mindestens ein Element (Regel) muss vorhanden sein, damit die Analysatorvorlage die Validierung im Analyse Management System bestehen kann.
Das **Event** beschreibt den Typ des Ereignisses, das die Analyse auslöst. Die Events werden vom Analyse Management System nicht fest vorgegeben, benötigen aber eine eindeutige Kennzeichnung, damit nicht zwei Events mit unterschiedlicher Interpretation dieselbe Bezeichnung verwenden. Das Event wird in der Analysatorvorlage als Eventbezeichner beschrieben und kann zusätzliche optionale Parameter enthalten, die das Event anwenderspezifisch parametrisieren. Der Eventbezeichner besteht aus einer beliebig langen Folge von Zeichen - mit mindestens einem Zeichen. Somit besteht das Event aus einer eindeutigen Bezeichnung (Eventbezeichner) als String (Zeichenfolge) und dem Parameterfeld als Liste mit Elementen, die per Index erreichbar sind. Das Event wird von einem externen System initiiert und informiert damit das Analyse Management System über den Eventtyp, welcher die Analysatoren auslöst, die mit der eindeutigen Eventbezeichnung registriert sind. Wird in einem Analysatorvorlage das Event „NeueNachricht“ registriert und das Analyse Management System erzeugt einen Analysator, wird der Eventbezeichner „NeueNachricht“ hinterlegt. Eine Initiierung des Analysator erfolgt dann ausschließlich durch den Eventbezeichner „NeueNachricht“.

Außerdem ist in der Analysatorvorlage die **Aktion** enthalten, die ausgeführt werden soll, wenn die Analyse die Regeln erfüllt. Sie wird ebenfalls eindeutig über einen Bezeichner gewählt und ist vom Analyse Management System fest vorgegeben. Das ist erforderlich, weil im Gegensatz zum Event die Aktion Nachrichten an externe System schicken kann, die ein bestimmtes Format oder Parameter enthalten müssen. Dies muss im Analyse Management System festgelegt werden, wie mit der Schnittstelle des externen Systems zu sprechen ist. Das Feld für die Bezeichnung der Aktion (Aktionsbezeichner) enthält die eindeutige Zeichenfolge als String der Aktion und optional weitere Parameter zum Konfigurieren der auszuführenden Aktion. Der Bezeichner darf - wie beim Event - nicht doppelt verwendet werden, um die Aktion auswählen zu können, und muss mindestens ein Zeichen enthalten. Die Parameter erfolgen als beliebig lange Liste. Als Aktion kann ein anwenderspezifischer Text als Zeichenfolge dienen, der bei der Ausführung der Aktion versendet werden soll. Dieser Text kann Tokens in der Form „{Token-ID}“ enthalten, die vom Analyse Management System und dem ausgewählten Analyseverfahren vorgeben sind, die bei der Aktion mit Variablenwerten ersetzt werden, um dynamische Werte mit zu versenden. Alternativ können externe Systeme über Schnittstellen aufgerufen werden, um beliebige Aktionen auszuführen. In Abbildung 3.4 ist eine Aktion dargestellt, die in der Analysatorvorlage angelegt werden kann. Als Aktionsbezeichner wird die eindeutige Zeichenfolge „NachrichtSenden“ verwendet, die vom Analyse Management System ausgewählt werden soll, was den Versand einer Benachrichtigung auslöst. Der Benachrichtigungstext wird aus dem „Message“-Feld extrahiert. In Parameter wird das „Message“-Feld deklariert mit der zu sendenden Nachricht, im Beispiel „Test {id}“. Der Token „id“ wird dann zur Laufzeit mit dem in der Datenanalyse assoziierten Feld verknüpft. Die Verknüpfung und Deklarierung erfolgt ebenfalls im Analyse Management System.

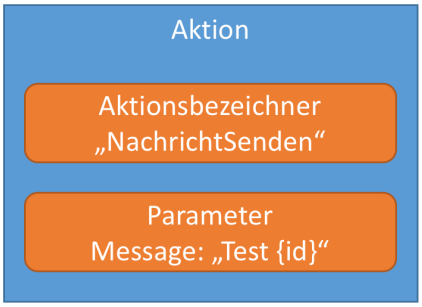

Abbildung 3.4: Beschreibung einer Aktion in der Analysatorvorlage.

Optional kann in der Analyse ein **Filter** gesetzt werden, der Datensätze filtert, die nicht einem bestimmten Muster entsprechen, um die Datensätze schon vor der Analyse zu reduzieren. Dazu kann ein Datenfeld, ein Vergleichsoperator und ein Vergleichswert angegeben werden, mit dem die Datensätze geprüft werden. Es werden nur Datensätze in die Analyse verschoben, die diesem festgelegten Muster entsprechen. Die Abbildung 3.5 zeigt die Funktionsweise des Filters. Dabei werden einzeln die Datensätze vom Filter eingelesen, mit dem Vergleichsoperator (festgelegt in Filterbeschreibung in Analysatorvorlage), dem Vergleichswert und dem Wert im Datenfeld verglichen. Das Datenfeld ist dabei der Index bzw. die Referenz eines Feldes im Datensatz zur Adressierung. Im Beispiel (Abbildung 3.5) wird das Datenfeld „wert1“ mit dem Vergleichsoperator „==“ und dem Vergleichswert „2“ verglichen. Ist im Feld „wert1“ der Wert gleich dem Wert „2“, ist der Filter erfüllt

und leitet den Datensatz an die Analyse weiter. Ist er nicht erfüllt, wird der Datensatz verworfen und nicht zu Analyse weitergeleitet.

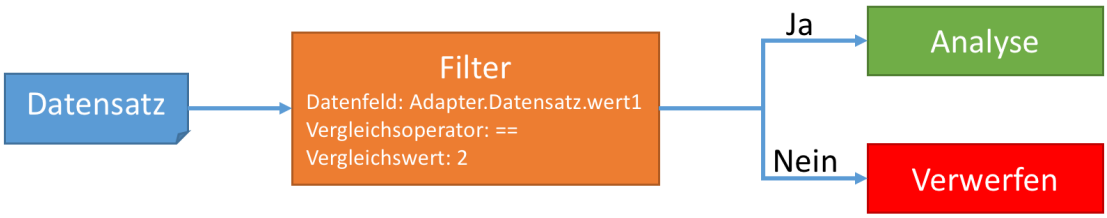

Abbildung 3.5: Filterung von Datensätzen mit Vergleichsoperator und Referenzwert.

Ein bestehendes Datenformat wie RuleML oder JessML wurde nicht verwendet, da alle anderen Konzepte einen erheblich größeren Umfang abdecken müssen und somit eine Vielzahl von Optionen und Parametern erfordern, die auch in einem verschachtelten Konzept abgebildet werden, was die Komplexität der Vorlage erhöhen. Aus diesem Grund wurde ein eigenes Format für die Analyseregeln entwickelt, das nur den reduzierten Umfang abdeckt, sich jedoch einfach konfigurieren lässt und Anwendern ohne IT-Kenntnissen die Möglichkeit gibt, Regeln ohne komplizierte Formate zu beschreiben. Das Analysatorformat benötigt für die Beschreibung der einfachsten Regel (den Vergleich eines Datenwertes) nur eine Regel in der Regelsyntax, das auszulösende Event und die ausführende Aktion. Der Filter ist optional und kann entfernt werden. Damit lässt sich eine Analyse mit nur drei Elementen abgedecken. JessML benötigt für dieselbe Regel 7 Elemente im XML-Format und RuleML mindestens 4. Die Komplexität deckt den Zweck für die Analyse von Gesundheitsdaten ab, die das Ziel definieren. Siehe dazu mehr im Kapitel Implementierung und Evaluierung.

3.3 Regelsyntax

Die Regelsyntax beschreibt grundsätzlich die Eigenschaften einer Datenanalyse in einer maschinenlesbaren Form, die sich in einem Parser interpretiert lässt. Die Regelsyntax ist dabei unabhängig von Event oder Aktion. Das bedeutet, dass die Regelsyntax nicht beeinflusst wird von Event oder Aktion und beliebig ausgetauscht werden kann. Eine Regel ist immer eine Eigenschaft einer Datenanalyse, etwa der Vergleich eines Wertes oder das Ergebnis einer Berechnung. Eine Datenanalyse kann dabei aus einer oder mehreren Regeln bestehen. Die Regelsyntax beschreibt die für die Analyse notwendigen Eigenschaften und Optionen, die bei der Analyse angewendet werden sollen, also das Verhalten einer Regel bei der Anwendung. Der Hauptbestandteil der Analysatorvorlage ist die Regelsyntax. Sie bestimmt deren Analyseeigenschaft, die an die spezifizierten Vergleichswerte und Datenfelder angewendet werden soll.
Für eine Datenanalyse wird die Beschreibung bzw. Bezeichnung benötigt, welche Analyse durchgeführt werden soll. Also ist die Bezeichnung des Analyseoperators gefordert. Der Analyseoperator kann für die Ausführung oder den Vergleich einen Vergleichswert benötigen. Vergleiche brauchen immer einen Vergleichswert, der in den Analysen der ECHO-Plattform verwendet wird, weshalb der Vergleichswert als erforderlich gekennzeichnet wird. Außerdem wird benötigt, welcher Datensatz bzw.

welches Feld im Datensatz analysiert werden soll. Weitere Optionen sind nicht zwingend erforderlich, da sich mit den drei Parametern eine Analyse deklarieren lässt. Aus dieser Anforderung leitet sich die Anzahl der erforderlichen Parameter einer Regel ab. Die Beschreibung einer Analyse benötigt somit den Analyseoperator, welcher das Analyseverfahren beschreibt, den Vergleichswert für den Analyseoperator und das Datenfeld, das analysiert werden soll.

Die Regelsyntax, die in Abbildung 3.6 dargestellt ist und im Analysesystem eingesetzt wird, wurde so einfach wie möglich gestaltet. Sie leitet sich aus den Zielen der Anforderung ab, die eine Analyse auch durch normale Anwender ohne IT-Kenntnisse ermöglichen soll. Aus diesem Grund wurde die Regelsyntax mit lediglich drei Parametern definiert, die für eine Beschreibung als Regel einer Datenanalyse ausreichen. Drei Felder genügen, da sich damit eine Datenanalyse mit einem Datenfeld, Analyseoperator und Vergleichswert beschreiben lässt, was die minimale Form einer Analyse darstellt, wie im vorherigen Absatz diskutiert wurde. Bei den Parametern handelt es sich um das Datenfeld in einem Datensatz, den Analyseoperator und den Vergleichswert. Optional sind noch die Parameter Anzahl, Datensätze und der Verknüpfungsoperator konfigurierbar.

Die Abbildung 3.6 zeigt im unteren Bildabschnitt ein Beispiel auf Basis der Regelsyntax. Es beschreibt die Regel X mit dem Datenfeld „Adapter.Tabelle.Spalte“, dem Analyseoperator „==“ und dem Vergleichswert „true“ mit Wertebereich von drei Datensätzen und der Regelverknüpfung mit „AND“. Die Regel vergleicht drei Datensätze, bei diesen muss das Datenfeld „Zeile.Spalte“ mit dem Vergleichswert „true“ übereinstimmen, welches der Analyseoperator „==“ prüft. Stimmt dieser Vergleich bei den drei Datensätze überein, liefert die Regel „Wahr“ zurück, was per „AND“ mit der nächsten Regel verknüpft wird. Stimmt der Datenwert mit dem Vergleichswert durch den Analyseoperator „==“ nicht überein, bei den anderen zweien jedoch schon, liefert die Regel den Zustand „false“ zurück. Dies geschieht ebenfalls, wenn alle Datensätze nicht übereinstimmen.

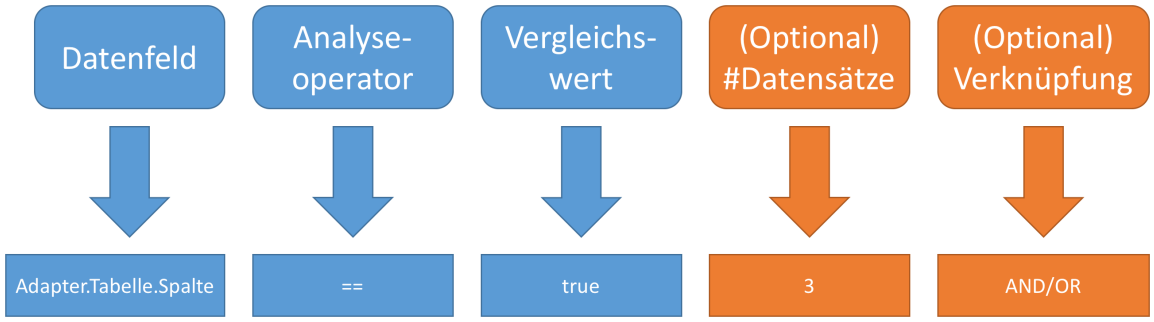

Abbildung 3.6: Spezifikation einer Regel in der Regelsyntax mit einer Beispielregel.

Das **Datenfeld** beschreibt den Standort, an dem der Wert hinterlegt ist, der mit dem Analyseoperator und dem Vergleichswert analysiert werden soll. Dabei kann es sich um einen einzelnen Wert handeln oder um eine Vielzahl von Datensätzen mit eigenen Datenfeldern. Es muss nur klar und eindeutig den Datensatz kennzeichnen. Deshalb kommt der Separator „.“ zum Einsatz. Er trennt die einzelnen Parameter des Standorts. Die Parameter beschreiben dabei den genauen Weg zum Datenwert. Die erste Zeichenfolge (Bezeichner) vor dem Separator „.“ beschreibt den Adapter der Datenquelle. Eine Datenquelle kann dabei eine Datenbank, ein Dateipfad, ein Objekt oder eine andere Schnittstelle sein. Der Bezeichner muss nur eindeutig den Adapter kennzeichnen. Verfügbare Adapter sind da-

bei im Analyse Management System deklariert. Die nachfolgenden Zeichenfolgen zwischen dem Separator beschreiben die weiteren Stufen des Pfads zum Datenwert innerhalb des Adapters. Die Abstufung des Pfads kann dabei beliebig lang sein. Der Pfad beschreibt nur den Weg zum Datenwert im Datenmodell des Adapters, nicht den Pfad zu einem spezifischen Datensatz. Das in Abbildung 3.6 dargestellte Beispiel einer Regeldeklaration zeigt den Pfad zum Datenwert mit der Zeichenfolge „Adapter.Tabelle.Spalte“. Der Adapter kann dabei eine relationale Datenbank sein und wird im Beispiel als „Adapter“ bezeichnet. Der nachfolgende Pfad beschreibt dann im Beispiel die Tabelle mit dem Namen „Tabelle“ in der Datenbank und die Spalte mit dem Namen „Spalte“. Eine Auswahl des Datensatzes findet nicht statt, sondern soll unabhängig vom einzelnen Datensatz den Pfad definieren. In der SQL-Syntax würde dies der folgenden Abfrage entsprechen:

```
SELECT Spalte FROM Tabelle;
```

Der **Analyseoperator** beschreibt das Verfahren bzw. die Methode, die angewendet werden soll, um das Datenfeld und den Vergleichswert zu analysieren und muss ebenfalls eindeutig kennzeichnet sein. Als Bezeichner für die Kennzeichnung wird ebenfalls - wie bei anderen Bezeichnern - eine Zeichenfolge verwendet. Für die Analyseregeln stellt das System die üblichen logischen Vergleichsoperatoren bereit, die vom Ausführungssystem zur Verüfügung gestellt werden. Dabei handelt es sich um:

Vergleichsoperatoren

- `==` Datenwert und Vergleichswert gleich
- `!=` Datenwert und Vergleichswert ungleich
- `>` Datenwert größer als Vergleichswert
- `<` Datenwert kleiner als Vergleichswert
- `>=` Datenwert größer oder gleich als Vergleichswert
- `<=` Datenwert kleiner oder gleich als Vergleichswert

Der Vergleichswert wird verwendet, um über den Analyseoperator mit dem Datenfeld zu vergleichen und kann ebenfalls jede Art von kompatiblem Datentyp des Datenfelds enthalten. Dazu stellt das Analyse Management System fest, dass nur Vergleichswert und Datenfeld desselben Datentyps analysiert werden. Dazu verwendet es den Datentyp des Vergleichswerts bei der Eingabe und ermittelt aus dem Datenmodell des Adapters den Datentyp des Datenfelds. Stimmen beide überein, kann eine Analyse beginnen. Damit ist sichergestellt, dass bei der Analyse nur Daten verglichen werden können, die sich vergleichen lassen, und es nicht zu einem unsicheren Vergleich kommt. Unsichere Vergleiche bedeuten dabei, dass Vergleichswert und Datenfeld vom Wert unterschiedlich sind, aber bei der Analyse wegen des Datentyps als gleich erkannt werden, oder wenn Vergleichswert und Datenfeld vom Wert her gleich sind, aber als ungleich erkannt werden, da der Datentyp in der Analyse zu einer Falschinterpretation führt.

Optional stehen noch zwei Parameter zur Verfügung: Zur Eingrenzung der Anzahl von Datensätzen, die geprüft werden sollen, und ein Verknüpfungsfeld zur Verbindung von mehreren Regeln.

Das Verknüpfungsfeld enthält dabei den Verknüpfungsoperator. Der erste Parameter spezifiziert den Auswahlwahlbereich an Datensätzen, mit dem sich festlegen lässt, wie viele Datensätze mit der Regelsyntax analysiert werden sollen. Die Angabe erfolgt hierbei immer als ganzzahlige Zahl. Wird dieser Parameter nicht angegeben, soll nur der aktuellste Datensatz analysiert werden. Mit Angabe des Auswahlbereichs wird die Anzahl an Datensätzen gewählt, die ebenfalls überprüft werden sollen. Mit dieser Angabe wird die Anzahl an Datensätzen gewählt, die über die Schnittstelle geladen werden. Die Regel ist nur erfüllt, wenn der Vergleichsoperator für alle ausgewählten Datensätze übereinstimmt.
Der zweite optionale Parameter für die Regeldefinition ist die Verknüpfung von Regeln. Dies ist notwendig, wenn die Analyse aus mehreren Regeln besteht. Die Abbildung 3.7 zeigt die Verknüpfung von einzelnen Regeln mit mehreren Verknüpfungsoperatoren. Wenn die Regelsyntax mehrere Regeln enthält, folgt auf eine Regel der festgelegte Verknüpfungsoperator und darauf eine weitere Regel. Mit dem Verknüpfungsoperator, der die booleschen Operatoren AND und OR erlaubt, können mehrere Regeln nacheinander verknüpft werden. Die Anzahl der verknüpften Regeln ist dabei beliebig. Die booleschen Operatoren AND und OR besitzen bei der Verknüpfung dieselbe Bindungskraft. Das bedeutet: Die Auswertung der booleschen Verknüpfung erfolgt immer von links nach rechts, also beginnend mit Regel 1 bis Regel n. Klammerung oder Änderung der Auswertungslogik (Auswertung der booleschen Verknüpfung) ist nicht möglich. Dies wurde gewählt, um die Verknüpfung der Regeln so einfach wie möglich zu beschreiben.
Eine Menge an Regeln wird nur erfüllt (boolescher Ausdruck „wahr“), wenn die komplette Verknüpfung der Regeln der logischen Verknüpfungsoperatoren ein erfüllt (wahr) liefert. Somit muss die Kombination aus den Verknüpfungen immer erfüllt liefern. Die verknüpften Regeln werden hintereinander ausgeführt - in der Reihenfolge, in der sie deklariert sind. Das boolesche Endergebnis aus der Verknüpfung beschreibt, ob die Regel erkannt wurde oder nicht. Zwischen zwei Regeln ist immer ein Verknüpfungsoperator, der die Regeln miteinander verbindet, um eine boolesche Auswertung zu erlauben. Die letzte Regel in der Syntax enthält keinen weiteren Verknüpfungsoperator, das bedeutet, dass die letzte Regel die Syntax terminiert. Enthält eine Analyse n Regeln, dann enthält die Regelsyntax n-1 Verknüpfungsoperatoren für die Verbindung der einzelnen Regeln.

Abbildung 3.7: Beschreibung der Syntax zur Verknüpfung mehrerer Regeln in der Regelsyntax.

Im Vergleich zu RuleML, JessML oder Jess, die in den Grundlagen vorgestellt wurden, bietet der reduzierte Umfang an Parametern und Optionen die Möglichkeit, einfach über drei Parameter eine Regel zu erstellen, die auch von Anwendern ohne IT-Kenntnisse gesetzt werden kann. Es reichen für eine einzelne Regel in der Regelsyntax drei Parameter, die das Datenfeld, den Vergleichsoperator und den Vergleichswert beschreiben. Jess bietet mit der Lisp-Syntax eine vom restlichen Java oder Javascript Framework unterschiedliche Syntax. Der Anwender muss sich mit der Lisp-Syntax auseinandersetzen die nicht üblich ist in den beliebten Programmiersprachen. JessML und RuleML verwenden XML-Syntax zur Deklaration der Regeln, erfordern aber für eine einzelne Regel ein kompliziertes Konstrukt aus Elementen, Subelementen und Verschachtelungen. Sie bieten im Gegensatz zu dieser Regelsyntax aber eine breitere Funktionalität, die jedoch für Anwender ohne IT-Kenntnisse (sprich die Zielgruppe dieser Lösung) nicht nötig ist, da nur einfache Regeln erzeugt werden sollen. Bei beiden müsste dies

über weitere Konzepte abstrahiert werden.
Größter Nachteil von Jess (JessML) ist jedoch, dass es nur als Java Framework verfügbar ist. Das bedeutet, dass in Jess die zu analysierenden Daten in Objekten vorliegen müssen. Somit muss bei Jess erst das Datenmodell in Java-Objekte konvertiert werden, was keinen generischen Ansatz erlaubt, da immer erst eine teilweise manuelle Konvertierung erfolgen muss. Der größte Nachteil von RuleML ist die Einbindung von Aktionen und Events, die mit externen Systemen erfolgen. Dies ist bei RuleML nur in beschränkter Form möglich. RuleML kann als Aktion die Änderung von Werten im Datensatz durchführen aber nicht generische Aktionen ausführen wie einen REST API-Aufruf. Auch kann RuleML nicht auf externe generische Events reagieren. Es reagiert nur auf Änderungen bei einem Datensatz. Beide Nachteile können von der Regelsyntax abgedeckt werden.

3.4 Erweiterbare Funktionalität

Die Analysearchitektur verwendet Analyseoperatoren, Datenmodelle mit Datentypen der zu analysierenden Daten, Verknüpfungsoperatoren, auslösende Events, wählbare Texttokens aus dem Datenmodell und ausführende Aktionen. Diese Eigenschaften des Analysesystems werden zentral im Ablaufplan des Parser deklariert oder programmiert und bieten somit eine Erweiterung an das Umfeld des Analysesystems. Es wird damit gewährleistet, wechselnde Datenmodelle zu erkennen, neue Operatoren zu verwenden oder auf erweiterte Events zu reagieren - welche aus veränderten Eigenschaften der externen Daten oder dem Umfeld entstehen. Folgende Optionen sieht die Analysearchitektur vor:

Anpassbare Eigenschaften

a) Datenmodell und Datentyp

b) Vergleichsoperatoren

c) Verknüpfungsoperatoren

d) Events

e) Aktionen

f) Texttokens

Zum Auslesen des Datenmodells ist ein Adapter im Parser vorhanden, der zur Laufzeit das Datenmodell der zu analysierenden Daten interpretiert und die vorhandenen Datenfelder und deren Datentypen extrahiert. Dies erfolgt immer bei der Erstellung einer Analyse und wird somit dynamisch zur Laufzeit interpretiert. Das ermöglicht ein dynamisches Datenmodell der zu analysierenden Daten, welches jederzeit angepasst werden kann und nur eine Aktualisierung des Analysators benötigt. Dazu wird das Metamodell bzw. die Datenbeschreibung ausgelesen, die von der externen Plattform zur Verfügung gestellt wird. Der Adapter für das Datenmodell kann ebenfalls erweitert werden, um neue Datenmodelle parsen zu können oder eine Kommunikation mit anderen System vorzunehmen, indem die Metamodellbeschreibung, eine Transformierungsfunktion zur Umwandlung der Datenmodelle in ein vom Parser lesbares Format und der Standort des externen Systems im Analyse Management System hinterlegt wird.

Analyseoperatoren und Verknüpfungsoperatoren können ebenfalls beliebig erweitertet werden, indem sie im Parser eindeutig deklariert werden. Außerdem wird die Funktionalität in der verwendeten Programmiersprache des Ausführungssystems programmiert. Dies muss manuell erfolgen, kann aber über einen externen Adapter vom Parser abgerufen werden, welcher die Deklaration bereits enthält. Die Beschreibung eines Operators besteht aus zwei Feldern, dem verwendeten **Operatornamen**, welcher auch als Index dient, und der programmierten **Funktion**. Dazu kennzeichnet die eindeutige Deklarierung den Operatornamen, der in der Analysatorenvorlage verwendet wird, um in der Regelsyntax den verwendeten Operator zu beschreiben. Zur Beschreibung des Operatornamen kann ein beliebiger String verwendet werden, der noch nicht vergeben wurde und der Syntax entspricht, die die Ausführungssprache erlaubt. Die Funktionalität wird als Ausführungssprache des Ausführungssystems beschrieben und kann in vollen Umfang eingesetzt werden, um den Operator zu programmieren. Dies erlaubt auch die Einbindung externer Programme, Verwendung von Schnittstellen oder das Einbinden von Bibliotheken. Die Programmsprache wird ebenfalls als String im zweiten Feld hinterlegt und muss der Syntax der Ausführungssprache entsprechen. Der Programmteil wird bei der Analyse vom Ausführungssytem interpretiert und ausgeführt.

Events und Aktionen können im Parser deklariert und erweitert werden. Die bereits beschriebenen Events und Aktionen sind darin bereits enthalten. Zur Deklaration muss bei Events der **Eventbezeichner** deklariert sein. Bei Aktionen muss der **Aktionsbezeichner** und die auszuführende **Aktion** enthalten sein. Wie bei den Operatoren muss eine eindeutige Bezeichnung angegeben werden, die aus einem beliebigen String besteht und in der Analysatorvorlage zur Kennzeichnung verwendet wird. Diese eindeutige Bezeichnung des Events wird auch als eindeutiger Eventbezeichner verwendet. Der Eventbezeichner ist beim Analysator hinterlegt und wird beim Eintreffen des spezifischen Events aktiviert. Eine weitere Definition ist bei Events nicht notwendig, die Events müssen nur mit dem von externen Systemen erzeugten Events übereinstimmen. Bei Aktionen ist - wie bei den Operatoren - ein zweiter Deklarationsbestandteil vorhanden, der mit der Ausführungssprache die Funktionalität der Funktion beschreibt. Die Funktionalität muss als String hinterlegt sein, der vom Ausführungssystem gestartet werden kann. Darin kann weiterer Programmcode enthalten sein, der über eine deklarierte Schnittstelle (welche ebenfalls in der Programmbeschreibung enthalten ist) externe Programme, Bibliotheken oder Systeme anspricht, die nicht im Analysesystem enthalten sind.
Texttokens kommen nur bei Aktionen vor, welche ein Datenobjektfeld des Datenfelds bezeichnen, die bei der ausführenden Aktion im Textstring durch die Werte des Datenfeldes ersetzt werden. Die Tokens müssen im Parser deklariert werden, um die einzelnen Datenwerte für die Benachrichtigung freizugeben, die generell wegen des Datenschutzes deaktiviert sind. Zu Deklaration muss der Tokenstring und der Datenfeldbezeichner angegeben werden, der auf das Token referenziert werden soll. Das Token besteht aus einem Textstring, beginnend mit dem Öffnen der geschweiften Klammer „{“, dem Datenfeldnamen und der abschließenden geschweiften Klammer „}“. Damit ergbit sich das Token mit folgender Bezeichnung: „{Token-Bezeichner}“.

3.5 Data Mining

Unabhängig von den generellen Analysefunktionen, die Standard-Operatoren enthalten und von Events ausgeführt werden, verfügt die Analysearchitektur noch über die Funktion zur Ausführung

von Data Mining-Analysen auf bereits existierende Daten, die vom Analysesystem über einen Adapter abgerufen werden. Dies erfolgt, um der Analysearchitektur ein komplettes Spektrum an Analysen bereitzustellen, ohne dass große Änderungen an dem System erfolgen müssen.
Bei Data Mining-Analysen wird nicht per Analysatorvorlage die Analyse selbst als Regelsyntax beschrieben, sondern nur das verwendete Analyseverfahren und die notwendigen Parameter. Es erfolgt somit in der Regelsyntax nur die Auswahl des Verfahrens. Die Verfahren sind im Analyse Management System deklariert. Die notwendigen Datenmodelle und die aufbereiteten Datensätze werden nicht vom Analyse Management System bereitgestellt, sondern werden manuell erzeugt. Das Analyse Management System erlaubt dann den Zugriff auf die aufbereiten Daten. Die Abbildung der Analyseverfahren erfolgt als Ablaufplan im Analysesystem und verwendet dieselben Funktionsblöcke der Standardanalysen - nur unter der Verwendung eines Data Mining-Algorithmus. Die Ausführung erfolgt bei Data Mining über eindeutig gekennzeichnete externe Events, wie sie bei den Standardanalysen ebenfalls eingesetzt werden um die Analyse zu aktivieren. Die Beschreibung des Data Mining-Analyseverfahrens erfolgt durch die Regelsyntax und die Analysatorvorlage, jedoch nur die Auswahl des Verfahrens, nicht die Funktion des Verfahrens. Die Deklaration verwendet auf Grund der Art der Analyse in der Vorlage eine andere Interpretation der Felder in der Regelbeschreibung, die Regelsyntax entspricht jedoch der der Standardanalysen. Die Syntax wird dazu verwendet, um die klar definierte Funktionsweise per Regelsyntax zu parametrisieren, aber nicht deren Funktionsweise zu beschreiben, wie es der Fall bei Standardanalysen ist. Die Entscheidung der Analysatorvorlage erfolgt bei der eindeutigen Auswahl eines Data Mining-Operators für das Analyseverfahren, welches in der Vorlage deklariert wird. Anders als bei Standardverfahren erfolgt bei der Data Mining-Analyse keine Transformation der Daten zur Laufzeit in dem Ausführungssystem für die Anpassung an den Algorithmus. Für das Ausführen des Data Mining-Algorithmus müssen die über den Adapter abgerufenen Daten bereits ein passendes Datenmodell enthalten, welches direkt im Algorithmus angewendet wird. Die Aufbereitung der Daten erfolgt somit bereits bei externen Quellen, die die Daten in passender Form bereitstellen.

3.6 Zugriffsschutz

Bei der Handhabe und Analyse von persönlichen Daten, besonders im Gesundheitsumfeld, sollte der Zugriff auf die Daten stark reglementiert erfolgen. [DPNB11] Damit wird verhindert, dass Analysen unautorisierten Zugriff auf Daten erhalten. [dat] Zu diesem Zweck sieht die Analysearchitektur eine strenge Kontrolle der Datenfunktionen vor. Da die Analysatoren und Events anwenderspezifisch erzeugt werden, kann eine Trennung der einzelnen Analysen erfolgen. Damit ist gegeben, dass eine anwenderspezifische Analyse nur auf die vom Anwender bezogenen Daten Zugriff erhält und so der Datenzugriff auf die Benutzerkennung reglementiert wird. Das bedeutet, dass die vom Anwender erstellte Analyse die Anwenderkennung verwendet, von welcher das erzeugte Event entstanden ist, sodass nur die Daten in die Analyse fließen, auf die der Anwender selbst Zugriff erhält, jedoch keine, die er nicht aufrufen kann. So wird von der Architektur sichergestellt, dass keine Daten in die Analysen fließen und möglicherweise über Aktionen ausgegeben werden, auf welche vom Benutzer nicht zugegriffen werden kann. Das Analysesystem verwendet dazu nur externe Schnittstellen, die vom System bereitgestellt werden, das die Datenverwaltung durchführt, um die Daten für die Analyse abzurufen. Dazu wird immer die Anwenderkennung des auszuführenden Benutzers verwendet, die

nur die für den Anwender bestimmten Daten abrufen kann. Diese Zugriffsbeschränkung muss vom Datenhaltungssystem gegeben sein. Mit dieser Architektur wird verhindert, dass das Analysesystem selbst vollständigen Zugriff auf die Daten hat. Außerdem hat auch nur die Analyse selbst auf die spezifischen Daten Zugriff. Nach Beenden der Analyse werden alle abgerufenen Daten wieder vollständig verworfen und müssen beim nächsten Aufruf erneut abgerufen werden. Über die Deklarierung der Texttokens lassen sich die auszugebenden Daten beschränken, die das Analysesystem verlassen, welche bei der Ausführung einer Aktion gesendet werden.

4 Implementation

Die ECHO-Plattform verwendet Datenanalysen mit SQL-Abfragen an die Datenbank, die fest in den Quellcode der Anwendung implementiert wurde. [Ste14] Die SQL-Anfragen werden durch den SQL Treiber von Nodejs ausgeführt. Diese Implementierung ermöglicht nur feste Analysen, die statisch in den Quellcode implementiert sind und die nur durch Anpassung der Anwendung erweitert werden können. Dazu musste die Anwendung um weitere SQL-Abfragen erweitert werden, die eine weitere Analyse abbilden. Anpassbar durch den Anwender selbst sind die Analysen mit diesem Verfahren nicht. Jede Anpassung erfordert den Eingriff in den Quellcode. Mit der Einführung des neuen Analysesystems erlaubt die ECHO-Plattform eine Vielzahl von Analysefunktionen zur Unterstützung von Ärzten und Patienten: Die Analysen können - ohne am Quellcode Anpassungen durchzuführen - hinzugefügt oder gelöscht werden, sind anwenderspezifisch mit einer für den Anwender einfachen Gestaltungsform anlegbar, sind generisch erweiterbar, erlauben Data Mining und eine Multi-Serverarchitektur für bessere Skalierung und Performance.

4.1 Ausführungssystem

Das Ausführungssystem ist notwendig zur Ausführung der Analysen und des Management Systems und ist die Laufzeitumgebung für alle Dienste. Dazu werden Ablaufpläne im Ausführungssystem erzeugt, die durch externe Events oder Anfragen gestartet werden. Ein Ausführungssystem wurde verwendet, weil es eine variable Funktionalität unterstützt und als Laufzeitsystem für die komplette Analyseplattform dient. Es kann beliebigen Programmcode ausführen, ist unabhängig vom Betriebssystem und erlaubt eine einfache Serverarchitektur, da eine Synchronisation zwischen den Instanzen nicht notwendig ist. Jede Instanz agiert mit dem Teil seiner Analysedaten selbst. Ebenfalls erlaubt das Ausführungssystem das Einbinden externer Bibliotheken für Analyseverfahren und die Ausführung externer Programme auf dem Host. Eine Rule Engine, eine Datenbank oder event-basierendes oder relationales Datenbanksystem wurden nicht verwendet. Die Rule Engine bietet zwar eine fertige Syntax zur Beschreibung der Sprachen, erfordert aber ein zusätzliches Laufzeitsystem und verwendet, wie bereits gezeigt, bei Jess eine komplizierte Regelsprache. Ziel ist aber die einfache Definition von Regeln zur Datenanalyse. Auch bietet das Ausführungssystem bereits vorhandene Funktionsobjekte für die Kommunikation mit Webdiensten und eine Programmiersprache, die eine breite Funktionsimplementierung von Analysen erlaubt. Das Analyse Management System, wie die Analysen selbst, können darauf ausgeführt werden, was bei den Alternativen nicht der Fall wäre. Die Daten werden in der ECHO-Plattform auf einem MySQL-System gespeichert. Das bedeutet, dass eine Änderung der Datenspeicherung nicht vorgenommen werden kann, ohne große Änderungen an der bestehenden Plattform durchzuführen, was verhindert werden sollte. Dies würde auch eine generische

Implementierung verhindern, um die Analyse anderer Daten zu ermöglichen. Deshalb ist ein Analysesystem auf einem Datenbanksystem per plattformabhängigen Prozeduren nicht möglich, diese würden das generische Prinzip des Analysesystems verletzen und die Konvertierung der Daten auf das Datenbanksystem erfordern, welches die Datenanalysen ausführt. Aus diesem Grund wurde das Analysesystem nicht in der bereits bestehenden MySQL-Plattform als Prozeduren integriert, sondern auf ein unabhängiges Ausführungssystem, das flexibel programmierbar, datenbankunabhängig und integrierbar in die Plattform ist.

Für das Ausführungssystem wurde in der Implementation Node-Red verwendet. Node-Red ist ein auf Nodejs basierendes Open-Source-Ausführungssystem zur Kommunikation mit Hardware, Verwendung von Schnittstellen oder zur Bereitstellung von Services, die als Ablaufplan definiert werden. [nod] Nodejs ist eine Laufzeitumgebung zur Ausführung von JavaScript als Serveranwendung. [Tei12] Node-Red bietet eine breite Platte an bereits verfügbaren Funktionsmodulen für die Webkommunikation, die Analyse von Daten und die Kommunikation mit Schnittstellen und eignet sich deshalb für die Anwendung als Ausführungssystem. Die Abbildung 4.1 zeigt die Administrationsoberfläche von Node-Red mit dem Parser-Ablaufplan, was eine Änderung des Analyse Management Systems oder einzelner Analysen zur Laufzeit erlaubt, ohne die Regelsyntax zu verwenden.

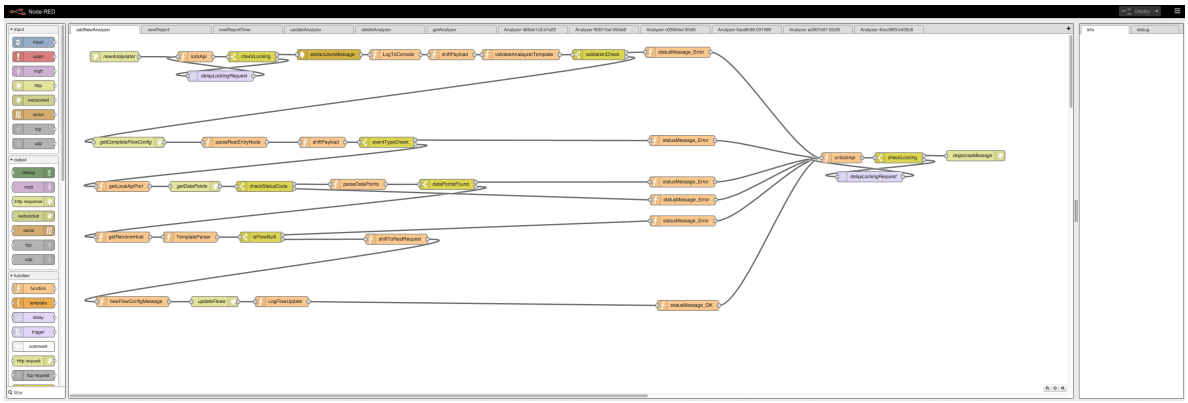

Abbildung 4.1: Administrationsoberfläche von Node-Red zum Management von Ablaufplänen.

Nodejs ist eine Plattform für Dienste in verteilten Systemen und ist in JavaScript geschrieben. Node-Red ist ebenfalls in JavaScript programmiert und verwendet in anwenderspezifischen Funktionsblöcken dieselbe Programmiersprache. So erreicht das Analysesystem mit der ECHO-Plattform eine homogene Umgebung mit denselben Technologien und einer einfacheren Wartung, da die Systeme nicht in unterschiedlichen Software-Komponenten existieren. Die ECHO-Plattform und das Analysesystem verwenden die Nodejs-Laufzeitumgebung, welche wiederum JavaScript nutzt. Damit ist eine gleiche Umgebung geschaffen mit derselben Laufzeitumgebung, was eine Integration in das Laufzeitsystem der ECHO-Plattform erlaubt sowie dieselbe Programmiersprache und Schnittstellen. Beide Systeme könne innerhalb einer Nodejs-Laufzeitinstanz auf unterschiedlichen Netzwerkports betrieben werden. Node-Red ist unabhängig vom Datenbanksystem, dem Datenbankmodell oder einem Regelsystem. Es kann somit über Erweiterungen, Deklaration von Analysen und Datentransformation per Adapter an andere Plattformen angepasst werden, was eine Rule Engine oder ein Datenbanksystem nicht erlaubt. Daraus folgt auch die Einschränkung der Funktionalität auf die

Haupteigenschaft. Diese Anpassung erfolgt nur an den Ablaufplänen im Ausführungssystem und benötigt keine oder nur geringe Änderung externer Plattformen, welche die Daten bereitstellen.

4.1.1 Funktionsbausteine

Node-Red verwendetet in den Ablaufplänen Funktionsbausteine, die eine bestimmte Aufgabe haben und von Node-Red bei der Ausführung des Ablaufplans gestartet werden. Die Abbildung 4.2 zeigt die Darstellung eines Funktionsbausteins in der Node-Red-Weboberfläche. Die Node-Red-Weboberfläche verwendet eine grafische Darstellung der Ablaufpläne.

Abbildung 4.2: Funktionsbaustein in Node-Red als grafische Darstellung.

Ein Funktionsbaustein besteht aus einer Funktionalität, die er ausführt, keinem oder einem Eingang und keinem, einem oder mehreren Ausgängen. Die Funktionalität kann per JavaScript programmiert werden. Zwischen den Funktionsbausteinen werden über die Verbindung zwischen Ausgang und Eingang Nachrichten ausgetauscht. Eine Nachricht ist dabei ein JavaScript-Objekt mit beliebigem Inhalt. Ein Funktionsbaustein wird nur aktiviert, wenn der Eingang eine Nachricht erhält. Dann führt er seine Funktion aus und kann die Nachricht anpassen, neu erzeugen oder weiterleiten. Das Versenden der Nachricht erfolgt dann über die Ausgänge. Sind mehr Ausgänge vorhanden, muss die Nachricht nicht an alle Ausgänge versendet werden. Alle Funktionsbausteine werden in einem eigenen Speicherbereich ausgeführt, und der Austausch von Daten ist nur über die Nachricht möglich. Nach der Ausführung des Funktionsbausteins werden alle im Funktionsbaustein anfallenden Daten wieder gelöscht. Das bedeutet: Die Variablen und Daten in einem Funktionsbaustein sind nur flüchtig vorhanden. Ein Speicherung durch eine Datei oder über einen globalen Speicherbereich oder eine Datenbank ist notwendig, wenn permanent Daten gespeichert werden sollen. Die Funktionsbausteine werden im Analysesystem verwendet, da sich die Funktionalität über den eingefügten JavaScript-Programmcode jederzeit ändern und bei der Erzeugung des Funktionsbausteins mit Parametern im JavaScript-Programmcode personalisieren lässt.
Zusätzlich zu dem Funktionsbaustein sind vorgefertigte Funktionsbausteine vorhanden, die bereits eine Funktionalität besitzen. Dabei kann es sich um boolesche Operatoren handeln, Zugriffe auf Schnittstellen oder das Speichern von Daten. Bei fertigen Funktionsbausteinen muss die Funktionalität nicht in JavaScript beschrieben werden, sondern ist bereits fertig implementiert. Vorgefertigte Funktionsbausteine haben je nach Funktionalität keinen Eingang, wenn es sich um einen Startknoten handelt, oder einen Ausgang, wenn es sich um einen Endknoten handelt. Der Startknoten startet den Ablaufplan, der Endknoten beendet einen Ablaufplan. Fertige Funktionsbausteine werden nur verwendet, wenn deren Funktionalität keine Änderungen erfordert oder auch angepasst werden kann.

Das Listing 4.1 zeigt die Darstellung eines Funktionsbausteins in der Node-Red-Konfiguration, welcher die aktuelle Nachricht auf die Konsole ausgibt. Der Funktionsbaustein besitzt einen Ausgang für die Weitergabe der Nachricht. Der Funktionsbaustein entspricht genau der Abbildung 4.2, nur dargestellt als Programmcode. Die einzelnen Module werden als JSON-Objekt mit dem Identifikator

Listing 4.1 Verknüpfungsoperator zur Verbindung von zwei Analyseregeln mit der Regelsyntax im Strukturteil der Analysatorvorlage.

```
{
      "id": "7b7a315b.8485d",
      "type": "function",
      "name": "LogToConsole",
      "func": "console.log(msg)\n\nreturn newMsg;",
      "outputs": 1,
      "valid": true,
      "x": 945,
      "y": 79,
      "z": "4dbe6746.b24198",
      "wires": [
            [
            "82dfe52f.7d2018"
            ]
      ]
}
```

(ID) des Ablaufplans in der Konfiguration gespeichert. Dies erfolgt mit dem „z“-Feld, welches mit dem eindeutigen Bezeichner den Ablaufplan deklariert. Das „id“-Feld beschreibt eindeutig die ID des Modules. Das „output“-Feld deklariert die Menge an Ausgängen. Die eindeutige ID wird ebenfalls im „wires“-Feld verwendet, um die Verbindung mit dem vorherigen Funktionsbaustein zu beschreiben. Das Feld wird als Liste abgebildet. Das „wires“-Feld ist abhängig von der Anzahl an Ausgängen. Je Ausgang ist im „wires“-Feld ein Feld für die ID des nachfolgenden Funktionsbausteines. Damit wird die Verkettung und Ausführungsfolge der Funktionsbausteine beschrieben. Node-Red folgt bei der Ausführung der IDs im „wires“-Feld, um den nächsten Funktionsbaustein zu ermitteln, der ausgeführt werden soll. Das „type“-Feld beschreibt die Art des Funktionsblocks, das „name“-Feld deklariert den Namen des Blocks, und die Koordinaten in den Feldern „x“ und „y“ beschreiben die Position des Moduls im Ablaufplan in einem Koordinatensystem mit zwei Dimensionen.

4.1.2 Aktualisierung der Ablaufpläne

Node-Red verwendet das JSON-Datenformat für die Speicherung der Ablaufpläne. Die Speicherung erfolgt dabei persistent auf dem nichtflüchtigen Speichermedium des Systems, auf dem Node-Red gestartet wurde. Die Konfiguration kann per Webschnittstelle aktualisiert werden. Dazu stellt Node-Red unter „/flows“ eine REST-API zur Verfügung, die die neue Konfiguration per JSON-Datenformat aktualisiert. Für die Aktualisierung der Konfiguration [nod] der Ablaufpläne in Node-Red stehen drei Arten des Aktualisierungsverfahrens zur Verfügung. Zur Aktualisierung muss im Hypertext Transfer Protocol (HTTP) Header des REST-Aufrufs das Feld „Node-RED-Deployment-Type“ mit einem der folgenden Aktualisierungsverfahren beschrieben sein:

Full stoppt alle Ablaufpläne auf dem Ausführungssystem. Nachdem alle gestoppt sind, führt Node-Red eine Aktualisierung aller Ablaufpläne und Blöcke durch und startet das System erneut.

Nodes liest die neue Konfiguration ein und vergleicht die Blöcke. Bei einem Unterschied zwischen neuem und altem Block wird der Block gestoppt und aktualisiert. Bei einer neuen Instanz des Ablaufplans wird die neue Konfiguration verwendet.

Flows vergleicht die Änderungen von Ablaufplänen und Blöcken. Wird ein Unterschied in einem Block oder einem Ablaufplan erkannt, wird dieser gestoppt und aktualisiert. Danach wird die neue Konfiguration gestartet und bei einer neuen Instanz verwendet.

Das Analyse Management System verwendet bei der Aktualisierung der Konfiguration den Parameter „Full“, damit es nicht zu einer Inkonsistenz bei der Bereitstellung kommt, wenn Blöcke zur Laufzeit ausgetauscht werden, während gerade eine Analyse läuft. Das könnte die Analyse verfälschen, da der Analyseoperator während der Laufzeit ausgetauscht wird, oder zu einem Fehler führen, weil der Operator auf Datenfelder zugreift, die nicht existieren, da der Ablaufplan soweit fortgeschritten ist und schon die Analysedaten eingelesen hat, aber die eigentliche Analyse mit dem neuen Operator startet. Werden alle Analysen gestoppt und dann aktualisiert, ist dies ausgeschlossen. Das führt jedoch nicht zum Verlust möglicher Events oder Beenden des Analyse Management Systems, da diese Aktualisierung nur wenige ms erfordert und in dieser Zeit der Timeout des HTTP-Request noch nicht abgelaufen ist. Die TCP-Verbindung, welche für HTTP verwendet wird, verhindert den Verlust. Erst wenn die Aktualisierung mehrere Sekunden dauert, kann dies zu einem Verlust führen.

4.1.3 Erweiterungen für Node-Red

Node-Red verwendet die Laufzeitumgebung Nodejs und ist in JavaScript programmiert. Damit können externe Bibliotheken oder Erweiterungen von JavaScript oder Nodejs auch von Node-Red verwendet werden. Node-Red bietet die Möglichkeit zur Einbindung weiterer Bibliotheken über die Konfiguration. Damit lassen sich die Bibliotheken auch in den Funktionsbausteinen verwenden. Das ist notwendig, da die für Nodejs verwendete Methode „require()“ zum Laden von Erweiterungen nicht in Funktionsbausteinen zur Verfügung steht. Die Erweiterungen müssen bei Node-Red global in der Konfigurationsdatei geladen werden. Sie stehen dann als globale Variablen unter „context.global.<variable>“ in allen Funktionsbausteinen zur Verfügung. Damit erlaubt auch Node-Red einen gemeinsamen Speicherbereich zwischen Funktionsbausteinen oder Ablaufplänen über die Verwendung der Konfiguration als Speichermedium. Für die Implementation des Analysesystems müssen folgende Erweiterungen in der Laufzeitumgebung Nodejs installiert und in die Konfiguration eingetragen werden:

Verwendete Erweiterungen

a) jsonwebtoken

b) lockfile

c) fs

d) c4.5

e) mysql

Die Erweiterung „jsonwebtoken“ wird für die Erzeugung von Authentifikationstokens benötigt. Die Bibliothek stellt die Methoden zur Erzeugung der von der ECHO-Plattform verwendeten Authentifikationsverfahren bereit. Die „lockfile“-Erweiterung wird für die Synchronisation parallel laufender Ablaufpläne gebraucht. Die Erweiterung stellt das Erzeugen von Sperrungsdateien auf der Festplatte zur Verfügung, um die Inkonsistenz der Node-Red-Konfiguration zu verhindern. Zur Erzeugung der Dateien auf der Festplatte wird die Erweiterung „fs“ benötigt. Sie stellt die Methoden zur Interaktion mit dem Dateisystem zur Verfügung. Die Erweiterung „c4.5“ stellt den Data Mining-Algorithmus C4.5 bereit. C4.5 verwendet Entscheidungsbäume, um Datensätze zu klassifizieren. Die „mysql“-Erweiterung wird genutzt, um für den Data Mining-Algorithmus eine native MySQL-Datenbank-Verbindung aufbauen zu können. Diese ist erforderlich, um die für Data Mining aufbereiteten Daten direkt von der MySQL-Datenbank der ECHO-Plattform zu laden.

4.2 Analyse Management System

Der Parser transformiert die Analysatorvorlage in einen vom Ausführungssystem ausführbaren Ablaufplan. Der Parser ist Hauptbestandteil des Analyse Management Systems, das ebenfalls im Ausführungssystem als Ablaufplan implementiert ist. Somit besteht der Parser selbst, wie alle erzeugten Analysen und das Analyse Management System, aus einem im Ausführungssystem erzeugten Ablaufplan. Diese Implementierung wurde zum einen gewählt, weil sie kein weiteres Laufzeitsystem erfordert. Die Analysen benötigen ein Ausführungssystem, somit ist dieses bereits vorhanden und kann auch für das Analyse Management System verwendet werden. Dies spart eine weitere Laufzeitumgebung für das Analyse Management System ein. Zweitens kann das Analyse Management System auf allen Ausführungssystemen gestartet werden. Wenn mehrere Systeme vorhanden sind, bietet es damit eine verteilte Serverarchitektur, die bei n Ausführungssystemen n-1 Ausfälle toleriert oder eine Skalierung der Anfragen ermöglicht. Außerdem kann auf jedem Ausführungssystem gleichzeitig zu den Analysen das Analyse Management System betrieben werden - man benötigt so für die Ausführung minimal ein System. Auf dem Ausführungssystem werden der Parser und alle Analysefunktionsblöcke, die durch den Parser erzeugt werden, in JavaScript implementiert. Das ist erforderlich, da das Ausführungssystem Node-Red nur eine Unterstützung von JavaScript als Programmcode in den Funktionsblöcken erlaubt. Folgende Ablaufpläne sind im Analyse Management System implementiert zur Verwaltung und Steuerung des Analysesystems:

Ablaufpläne im Analyse Management System

a) Parser

b) Webschnittstellen zum Anlegen, Löschen und Aktualisieren von Analysatoren

c) Webschnittstelle zum Abruf von aktiven Analysatoren

d) Webschnittstelle für eingehende Events

Der Parser im Analyse Management System wird für die Erzeugung der Ablaufpläne aus der Analysatorvorlage benötigt. Die Webschnittstellen für das Anlegen, Löschen und Aktualisieren sind notwendig, um die vom Anwender gesendeten Analysatorvorlage zu empfangen und die jeweilige Aktion auszuführen. Die verwendete Webschnittstelle entscheidet die ausgeführte Aktion. Die

empfangenen Analysatorenvorlagen werden an den Parser weitergeleitet. Die Webschnittstelle für Events wird verwendet, um die von Events gesendeten Events zu empfangen und an die jeweiligen Ablaufpläne weiterzuleiten.
Alle externen Schnittstellen des Analyse Management Systems sind als HTTP-Webservice implementiert, der JSON als Datenformat für die Übertragung verwendet und sich an REST orientiert. Das Datenformat wurde gewählt, weil die ECHO-Plattform ebenfalls eine REST-API für die Kommunikation bietet und Node-Red selbst eine HTTP-REST-API mit JSON als Datenformat nutzt. Damit hat das Analyse Management System ein zur ECHO-Plattform und zu Node-Red homogenes System für Schnittstellen. Die REST-Webschnittstellen des Analyse Management System sind als Ablaufplan des Ausführungssystem implementiert. Die Ablaufpläne verwenden dazu die fertige HTTP-Anfrage und Antwort-Funktionsbausteine, die von Node-Red aus dem Modulbaukasten bereitgestellt werden und den Webdienst (Webserver) von Node-Red nutzen. Das erfordert keinen zusätzlichen Webserver auf dem System und spart Ressourcen. Die anderen Funktionsblöcke setzen ebenfalls fertige Module aus dem Modulbaukasten ein oder verwenden direkt JavaScript als Programmcode. Der Parser generiert aus der Analysatorvorlage den Ablaufplan der Analyse, die Webschnittstellen fürs Anlegen, Löschen und Aktualisieren der Analysatoren nehmen die Analysatorvorlage entgegen und führen die jeweilige Aktion aus. Die Anfrage wird mit der aktuellen bzw. neuen Analysator-ID beantwortet für den späteren Zugriff auf den Analysator durch das Analyse Management System. Dazu wird der Ablaufplan des Analysators erzeugt und durch die von Node-Red bereitgestellte Schnittstelle aktiviert. Die Webschnittstelle für die Anzeige der Analysatoren liefert die aktuellen aktiven Analysatoren mit deren ID zurück, die aktiv auf dem Ausführungssystem sind. Sind mehrere Systeme aktiv, antwortet die Webschnittstelle nur mit den lokal ausgeführten Analysatoren.

Da Node-Red bei einer neuen Anfrage eine neue Instanz erzeugt, laufen bei mehreren kurz hintereinander folgenden Anfragen mehrere aktive Instanzen des gleichen Ablaufplans parallel und unabhängig voneinander. Node-Red betreibt dabei jeden Funktionsblock als einzelne Instanz, die nach Beenden alle lokalen Daten wieder vernichtet. Es wird kein Datenaustausch zwischen mehreren aktiven Instanzen zugelassen. Es ist erlaubt, dass Analysen parallel auf einem System ausgeführt werden. Auch die parallele Ausführung der Funktionsblöcke ist möglich.

4.3 Analysatorvorlage und Parser

Die Vorgabe eines Systems für die Implementation von Analysevorlage oder Analysesystem gab es nicht. Die Analyseplattform sollte sich jedoch in die ECHO-Plattform integrieren lassen, dynamische Analysen ermöglichen, über einfache Regeln durch den Anwender erstellen lassen und, wenn möglich, die verfügbaren Schnittstellen für die Kommunikation verwenden. Für die Abbildung der Analysen nutzt das Analysesystem eine Regelsyntax für die Beschreibung der Analysebausteine. Diese Syntax wird gekapselt in einer Analysatorvorlage, die der Parser in eine für das Ausführungssystem interpretierbare Form konvertiert. Als interpretierbare Form wird ein Ablaufplan verwendet. Das Ausführungssystem nutzt den Ablaufplan zur Ausführung der Analysen. Die Daten werden in der ECHO-Plattform in einer MySQL-Datenbank gespeichert, die fest für die Plattform vorgegeben ist. Die Implementierung der Analysen muss den Zugriff auf die bestehenden Daten ausführen und zwecks Datenschutz nicht selbst speichern.

4.3.1 Analysatorvorlage

Für den Container der Analyseregeln und die Beschreibung des Analysators wurde das JSON-Datenformat verwendet, statt pures XML oder ein anderes Datenformat. Grund dafür ist das in der ECHO-Plattform verwendete Datenformat, das ebenfalls JSON für die Übertragung von Daten über die bereitgestellte Webschnittstellen einsetzt. Dabei verwendet die Plattform die Swagger-API, die eine RESTful-API in der Nodejs-Laufzeitumgebung darstellt. Die Nodejs-Laufzeitumgebung verwendet JavaScript als Implementierungssprache, welches häufig JSON als Datenformat nutzt. Da sich das Analysesystem integrieren lassen soll und somit alle Schnittstellen über ein System erreichbar sein sollen, werden die Analyseschnittellen ebenfalls in der Swagger-API angeboten, um ein homogenes Datenformat für alle Schnittstellen zu haben und alle Funktionen über einen Endpunkt bereitzustellen. Diese Verwendung von Swagger und Nodejs mit JavaScript führt zur Verwendung von JSON, da sonst ein zweites Datenformat in der gleichen Schnittstellenumgebung erforderlich wäre, was kein homogenes Format erlaubt. Durch die in Nodejs verfügbaren JSON-Parser-Standardwerkzeuge lässt sich die Analysatorvorlage einfach einlesen und per JSON-Schema validieren und benötigt somit keine zusätzlichen Werkzeuge, was ein eigenes Format erfordern würde.

Als Regelbeschreibungssprache wurde eine in JSON abgebildete Objektstruktur verwendet, welche die Analysatorbeschreibung im JSON-Datenformat repräsentiert, an Stelle einer bereits vorhandenen Regelbeschreibungssprache wie RuleML oder JessML. Dazu werden die Parameter der Analyseregel als JSON-Objekt in einem JSON-Array dargestellt, was für die Beschreibung der Analyse ausreicht. Grund dafür ist, dass als Container JSON verwendet wird. Die JSON-Objektbeschreibung erlaubt ein homogenes Datenformat in der Analysatorvorlage statt der Einbettung eines weiteren Datenformats oder des Wechsels des Schnittstellenformats. So wird die komplette Analysatorvorlage als JSON-Datei mit JSON-Objekten beschrieben, die über einen JSON-Parser vollständig eingelesen werden kann. Ein separater Parser zum Einlesen der Regeldefinition ist damit nicht notwendig, wie es mit RuleML oder JessML der Fall wäre. Alles erfolgt komplett per JSON-Parser.
Auch wurde keine Regelsprache wie JessML oder RuleML zur Beschreibung der Regeln verwendet, weil die Deklaration im JSON-Format eine Beschreibung der einzelnen Regeln möglich macht. Die Regeldefinition verwendet drei Parameter, die in der Regelsyntax definiert wurden. Durch die Verwendung von RuleML oder einer anderen Regelbeschreibungssprache hätte die Regelsyntax weit umfangreicher ausfallen müssen. Sie hätte vom Anwender deklariert werden müsssen, was somit in der Verwendung die Komplexität erhöhen würde und eine Transformation von Regelsyntax zu einer Regelbeschreibungsprache erforderlich machen würde. Das in den Grundlagen beschriebene Beispiel zu RuleML in Listing 2.1 (beschreibt Regel zur Prüfung eines Wertes) zeigt, dass die Regelsyntax ein umfangreicheres Konstrukt zur Regelbeschreibung erfordert und sich nicht in drei Regeleigenschaften abbilden lässt. Dies ist mit der Regelsyntax möglich, da nur das Datenfeld, der Vergleichsoperator und der vergleichende Wert beschrieben werden müssen. Im Vergleich dazu zeigt die Listing 4.2 ebenfalls die Prüfung eines Datenwertes eines einzigen Datenfeldes.

Zur Beschreibung der Regel muss nur das JSON-Array im „structure“-Bereich der Analysatorvorlage mit einer Regel befüllt werden, welche die erforderlichen Parameter der Regelsyntax enthält. Das ist im dem gezeigten Listing 4.2 der Vergleich eines Datenwertes, welcher im „field“-Feld beschrieben wird, mit dem Vergleichungsoperator im „exp“-Feld mit dem Wert dem booleschen Datenwert „true“ im „value“-Feld. Das Feld „type“ beschreibt den Typ des Regelknotens. In der Implementierung sind zwei

Listing 4.2 Beschreibung eines Analysators mit einer Analyseregel in der Regelsyntax und der Verwendung des JSON-Datenformats zur Übertragung und Interpretation.

```
{
      "structure": [
            {
                  "type": "node",
                  "field": "daily_reports.q1",
                  "exp": "==",
                  "value": true,
                  "range": "NULL"
            }
      ],
      "event": {
            "type": "newDailyReport",
            "parm": {}
      },
      "action": {
            "type": "sendMessage",
            "parm": {"message": "Testnachricht mit Token {id}"}
      }
}
```

Typen vorgesehen, „node“ und „operator“. Der „node“-Typ beschreibt den Knoten als Regeldefinition nach der Regelsyntax, die in Kapitel 3.3 deklariert wurde. „Operator“-Typ beschreibt den Knoten als Verknüpfungsoperator, der auf einen „node“-Typ folgt, wenn zwei Node-Knoten verknüpft werden sollen. Der Typ wird vom Parser interpretiert, um zu erkennen, was der Typ des Knotens in der Strukturbeschreibung der Analyseregel ist. Somit lässt sich eine einfache Erkennung der einzelnen Knoten im Parser ermöglichen, ohne auf eine statische „node“- und „operator“-Folge zu setzen. Das „field“-Feld beschreibt den Datenpunkt bzw. die Datentabelle mit der Spaltenkennung. Tabelle und Spalte werden mit einem Verknüpfungszeichen „.“ verbunden und erlauben eine zweidimensionale Identifikation des Datenpunktes, der verglichen werden soll. Somit kann die Tabelle deklariert werden, die den Datensatz enthält, und die Spalte in der Tabelle, die den Datenwert zur Analyse enthält. In dem gezeigten Beispiel wird auf die Tabelle „daily_reports“ zugegriffen und die Spalte „q1“ zur Analyse herangezogen. Diese Deklaration wird genutzt, da das Backend der Plattform eine MySQL-Datenbank verwendet, die Tabellen- und Spaltenkennzeichnung von Datensätzen einsetzt, um ein Datenfeld zu identifizieren. So lässt sich die MySQL-Bezeichnung in der Analysevorlage übernehmen. Beim Wechsel des Backends für die Datenhaltung kann das Format adaptiert werden, um etwa die Indexe eines Arrays zu beschreiben. Das erlaubt eine einfache Wiederverwendung statt konkreter Beschreibung des Feldes, das sich nicht übertragen lassen kann. Das „exp“-Feld beschreibt den Analyseoperator, der verwendet werden soll. In diesem Beispiel ist es der Vergleichsoperator „==“, um den Datenwert mit dem gegebenen Wert zu vergleichen und bei Übereinstimmung die Aktion auszulösen. Die Art der Operatorbeschreibung wird verwendet, da sie im Analyse Management System bereits als String-Bezeichnung zur Identifikation hinterlegt ist. Das Feld „value“ enthält den Wert, der vom Analyseoperator verwendet wird, um die gewählte Analyse mit dem im Datensatz enthaltenen Wert durchzuführen. In diesem Beispiel ist es der boolesche Wert „true“. Das letzte Feld „range“ beschreibt die Menge an Datensätzen, die für die Analyse verwendet werden sollen. Die Range „NULL“, die dem Parser die Anzahl 0 an Datensätze mitteilt, legt fest, dass nur ein Datensatz

Listing 4.3 Verknüpfungsoperator zur Verbindung von zwei Analyseregeln mit der Regelsyntax im Strukturteil der Analysatorvorlage.

```
{
      "type": "operator",
      "op": "AND"
}
```

verglichen werden soll. Bei der Angabe einer Ganzzahl größer 1 wird die im „range“-Feld enthaltene Zahl verwendet, um die Menge an Datensätze abzurufen und der Analyse zuzuführen. Die Felder type, field, „exp“, „value“ und „range“ sind bei dem Typ „node“ verpflichtend. Sind sie nicht enthalten oder fehlen teilweise, dann kann die Analysatorvorlage nicht vom Analyse Management System validiert werden. Der Typ „operator“, der den Verknüpfungsoperator beschreibt, erfordert nur zwei Felder in der Regelsyntax. In Listing 4.3 wird der Verknüpfungsoperator in der Regelsyntax gezeigt. Für die korrekte Beschreibung der Verknüpfung sind die Felder type und op erforderlich. Beide Felder müssen deklariert sein, um eine Validierung zu bestehen. Das Feld type kennzeichnet die Struktur, die Bezeichnung erfolgt bei einem Verknüpfungsoperator mit der Kennzeichnung „operator“, im Gegensatz zu „node“ für eine Regelbeschreibung. Das zweite Feld op enthält die Art der Verknüpfung. Implementiert sind die logischen Verknüpfungen AND und OR für eine UND-Verknüpfung und für eine ODER-Verknüpfung. Im Feld op muss bei einer Verknüpfung ein AND oder ein OR deklariert sein. Fehlende oder unkorrekte Deklarierung der Felder verhindet eine Validierung und führt zum Abbruch durch den Parser. Das stellt sicher, dass keine fehlerhaften Analysen erzeugt werden kann.

Die Analysatorvorlage enthält, nach der Regelbeschreibung im Strukturteil, auch das Event, das zur Ausführung der Analyse führt. Das bedeutet, dass bei Eintreffen des spezifischen Events die Analyse ausgeführt wird. Für eine homogene Beschreibung der Analyse ist diese Deklaration ebenfalls im JSON-Datenformat, so dass der Analysator komplett von einem JSON-Parser in eine Objektstruktur transformiert werden kann. Das Event, das zum Ausführen des Analysators führt, wird in der „event“-Struktur der Analysatorvorlage deklariert. Der Kennzeichner „event“ bestimmt, dass im Parser die nachfolgende Datenstruktur, welche ebenfalls in JSON deklariert ist, für die Beschreibung des Events verwendet werden soll. In der Datenstruktur sind dafür die Felder „type“ und „parm“ vorgesehen. Das „type“-Feld enthält die eindeutig im System gekennzeichnete Event-ID. Mit dieser eindeutigen ID kann der Parser den Funktionsblock für dieses Event abrufen und generieren. Zur Parametrisierung des Events enthält die Eventstruktur das Feld „parm“, welches Parameter umfassen kann, um das Event weiter einzuschränken oder weitere Informationen mitzuliefern, die die Analyse spezifizieren. Das Parameterfeld wurde gewählt, da sich mit dieser Datenstruktur beliebig viele Parameter übertragen lassen, die für spätere Analysen notwendig werden könnten. Somit bietet es eine generische Anpassbarkeit des auszuführenden Events oder des Analysators. Für die Standardanalysen wird das „parm“-Feld nicht benötigt. Das Feld wird in der jetzigen Implementierung dafür eingesetzt, in Data Mining-Analysen weitere Parameter zu übertragen, um die Analyse weiter zu spezifizieren. Zur generischen Anpassbarkeit ist das Event frei festlegbar implementiert, muss aber eine eindeutige Bezeichnung nutzen. Dies erlaubt es dem Analyse Management System, ohne Anpassung des Ausführungssystems auf beliebige Events zu reagieren. Die Events müssen bei der Ausführung nur von einem externen System ausgelöst werden, damit die Analyse startet.

Der letzte Bestandteil der Analysatorvorlage ist die „action“-Datenstruktur, die - wie alle anderen Strukturteile der Analysatorvorlage - ebenfalls in JSON deklariert ist. Darin sind wie bei der „event“-Struktur die Felder „type“ und „parm“ enthalten. Das „type“-Feld beschreibt mit einem eindeutigen Bezeichner die vom Analysesystem auszuführende Aktion, wenn die Analyseregel eine Übereinstimmung findet. Dieser Bezeichner muss im Parser deklariert sein, um vom Ausführungssystem die notwendige Aktionsdeklaration im Analyse Management System zu laden. Das ist notwendig, da ein externes System bei der Ausführung der Aktion beteiligt sein kann, das eine Datenstruktur und Parameter für die Kommunikation erfordert. Die Parameter für die Aktion sind im Parser hinterlegt, um eine generische Aktion bereitzustellen, die dynamisch über den Analysator angepasst wird oder die von der Schnittstelle erforderliche Struktur enthalten kann. Diese Parameter und die Datenstruktur werden im Analyse Management System unter dem eindeutigen Bezeichner der Aktion deklariert, welcher dann für die jeweilige Aktion in der Analysatorvorlage unter dem „type“-Feld eingetragen wird. Das zweite Feld „parm“ im Aktionsteil wird dafür genutzt, notwendige Parameter für die Aktion zu übertragen und die Aktionsausführung zu parametrisieren. Das ist nötig, um an den Anwender eine spezifische Nachricht zu verschicken, die im Aktionsteil ausgeführt wird. Die Nachricht kann der Benutzer bei der Erstellung in der Analysatorvorlage im „parm“-Feld unter dem Feld „message“ deklarieren. Sie kann beliebigen Text enthalten. Diese Nachricht wird dann über die ausführende Aktion versendet. In der Nachricht können Tokens enthalten sein. Die Tokens werden bei der Ausführung durch das Variablenfeld ersetzt. Tokens und Nachrichten wurden implementiert, um den Nutzern der ECHO-Plattform anwenderspezifische Nachrichten zu senden, die Datenwerte zur Information enthalten können.

4.3.2 Analysatorvorlageübersetzung im Parser

Die Analysatorvorlage beschreibt eine Analyse in maschinenlesbarem Text. Das Ausführungssystem im Analysesystem verwendet jedoch eine im Ablaufplan dargestellte Programmbeschreibung. Die Analysatorvorlage muss im Analyse Management System von JSON als Dateiformat in einen Ablaufplan übersetzt werden. Die Analysatorvorlageübersetzung erzeugt aus einer Analysatorvorlage den Analyseablaufplan, der wiederum auf einem Ausführungssystem aktiviert wird. Alle dargestellten Methoden sind im Analyse Management System integriert, das ebenfalls als Ablaufplan arbeitet.

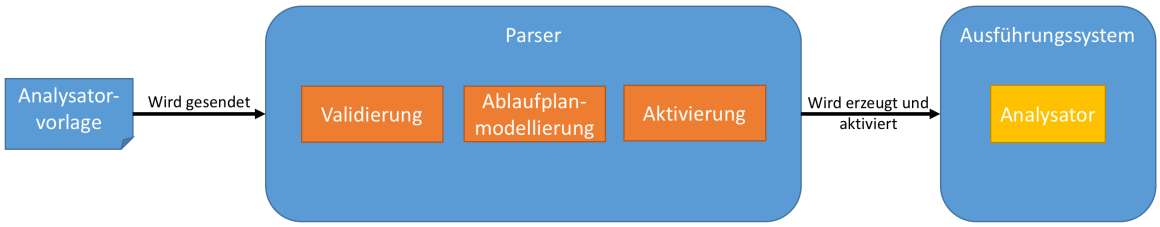

Abbildung 4.3: Ablaufende Arbeitsschritte bei der Aktivierung eines Analysators über die Analysatorvorlage.

Die Analyseflussmodellierung besteht dabei, wie in Abbildung 4.3 zu sehen, aus drei Schritten, die innerhalb des Analyse Management Systems ablaufen, wenn ein neuer Analysator durch eine Analysatorvorlage erzeugt werden soll. Die Analysatorvorlage wird vom Anwender an das Analyse

Management System gesendet. Das Analyse Management System leitet die Analysatorvorlage an den Parser weiter. Im Parser erfolgt dann die Übersetzung der Analysatorvorlage zum Analyseablaufplan. Als erster Schritt der Übersetzung passiert die Validierung der Analysatorvorlage. Im zweiten Schritt folgt die Analysatorvorlageübersetzung der Analyse in das vom Ausführungssystem verwendete Programmformat. Es kommen bei Node-Red JavaScript-Objekte im JSON-Datenformat mit JavaScript-Programmcode zum Einsatz. Der letzte Schritt verwendet den fertigen Ablaufplan und sendet ihn über die Aktualisierung an ein aktives Ausführungssystem, das vom Analyse Management System ausgewählt wurde, und aktiviert den Analysator.

Parser

Der Parser zur Interpretation der Analysatorvorlage ist der zentrale Bestandteil der Übersetzungsarchitektur. Er übernimmt die komplette Übersetzung der Syntax in die für das Ausführungssystem notwendige Programmiersprache bzw. deren Ablaufplan. Der Parser besitzt dabei, wie in Abbildung 4.4 zu sehen, 8 Stufen, die nötig sind, um den Ablaufplan zu erzeugen, und ist ebenfalls im Ausführungssystem implementiert. Die Funktionsblöcke sind in dynamischer, anpassbarer Form innerhalb des Parsers deklariert und bilden deren Funktionalität ab.

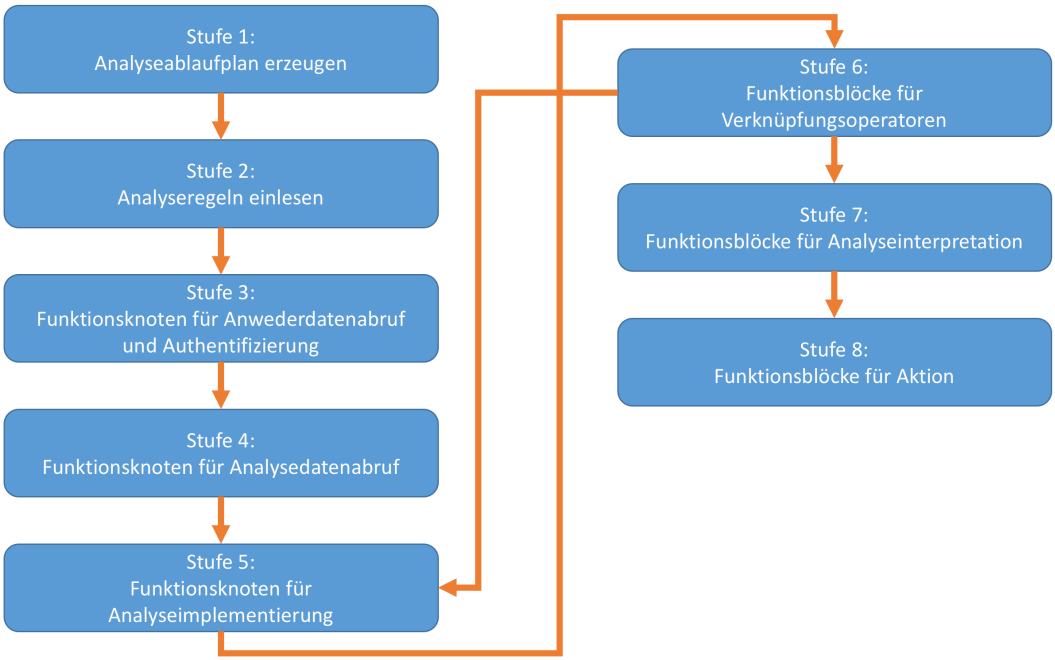

Abbildung 4.4: Die Stufen des Parsers zur Erzeugung des Ablaufplans einer Analyse von einer Analysatorvorlage.

Stufe 1 des Parsers erzeugt für das Ausführungssystem einen neuen Ablaufplan, so dass jede Analyse in einem separaten Abschnitt dargestellt wird. Jede Analyse wird in einem neuen Ablaufplan erzeugt, der eine eindeutige Bezeichnung erhält. Dazu bekommt jeder Analysator seine eigene, eindeutige Identifizierungsmarkierung, die auch für die Referenzierung des Analyse Management Systems

verwendet wird. Mit dieser ID kann das Analyse Management System den Analysator auf Anwenderwunsch aus dem Ausführungssystem löschen. Die ID wird ebenfalls verwendet, um eine Separierung zu ermöglichen, damit die einzelnen Analysen im Ausführungssystem voneinander getrennt sind und leichter von Administratoren betrachtet werden können. In diesem Schritt wird zudem der Startknoten erzeugt, der beim Eintreffen des Events, auf das der Analysesators reagieren soll, den Ablaufplan startet. Dieser Knoten ermöglicht es, eventgesteuert die einzelnen Analysatoren vom zentralen Analyse Management Systemes zu starten.

Stufe 2 liest die Analysatorvorlage ein und erzeugt aus der Analysatorvorlage ein JavaScript-Objekt, das Node-Red interpretieren kann. Hierzu wird in Node-Red die Methode **JSON.parse()** von Nodejs verwendet, um JSON in ein JavaScript-Objekt zu konvertieren. Das hat den Vorteil, dass der JSON-Text auf Syntaxfehler geprüft wird. Außerdem wird die Funktion direkt von der Bibliothek (Nodejs) bereitgestellt und benötigt keine eigene Programmierung.

In der **Stufe 3** werden die Funktionsknoten für den Abruf der Anwendungsdaten und die Authentifizierung des Zugriffs aus einer externen Quelle abgebildet. Anwendungsdaten beschreiben dabei die Patienten-ID, mit welcher der Ablaufplan später ausgeführt werden soll. Die Patienten-ID folgt dabei aus dem Datensatz, der erzeugt wurde. Der Funktionsblock extrahiert aus der Zeichenfolge die Parameter mit dem Separator „/“. Die Zeichenfolge enthält die Patienten-ID, das Datensatz-Datenmodell und die Datensatz-ID. Die IDs werden aus der folgenden übertragenen Zeichenkettenstruktur extrahiert:

/<Patienten-ID>/<Datensatz-Datenmodell>/<Datensatz-ID>

Die Patienten-ID beschreibt die ID, mit welcher der Datensatz erzeugt wurde. Sie wird benötigt, um den Arzt zu ermitteln, der dem Patienten zugeordnet ist. Dazu werden Funktionsblöcke angelegt, die über die Patienten-ID die Arzt-ID über einen REST-API-Aufruf zur ECHO-Plattform ermitteln. Die Arzt-ID ist erforderlich, um später die Daten des Patienten für die Analyse zu erhalten. Das Datensatz-Datenmodell beschreibt den Typ des Datensatzes, der erzeugt oder verändert wurde. Die Datensatz-ID beschreibt die ID, mit welcher der Datensatz in der Datenbank gespeichert wurde.
Die Authentifizierungknoten, die ebenfalls in Stufe 3 erzeugt werden, verwenden die Patienten-ID und Arzt-ID zur Erzeugung der Authentifizierungstokens. Im Funktionsblock werden aus den IDs mit dem geheimen Schlüssel die Authentifizierungstokens erzeugt. Der geheime Schlüssel muss vorher in der Analyse Management System-Konfiguration hinterlegt werden. Die Authentifizierungstokens werden verwendet, um mit der ECHO-Plattform zu kommunizieren. Das Authentifizierungstoken besteht aus einer Zeichenfolge, mit der sich das Analyse Management System als spezifischer Benutzer authentifiziert. In Stufe 3 werden Authentifizierungstokens für den Arzt und für den Patienten erzeugt. Das Patient-Authentifizierungstoken wird benötigt für den Abruf der Arzt-ID und die Versendung von Benachrichtigungen. Das Arzt-Authentifizierungstoken wird für den Abruf der Analysedaten verwendet.

In **Stufe 4** werden die Funktionsblöcke des Ablaufplans zum Abruf der Analysedaten erzeugt, die vorher in der Regelsyntax als Datenfeld festgelegt wurde. Die Daten für die Analyse werden aus einer externen Quelle abgerufen. Dazu wird vom Parser ein REST-API Abrufknoten erzeugt, der über eine Webschnittstelle die Datensätze für die Analyse von der ECHO-Plattform aufruft. Der REST-API-Abrufknoten wird als fertiger Funktionsbaustein von Node-Red bereitgestellt, der einen

Webaufruf ausführt. Der Webaufruf verwendet bei HTTP GET als REST-Befehl. Mit GET wird eine Ressource abgerufen. Die Ressource enthält den Datensatz. Die Quelle und der Adapter für die Transformation ist im Analyse Management System hinterlegt. Das Datenmodell gibt das Datenformat, das Übertragungsprotokoll und den Endpunkt der externen Quelle an, die - wie in Sektion 3.3 beschrieben ist - automatisiert aus den externen Quellen ausgelesen wird. Im Fall der ECHO-Plattform wird ein REST-API-Aufruf gestartet, der JSON als Datenformat und das Übertragungsprotokoll HTTP verwendet. Abhängig vom verwendeten Adapter kann auch ein anderes Datenformat, eine andere Quelle und ein anderes Übertragungsprotokoll eingesetzt werden. Im Ablaufplan sind ebenfalls Funktionsknoten für die Aufbereitung der Datenanalyse enthalten, die notwendig sind, um die verwendete Datenanalyse mit Daten zu versorgen. Das wird über einen Adapter im Analyse Management System ermöglicht, der das JSON-Datenformat in ein JavaScript-Objekt transformiert, das die Datensätze enthält, die über den REST-API-Aufruf empfangen wurden. Die Transformation der externen Daten des Datenabrufs erfolgt dabei immer zur Laufzeit des Analysators, nachdem die notwendigen Daten für die Analyse abgerufen wurden.

Die **Stufe 5** implementiert im Ablaufplan den Funktionsblock für die eigentliche Datenanalyse. Dazu sind im Parser die Analyseoperatoren deklariert und deren Funktion beschrieben, die das System zur Verfügung stellt. Die Datenanalyse ist dabei als JavaScript-Programmcode umgesetzt, der aus der Regelsyntax generiert wird. Nur bekannte Analyseoperatoren können verwendet werden, welche im Analyse Management System deklariert wurden. Jede Regel in der Regelsyntax wird als Analysefunktionsbaustein erzeugt. Existieren zum Beispiel drei Regeln in der Regelsyntax, werden drei Analysefunktionsbausteine erzeugt. Die Analysefunktionsbausteine werden mit den in der Regelsyntax deklarierten Verknüpfungsoperatoren verbunden. In den Analysefunktionsbausteinen werden vom Parser der in der Regelsyntax deklarierte Analyseoperator, das Datenfeld und der Vergleichswert hinterlegt und daraus JavaScript-Programmcode für die Ausführung erzeugt. Der JavaScript-Programmcode ist dabei im Analyse Management System hinterlegt, der dynamisch mit dem Datenfeld und dem Vergleichswert eingefügt wird. So wird dynamisch für jede Analyse der passende Analysefunktionsbaustein erzeugt, der die spezifischen Werte der Analysatorvorlage enthält. Der Funktionsblock für die Analyse liefert nach Durchlauf des JavaScript-Programmcodes abhängig vom Analyseergebnis den booleschen Wert. Ist die Analyseregel erfüllt, liefert der Analysefunktionsbaustein das Ergebnis „Wahr“, falls die Regel nicht erfüllt ist „Falsch“.

In der **Stufe 6** werden die Funktionsblöcke für die Verknüpfung der einzelnen Analyseoperatoren hinzugefügt, wenn in der Regelsyntax mehr als eine Analyseregel deklariert wurde. Ist nur eine Regel vorhanden, wird der Verknüpfungsbaustein durch den Parser übersprungen. Der Verknüpfungsbaustein repräsentiert dabei den Verknüpfungsoperator. Sind mehrere Regeln vorhanden, wird der in der Regelsyntax beschriebene Verknüpfungsoperator mit dem Analysebaustein im Ablaufplan verknüpft. Dazu wird der Analysefunktionsbaustein mit dem nächsten Analysefunktionsbaustein mit dem Verknüpfungsoperator verknüpft. Der Verknüpfungsbaustein extrahiert dabei aus der Datenausgabe der Analysebausteine die Analyseergebnisse. Alle booleschen Werte werden mit dem Verknüpfungsoperator verknüpft, so dass nach allen Analyse- und Verknüpfungsbausteinen das Endergebnis als einzelner boolescher Wert zurückgeliefert wird. Die Funktionsweise des Verknüpfungsbausteins wird durch die Regelsyntax mit „AND“ oder „OR“ dargestellt und als einfache boolesche Verknüpfung der Werte der Analysebausteine und mit dem Verknpüfungsoperator verwendet. Ist der Verknüpfungsbaustein im Ablaufplan integriert, springt der Parser zurück zur Stufe 5, um den nächsten Analysebaustein zu

erzeugen. Ist der letzte Analysebaustein gesetzt, und es folgt keine weitere Regel in der Regelsyntax, dann folgt die nächste Stufe des Parsers.

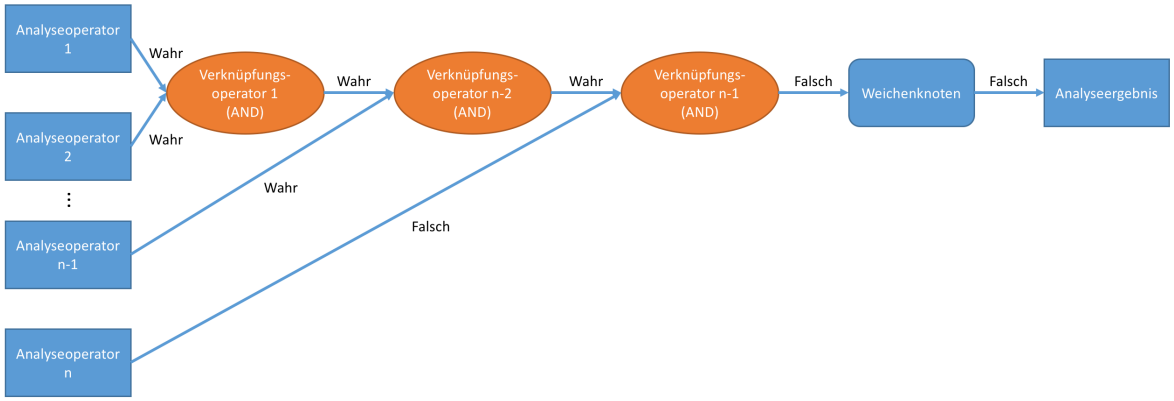

Abbildung 4.5: Berechnung des Analysergebnisses aus den Einzelergebnissen der Analysebausteine.

Die **Stufe 7** erzeugt den Funktionsbaustein für die Ergebnisinterpretation. Die Abbildung 4.5 zeigt beispielhaft die Berechnung des Analyseendergebnisses aus allen einzelnen Analyseergebnissen der Analysefunktionsbausteine (Analyseoperatoren), die für die Ergebnisinterpretation notwendig sind. Die Verknüpfung der Einzelergebnisse erfolgt durch die Verknüpfungsoperatoren, die in der Darstellung mit „AND“ verknüpft werden. Die Analyse- und Verknüpfungsoperatoren sind in der Regelsyntax beschrieben. Der Weichenknoten führt die Interpretation des Ergebnisses der Analyse durch, indem er - wie auf der Abbildung 4.5 zu sehen - das Ergebnis der Verknüpfungsoperatoren mit dem booleschen Wert „Wahr“ prüft. Das Ergebnis folgt aus den einzelnen Analysefunktionsbausteinen im Ablaufplan. Die Analysebausteine liefern einen booleschen Wert, welcher für die Ergebnisinterpretation verwendet wird. Ist die Regel in einem Analysefunktionsbaustein erfüllt, liefert dieser den booleschen Wert „Wahr“, sonst „Falsch“. Die Weitergabe des Werts erfolgt in Node-Red über die Nachricht als JavaScript-Objekt. Der Wert wird im Feld „state“ des Objekts übertragen. Die Ergebnisse der Analysebausteine werden verknüpft mit den Verknüpfungsoperatoren. Der Verknüpfungsoperator liefert nach der Verknüpfung als Endergebnis ebenfalls einen booleschen Wert, der vom Funktionsbaustein für die Ergebnisinterpretation verwendet wird. Dazu erzeugt der Parser einen Weichenknoten. Der Weichenknoten ist ein fertiger Funktionsbaustein von Node-Red, der den Wert eines Feldes im JavaScript-Objekt mit einem hinterlegten Wert prüft. Jeder Wert für die Prüfung erhält einen eigenen Ausgang. Der Weichenknoten sendet bei Übereinstimmung die Nachricht (JavaScript Objekt) an den jeweiligen Ausgang. Der Endwert des Verknüpfungsoperatoren wird im Weichenknoten mit dem „Wahr“-Wert verglichen, indem der Vergleichswert mit dem booleschen Wert „Wahr“ geprüft wird. Eine Weiterleitung erfolgt, wenn das Endergebnis ein „Wahr“ liefert. Ergibt die boolesche Verknüpfung aller Analysefunktionsbausteine den Wert „Wahr“, sind alle Regeln erfüllt und der Wert wird vom Weichenknoten akzeptiert. Die Nachricht mit dem Analyseergebnis wird dann vom Weichenknoten an die Funktionsbausteine für die Aktion weitergeleitet. Wird ein „Falsch“ als Ergebnis erkannt, wird der Ablaufplan beendet. Als weitere Alternative kann der Weichenknoten auch einen beliebigen Wert, der bei der Analyse deklariert wurde, mit einem in der Regelsyntax spezifizierten Wert vergleichen. Er leitet dann ebenfalls die Nachricht an den passenden Ausgang weiter, wenn die Bedingung übereinstimmt. Das erfolgt jedoch nur, wenn der Analyseoperator im Analyse Management System entsprechend programmiert wurde.

Die letzte **Stufe 8** im Parser erzeugt die Aktionsknoten für eine Benachrichtigung oder Ausführung von Aktionen, die in der Analysatorvorlage im Aktionsteil beschrieben wurde. Dazu wertet der Parser die eindeutige Bezeichnung der Aktion aus, wählt die im Parser deklarierte Aktion und erstellt die dafür notwendigen Funktionsbausteine, die die Aktion repräsentieren. Der Programmcode für die Funktionsbausteine ist im Analyse Management System hinterlegt. Zur Erkennung der Aktion wertet der Parser die Parameter im Aktionsteil der Analysatorvorlage aus. Er kann dynamisch anpassbare Aktionsbausteine erzeugen, die spezifische Nachrichten oder parametrierende Aktionen enthalten. Ebenfalls vom Parser wird ein Baustein erzeugt, der im Text vorhandene Tokens mit Datenwerten ersetzt. Die verfügbaren Tokens sind dabei im Parser spezifiziert und werden im Funktionsbaustein hinterlegt. Nach Abschluss der Stufe 8 ist die Analysatorvorlage komplett in einer vom Ausführungssystem interpretierbaren Form implementiert.

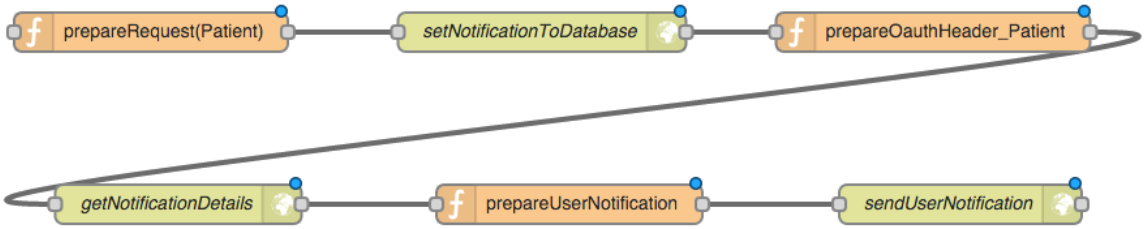

Abbildung 4.6: Funktionsbausteine für die Ausführung der Aktion zum Versenden einer Nachricht und Speicherung der Nachricht in der Datenbank.

Die Abbildung 4.6 zeigt die Funktionsbausteine für die Ausführung der Aktion zum Speichern der Benachrichtigung in der Datenbank und Versenden der Benachrichtigung an den Patienten. Die Aktion besteht aus dem Funktionsbaustein „prepareRequest“ zur Erzeugung der Benachrichtigung für die Speicherung der Nachricht in der Datenbank. Dabei wird das JavaScript-Objekt per „JSON.stringify()“-Methode in das JSON-Format konvertiert und an den nächsten Funktionsbaustein übertragen. Der Funktionsbaustein „setNotificationToDatabase“ ruft die REST-API der ECHO-Plattform und überträgt die Benachrichtigung im HTTP Body für die Speicherung. Zur Authentifizierung werden die in Stufe 3 erzeugten Authentifikationstokens verwendet. Der Funktionsbaustein „prepareOauthHeader_Patient“ erzeugt den HTTP Header für die Kommunikation mit der Webschnittelle der ECHO-Plattform. Dazu werden die Authentifikationstokens im Header hinterlegt. Der HTTP Header wird vom „getNotificationDetails“ verwendet, um über die ECHO-Plattform die Benachrichtigungseinstellungen des Patienten abzurufen. Die Benachrichtigungseinstellungen sind notwendig und werden nicht statisch hinterlegt, da der Patient diese jederzeit ändern kann und die Änderung nicht an das Analyse Management System weitergeleitet wird. Dafür ruft der Ablaufplan immer bei der Ausführung der Aktion die aktuellen Benachrichtigungseinstellungen ab. Die Benachrichtigungseinstellungen enthalten unter anderem den Benachrichtigungstyp. Sie werden verwendet, um die Benachrichtigung zu konfigurieren. Der Funktionsbaustein „prepareUserNotification“ nutzt die Benachrichtigungseinstellungen für die Konfiguration der Aktion und der Nachricht. Die in der Nachricht enthaltenen Tokens werden mit den Datenwerten der Analyse ersetzt. Dies erfolgt durch das Durchsuchen des Textes nach den Token-Bezeichnern, die im Analyse Management System hinterlegt sind. Wird ein Token gefunden, wird das im Analyse Management System festgelegte Feld des Ergebnisses verwendet, um das Token zu ersetzen. Je nach ausgewähltem Benachrichtigungstyp wird der im Analyse Management System

hinterlegte Endpunkt für die Ausführung der Aktion genutzt. Endpunkt bedeutet in diesem Kontext die URL der Webschnittstelle zur Ausführung der Aktion. Die Ausführung des Versands der Nachricht erfolgt dann über den Funktionsbaustein „sendUserNotification“, welcher einen HTTP-POST-Aufruf an die Schnittstelle der Aktion sendet.

Validierung

Die Validierung erfolgt als erstes, wenn eine Analysatorvorlage an das Analyse Management System gesendet wird, um die Korrektheit der Vorlage zu prüfen. Nur von korrekten Vorlagen kann ein Ablaufplan erzeugt werden, weil der Parser nur aus korrekten Vorlagen die notwendigen Bausteine und Verbindungen im Ablaufplan zusammensetzen kann. Er bricht sonst mit einer Fehlermeldung ab. Dazu wird erst die Analysatorvorlage validiert, inwieweit alle Bestandteile in der korrekten syntaktischen und semantischen Syntax dargestellt werden und alle Optionen korrekt gesetzt sind. Die Abbildung 4.7 zeigt die drei Stufen der Validierung der Vorlage mit der Prüfung der JSON-Syntax, prüft die erforderlichen Bestandteile der Analysatorvorlage wie Event, Aktion und Regelsyntax, und prüft zum Schluss die verwendeten Analyseoperatoren, das Datenmodell und deren Datentypen.

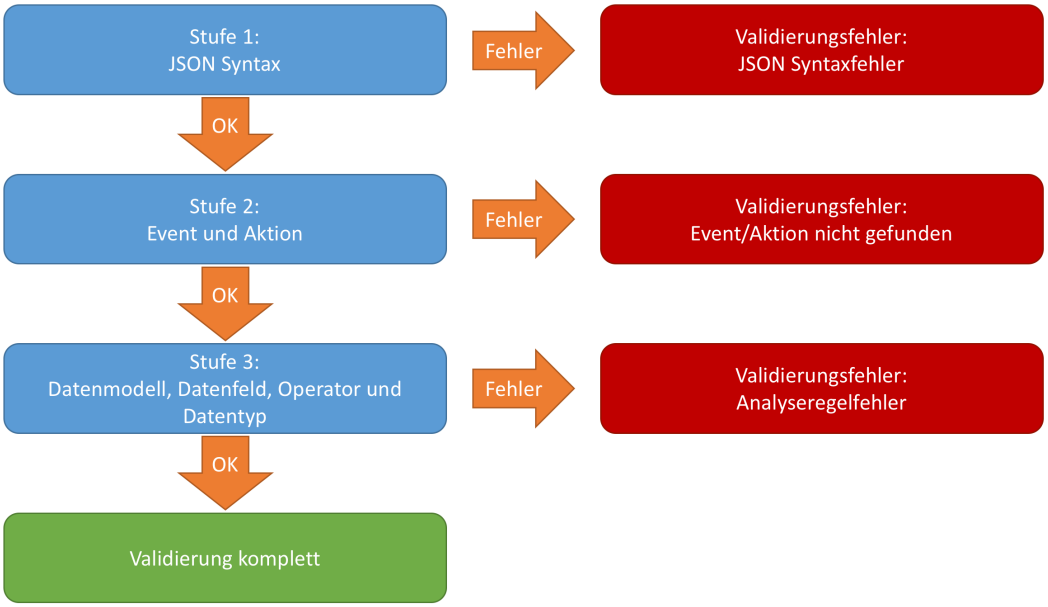

Abbildung 4.7: Die ablaufenden Validierungsschritte zur Überprüfung von Analysatorvorlage und Fehlerausgaben.

Die **erste Stufe** überprüft den Container mit einem JSON-Schema, das den konkreten Aufbau der Analysatorvorlage beschreibt. Das JSON-Schema wird nur für den korrekten Aufbau der Analysatorvorlage verwendet, weil sich die Datenquelle und Datentypen sowie die verfügbaren Aktionen und Events nicht mit einer JSON-Schema-Prüfung validieren lassen, die vom System dynamisch erzeugt und deklariert wird. Somit lässt sich kein statisches JSON-Schema hinterlegen. Das JSON-Schema validiert nur die JSON-Syntax, den korrekten Aufbau der Analysatorvorlage und ob es sich bei der

Analysatorvorlage um ein JSON-Format handelt, nicht jedoch die Zusammensetzung der Datenanalyse.
In der **zweiten Stufe**, nach erfolgter Validierung des JSON-Containers, kann dessen Inhalt ausgelesen werden. Dann erfolgt die Prüfung der in der Analysatorvorlage definierten Events und Aktionen. Das Event und die Aktion werden mit den im System verfügbaren Events und Aktionen abgeglichen. Ist das nicht der Fall, wird mit einer Fehlermeldung abgebrochen. Ist das deklarierte Event und die Aktion vorhanden, kann die Validierung die nächste Stufe aktivieren. Eine Validierung ist notwendig, da der Anwender beliebige Events und Aktionen in die Analysatorvorlage eintragen kann, das System aber nur die im System vorhandenen Events und Aktionen interpretieren kann. Aus diesem Grund sind Aktionen und Events auch eindeutig benannt.
In der **dritten Stufe** erfolgt die Prüfung der mit der Regelsyntax deklarierten Analyseregeln. Dazu wird jede Regel daraufhin überprüft, ob das Datenfeld, welches analysiert werden soll, vorhanden ist. Dazu wird vom Analyse Management System das Datenmodell der vorhandenen Datenstruktur ausgelesen und überprüft, ob dieses existiert. Die Extraktion der Datenmodelle erfolgt über den in der Regelsyntax festgelegten Adapter. Über den Adapter im Analyse Management System werden die Datenmodelle transformiert und aufbereitet, sodass die Validierungskomponente eine Interpretation durchführen kann. Gibt es eine Übereinstimmung, wurde das gewählte Datenmodell im System gefunden. Danach wird die zweite Komponente des Datenfelds geprüft, das den Pfad zum Datenwert deklariert. Diese wird ebenfalls im Datenmodell auf Existenz geprüft: durch Vergleich des im Datenmodell angegebenen Datentyps mit dem des Vergleichswerts. Dazu wird der Datentyp des Vergleichswerts ermittelt und mit der Angabe des Datentyps im Datenmodell verglichen. Stimmen beide überein, sind die Datentypen von Datenfeld und Vergleichswert kompatibel. Im Anschluss folgt noch die Prüfung der angegebenen Operatoren, ob diese im Analyse Management System vorhanden sind und genutzt werden können. Ein Fehler in der Validierung erzeugt einen direkten Abbruch der Validierung und der Analyseflussmodellierung. Ist die Validierung erfolgreich abgeschlossen, kann die Erzeugung des Analyseablaufplans beginnen.

Übersetzung

In der Übersetzungphase wird aus der Analysatorvorlage ein Analyseablaufplan erzeugt, der mit der Programmsprache des Ausführungssystems umgesetzt wird. Bei der Verwendung von Node-Red erfolgt dies als JavaScript-Objekt in JSON mit JavaScript Programmcode. Zur Anwendung kommt ein Parser, der die Vorlage mit einem Iterationsverfahren durchläuft und den Analyseablaufplan mit Modulbausteinen aufbaut. Der Parser iteriert somit über die Analyseregeln und erzeugt für jede Regel die notwendigen Funktionsbausteine. Der Ablaufplan besteht dabei aus Funktionsbausteinen, die eine Funktionalität repräsentieren, und Verbindungen zwischen den Bausteinen, die den Kontroll- und Datenfluss repräsentieren, der im Ablaufplan erfolgt. Bei Node-Red erfolgt der Kontroll- und Datenfluss über JavaScript-Objekte. Die Objekte werden als Nachricht zwischen den Funktionsblöcken übertragen. Je nach Auswahl des Ausgangs des Funktionsblocks folgt der Ablaufplan dem Kontrollfluss, indem die Nachricht über den ausgewählten Ausgang übertragen wird. Der Analyseablaufplan stellt dabei die Eigenschaft dar, die in der Analysatorvorlage definiert ist. Die Eigenschaften werden mit Funktionsblöcken und Übergangsbedingungen der Ausgänge des Funktionsblocks beschrieben, sodass die zuvor definierte Datenanalyse und deren Analysemethode als Ablaufplan dargestellt werden können.

Die Abbildung 4.8 zeigt die Darstellung eines Analysators im Ausführungssystem. Die Bausteine in der Abbildung 4.8 sind dabei abstrahiert dargestellt und können im Ausführungssystem aus mehreren Funktionsbausteinen bestehen. Die Funktionsbausteine sind in einzelne Blöcke zusammengefasst, um die Abgrenzung zwischen den einzelnen Funktionen des Ablaufplans darzustellen. Der Ablaufplan besteht dabei aus einem Startknoten, der für die Identifizierung des Analysators sowie zum Start des Ablaufplans dient. Der Startknoten startet beim Eintreffen des Events, worauf er programmiert wurde. Dazu wird das im Analysatorvorlage hinterlegte Event ebenfalls im Startknoten hinterlegt und startet nur bei diesem Event. Für jedes am Startknoten empfangene Event wird der Ablaufplan gestartet. Der Ablaufplan ist dabei vollständig unabhängig von vorherigen oder nachfolgenden Starts des Ablaufplans. Das bedeutet: Der Ablaufplan läuft immer in einer eigenen Instanz, bis er beendet wurde. Nach dem Startknoten folgen die Knoten zur Extraktion der Anfragedaten. Anfragedaten sind dabei die im Event enthaltenen Informationen. Dabei handelt es sich um die Datensatz-ID und die Benutzer-ID des Datensatzes. Die IDs werden benötigt, um anwenderspezifisch die Analyse zu parametrisieren.

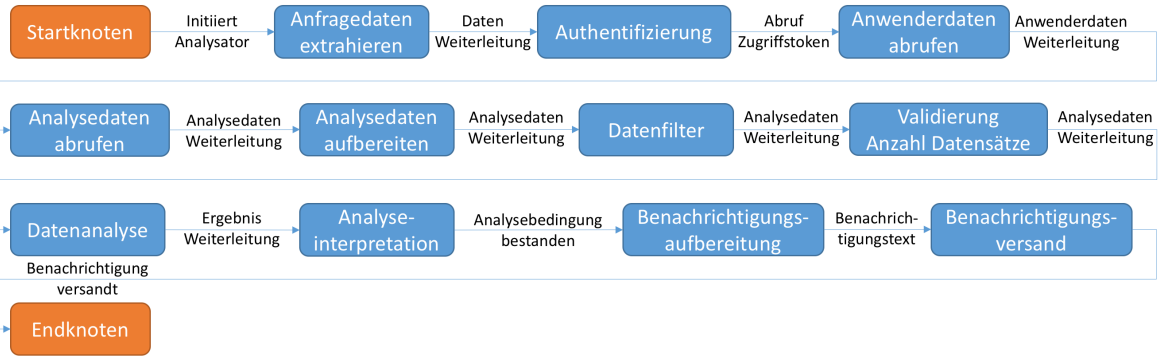

Abbildung 4.8: Ausführungsdarstellung eines Analysators im Ausführungssytem.

Bei der Analyse mit der ECHO-Plattform werden durch die Parameter die Patienten-ID und Arzt-ID übertragen. Die IDs werden benötigt, um Authentifizierungstokens zu generieren. Die Authentifizierungstokens bestehen aus einer Zeichenfolge und dienen beim Abruf von Daten zur Authentifizierung an der Webschnittstelle. Die IDs werden an den Authentifizierungsknoten weitergeleitet, der die Authentifizierung gegenüber dem externen System vornimmt, um auf zugriffsgeschützte Daten zuzugreifen und weitere Parameter für die Analyse zu laden. Im Fall der ECHO-Plattform werden die Authentifikationstokens für Patient und Arzt generiert. Die Authentifizierungstokens werden für den weiteren Abruf von personalisierten Daten gespeichert und nicht immer neu generiert. Die Speicherung erfolgt dabei im Anwenderdatenabruf-Knoten. Die Tokens werden ebenfalls im Knoten Analysedatenabruf verwendet, welcher die für die Analyse notwendigen Daten von externen Quellen abruft. Der Abruf der Analysedaten kann in der ECHO-Plattform nur mit autorisiertem Zugriff erfolgen. Die abgerufenen Daten werden in den Knoten für die Aufbereitung der Analysedaten weitergeleitet, passend zur festgelegten Analyse. Der Knoten Analysedatenaufbereitung bereitet die Daten auf, transformiert die Daten in ein für die Analyse verwendbares Format und testet, ob die erforderlichen Daten vorhanden sind. Dies ist erforderlich, da die Daten aus externen Quellen nicht in passender Form verfügbar sind oder vorherige Berechnungen durchgeführt werden müssen. Bei der

ECHO-Plattform werden immer die letzten Datensätze abgefragt oder diejenigen, welche über eine ID oder einen Wertebereich spezifiziert werden. Die Implementierung des Abrufs, welcher Datensatz ausgeliefert wird, ist abhängig von der Schnittstelle der externen Datenquelle.

Nach der Aufbereitung werden die Analysedaten an den Datenfilter weitergeleitet, wenn dieser in der Analysatorvorlage definiert wurde. Ist ein Filter nicht vorhanden, verfügt der Ablaufplan auch nicht über einen Filterknoten. Der Datenfilterknoten filtert aus den aufbereiteten Analysedaten diejenigen aus, die in der Filterbedingung keine Übereinstimmung mit der in der Analysatorvorlage enthaltenen Bedingung liefern. Dazu wird der im Filter spezifizierte Datenwert mit dem Wert im Datensatz verglichen. Ist die Bedingung nicht erfüllt, wird der Datensatz aus der Menge gelöscht. Damit lässt sich die Menge an Datensätzen vor der Datenanalyse von nicht gewünschten Elementen bereinigen, bevor die Datensätze vom Analysealgorithmus bearbeitet werden. Mit der Regelsyntax lässt sich die Anzahl an Datensätzen, die analysiert werden sollen, bestimmen. Dieser Parameter wird für die Abfrage der Daten aus den externen Quellen verwendet. Nach dem Abruf der Daten wird die Anzahl der Datensätze geprüft. Dies erfolgt über den Baustein zur Validierung der Anzahl von Datensätzen, der bei der Übersetzung des Ablaufplans eingesetzt wird. Der Baustein überprüft die Anzahl an Datensätzen mit dem in der Analysatorvorlage angegeben Grenzwert. Nur wenn die Anzahl übereinstimmt, wird mit der Analyse begonnen. Nach der Aufbereitung und Filterung der Datensätze erfolgt die Analyse, die für den Analysator konfiguriert wurde. Dazu beinhaltet der Ablaufplan einen Baustein mit der Bezeichnung Datenanalyse, der die eigentliche Datenanalyse enthält und mit allen Datensätzen in der Datensatzmenge durchlaufen wird. Eine erfolgreiche Analyse ist nur möglich, wenn alle Datensätze in der Analysedatenmenge die Analysekriterien erfüllen. Die Analysedatenmenge ist die Menge an Datensätzen, die abgerufen und analysiert wurden. Die Analysekriterien sind vorher in der Regelbeschreibung deklariert worden. Der Analysebaustein liefert „Wahr“ oder „Falsch“ als booleschen Wert zurück - oder einen Datenwert sowie die im Algorithmus berechneten Parameter als Ergebnisobjekte, die abhängig vom implementierten Algorithmus sind und beliebig erweitert werden können. Das Analyseverfahren, welches zum Einsatz kommen soll, kann aus einem völlig beliebigen Datenanalysealgorithmus bestehen, der vorher eindeutig gekennzeichnet und im Analyse Management System einprogrammiert wurde. Es ist auch möglich, Bibliotheken einzubinden oder die Analyse über einen Aufruf auf einem externen, völlig unabhängigen System zu starten und den Kontrollfluss so lange zu blockieren, bis das externe System das Ergebnis der Datenanalyse zurückliefert. Die Prüfung, ob die Datensätze den Analysekriterien entsprechen, erfolgt bei der Analyseinterpretation und ist nicht im Datenanalysebaustein integriert. Der Datenanalyseknoten liefert einen booleschen Wert oder einen Datenwert zum Interpretationsknoten, der als Kontrollstruktur dient und die zurückgelieferten Werte mit den Referenzwerten im Interpretationsknoten vergleicht. Ist die Bedingung falsch, endet der Analyseablauf am Interpretationsknoten. Stimmen die Bedingungen überein, erfolgt die Freigabe des Kontrollflusses für den nächsten Knoten.

Entspricht das Ergebnis der Interpretation der Analyseregeln, dann wird die in der Analysatorvorlage festgelegte Aktion ausgeführt. In der aktuellen Implementation sieht die Plattform den Versand von Benachrichtigungen über eine externes Schnittstelle vor. Die Benachrichtigungen werden dabei ebenfalls in einer Datenbank gespeichert. Andere Aktionen sind im Analyse Management System nicht hinterlegt. Aktionen können jederzeit durch das Analyse Management System erweitert werden. Für die Aktionen enthält jeder Analyseablaufplan Funktionsblöcke für die Benachrichtigungsaufbereitung, die die Aktion für die Ausführung vorbereitet, indem die notwendigen Anwenderdaten aus externen

Quellen abgerufen werden. Die Parameter passen die Aktionsvorbereitung zur Laufzeit mit anwenderspezifischen Eigenschaften an. Darunter fällt auch die Anpassung der Benachrichtigungsart, die vom Anwender zur Laufzeit bestimmt wird und die die notwendigen Parameter für die Kommunikation mit den externen Benachrichtigungssystemen enthält. Ebenfalls werden in der Vorbereitungsphase eventuelle Tokens im Benachrichtigungstext, der in der Analysatorvorlage festgelegt wurde, mit den aus der Analyse erzeugten Daten ausgetauscht, um eine dynamische Antwort mit aus der Analyse entstehenden Daten zu generieren. Ist die Aufbereitung der Aktion erfolgt, folgt die Ausführung der Aktion mit den vorher aufbereiteten Parametern. Hierzu kann man über die Funktionsblöcke jede Art von externen REST-Aufrufen starten - zur Auslösung der Aktion, die im Analyse Management System hinterlegt sind. Nach der Ausführung der Aktion wird das System mit dem Endknoten beendet. Alle zur Laufzeit anfallenden Daten werden nach dem Beenden der Instanziierung des Ablaufplans verworfen. Es erfolgt somit keine persistente Speicherung von zur Laufzeit anfallenden Daten innerhalb des Analyseablaufplans. Daten können nur durch die auszuführende Aktion auf externe Systeme übertragen werden, jedoch nicht innerhalb der Instanz, welche nach der Ausführung beendet wird.

4.3.3 Aktivierung

Die Aktivierung ist der letzte Schritt bei der Erzeugung eines neuen Analysators aus einer Analysatorvorlage und erfolgt durch die Aktualisierungskomponente im Parser des Analyse Management Systems. Bei der Aktualisierungskomponente handelt es sich um einen Ablaufplan, der die Konfiguration des Analysators als Ablaufplan an die Webschnittstelle **/flows** per HTTP POST sendet. Zur Übertragung der Konfiguration wird JSON eingesetzt. Bis zur Aktivierung wurde nur in der vom Ausführungssystem verwendeten Sprache das Ablaufsystem des Analysators generiert, ohne dass der Ablaufplan auf dem Ausführungssystem gestartet wurde. Das bedeutet: Der Ablaufplan wurde im flüchtigen Zwischenspeicher erzeugt und nicht persistent zwischengespeichert. Das erfolgt erst bei der Aktivierung durch das Analyse Management System in einem Ausführungssystem, das den persistenten Speicher zur Verfügung stellt. Das ist dadurch bedingt, dass das Analyse Management System auch als Ablaufplan aufgebaut ist. Das Analyse Management System wird durch den Aufruf zur Analysatorerzeugung neu instanziiert und nach der Generierung des Ablaufplans wieder komplett verworfen. Somit erfolgt die Aktivierung der Analyse erst nach dem Aktualisieren des Ausführungssystems.

Dieser Vorgang erfolgt durch die Aktualisierungskomponente, in dem der vom Parser erzeugte Ablaufplan im Ausführungssystem aktiviert wird. Das Analysesystem verfügt dabei über eine Multi-Serverarchitektur, die es ermöglicht, die Analysatoren auf verschiedenen Ausführungssystemen zu aktivieren, indem es eines aus der Liste der aktiven Systeme wählt. Das Analyse Management System führt eine Ermittlung des Zielsystems für die Aktivierung des Analysators durch. Dazu wählte das Analyse Management System zufällig ein Ausführungssystem, das aktiv ist und zur Verfügung steht, um eine breitgefächerte Verteilung der Analysatoren auf unterschiedliche Ausführungssysteme zu erreichen. Die zufällige Auswahl verbessert die Performance und erlaubt eine Skalierung, indem die Analysatoren zufällig auf den Ausführungssystemen verteilt werden. Das Ausführungssystem wird per folgender SQL-Abfrage ermittelt:

```
SELECT * FROM analyserHost WHERE enabled = '1' ORDER BY RAND() LIMIT 1;
```

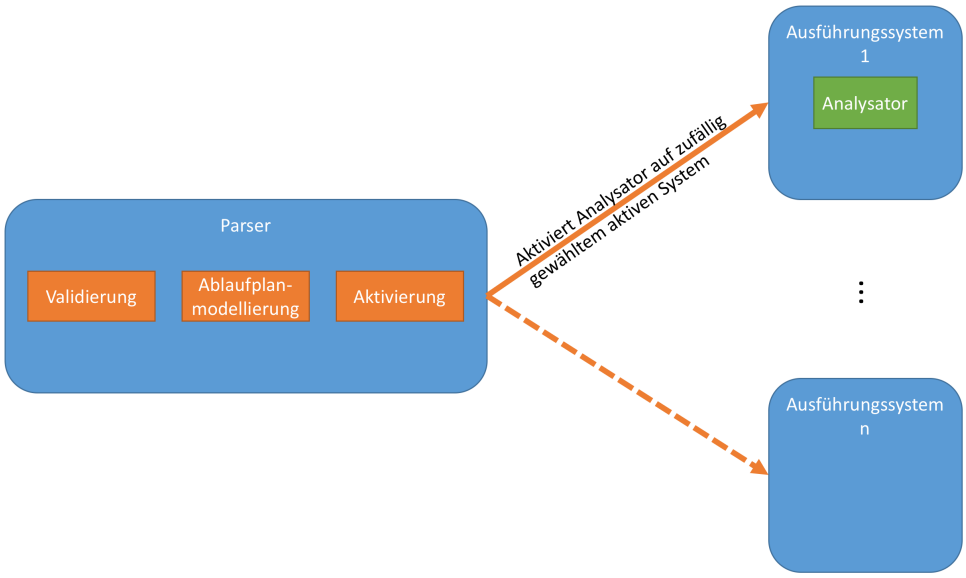

Abbildung 4.9: Aktivierung des Analysators durch das Analyse Management System mit der zufälligen Auswahl eines Ausführungssystems.

Dabei wird aus der Tabelle **analyserHost**, welche alle Ausführungssysteme enthält, zufällig ein System gewählt, das aktiv ist. Die MySQL-Datenbank sortiert dabei zufällig die Einträge der aktiven Ausführungssysteme und wählt den obersten in der Tabelle. Die Abbildung 4.9 zeigt die Auswahl des Zielsystems, welches für den zu aktivierenden Analysator genutzt werden soll. Das primäre System ist dabei immer das Ausführungssystem, welches das Analyse Management System enthält, was aber auch für die Aktivierung von Analysen genutzt werden kann. Somit lässt sich das Analysesystem mit einem Ausführungssystem starten. Das Analyse Management System sendet nach der Auswahl des Ausführungssystems den in JSON transformierten Ablaufplan. Für die Aktivierung wird nicht nur der transformierte Ablaufplan verwendet. Die Webschnittstelle zur Aktualisierung der Konfiguration von Node-Red benötigt nicht nur den neuen Ablaufplan als JSON, sondern erfordert die komplette Konfiguration inklusive aller bereits aktiven Ablaufpläne. Dazu wird bei der Aktivierung des Analysators die aktuelle Konfiguration des Ausführungssystems per HTTP GET als JSON abgefragt. Der bereits in JSON transformierte Ablaufplan wird der aktuellen Konfiguration hinzugefügt und die neue Konfiguration mit dem Analysator an das Ausführungssystem gesendet. Vom Ausführungssystem werden die Änderungen der Konfiguration ermittelt und alle geänderten Ablaufpläne beendet, aktualisiert und neu gestartet. Nach Abschluss der Aktualisierung ist der Analysator aktiv.

Die von Node-Red bereitgestellte Webschnittstelle zur Aktivierung einer neuen Konfiguration kann parallel mehrmals zur selben Zeit aufgerufen werden. Die Konfiguration, die zum Schluss aktualisiert wird, wird aktiviert. Da bei Node-Red die komplette Konfiguration ausgelesen und erweitert werden muss, kann es bei paralleler Aktualisierung zu Inkonsistenzen kommen, weil zwischen Auslesen der Konfiguration, der Aktualisierung der Konfiguration und dem Senden der neuen Konfiguration sich die aktuelle Konfiguration von Node-Red wieder ändern könnte. Um die Aktualisierung der Konfiguration von Node-Red weitgehend vor Inkonsistenzen zu sichern, wurde ein Sperrmechanismus implementiert, der dies verhindert. Der Sperrmechanismus, wie in Abbildung 4.10 dargestellt,

verhindert Inkonsistenzen, indem eine Sperre erzeugt wird, wenn eine Aktualisierung der Konfiguration durchgeführt wird. Der Sperrmechanismus wird beim Aktualisieren, dem Erzeugen und beim Löschen von Analysatoren verwendet. Der Funktionsbaustein „lockApi“ in der Abbildung 4.10 versucht eine Sperrdatei zu erzeugen. Der Weichenknoten „checkLocking“ prüft das Ergebnis, ob eine Sperre erzeugt werden konnte. Ist das der Fall, wird der Ablauf fortgesetzt. Wenn die Sperre nicht erzeugt werden konnte, startet der Funktionsbaustein „delayLockingRequest“ und verzögert den nächsten Versuch.

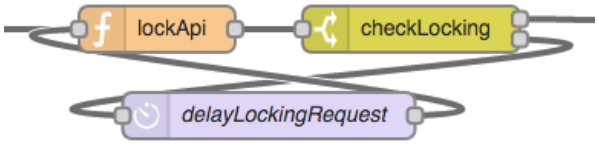

Abbildung 4.10: Sperrmechanismus zum Schutz vor Inkonsistenzen des Ausführungssystems.

Wird die Konfiguration aktualisiert, wird zuerst geprüft, ob eine Sperrdatei vorhanden ist. Die Sperrdatei besitzt keinen Inhalt, sondern dient lediglich zur Prüfung, ob diese vorhanden ist. Wurde die Sperrdatei bereits erzeugt, wird ein Timer gestartet, der die Abfrage zufällig zwischen 100 und 150ms verzögert. Der Timer wurde zufällig gewählt, damit es bei mehrfachen, parallel laufenden Anfragen nicht zu einem gleichzeitigen Zugriff auf die Datei kommt. So wird verhindert, dass mehrere Anfragen gleichzeitig feststellen, dass keine Datei verfügbar ist, und es zu einer Inkonsistenz kommt. Nach Ablauf des Timers wird nochmals geprüft, ob die Sperrdatei vorhanden ist. Ist das der Fall, startet der Timer mit einer weiteren zufälligen Verzögerung. Wurde die Datei gelöscht, kann die Anfrage die Konfiguration des zu aktualisierenden Ausführungssystems einlesen, aktualisieren und wieder an das Ausführungssystem senden. Nachdem die Aktualisierung bestätigt wurde, indem die HTTP-Anfrage abgeschlossen wurde, wird die Sperrdatei wieder gelöscht, und weitere Anfragen können eine Aktualisierung durchführen. Zum Verhindern eines Deadlock wird die Sperrdatei beim Starten des Ausführungssystems gelöscht. Deadlock bedeutet in diesem Zusammenhang, dass eine Sperrdatei angelegt wurde, aber nicht gelöscht. Der Prozess, der die Sperrdatei angelegt hat, wurde bereits beendet, die Datei aber nicht gelöscht. Dies wird dadurch verursacht, dass das Ausführungssystem bei der Aktualisierung beendet wurde, ohne dass der Prozess abgeschlossen war. Eine Löschung der Sperrdatei ist dann nicht mehr möglich. Nach dem Neustart des Ausführungssystems kann keine Aktualisierung mehr erfolgen, da die Sperrdatei nicht mehr gelöscht wird, weil die Anfrage, die zur Laufzeit unterbrochen wurde, bereits beendet ist.

4.4 Event-Kommunikation

Die Ausführung der Datenanalysen im Analysesystem erfolgt immer nur nach Eintreffen eines Events von einem externen System. Dabei kann es sich um ein automatisch versendetes Event handeln, oder um ein vom einem Anwender selbst versandtes. Die Annahme der Events erfolgt zentral vom Analyse Management System und nicht vom Analysator selbst. Das Analyse Management System prüft die Events daraufhin, ob alle notwendigen Parameter korrekt gesetzt sind, und verteilt sie an die darauf wartenden Analysatoren. Bei den Parametern handelt es sich um den Event-Typ und optional eine

Listing 4.4 Eventstruktur für die Aktivierung von Analysatoren durch das Analyse Management System.

```
{
      "type": "Event-Typ",
      "url": "Parameterzeichenfolge"
}
```

Zeichenfolge. Die Analysatoren erhalten nur das Event, wenn der Event-Typ mit dem im Analysator hinterlegten übereinstimmt. Die Kontrolle für die Verteilung übernimmt das Analyse Management System. Damit ist gegeben, dass nur die Events an die Analysatoren übertragen werden, die die Analysatoren verwenden, und nicht alle Events selbst vom Analysator geprüft werden müssen. Das hätte zur Folge, dass alle Analysatoren bei einem Eintreffen eines Events gestartet und nach der Prüfung wieder beendet würden. Das könnte bei einer hohen Anzahl an Events und Analysatoren zu einer hohen Last und schlechten Skalierung führen, weil nicht zentral die Events geprüft werden und nur die Analysatoren gestartet werden, die das Event verwenden.

4.4.1 Event-Aufbau und Event-Typen

Ein Event besteht immer aus einer Zeichenfolge mit dem JSON-Datenformat. Das Listing 4.4 zeigt den Aufbau der Eventstruktur. In der Eventstruktur ist das Feld „type“ enthalten, das den Event-Typ beschreibt. Damit kann das Analyse Management System den Typ des Events erkennen. Die Erkennung des Typs erfolgt über zwei Schritte: Je nach Aufruf der URL der Webschnittstelle wird das Event einem Event-Typ zugeschrieben, das Feld „type“ beschreibt dann in der zweiten Stufe den Event-Typ noch genauer. Das Feld „url“ wird dazu verwendet, in der Eventstruktur eine beliebige Zeichenfolge zu übertragen. Die Zeichenfolge wird an die Analysatoren weitergeleitet und kann für das Parametrisieren der Analysatoren auf Eventbasis verwendet werden. Das Feld wird in der ECHO-Plattform dazu genutzt, um die Bezeichner des erzeugten Datensatzes zu übertragen. Bei den Bezeichnern handelt es sich um die Patienten-ID, den Datensatztyp und die Datensatz-ID.

Das Analysesystem verwendet externe Events, die an die Webschnittstelle des Analyse Management System versendet werden, um Analysen, die auf das jeweilige Event warten, zu starten. Für die Übertragung zum Analyse Management System wird eine HTTP-REST-API verwendet, die das Event als JSON-Zeichenfolge entgegennimmt. JSON und eine HTTP-REST-API werden eingesetzt, um eine homogene Schnittstellenbeschreibung zu erreichen. Das Analyse Management System verwendet zwei unterschiedliche Typen von Events:

Event-Typen

a) Neuer Datensatz

b) zeit-/intervall-basierend

Der Event-Typ „Neuer Datensatz“ wird verwendet, wenn ein neuer Datensatz in der ECHO-Plattform erzeugt wurde. Es teilt dem Analyse Management System mit, dass ein Datensatz erzeugt wurde und von welchem Typ er ist. Damit können alle Analysatoren aktiviert werden, die auf den jeweiligen

neuen Datensatz warten. Alle Events vom Typ „Neuer Datensatz“ werden an die Webschnittstelle des Analyse Management System gesendet, das unter folgender URL erreichbar ist:

http://<host-vom-ams>:<ams-port>/analyzer/new_report

Der Host beschreibt das System, auf dem das Analyse Management System arbeitet, und den Port, unter dem der Webserver aufgerufen werden kann. Der nachfolgende Pfad beschreibt die Webschnittstelle, unter dem die REST-API für den Event-Typ erreichbar ist. Bei dem Event-Typ „Neuer Datensatz“ kann das Feld „type“ eine beliebige Zeichenfolge enthalten und mit dem Feld den Datensatz-Typ beschreiben. Wird zum Beispiel ein Datensatz vom Typ DailyReport erzeugt, enthält das Feld „type“ die Zeichenfolge „DailyReport“.

Der Event-Typ „zeit-/intervall-basierend“ wird verwendet, wenn Analysatoren aktiviert werden sollen, die auf ein zeitliches Intervall warten und unabhängig von einem Datensatz sein sollen. Das zeitliche Intervall wird von einem externen System ausgelöst, das Analyse Management System verwendet intern keine Funktionsblöcke für die zeitliche Ausführung von Analysatoren. Der Grund dafür ist, dass der Ablaufplan für die Analyse nur feste Intervalle verwenden kann. Eine Änderung am Intervall würde das Löschen und Neuerzeugen des Ablaufplans erfordern. Über die Event-Kommunikation können die Analysatoren in beliebigen Intervallen aktiviert werden. Die ECHO-Plattform sieht aktuell nur eine Analyse auf Intervallebene vor. Dabei handelt es sich um die tägliche Überprüfung, ob ein Bericht vom Patienten erzeugt wurde. Deshalb werden bei zeit-/intervallbasierenden Events alle Events ausgelöst, ohne dass für einzelne Analysen ein dynamisches Intervall vorgegeben werden kann. Das Versenden des Events erfolgt über die Webschnittstelle des Analyse Management System unter der URL:

http://<url-zu-ams>:<ams-port>/analyzer/time_based

Bei Event-Typ „Zeit/Intervall“ wird der Typ direkt über die URL der Webschnittstelle bestimmt. Im Feld „type“ wird dann die Zeichenfolge „timeBased“ übertragen. Das wird so gelöst, weil es in der aktuellen Implementierung keine weiteren Untertypen der zeit-/intervallbasierenden Events gibt. Für den generischen Ansatz und den späteren Fall, wenn weitere Events hinzukommen, ist die Auswahl des Typs möglich.

Für den Event-Typ „Neuer Datensatz“ stehen in der ECHO-Plattform 8 Events zur Verfügung, für alle Datensätze, die Patientendaten beinhalten, die für eine Analyse benötigt werden können. Die einzelnen Events sind abhängig von der Datenbank-Tabelle, in der der neue Datensatz angelegt wurde. Pro Tabelle ist ein spezifisches Event vorhanden. Beispielhaft werden mögliche Events beim Anlegen eines Datensatzes in der Tabelle 4.1 dargestellt. Die Tabelle stellt nicht alle verfügbaren Events beispielhaft dar.

In der Datenbank Tabelle „dailyReports“ werden neue tägliche Berichte gespeichert. Beim Anlegen des Datensatzes wird das Event **daily_report** erzeugt. Deklariert wird das Event in Analysatorvorlage durch die Angabe der Zeichenfolge „newDailyReport“. Bei der Datenbank Tabelle „treatments“ werden die Behandlungen gespeichert. Beim Anlegen eines Datensatzes wird das Event **treatments** ausgelöst.

Tabelle 4.1: Beispielhafte Darstellung von Events beim Erzeugen eines Datensatzes.

Tabelle	Datensatz	Event-Typ	Beschreibung in Analysatorvorlage
dailyReports	Täglicher Bericht	daily_report	newDailyReport
treatments	Behandlung	treatments	newTreatments
readings	Messung	readings	newReadings

Das Event „treatments“ wird in der Analysatorvorlage durch die Angabe der Zeichenfolge „newTreatments“ beschrieben. Für Messung steht die Datenbank Tabelle „readings“ zur Verfügung. Beim Anlegen des Datensatzes wird das Event **readings** ausgelöst. Soll eine Ausführung des Analysators bei der Erzeugung des Datensatzes für Messungen erfolgen, muss in der Analysatorvorlage „newReadings“ hinterlegt werden. Die Beschreibung in der Analysatorvorlage und dem Event-Bezeichner ist unabhängig. Das ermöglicht eine interne Beschreibung und eine für den Anwender lesbare Deklaration.

4.4.2 Event-Verteilung

Die Verteilung der eingehenden Events wird durch das Analyse Management System durchgeführt. In dem das Analyse Management System nur die Events an die Analysatoren weiterleitet, auf die die Analysatoren warten. Dies könnte ebenfalls über eine HTTP-REST-API erfolgen für einen homogenen Ansatz. Dabei würden die Events per HTTP Aufruf an die Analysatoren weitergeleitet. Jedoch ist das in der Node-Red-Implementierung aktuell nicht möglich. Die dafür notwendige HTTP-REST-API kann von Node-Red angelegt werden, in Tests wurde jedoch festgestellt, dass die Aktivierung der Schnittelle erst bei einer weiteren Erweiterung der Node-Red Konfiguration erfolgt. Das hätte zur Folge, dass der letzte erzeugte Analysator erst Events erhalten kann, wenn ein weiterer Analysator angelegt wurde. Zudem hat der HTTP-Ansatz die Problematik, dass er bei einer hohen Anzahl an Events und Analysatoren schlecht skaliert. Für jedes Event müsste das Analyse Management System das Event per HTTP-Aufruf an den Analysator übertragen. Dies erfolgt sequentiell. Sind viele Analysatoren aktiviert, wird der Versand durch die sequentielle Übertragung verlangsamt.
Als Ersatz für eine HTTP-Übertragung kommt das Publish-Subscribe-Verfahren zum Einsatz. Das Verfahren wird eingesetzt, um eine große Menge an Daten von Produzenten an eine beliebige Menge von Konsumenten zu verteilen, die sich mittels Registrierung beim Produzenten anmelden und dem Produzenten mitteilen, welche Daten sie erhalten wollen. Das hat den Vorteil, dass die Konsumenten nur die Daten erhalten, für die sie eine Registrierung durchgeführt haben, was eine Reduzierung des Datenvolumens zur Folge hat. Als Publish-Subscribe-Verfahren wird das offene Nachrichten-Protokoll MQ Telemetry Transport (MQTT) für Maschinen-zu-Maschinen-Kommunikation verwendet, welches von Andy Stanford-Clark (IBM) und Arlen Nipper (Cirrus Link Solutions) entwickelt wurde. [mqt] MQTT ist standardisiert von der Organization for the Advancement of Structured Information Standards (OASIS). Es wird eingesetzt, da es als Open-Source-Protokoll frei verfügbar ist und bereits fertige Funktionsbausteine in Node-Red vorhanden sind. Damit muss ein Publish-Subscribe-Verfahren nicht als Ablaufplan umgesetzt werden, sondern der fertige Funktionsbaustein kann verwendet werden. MQTT benötigt ein Serversystem namens Broker, bei dem sich die Konsumenten (Analysatoren) registrieren. Die Produzenten von Events leiten diese an den Broker weiter. Der Broker verteilt dann

die empfangen Events an die beim Broker registrierten Konsumenten. Das hat den Vorteil, dass sich die Konsumenten nicht bei jedem Produzenten registrieren müssen, sondern nur am zentralen Broker. Als Broker kommt der Open Source Server Mosquitto zum Einsatz, weil er frei verfügbar und ohne große Konfiguration direkt einsatzbereit ist. Der Broker kann auf dem System des Analyse Management System betrieben werden oder separat auf einem anderen System.

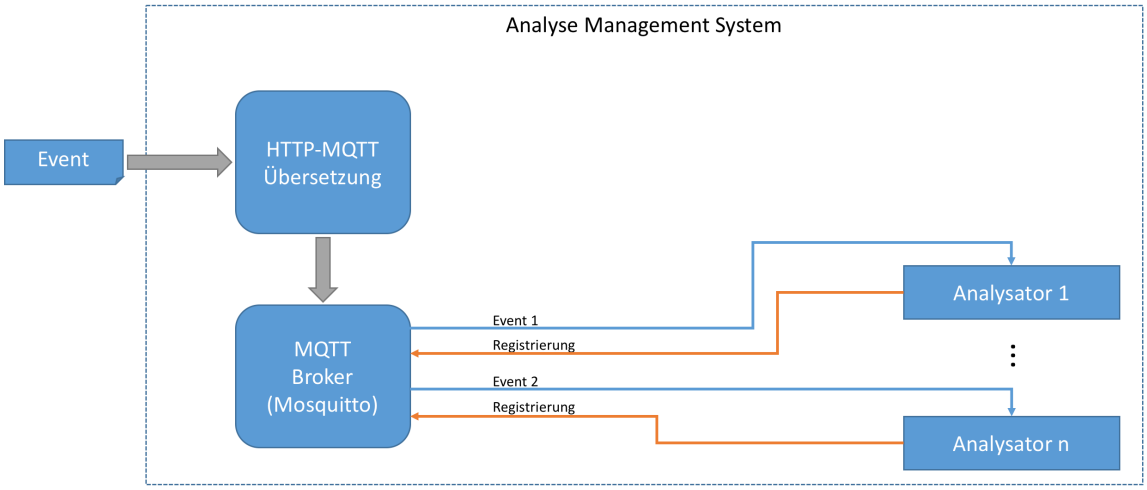

Abbildung 4.11: Verteilung der Events durch den MQTT Broker an die Analysatoren.

Die Abbildung 4.11 zeigt die Integration der Verteilung von Events im Analyse Management System. Eingehende Events werden über die REST-API entgegengenommen. Es wird nicht MQTT für die Übertragung von externen Systemen zum Analyse Management System verwendet, weil sonst eine einheitliche REST-API für die Webschnittstellen nicht angeboten werden kann. Mit einer REST-API kann eine hohe Verfügbarkeit erreicht werden, da eine breite Unterstützung vorhanden ist. Die eingehenden Events werden an die HTTP-MQTT-Übersetzung weitergeleitet. Die HTTP-MQTT-Übersetzung ist als Ablaufplan implementiert und konvertiert den JSON Body, der das Event enthält, in ein MQTT-Datenpaket. Der Ablaufplan für HTTP-MQTT-Übersetzung agiert dabei als Produzent und registriert sich beim Broker. Das MQTT-Protokoll beinhaltet einen Header für die Kontrollstruktur, um die Pakete zu vermitteln, optional einen weiteren variablen Header für die weitere Kontrollstruktur und den Payload. Der Payload enthält die Daten, die mit dem MQTT-Protkoll übertragen werden. Bei der HTTP-MQTT-Übersetzung wird im MQTT Header das Topic gesetzt. Das Topic enthält den Event-Typ, welches auch von Konsumenten verwendet wird, um sich beim Broker zu registrieren. So ermittelt der Broker beim Eintreffen des Events die Konsumenten, an welche das Event weitergeleitet werden soll. Im Payload wird die Zeichenfolge des Felds „url“ eingefügt. Das MQTT-Paket wird vom Ablaufplan für die HTTP-MQTT-Übersetzung an den Broker gesendet. Für die Übertragung wird TCP verwendet, was sicherstellt, dass kein Event bei der Übertragung verloren geht.
Die Analysatoren registrieren sich als Konsument beim Broker. Dazu verwenden sie den Event-Typ zur Registrierung als Topic. Damit ist sichergestellt, dass der Broker alle Events des Typs an den Konsumenten weiterleitet. Sendet der Ablaufplan für HTTP-MQTT-Übersetzung ein Event an den Broker, ermittelt der Broker alle Analysatoren, die das Event empfangen wollen und leitet es an die Analysatoren weiter. Ein Analysator registriert sich dabei immer mit einem Event-Typ, bedingt

dadurch, dass ein Analysator nur für ein Event erzeugt werden kann. Unterschiedliche Analysatoren können sich - wie in der Abbildung 4.11 dargestellt - auf unterschiedlichen Event-Typen registrieren.

Der Einsatz von MQTT erlaubt es, das Analysesystem auf mehreren unabhängigen Ausführungssystemen zu betreiben. Das wird dadurch ermöglicht, dass der Broker auch von externen Systemen erreicht werden kann und die Verteilung der Events übernimmt, ohne dass eine Verwaltung des Analyse Management System erfolgen muss. Die Analysatoren müssen sich dazu nur am Broker als Konsumenten registrieren.

4.5 Datenmodell des Analyse Management Systems

Das Ausführungssystem Node-Red speichert in JSON die Konfiguration der Ablaufpläne. Das bedeutet, dass das Analyse Management System und die Analysatoren in der Konfiguration abgespeichert werden. Bei einer fehlerhaften Aktivierung einer Konfiguration oder Inkonsistenzen bei der Aktualisierung der Konfiguration durch parallele Zugriffe in Node-Red kann es jedoch dazu führen, dass die Konfiguration nicht mehr korrekt ist. Das geschieht unter anderem deshalb, weil die Webschnittstelle von Node-Red für das Aktualisieren nicht geschützt ist vor parallelen Zugriffen. Das kann dazu führen, dass es bei zwei parallelen Aktualisierungen zu einer Überschreibung der Konfiguration kommt, weil die Aktualisierung zur Laufzeit unterschiedliche Konfigurationsstände verwendet. Zur Verhinderung dieser Problematik werden unter anderem die Analysatoren in der ECHO-Plattform gespeichert. Dazu wird die gesendete Analysatorvorlage in die einzelnen Bestandteile zerlegt und in MySQL-Tabellen gespeichert. Eine Wiederherstellungsfunktion erzeugt aus den zerlegten Bestandteilen der Analysatorvorlage wieder eine im JSON-Datenformat beschriebene Analysatorvorlage. Die zusammengefügte Analysatorvorlage wird zur Erzeugung und Aktivierung an das Analyse Management System gesendet.

Die Abbildung 4.12 zeigt das ER-Diagramm des Analyse Management Systems zur Speicherung der Analysatoren in der ECHO-Plattform. Die Daten des Analyse Management System werden dazu in der MySQL-Datenbank der ECHO-Plattform hinterlegt. Die Tabelle **analyser** speichert das Grundgerüst der Analysatorvorlage. Grundgerüst bedeutet dabei die Haupteigenschaften des Analysators ohne Analyseregeln. Jede Analysatorvorlage erhält eine eindeutige ID zur Identifizierung und eine Zuordnung des Accounts, von dem die Analysatorvorlage erzeugt wurde. Weiter sind der Event-Typ und die Aktion gespeichert, die von der Analysatorvorlage verwendet wird. Für die Aktion kann auch die Nachricht hinterlegt werden, die in Analysatorvorlage festgelegt wurde. Für die Zuordnung des Analysators im Analyse Management System zur Analysatorvorlage in der Tabelle wird die eindeutige Kennzeichnung des Analyse Management System gespeichert. Damit kann auch geprüft werden, ob der Analysator auf dem Analyse Management System erzeugt wurde. Desweiteren wird der Zeitpunkt der Erzeugung und eine mögliche Änderungszeit gespeichert. In der Tabelle **accounts** werden die Benutzer der ECHO-Plattform hinterlegt. Die eindeutige ID des Benutzers wird dazu verwendet, den Analysator einem Benutzer zuzuordnen.
Die Tabelle **analyserNotification** enthält die Aktion, die vom Analyse Management System und der ECHO-Plattform ausgeführt werden kann. Daraus erhält die Aktion eine eindeutige ID sowie eine Textbeschreibung der Aktion und die eindeutige Kennzeichnung der Aktion im Analyse Management System. Diese eindeutige Kennzeichnung wird auch in der Analysatorvorlage verwendet. Simultan

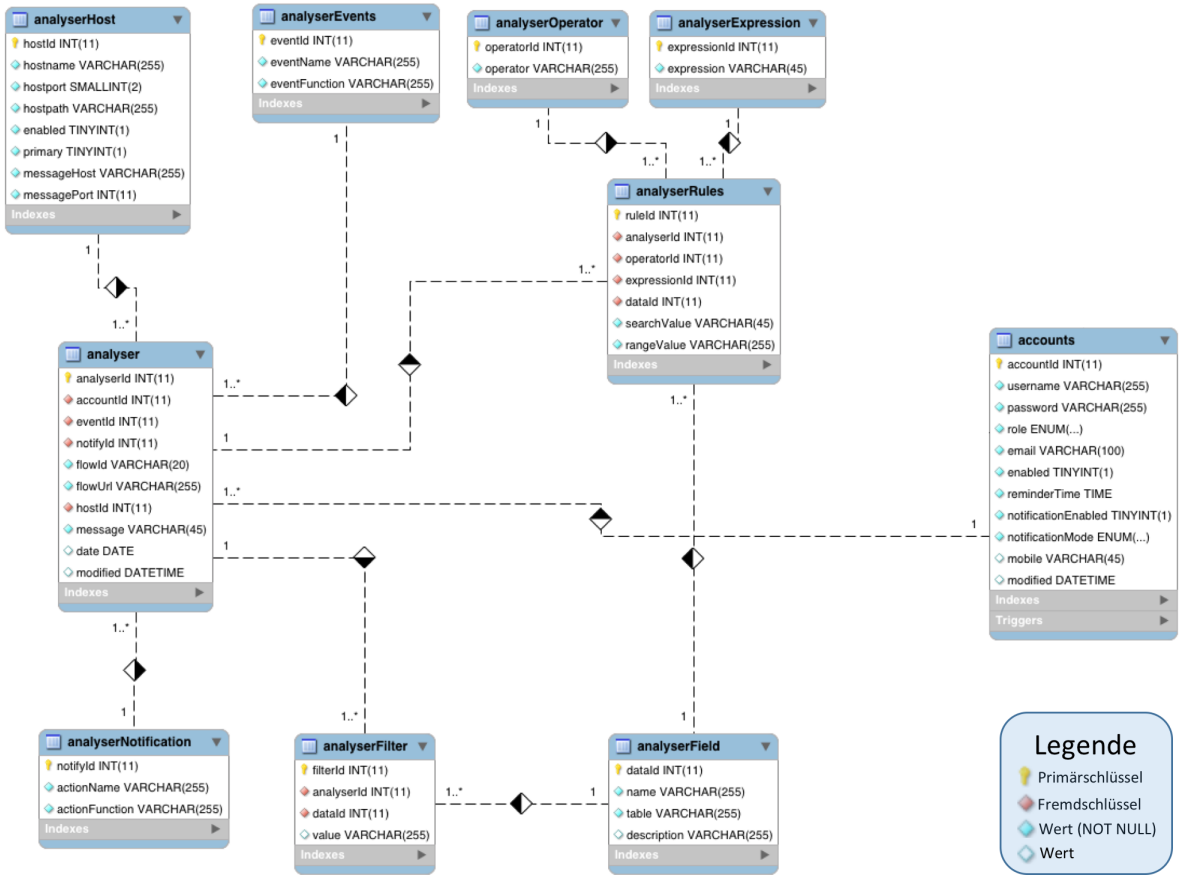

Abbildung 4.12: Darstellung des ER-Diagrammes des Analyse Management Systems.

dazu enthält die Tabelle **analyserEvents** die eindeutige ID des Events, welcher vom Analysator verwendet werden soll, die Bezeichnung in Textbeschreibung und den Event-Typ, der im Analysatorvorlage deklariert wird.
In der Tabelle **analyserHost** wird das Ausführungssystem hinterlegt, auf dem der Analysator erzeugt wird. Dazu erhält jedes Ausführungssystem eine eindeutige ID, den Hostnamen und Port, über den das Ausführungssystem erreichbar ist, sowie den Pfad zur Webschnittstelle. Hinterlegt ist außerdem der Broker und der Port zum Broker, der für das MQTT-Protokoll verwendet werden soll. Außerdem kann festgelegt werden, ob das Ausführungssystem als Analyse Management System agieren und für das Erzeugen des Analysators verwendet werden soll. Das Ausführungssystem mit dem Analyse Management System wird ebenfalls in der Tabelle festgelegt. Nur ein Ausführungssystem wird für das Analyse Management System verwendet.
Die Tabelle **analyserRules** enthält die Analyseregeln der Analysatorvorlage. Die Regeln werden einzeln zerlegt in der Tabelle gespeichert. Jede Regel erhält dabei einen Eintrag. Die Zuordnung zum Analysator erfolgt über die eindeutige ID des Analysators. Des Weiteren werden darin der Typ des Analyseoperators, das Datenfeld und der Vergleichswert für die Analyse gespeichert. Verfügt die Analysatorvorlage über mehrere Regeln, wird der Verknüpfungsoperator hinterlegt. Sollen mehrere Datensätze überprüft werden, wird die Anzahl der Datensätze ebenfalls gespeichert. In der Tabelle **analyserExpression** werden die Analyseoperatoren gespeichert, die im Analyse Management

System verfügbar sind. Dazu erhält jeder Analyseoperator eine eindeutige ID und die eindeutige Kennzeichnung des Analyseoperatoren, welcher im Analyse Management System verwendet und in der Analysatorvorlage deklariert wird. Die Verknüpfungsoperatoren, die im Analyse Management System verfügbar sind, werden in der Tabelle **analyserOperator** gespeichert. Jeder Verknüpfungsoperator erhält dabei eine eindeutige ID und die eindeutige Kennzeichnung des Verknüpfungsoperators, die im Analyse Management System verwendet wird. Wird in der Analysatorvorlage ein Filter angelegt, wird dieser bei der Speicherung in der Tabelle **analyserFilter** abgelegt. Jeder Filter erhält eine eindeutige ID, die eindeutige ID des Analysators, welcher den Filter verwendet, und das Datenfeld, auf dem der Filter angewendet werden soll. Zusätzlich wird noch der Vergleichswert für den Filter gespeichert. Die Tabelle **analyserField** wird für die Speicherung der Datenfelder verwendet, die vom Analyse Management System genutzt werden können. Jedes Datenfeld erhält eine eindeutige ID, die Tabelle des Datenfelds sowie die Spalte der Tabelle, die das Datenfeld beschreibt. Zusätzlich kann für das Datenfeld noch eine Beschreibung hinterlegt werden.

4.6 Datenzugriff

Für die Kommunikation mit dem Analysesystem wird eine Webschnittstelle per REST-API angeboten. Die Schnittstellen sind in die Swagger-API integriert, die die ECHO-Plattform für die Kommunikation für andere System anbietet. Damit wird erreicht, dass alle Schnittstellen der ECHO-Plattform und des Analysesystems zentral erreichbar sind und von Anwender verwendet werden können.

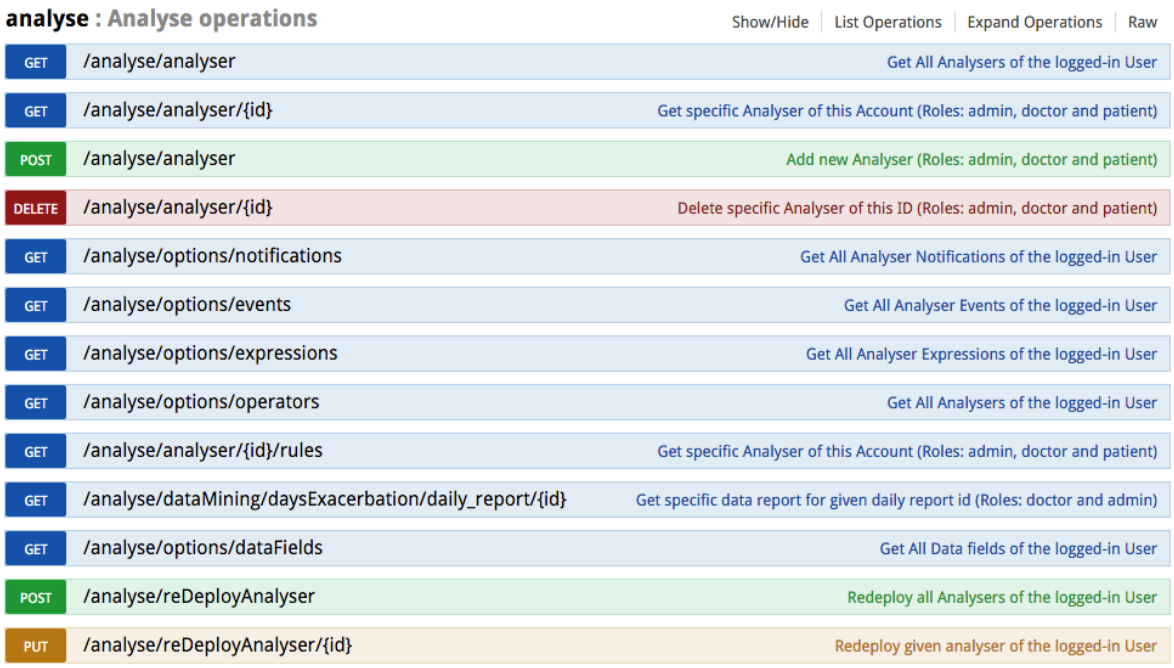

Abbildung 4.13: Angebotene Webschnittstellen des Analysesystems für die Kommunikation mit externen Systemen.

Die Abbildung 4.13 zeigt die angebotenen Webschnittstellen des Analysesystems, die über die ECHO-Plattform zur Verfügung gestellt werden. Per HTTP GET können die Analysatoren abgerufen werden, die unter dem verwendeten Benutzer erzeugt wurden und aktiv sind. Über die Angabe der ID des Analysators können spezifisch die Daten des angegebenen Analysators abgerufen werden. Per HTTP POST kann mit einer Analysatorvorlage in JSON ein Analysator erzeugt werden. Die Zuordnung des Analysators erfolgt über den verwendeten Benutzer. Als Antwort wird die Analysator-ID und das verwendete Ausführungssystem zurückgegeben. Mit der Angabe der Analysator-ID können die im Analysator verwendeten Regeln nach der Erzeugung über einen HTTP GET abgerufen werden. Durch HTTP DELETE mit der Angabe der Analysator-ID kann der betreffende Analysator vom Analyse Management System und aus der Datenbank gelöscht werden. Für die Erzeugung von Analysatorvorlagen werden Schnittstellen bereitgestellt, mit denen spezifische Daten des Analysesystems abgerufen werden können. Zum Abruf stehen die bereitgestellten Events, die Aktionen, die das Analyse Management System ausführen kann, die angebotenen Analyseoperatoren, Verknüpfungsoperatoren und die verfügbaren Datenfelder. Werden von einem Analysator Benachrichtigungen erzeugt, dann kann der Benutzer diese per HTTP GET abfragen. Jeder Benutzer kann die ihm zugeordneten Analysatoren per HTTP PUT neu erzeugen lassen. Dazu werden automatisch die Analysatoren auf der Ausführung gelöscht und neu erzeugt. Alternativ kann nur ein einzelner Analysator per HTTP PUT aktualisiert werden.

4.7 Data Mining

In das Analysesystem wurde Data Mining als „Proof of Concept“ implementiert. Es soll demonstrieren, dass das Analysesystem zur Ausführung von Data Mining verwendet werden kann. Weitere Analysen müssen jedoch separat in das Analyse Management System implementiert werden, da eine generische Einbindung nicht möglich ist. Der Grund dafür ist, dass Data Mining-Analysen aufbereitete Daten aus der MySQL-Datenbank benötigen, die nicht selbst vom Analyse Management System erzeugt werden können. Ziel der implementierten Data Mining-Analyse ist es, eine Abschätzung zu geben, wann eine Exazerbation des Patienten auftreten kann. Die Ausgabe soll als grobe Einteilung in Wahrscheinlichkeiten erfolgen. Dies erfolgt durch die Ermittlung, wann die Wahrscheinlichkeit für eine Exazerbation steigt und wann eine mögliche Exazerbation auftreten kann. Der Zeitraum, der in Gruppen unterschiedlicher Länge dargestellt wird, wird dem Patienten als Benachrichtigung verschickt. Die Zeiträume der Einteilung kann variabel über die Analysatorvorlage vorgegeben werden. Die Zeiträume in der Analysatorvorlage werden vom Analysesystem dazu verwendet, um bestehende Datensätze eine Klasse zuordnen zu können. Durch die Data Mining-Analyse kann für neue Datensätze, auf die noch keine Exazerbation folgte, die Klasse abgeschätzt werden. Die Klasse stellt dann den Zeitraum dar, wann eine Exazerbation auftreten kann. Damit kann der Patient frühzeitig den Arzt aufsuchen oder Gegenmaßnahmen treffen. Für die Analyse werden die in der ECHO-Plattform hinterlegten Daten aller Patienten verwendet. Dazu gehören die täglichen Berichte, die die Patienten ausfüllen, Arztbesuche, Berichte aus Gesundheitsanalysen wie etwa Lungenvoluminas, medizinische Hintergrunddaten von weiteren Erkrankungen und die Stammdaten des Patienten. Die Ausführung der Analyse erfolgt direkt im Ausführungssystem als Ablaufplan, ähnlich dem der anderen beschrieben Analysen mit der Einbindung externer Bibliotheken.

Die Klassenbezeichnung erfolgt bei der Analyse in Labels. Labels repräsentieren dabei die Klasse. Jeder Datensatz, der vor einer Exazerbation erzeugt wurde, kann für die Analyse verwendet werden, weil eine Einteilung in eine Klasse möglich ist. Das kann zum Beispiel mit drei Klassen niedrig, mittel und hoch erfolgen. Wobei die Klasse niedrig den Zeitraum mit größer 30 Tage beschreibt, mittel 29 bis 10 Tage und hoch mit 9 bis 0 Tage. Wird bei einem Datensatz der Zeitraum von 12 Tagen zwischen letztem Artzbesuch und Exazerbation berechnet, wird der Datensatz der Klasse mittel zugeordnet, indem er als Klassifikationsmerkmal dient. Dieser Status wird als Klasse verwendet, um einen Entscheidungsbaum aufzubauen. Beispielhaft kann der folgende Datensatz für die Erzeugung des Entscheidungsbaums verwendet werden:

P1 | Stammdaten | Täglicher Bericht | 17 Tage zur Exazerbation | Klasse: Mittel

Auf den Datensatz folgt bereits ein Eintrag mit der Exazerbation. Das ist erforderlich, um die Tage zur nächsten Exazerbation zu berechnen. Das „P1“ bezeichnet den Patienten. Die Stammdaten enthalten Informationen über den Patienten wie Geschlecht oder Alter in Jahren. Die Daten des täglichen Berichts enthalten Informationen, die der Patient täglich zu seinem Gesundheitszustand liefert. Die 17 Tage zur Exazerbation beschreibt die Anzahl an Tagen zwischen dem Tag, an dem der tägliche Bericht erzeugt wurde, und dem Tag, als die nächste Exazerbation stattfand. Die 17 Tage zur Exazerbation entsprechen mit der oberen Einteilung die Klasse „mittel“, weil die 17 Tage in den 29 bis 10 Tagen der Klasse „mittel“ liegen. Weitere Datensätze können nun anlog dazu ebenfalls einer Klasse zugeordnet werden, wenn nach dem Eintrag eine Exazerbation stattfand. Die daraus einstehenden Daten mit Klassenzuordnung werden bei der Data Mining-Analyse verwendet, um für Datensätze die Klasse zu bestimmen, auf die noch keine Exazerbation folgte.

Für die Data Mining-Analyse der Exazerbationwahrscheinlichkeit wird der Algorithmus C4.5 von Ross Quinlan beschrieben in [Qui14] eingesetzt. C4.5 verwendet Entscheidungsbäume zur Einteilung von Datensätzen in Klassen. Durch Klassifikation sollen die aufbereiteten Daten Klassen zugeordnet werden. C4.5 erzeugt aus Trainingsdaten einen Entscheidungsbaum. Die Trainingsdaten sind Daten aus dem laufenden System, die für die Modellerzeugung des Entscheidungsbaums verwendet werden, der dann die Einteilung der neuen Datensätze in Klassen ermöglicht. Die Trainingsdaten enthalten bereits die zugehörige Klasse, bei neuen Datensätzen soll diese ermittelt werden. Bei der Erzeugung des Entscheidungsbaums folgt man der Annahme, dass die Verteilung der Trainingsdaten gleich der der neuen Datensätze ist. Das bedeutet wiederum, dass die neuen Datensätze durch das Hinabsteigen des Entscheidungsbaums zu einer hohen Wahrscheinlichkeit der passenden Klasse zugeordnet werden können. Umso geringer die Abweichung der Trainingsdaten ist, umso besser wird die Einteilung der Klasse von neuen Datensätzen. Aus diesem Grund werden immer alle verfügbaren Daten als Trainingsdaten verwendet. Je mehr Daten verfügbar sind, umso besser kann die Klassifikation neuer Datensätze durchgeführt werden. Der Algorithmus wird per externer JavaScript-Bibliothek names C4.5 in das Ausführungssystem über den Nodejs Package Manager (npm) eingebunden. Das Ausführungssystem stellt dann die Bibliothek in den Funktionsbausteinen zur Verfügung.

Die Data Mining-Analysen unterscheiden sich in drei Punkten von den anderen implementierten Analysen:

(i) Die Analyse wird zwar in der Analysatorvorlage beschrieben, inklusive der ausführenden Aktion und dem Event, jedoch wird in der Analysatorvorlage nur festgelegt, welche Analyse zum Einsatz kommt. Die Regelsyntax beschreibt nicht die Analyse in der Funktionsweise bzw. deren Regeln, sondern deklariert nur Parameter der Data Mining-Analyse, die notwendig für die Ausführung sind. Somit lässt sich bei Data Mining-Analysen nicht die Funktionalität der Analyse ändern oder beschreiben.

(ii) Die Data Mining-Analysen verwenden eine externe Bibliothek, die den Data Mining-Algorithmus implementiert, und führt diesen aus. Die Data Mining-Analyse wird nicht im Ablaufplan mit Funktionsbausteinen erzeugt, wie es bei anderen Analysen durchgeführt wird.

(iii) Die Aufbereitung der Daten erfolgt bei Data Mining-Analysen nicht im Ablaufplan der Analyse, sondern wird extern bereits aufbereitet zur Verfügung gestellt. Der Abruf der Daten für die Analyse erfolgt direkt über die MySQL-Datenbank und nicht über eine Webschnittstelle. Das folgt aus der Tatsache, dass die aufbereiteten Daten alle Datensätze aller Patienten enthalten. Das würde der Zugriffsschutz der Webschnittstelle verhindern, der nur die Übertragung der Daten erlaubt, auf die der Benutzer Zugriff hat. Das würde jedoch wiederum die Data Mining-Analyse verschlechtern, da nur die Daten des Benutzers zur Verfügung stehen, auf die der Benutzer Zugriff hat. Die Anzahl der Daten wäre somit geringer.

4.7.1 Aufbereitung

Die Erzeugung des Entscheidungsbaums erfolgt immer zur Laufzeit der Analyse. Dazu werden bei jeder Analyse alle Daten abgerufen, die für die Analyse notwendig sind. Das erhöht den Aufwand bei der Ausführung. Das ist dadurch begründet, dass die Bibliothek aktuell den Entscheidungsbaum nicht zwischenspeichern kann und somit immer ein neuer generiert werden muss. Dazu müssen die Trainingsdaten abgerufen werden. Der Abruf der Daten erfolgt über eine direkte MySQL-Verbindung zur Datenbank. Die Trainingsdaten wie auch der Datensatz, der klassifiziert werden soll, muss bereits in der ECHO-Plattform aufbereitet in der Datenbank gespeichert sein. Bei der Aufbereitung werden Datensätze aus unterschiedlichen Tabellen verknüpft, Datenwerte konvertiert und Datenfelder entfernt, wenn diese nicht notwendig sind, so dass Datensätze vorhanden sind, die direkt im C4.5-Algorithmus eingelesen werden können. Dazu wird ein MySQL View mit dem Namen **daysExacerbationTraining_view** verwendet. Ein MySQL View ist eine logische Tabelle in der MySQL-Datenbank, bei der die Datensätze aus einer oder mehreren Tabellen stammen, und die eine abstrahierte Sicht auf die Tabelle verwendet. [Sue02] Für die Aufbereitung der Daten wird der View verwendet, um die benötigten Daten der Analyse in einer Tabelle darstellen zu können.

Die Abbildung 4.14 zeigt die Datenquellen, die in dem View aus den unterschiedlichen Tabellen verknüpft werden. Der Datensatz im View ist grundsätzlich abhängig vom Patienten (Tabelle: patients) mit den Patientendaten und dem täglichen Bericht (Tabelle: dailyReports). Pro Patient und täglichem Bericht wird ein Eintrag im View erzeugt. Das bedeutet: Wenn der Patient einen täglichen Bericht erzeugt, existiert auch ein Datensatz im View für die Analyse. Jeder dieser Einträge enthält die Patientendaten (Tabelle: patients) wie Geschlecht und Alter (in Jahren) sowie die im täglichen Bericht beantworteten Fragen (Tabelle: dailyReports). Der Datensatz enthält zusätzlich die Daten des nächsten Arztbesuchs (Tabelle: visits), die Messwerte (Tabelle: treatments) und andere Erkrankungen des

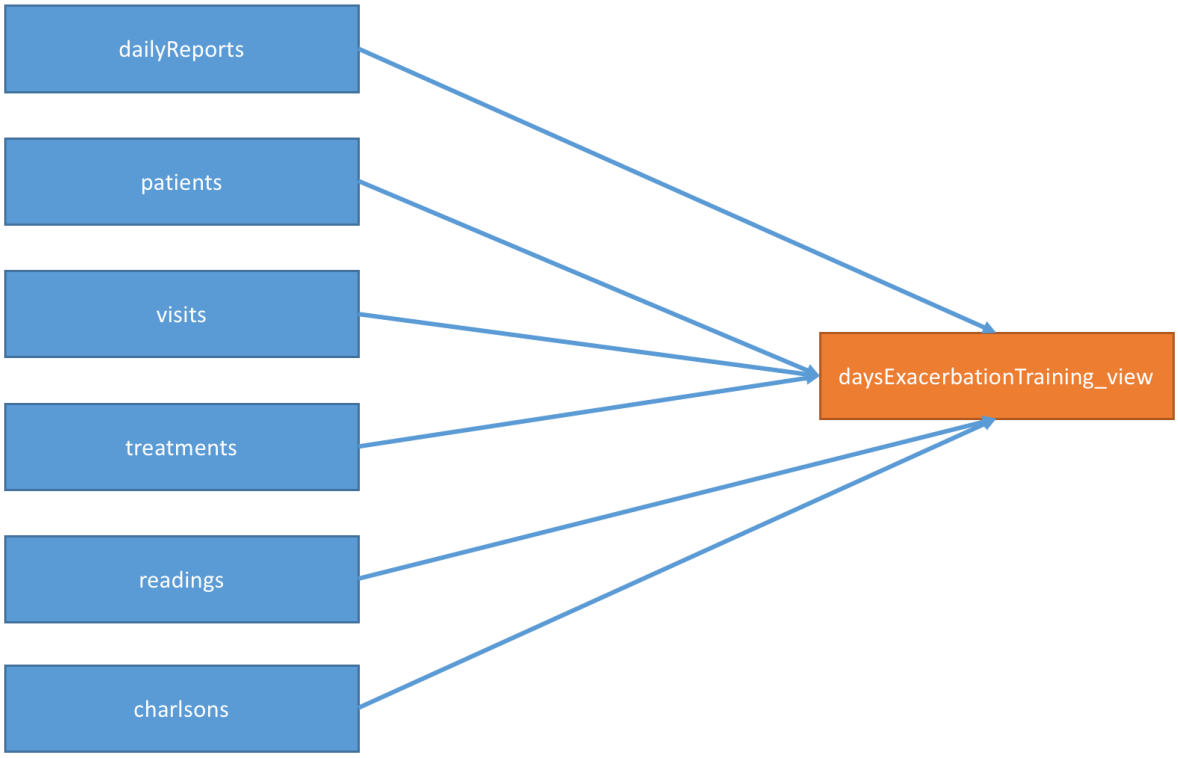

Abbildung 4.14: Aufbereitung der Daten für das Data Mining.

Patienten (Tabelle: charlsons). Des weiteren werden die vom Arzt beim Besuch ermittelte Werte des Patienten (Tabelle: readings) hinzugefügt. Der Status der Exazerbation wurde in der ECHO-Plattform in mehreren Tabellen gespeichert und der Arztbesuch wurde ebenfalls in den einzelnen Tabellen hinterlegt. Für die Data Mining-Analyse wurde dies geändert, um zentral die Daten abrufen zu können die alle auf denselben Datensatz des Arztbesuchs referenzieren. Der Arztbesuch wird in der Tabelle „visits“ gespeichert. Die Tabellen, die vorher den Arztbesuch enthielten, referenzieren nach der Änderung auf die Tabelle „visits“. So wird der Status des Arztbesuchs nicht mehr mehrfach gespeichert. In der Tabelle „visits“ ist nun auch zentral der Status der Exazerbation hinterlegt.

Abbildung 4.15: Abhängigkeit zwischen den Artzbesuchen und der Erzeugung von täglichen Berichten.

Die Abbildung 4.15 zeigt in zeitlichem Verlauf die Erstellung von täglichen Berichten, Arztbesuchen und Exazerbation. Beim Arztbesuch wird der Status festgelegt, ob eine Exazerbation stattfand oder nicht. Tritt eine Exazerbation bei einem Patient auf, wird dies beim Arztbesuch vermerkt, sonst erhält der Eintrag den Status „baseline“. Der Eintrag im View ist nur verfügbar, wenn der darauf folgende Arztbesuch vorhanden ist und der Status Exazerbation gesetzt wurde. Das ist notwendig, weil die Einträge des Views sich immer auf den kommenden Arztbesuch beziehen, um festzustellen, ob und wann eine Exazerbation stattfand. Das bedeutet, dass die in der Abbildung 4.15 dargestellte Exazerbation den Status für alle täglichen Berichte zwischen dem Arztbesuch 2 und dem Arztbesuch 3 bestimmt. Der Status kann aber nur gesetzt werden, wenn er beim Arztbesuch hinterlegt wurde. Somit erhalten die Datensätze im View den Status, ob eine Exazerbation stattfand oder nicht. Das ist aber nur dann entscheidbar, wenn der Patient einen Arztbesuch durchführt. Ist das nicht passiert, besitzen die Datensätze noch keinen Status und somit keine Klasse. Diese Einträge dürfen nicht als Trainingsdaten verwendet werden, da die Klasse fehlt. Der C4.5-Algorithmus wird für diese Datensätze angewendet, um den Status zu bestimmen. Nur Einträge mit dem Status Exazerbation sind von Interesse als Trainingsdaten, weil sie den Zeitraum zwischen Exazerbation und letzten Artzbesuch enthalten, der für die Berechnung der Klasse notwendig ist. Ist eine Exazerbation vorhanden, wird die Dauer zwischen dem vorherigen Arztbesuch und dem Artzbesuch mit dem Exazerbationseintrag berechnet.

4.7.2 Analyse

Die Analyse zur Berechnung der Exazerbationswahrscheinlichkeit wird immer ausgeführt, wenn ein täglicher Bericht erzeugt wird. Der Erzeugung der Analyse erfolgt ebenfalls über eine Analysatorvorlage. Das Listing 4.5 zeigt beispielhaft eine Analysatorvorlage zur Erzeugung einer Data Mining-Analyse. Dabei wird im „structure“-Feld die Klassen beschrieben. Jede Klasse beschreibt den Zeitraum der Tage und den Bezeichner der Klasse. Klasseneinteilung wird mit „LOW“, „MIDDLE“ und „HIGH“ deklariert. Die Angabe „LOW“ beschreibt dabei eine Wahrscheinlichkeit einer Exazerbation innerhalb der nächsten 30 bis 100 Tage im Beispiel. Der Zeitraum wird im Feld „range“ hinterlegt. Die Klasse wird im Feld „value“ beschrieben. Für die Data Mining-Analyse muss im Feld „exp“ der Analyseoperator „between“ verwendet werden. Dieser beschreibt nicht direkt das Analyseverfahren, sondern dass der Zeitraum im Feld „range“ für die Klassifikation verwendet werden soll. Simultan dazu wird im Beispiel die Klasse „MIDDLE“ mit 10 bis 29 Tagen und „HIGH“ mit 0 bis 9 Tagen beschrieben. Zusätzlich wird im Datenfeldbeschreibung „field“ die Data Mining-Analyse beschrieben. Jede Klasse ist mit einem AND-Verknüpfungsoperator verbunden. Das Event, das die Ausführung des Analysators beschreibt, muss bei der Analyse Exazerbationswahrscheinlichkeit mit **dataMining.newDailyReport** beschrieben werden. Es legt fest, dass die Analyse immer beim Erzeugen eines täglichen Berichts ausgeführt wird. Im Parameterfeld des Events wird der Grenzwert beschrieben, dem eine Auslösung der Aktion folgt. Dabei sind die Klassen enthalten, die eine Aktion auslösen sollen. In der Aktionsbeschreibung der Analysatorvorlage wird die Aktion ausgewählt, die ausgeführt werden soll, sowie die Nachricht, die versendet werden soll. Zusätzlich ist in der Aktionsbeschreibung die Nachricht, die verschickt wird. In der Nachricht kann das Token „threshold“ enthalten sein, das die erkannte Klasse des täglichen Berichts in die Nachricht einfügt.

Listing 4.5 Analysatorvorlage zur Beschreibung einer Data Mining-Analyse.

```
{
      "structure": [
            {
                  "type": "node",
                  "field": "classification.daysExacerbation",
                  "exp": "between",
                  "value": "LOW",
                  "range": "30 to 100"
            },
            {
                  "type": "operator",
                  "op": "AND"
            },
            {
                  "type": "node",
                  "field": "classification.daysExacerbation",
                  "exp": "between",
                  "value": "MIDDLE",
                  "range": "10 to 29"
            },
            {
                  "type": "operator",
                  "op": "AND"
            },
            {
                  "type": "node",
                  "field": "classification.daysExacerbation",
                  "exp": "between",
                  "value": "HIGH",
                  "range": "0 to 9"
            }
      ],
      "event": {
            "type": "dataMining.newDailyReport",
            "parm": {"threshold": ["MIDDLE", "HIGH"]}
      },
      "action": {
            "type": "sendMessage",
            "parm": {"message": "Warnung: die Wahrscheinlichkeit auf eine kommende
                Exazerbation ist {threshold}."}
      }
}
```

Die Analyse wird über das Event **daily_report** angestoßen. Extrahiert wird aus dem Event die Patienten-ID und die Bericht-ID. Die IDs werden benötigt, um aus dem View **daysExacerbation_view** den Datensatz zu laden. Der View **daysExacerbation_view** unterscheidet sich vom View **daysExacerbationTraining_view** nur in einem Merkmal: In dem View **daysExacerbation_view** sind nur Datensätze vorhanden, die noch keine Verbindung zum nächsten Arztbesuch besitzen, weil dieser noch nicht erfolgte und somit der Status, ob eine Exazerbation stattfand oder nicht, noch nicht entschieden werden kann. Die Klassifikation über C4.5 wird verwendet, um den möglichen Status zu ermitteln. Dazu erzeugt der Algorithmus aus den Trainingsdaten des Views **daysExacerbation-Training_view** zur Laufzeit den Entscheidungsbaum und wendet diesen auf den neuen Datensatz an, der aus dem neuen, täglichen Bericht erzeugt wurde. Die berechnete Zuteilung der Klasse wird verwendet, um zu prüfen, ob eine Benachrichtigung des Nutzers erfolgen soll. Dazu wird die im Grenzwert angegebene Klasse mit der erkannten verglichen. Stimmt die Klasse mit einer Klasse aus der Liste überein, wird die Aktion ausgeführt. Der Grenzwert wird verwendet, damit der Arzt, der die Analyse konfiguriert, festlegen kann, wann der Patient benachrichtigt werden soll.

5 Evaluierung

In diesem Kapitel wird das Analysesystem dahingehend geprüft, ob die regelbasierten Analysen korrekt durchgeführt werden können. Dazu werden die implementierten Analyseoperatoren mit Beispielanalysen getestet und validiert, ob das vorher berechnete Ergebnis mit der Analyse übereinstimmt. Zusätzlich wird die Performance und Skalierung überprüft zwischen dem vorgestellten Analysesystem und einer MySQL-Datenbank. Abschließend erfolgt eine Abschätzung mit zufällig generierten Gesundheitsdaten, inwieweit die implementierte Data Mining-Analyse eine Früherkennung einer Exazerbation ermöglicht.

5.1 Validierung

Für die Validierung wurde ein Anwendungsprogramm implementiert, das die verfügbaren Analyseoperatoren, Datenfelder, Verknüpfungsoperatoren, Events und Aktionen des Analyse Management System verwendet, um regelbasierte Analysen grafisch zu generieren. Die Abbildung 5.1 zeigt die Oberfläche des Generators.

Der Generator verwendet die Webschnittstellen des Analyse Management Systems, um mit den in der Anwendung hinterlegten URL und Benutzerdaten alle notwendigen Daten zu laden. Sobald die Verbindung zum Analyse Management System aufgebaut wurde, kann der Anwender grafisch die Analysatorvorlage erzeugen, indem er die Regeln deklariert und das Event sowie die Aktion auswählt, die zur Verfügung stehen. Anschließend kann der Anwender die erzeugte Analysatorvorlage im Klartext prüfen und auf dem Analyse Management System aktivieren. Der Generator zeigt, dass das Analysesystem von Anwendern verwendet werden kann, die keine umfassenden IT-Kenntnisse haben. Der Anwender muss nur einen Zugang besitzen und kann nach Laden aller Daten Analyseregeln erzeugen, ohne die Regelsyntax zu verwenden.

Für die Validierung des Analysesystems wurden mit dem Analysegenerator Regeln erstellt. Jeder Analysator verwendet einen einzelnen Analyseoperator. Das bedeutet: Jede Analysatorvorlage verwendet eine einzige Regel für den Vergleich des Datenwerts mit dem Vergleichswert. Für die Überprüfung werden mit den Analyseoperatoren Vergleichsanalysen durchgeführt, die den Datenwert mit einem Vergleichswert abgleichen. Anschließend wird das Ergebnis der Analyse mit dem vorher manuell berechneten Wert verglichen. Stimmen beide überein, kann sichergestellt werden, dass der Analyseoperator korrekt umgesetzt wurde. Zusätzlich wurde die Datenmodellüberprüfung getestet. Dies erfolgte, indem ein Analysator erzeugt wurde, der beim Vergleichswert falsche Datentypen enthielt. Wird bei der Erzeugung des Analysators mit einem Datentypfehler abgebrochen, kann sichergestellt werden, dass die Überprüfung des Datentyps und somit das Datenmodells korrekt implementiert wurde.

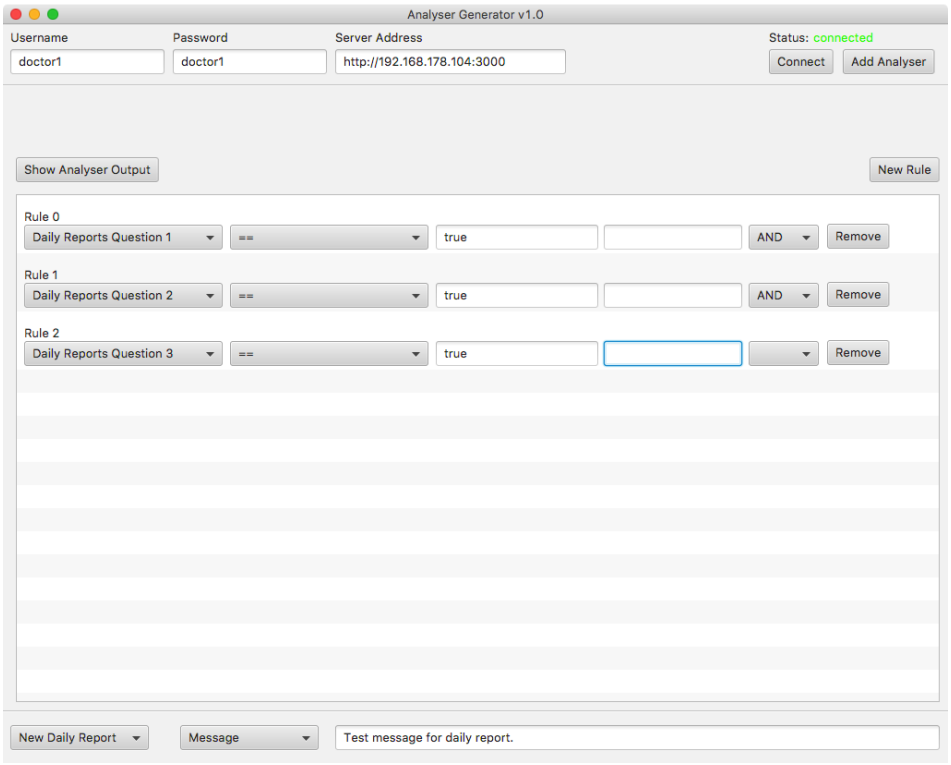

Abbildung 5.1: Analysatorvorlagegenerator zur grafischen Generierung von Analysen.

Die Tabelle 5.1 zeigt die Auswertung der Validierung der Analyseoperatoren. In der Spalte „Analyseoperator“ wird der verwendete Analyseoperator gekennzeichnet. Die Spalte „Expression“ beschreibt die Analyse. Die Spalte „Datenwert“ beschreibt den Wert des Datensatzes, die Spalte „Vergleichswert“ die Spalte mit dem zu überprüfenden Wert. In den Spalten „Analysatorergebnis“ und „Berechnetes Ergebnis“ sind die Ergebnisse der Analysen hinterlegt. Beide müssen übereinstimmen, damit die Analysatoroperatoren korrekt arbeiten. Ist dies der Fall, wird das in der Spalte „Auswertung“ mit OK bezeichnet, sonst FALSCH. Wie in der Auswertung zu sehen bestehen alle Testanalysen die Validierung, da das Ergebnis der Datenanalyse im Analyse Management System mit dem vorher berechneten übereinstimmt. Jede der Testanalysen zeigt eine Übereinstimmung des berechneten Ergebnisses mit dem des Analyseergebnisses.

Das vorher integrierte Analysesystem in der ECHO-Plattform besaß mehrere eingebettete Datenanalysen. Diese Datenanalysen wurden durch die regelbasierten Analyse ersetzt. Dabei bilden die Analysen dieselben Eigenschaft ab. Das bedeutet, beide Analysen müssen bei der Ausführung mit demselben Datenbestand dasselbe Ergebnis liefern. Zur Validierung dieser Analysen wurde mit Testdaten überprüft, ob beide Analysen dieselben Ergebnisse liefern. Die Tabelle 5.2 zeigt die Auswertung der Analysen in der statisch hinterlegten Version des alten Analysesystems mit der der regelbasierten Analysen. In der Spalte „Analysen“ werden die Analysen sprachlich beschrieben und in der obersten Zeile pro Feld die verwendete Tabelle der Datenbank. Die in Klammern hinterlegten Zahlen der Operatoren in der Spalte „Expression“ beschreibt die Anzahl an Datensätze bei der Analyse. Ist keine Zahl angegeben wird nur ein Datensatz überprüft. Hierbei muss die Expression von beiden Datensätzen erfüllt sein.

Tabelle 5.1: Auswertung der Überprüfung der Analyseoperatoren.

Analyse-operator	Expression	Daten-wert	Vergleichs-wert	Analysator-ergebnis	Berechnetes Ergebnis	Aus-wertung
==	Datenwert == Vergleichswert	1	1	true	true	OK
==	Datenwert == Vergleichswert	1	2	false	false	OK
!=	Datenwert != Vergleichswert	1	2	true	true	OK
!=	Datenwert != Vergleichswert	1	1	false	false	OK
<	Datenwert < Vergleichswert	4	5	true	true	OK
<	Datenwert < Vergleichswert	5	5	false	false	OK
<	Datenwert < Vergleichswert	5	4	false	false	OK
>	Datenwert > Vergleichswert	5	4	true	true	OK
>	Datenwert > Vergleichswert	5	5	false	false	OK
>	Datenwert > Vergleichswert	4	5	false	false	OK
<=	Datenwert <= Vergleichswert	5	5	true	true	OK
<=	Datenwert <= Vergleichswert	4	5	true	true	OK
<=	Datenwert <= Vergleichswert	5	4	false	false	OK
>=	Datenwert >= Vergleichswert	5	5	true	true	OK
>=	Datenwert >= Vergleichswert	4	5	false	false	OK
>=	Datenwert >= Vergleichswert	5	4	true	true	OK

Tabelle 5.2: Auswertung der regelbasierten Analysen mit den statisch hinterlegten Analysen.

Analyse	Expression	Datenwert	Regelbasierte Analyse	Statische Analyse	Auswertung
Daily Reports: 2 Reports alle 2 Tage	daily_reports. countReportsPerDate(2) == 2	count: 2	true	true	OK
Daily Reports: 2 Tage Temperatur über 38°	daily_reports. temperature(2) avgIncrease 38	temp: 38	true	true	OK
Daily Reports: Frage 1: Ja, 2 Tage	daily_reports.q1 == true	q1: true	true	true	OK
Daily Reports: Frage 1,2,3: Ja, 1 Tage	daily_reports.q1 == true AND daily_reports.q2 == true AND daily_reports.q3 == true	q1: true q2: true q3: true	true	true	OK
Daily Reports: Frage 3a, 3b: Ja, 1 Tag	daily_reports.q3a == true AND daily_reports.q3b == true	q3a: true q3b: true	true	true	OK
Daily Reports: Frage 3, 3c: Ja, 1 Tag	daily_reports.q3 == true AND daily_reports.q3c == true	q3: true q3c: true	true	true	OK
Daily Reports: Frage 5: Ja, 2 Tage	daily_reports.q5(2) == true	q5: true	true	true	OK

Die Werte in der Spalte „Datenwert“ beschreiben die vom Datensatz gesetzten Werte bei der Analyse. Für eine Übereinstimmung der Analysen müssen die Ergebnisse der Spalte „Regelbasierte Analyse“ und „Statische Analyse“ übereinstimmen. Erfolgt eine Übereinstimmung, dann folgt die Auswertung mit „OK“. Wie die Tabelle 5.2 zeigt stimmen alle Analysen der regelbasierten Analyse mit denen des alten Systems überein.

5.2 Data Mining

Für die Evaluierung der Data Mining-Analysen wurde ein Testgenerator von Datensätzen implementiert, welcher realistische Patienten- und Gesundheitsdaten erzeugt. Dies wurde erforderlich, weil zur

Zeit der Erstellung der Evaluierung nicht genug reale Datensätze für das Verfahren zur Verfügung standen. Für die Generierung wurden die mathematischen Berechnungen der Lungenvoluminas abhängig von Alter, Gewicht und Geschlecht aus [Baj07] verwendet, um Personen zu generieren sowie zufällige tägliche Berichte und Arztbesuche in Abhängigkeit vom Stammwert einer Person. Zum Beispiel ist die forcierte exspiratorische Vitalkapazität (FVC) von Größe in cm, von dem Alter in Jahren und vom Geschlecht abhängig. Sie beschreibt das Volumen, das nach maximaler Einatmung wieder mit maximaler Geschwindigkeit ausgestoßen werden kann. Die FVC wird im Generator folgend berechnet, um die Kennzahl zu erhalten, welchen Wert eine Person mit Geschlecht, Größe und Alter im Normalzustand aufweisen sollte:

Mann: FVC = (5.76 * L - 0.026 * A - 4.34) ±0.61
Frau: FVC = (4.43 * L - 0.026 * A - 2.89) ±0.43

Die Variable L steht für die Größe in cm, dass A für das Alter in Jahren. Die Zahlenkonstanten sind abhängig vom Geschlecht.

Ein anderes berechnetes Volumen ist das forcierte exspiratorische Volumen (FEV). Bei diesem Volumen handelt es sich um das maximal mögliche ausgeatmete Volumen pro Zeiteinheit. Wird das Volumen für eine Sekunde bestimmt, dann spricht man von FEV_1. FEV_1 wird wie folgt im Generator berechnet:

Mann: FEV_1 = 4.3 * L - 0.029 * A - 2.492
Frau: FEV_1 = 3.95 * L - 0.026 * A - 2.604

Die Variable L beschreibt wieder die Größe in cm, das A das Alter in Jahren. Die Zahlenkonstanten sind ebenfalls wieder abhängig vom Geschlecht.

Bei der Erstellung einer Person werden Stammwerte erzeugt, von denen die neuen Datensätze des Patienten sich nur um ± 15% unterscheiden. Somit wird ein eher realistischer Krankheitsverlauf simuliert, der keine extremen Sprünge enthält. Alle Werte liegen bei der Erzeugung in realistischen Wertgrenzen. Der Status, ob eine Exazerbation stattfand, wird anhand eines Punktesystems festgelegt. Sind mehr als 50% der Werte um 20% schlechter, führt dies zum Status, dass eine Exazerbation stattfand. Es kann jedoch angenommen werden, dass dieser Status bzw. die Berechnung nicht mit realen Szenarien übereinstimmt.
Für das Testverfahren der frühzeitigen Vorhersage von Exazerbationen wurden 100 Patienten generiert, die rund 15.000 tägliche Berichte erzeugt haben. Die erzeugten Berichte führen zu rund 15.000 Datensätzen für das Data Mining-Verfahren. Für die Evaluierung wurden bei der ersten Durchführung alle Datensätze mit der Wahrscheinlichkeit einer Exazerbation mit hoch markiert. Das bedeutet, dass alle Wertbereiche der täglichen Berichte ebenfalls zu einer Klassifizierung mit hoher Wahrscheinlichkeit führen. Alternativ wurden alle Datensätze mit niedriger Wahrscheinlichkeit markiert - das führte zu einer Klassifikation mit niedriger Wahrscheinlichkeit bei einem neuen täglichen Bericht. Abschließend wurde ein dritter Durchlauf gestartet mit zufälliger Klasseneinteilung. Die Klassen der Berichte sind jedoch von den Werten der Stammdaten des Patienten abhängig. Die Auswertung

Listing 5.1 MySQL-Abfrage für die Performance-Messung.

```
SELECT *
FROM dailyReports
WHERE dailyReports.recordId = (SELECT MAX(dailyReports.recordId) FROM dailyReports) AND
    dailyReports.q1 = true;
```

ergab eine Übereinstimmung von 17% mit der durch Data Mining bestimmten Wahrscheinlichkeit der Exazerbation zu dem von Stammdaten zufällig generierten Status der Exazerbation durch den Generator. Ob es sich dabei um realistische Werte handeln kann, ist nicht analysierbar. Die Daten für die Data Mining-Analyse wurden generiert und stellen keine Daten von realen Personen dar. Auch die Wahrscheinlichkeit der Erkennung, dass eine Exazerbation bevorsteht, lässt sich nicht überprüfen, da der Status ebenfalls vom Generator zufällig durch ein Punktesystem berechnet wurde, das die reale Abbildung nicht darstellen kann. Das Punktesystem wird auf alle Werte des Patienten angewendet. Jeder Wert der ± 10% des vorherigen Wertes des Datensatzes übersteigt, wird mit einem Punkt markiert. Bei booleschen Werten wird die Änderung des Wertes ebenfalls mit einem Punkt gewertet. Werden bei den ersten Datensätzen mehr als 10 Punkte festgestellt, wird der Status Exazerbation gesetzt. Ist das nicht der Fall, erhält der Status die Angabe Baseline. Der Wert pro Datensatz wird in einer Liste gespeichert. Für 10 als Datemnsatz wird als Grenzwert für die Punkteanzahl der Durchschnitt aller Datensätze verwendet, die bereits berechnet wurden.

Aus den zufällig berechneten Werten kann nur festgestellt werden, dass die Analyse per Data Mining über das Analyse Management System möglich ist. Inwieweit die Klassifikation mit realen Bedingungen übereinstimmt oder wie genau eine Exazerbationsvorhersage möglich ist, dazu kann keine Aussage getroffen werden. Das lässt sich erst mit realen Gesundheitsdaten ermitteln.

5.3 Performance & Skalierung

Für die Analyse der Performance und Skalierung wurde das Analyse Management System mit der ECHO-Plattform auf einer virtuellen Maschine betrieben. Das Datenbanksystem der ECHO-Plattform wurde ebenfalls auf derselben virtuellen Maschine ausgeführt. Der virtuellen Maschine stehen zwei 2 CPU-Kerne (i7-3615QM) zur Verfügung, 4GB RAM und eine SSD für die persistente Speicherung der Daten. Das Datenbanksystem dient zudem dazu, die Performance zwischen dem Analysesystem und einer Datenbank zu vergleichen. Der Vergleich zwischen einer Datenbank und dem Analysesystem wird auf der virtuellen Maschine ausgeführt. Dabei wird für den Vergleich eine Datenanalyse verwendet, die einen Datenwert mit einem Vergleichswert überprüft, ob die Werte übereinstimmen. Beide Systeme verwenden dieselbe Menge an Datensätzen: 5500 Stück. Es wird die Tabelle **dailyReports** mit den täglichen Berichten der ECHO-Plattform verwendet. Für den Vergleich der Performance wird die Testanalyse ebenfalls direkt auf der Datenbank durchgeführt, indem die Testanalyse als MySQL-Abfrage implementiert wird. Für die Implementation wird der WHERE-Bestandteil der MySQL-Abfrage dazu verwendet, die Regel zu implementieren. Im WHERE-Bestandteil der MySQL-Abfrage wird das Datenfeld mit dem Vergleichswert gleichgesetzt. Für die Testanalyse soll der letzte Datensatz dahingehend analysiert werden, ob er den Wert „true“ im Datenfeld „q1“ enthält. Für den Vergleichstest wurde die MySQL-Abfrage in Listing 5.1 verwendet.

Listing 5.2 Analysatorvorlage zur Prüfung der Performance mit einer Analyseregel.

```
{
      "structure": [
            {
                  "type": "node",
                  "field": "daily_reports.q1",
                  "exp": "=",
                  "value": true,
                  "range": "NULL"
            }
      ],
      "event": {
            "type": "newDailyReport",
            "parm": {}
      },
      "action": {
            "type": "sendMessage",
            "parm": {"message": "Regel erfuellt!"}
      }
}
```

Tabelle 5.3: Auswertung der Performance.

System	Ausführung einer Analyse	Ausführung von 10 Analysen (sequentiell)
MySQL Datenbank	91,29ms	726,43ms
Regelbasiertes Analysesystem	173,71ms	1857,65ms

Die MySQL-Abfrage verwendet die Tabelle **dailyReports** und ruft zuerst die „recordId“ des zuletzt hinzugefügten Datensatzes ab. Die „recordId“ wird dann verwendet, um den letzten Datensatz zu identifizieren und zu prüfen, ob der Datensatz in der Spalte „q1“ den Wert „true“ enthält. Ist das der Fall, wird der Datensatz zurückgeliefert. Liefert die MySQL-Abfrage keinen Datensatz zurück, ist die Regel nicht erfüllt. Liefert die Abfrage einen Datensatz zurück, ist die Regel erfüllt. Simultan dazu wird die Analysatorvorlage in Listing 5.2 verwendet, um die Analyse auf dem Analyse Management System auszuführen. Die Analysatorvorlage prüft den letzten Datensatz aus der Tabelle **dailyReports**, wenn ein neuer Datensatz erzeugt wird. Das Datenfeld „q1“ im Datensatz wird mit dem Vergleichswert „true“ überprüft. Sind beide Werte gleich, gilt die Regel als erfüllt und eine Nachricht wird versendet. Für die Messung der Performance wird die Ausführungsgeschwindigkeit gemessen, die das Datenbanksystem und das Analyse Management System benötigt. Die Messung der Ausführung erfolgt direkt am Server. Die Übertragung zum Client wird nicht gemessen.

Die Tabelle 5.3 zeigt die Auswertungsergebnisse der Performance zwischen dem Analysesystem und einer MySQL-Datenbank. Die Ergebnisse dokumentieren die Ausführungszeit der Analyse. Die Verarbeitung der MySQL-Datenbank ist deutlich schneller bei der Analyse der Daten als das Analyse Management System. Das Analyse Management System benötigt mit 173,71ms fast die doppelte Ausführungszeit. Das ist dadurch bedingt, dass das Analyse Management System die Daten über

Webschnittstellen abfragen und übertragen muss, was die Ausführungszeit durch mehr Operationen erhöht. Die Daten werden über die Webschnittstelle der ECHO-Plattform geladen. Dabei werden die Daten über die MySQL-Datenbank abgerufen. Zusätzlich werden die Datensätze über den Webserver übertragen, im Analyse Management System verarbeitet und die Aktion ausgeführt. Somit benötigt das Analyse Management System die Kommunikation über zwei Webschnittstellen für die Analyse. Benötigt wird die Abfrage der Daten und das Ausführen der Aktion. Das bedeutet erheblich mehr Aufwand und Laufzeit als die direkte Ausführung der Analyse auf der Datenbank, die bereits die Daten gespeichert hat. Der Vorteil des Analyse Management System ist nicht die Analysegeschwindigkeit. Auch bei der sequentiellen Ausführung von 10 Analysen benötigt das Analyse Management System mehr als doppelt so viel Zeit. Das bedeutet auch, dass je mehr Daten abgefragt werden müssen oder Daten aus unterschiedlichen Tabellen geladen werden müssen, der Abstand zur MySQL-Datenbank weiter steigt. Grund dafür ist die Tatsache, dass für jedes weitere Datenfeld eine weitere Verbindung über die Webschnittstelle aufgebaut werden muss, um die Daten zu übertragen. Zusätzlich verwendet das Analyse Management System keinen Cache. Wird in einer Regel dasselbe Datenfeld mehrmals verwendet, benötigt das Analyse Management System für jede Regel eine eigene Abfrage der Daten. Die geladenen Daten werden nicht wiederverwendet.

Für den Test der Skalierung des Analyse Management Systems wurde als Referenzwert eine Analyse ausgeführt. Die Regeln dieser Analyse wurden dann verwendet, um die Analyse zu duplizieren. Die Analysen wurde auf einem Analyse Management System erzeugt. Zur Überprüfung der Skalierung mit einer Analyse folgt der Test mit 50 Analysen desselben Typs und derselben Eigenschaft. Als Analyse wurde die Analysatorvorlage in Listing 5.2 verwendet. Der einzige Unterschied zur MySQL-Abfrage ist das Versenden einer Nachricht. Das Versenden der Nachricht wurde jedoch nicht mit in die Laufzeit einberechnet. Das Analyse Management System betreibt die Analysen getrennt, somit müssen die Daten für jede Analyse separat geladen werden. Ein Austausch der geladenen Daten zwischen den Analysen erfolgt nicht.

Tabelle 5.4: Skalierung der Ausführungszeit des Analyse Management Systems.

Anzahl Analysen (parallel)	**Anzahl Ausführungssysteme**	**Ausführungszeit**
1	1	165,44ms
10	1	1342,75ms
50	1	5513.51ms
50	2 (50:50)	2986,52ms

Die Tabelle 5.4 zeigt die Auswertung der Skalierung. Die Ausführung einer Analyse benötigt 165,44ms. Bei der parallelen Ausführung von 10 Analysen auf einem Ausführungssystem benötigt das Analyse Management System mit 1342,75ms fast die zehnfache Zeit einer Analyse. Bei der Ausführung von 50 Analysen verfünfzigfacht sich die Ausführungszeit. Der Grund dafür ist, dass Node-Red nur in einem Prozess (single-threaded) läuft und die Funktionsbausteine zwar parallel, aber nur auf einem CPU-Kern verarbeitet. [Ler11] Das wird dadurch verursacht, dass Nodejs, das von Node-Red zur Ausführung verwendet wird, zwar eventbasierend asynchron ausgeführt werden kann, aber ebenfalls nur single-threaded ausgeführt wird, was nur einen Kern verwendet. Das bedeutet, dass die Auslastung bei einem Kern bei 100% liegt und die restlichen CPU-Ressourcen ignoriert werden. Somit

können mehrere Analysen parallel zwar ausgeführt werden, blockieren jedoch die CPU-Ressourcen des einzig verfügbaren Kerns. Eine Verbesserung der Skalierung ist möglich mit der Verwendung mehrerer Node-Red Instanzen auf einer oder auf unterschiedlichen Maschinen. Die Anzahl der Node-Red-Instanzen muss mit der Anzahl an CPU-Kernen übereinstimmen, um diese Ressourcen verwenden zu können. Werden die Analysen auf zwei Node-Red-Instanzen verteilt, welche zwei unterschiedliche CPU-Kerne verwenden, halbiert sich fast die benötigte Ausführungszeit mit nur 2986,52ms zu 5513,51ms. Damit zeigt sich eine Skalierung der Ausführungszeit mit der Verwendung von mehreren Ausführungssystemen.

Eine besondere Einschränkung bilden Data Mining-Analysen. Bei der Ausführung der Ermittlung der Wahrscheinlichkeit einer Exazerbation führt es bei einer Analyse zu einer Laufzeit von über 60 Sekunden. Dies begründet sich aus der Menge von 15.000 Datensätzen, die übertragen werden müssen, der Erstellung des Entscheidungsbaums und der Klassifikation des Datensatzes. Dies führt bei der Ausführung der Analyse zu einer Auslastung von 100% des verwendeten CPU-Kerns. Außerdem hat es die Blockierung des Ausführungssystems und der Webschnittstellen durch die hohe Auslastung zur Folge. In dieser Zeit ist weder das Ausführungssystem ansprechbar, noch können Analysen erzeugt oder ausgeführt werden. Aus diesem Grund sollten Data Mining-Analysen immer auf einem Ausführungssystem erzeugt werden, das keine weiteren Analysen oder das Analyse Management System ausführt oder für neue Analysen verwendet wird. Alternativ kann statt der Verwendung vom JavaScript-Bibliotheken für Data Mining eine externe Anwendung verwendet werden, die über einen Funktionsblock angesteuert wird. Dann kann die Rechenlast außerhalb der Node-Red-Instanz ausgeführt werden. Beide Lösungen verhindern die Störung des Analyse Management System oder der Ausführung von Analysen.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein regelbasiertes Analysesystem für Gesundheitsdaten entwickelt, das in die Enhancing Chronic Patients' Health Online (ECHO) Plattform integriert werden kann. Bei der ECHO-Plattform handelt es sich um einen Onlinedienst zur Erfassung von Gesundheitsdaten von Patienten mit der chronisch obstruktiven Lungenerkrankung. Es entstand aus einem Forschungsprojekt, gefördert durch das Bundesministerium für Forschung und Bildung (BMBF). Das Analysesystem soll die Anwender mit spezifischen Analysen unterstützen, um die Durchführung von Diagnosen und Behandlungen zu verbessern. Es soll dem ärztlichen Personal wie den Patienten selbst bei der Früherkennung von Verschlimmerungen der chronisch obstruktiven Lungenerkrankung helfen und so die Lebensqualität der Patienten steigern.

Vor der Implementierung des regelbasierten Analysesystems verwendete die ECHO-Plattform Datenanalysen, die fest als MySQL-Abfrage in den Programmcode der Plattform integriert wurden. Eine Anpassung oder Erweiterung erforderte bisher immer die Anpassung des Quellcodes. Anwendungsspezifische Analysen oder das automatische Hinzufügen von weiteren Analysen durch das ärztliche Personal waren nicht möglich.
Deshalb wurde eine Analysatorvorlage entwickelt, die eine Analyse beschreiben kann. Die Analysatorvorlage verwendet eine Regelsyntax zur Beschreibung der Datenanalyseregeln mit Analyseoperator, Datenwert und Vergleichswert. Zusätzlich beschreibt die Analysatorvorlage, wann die Analyse ausgeführt werden soll und welche Aktion durchgeführt wird, wenn die definierten Regeln erfüllt werden. Zur Übertragung der Analysatorvorlage wurde das Datenformat JSON ausgewählt und für die Kommunikation mit anderen Systemen eine REST API bereitgestellt, um eine homogene Integration in die ECHO-Plattform zu erreichen. Für den Zugriffsschutz auf die sensiblen Gesundheitsdaten nutzt das Analysesystem nur die Berechtigungen der Anwender, welche das Analysesystem verwendet, und greift auf die Webschnittstellen der ECHO-Plattform zu. Für die Möglichkeit zur schnellen Erweiterung des Analysesystems wurde besonderer Wert auf das generische System gelegt. Das Analysesystem ist unabhängig von der Datenquelle und benötigt lediglich einen Transformationsadapter für die Daten, der direkt im Analysesystem programmiert werden kann. Es können Analyseoperatoren hinzugefügt werden, und es gibt die Möglichkeit der Erweiterung von Events und auszuführenden Aktionen, die über eine REST API ausgeführt werden können.

Das Analysesystem verwendet ein Ausführungssystem namens Node-Red an Stelle eines Datenbanksystems oder statt Complex Event Processing. Bei Node-Red werden Ablaufpläne ausgeführt, die als Funktionsbausteine deklariert werden. Node-Red ist Open Source und verwendet - wie die ECHO-Plattform - Nodejs als Laufzeitumgebung. Damit ist es möglich, ein Analysesystem auszuführen, das unabhängig von einem Datenbanksystem ist und zur Laufzeit den Austausch der Analyseregeln ermöglicht. Der Parser im Analysesystem auf der Node-Red-Instanz verwendet die Analysatorvorlage, um daraus den Ablaufplan für die Node-Red-Instanz zu generieren. Das Analysesystem kommuniziert

mit den Analysatoren über das MQTT-Protokoll, das dem Publish/Subscribe-Paradigma folgt, und erlaubt damit eine Multi-Serverarchitektur. Analysatoren können über mehrere Node-Red-Instanzen erzeugt werden und effektiv über das MQTT-Protokoll durch Events gestartet werden. Die Vermittlung der Events übernimmt dabei der Message-Broker Mosquitto.

Die Evaluierung des Analysesystems zeigt, dass die aktuelle Ausführungslaufzeit nicht mit einer Datenbank konkurrieren kann. Das ist bedingt dadurch, dass das Analysesystem selbst eine Datenbank verwendet, um die Daten abzurufen, was die Analyse erheblich verzögert. Auch zeigt es durch die Verwendung nur eines CPU-Kerns, bedingt durch die Plattform, eine langsamere Verarbeitung. Die Evaluierung beweist jedoch auch, dass durch eine Skalierung mit mehreren Node-Red-Instanzen eine erhebliche Steigerung der Performance möglich ist.
Zur Vorhersage der Exazerbationswahrscheinlichkeit wurde prototypisch eine Data Mining-Analyse als Klassifikation mit der Verwendung eines Entscheidungsbaums implementiert, die ebenfalls das Ausführungssystem Node-Red für die Berechnung verwendet. Es soll dazu dienen, zu demonstrieren, dass das Analysesystem Data Mining-Analysen ausführen und das Analysesystem zur Früherkennung einer Exazerbation dienen kann. Die Evaluierung der Data Mining-Analyse zeigt auch, dass eine Möglichkeit der Früherkennung besteht, jedoch auf Grund mangelnder Daten realer Patienten nur mit generierten Daten getroffen werden kann.

Ausblick

Die regelbasierte Analyse wurde im Rahmen dieser Arbeit entwickelt und bietet prototypisch die Möglichkeit der Verwendung für Data Mining-Analysen. Bei der Data Mining-Analyse wurde nur eine Analyse entwickelt, die eine Vorhersage der Exazerbationswahrscheinlichkeit ermöglicht. Zur Verbesserung muss die Exazerbationswahrscheinlichkeitsanalyse noch dahingehend mit realen Daten geprüft werden, inwieweit eine konkrete Abschätzung des Zeitraums möglich ist. Sollte eine Validierung der Analyse mit realen Daten möglich sein, kann evaluiert werden, ob weitere Data Mining-Analysen integriert werden sollten, um weitere Vorhersagen zu treffen, die für den Patienten die Lebensqualität verbessern können.

Die Data Mining-Analysen könnten auch auf externe Systeme ausgegliedert werden, statt Node-Red zu verwenden. Die Ausführung könnte aus einem Funktionsbaustein erfolgen. Dies würde die Performance steigern. Außerdem stünde eine Vielzahl von Data Mining-Werkzeugen zur Verfügung.

Das Analysesystem könnte mit weiteren Analyseoperatoren ausgestattet werden. Das würde dazu führen, dass die Regeln der Analyse vereinfacht werden können, weil sie zu einem Operator zusammengefasst werden. Weitere Analyseoperatoren würden auch die Fähigkeit zur Beschreibung zusätzlicher Analysen steigern, indem mehr Funktionen bereitstehen.

Besonderes Augenmerk muss auch auf das Ausführungssystem gelegt werden. Die Evaluierung der Performance zeigt, dass aktuell die Ausführungsgeschwindigkeit durch die Nodejs Laufzeitumgebung begrenzt wird. Eine Integration der Unterstützung von Mehrkernsystemen in Nodejs könnte die Leistung des Systems erheblich steigern. Die Leistung und besonders die Skalierung könnte

gesteigert werden, wenn jede Analyse auf einem eigenen Ausführungssystem erzeugt werden würde oder auf einer physischen Maschine Virtualisierung wie Docker zum Einsatz käme, um mehrere Kerne auslasten zu können. Auch könnte bei der Auswahl des Ausführungssystems eine neue Instanz auf einer Cloud-Plattform erzeugt werden und den Analysator darauf aktivieren. So könnte auch verhindert werden, dass eine Analyse andere Analysen blockiert.

Literaturverzeichnis

[Agg15] C. C. Aggarwal. *Data Mining - The Textbook*. Springer, Berlin, Heidelberg, 2015. (Zitiert auf den Seiten 22, 23, 24 und 25)

[Baj07] D. Bajraktarevic. *Pneumologie pocket*. Börm Bruckmeier, Grünwald, 2007. (Zitiert auf Seite 83)

[BEE+10] M. Blount, M. Ebling, J. Eklund, A. James, C. McGregor, N. Percival, K. Smith, D. Sow. Real-Time Analysis for Intensive Care: Development and Deployment of the Artemis Analytic System. *Engineering in Medicine and Biology Magazine, IEEE*, 29(2):110–118, 2010. doi:10.1109/MEMB.2010.936454. (Zitiert auf Seite 14)

[BKK+14] M. Bitsaki, C. Koutras, G. Koutras, F. Leymann, B. Mitschang, C. Nikolaou, N. Siafakas, S. Strauch, N. Tzanakis, M. Wieland. An Integrated mHealth Solution for Enhancing Patients' Health Online. In *Proceedings of the 6th European Conference of the International Federation for Medical and Biological Engineering (MBEC'14)*, S. 1–4. International Federation for Medical and Biological Engineering (IFMBE), 2014. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2014-61&engl=. (Zitiert auf Seite 29)

[BPS10] H. Boley, A. Paschke, M. O. Shafiq. RuleML 1.0: The Overarching Specification of Web Rules. In *Semantic Web Rules - International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings*, S. 162–178. 2010. doi:10.1007/978-3-642-16289-3_15. URL http://dx.doi.org/10.1007/978-3-642-16289-3_15. (Zitiert auf Seite 18)

[Cle08] T. Cleff. *Deskriptive Statistik und moderne Datenanalyse Eine computergestützte Einführung mit Excel, SPSS und STATA*. Betriebswirtschaftlicher Verlag Dr. Th. Gabler / GWV Fachverlage GmbH, Wiesbaden, Wiesbaden, 2008. (Zitiert auf Seite 16)

[cop] Chronic obstructive pulmonary disease (COPD). URL http://www.netdoktor.de/krankheiten/copd/. (Zitiert auf den Seiten 9 und 10)

[dat] Richtlinie zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr. URL http://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:31995L0046. (Zitiert auf Seite 38)

[DPNB11] M. Deng, M. Petkovic, M. Nalin, I. Baroni. A Home Healthcare System in the Cloud-Addressing Security and Privacy Challenges. In L. Liu, M. Parashar, Herausgeber, *IEEE CLOUD*, S. 549–556. IEEE, 2011. (Zitiert auf Seite 38)

[Hil03] E. F. Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003. (Zitiert auf Seite 19)

[HWS+15] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, F. Leymann. SitRS - A Situation Recognition Service based on Modeling and Executing Situation Templates. In C. Nikolaou, F. Leymann, Herausgeber, *Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing*, S. 35–49. IBM, 2015. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2015-34&engl=0. (Zitiert auf Seite 14)

[Kam] U. Kamps. Gabler Wirtschaftslexikon: Datenanalyse. URL http://wirtschaftslexikon.gabler.de/Archiv/1823/datenanalyse-v11.html. (Zitiert auf Seite 16)

[Lam13] A.-L. Lamprecht. *User-Level Workflow Design - A Bioinformatics Perspective.*, Band 8311 von *Lecture Notes in Computer Science*. Springer, 2013. URL http://dx.doi.org/10.1007/978-3-642-45389-2. (Zitiert auf Seite 21)

[Ler11] R. M. Lerner. At the Forge: Node.JS. *Linux J.*, 2011(205), 2011. URL http://dl.acm.org/citation.cfm?id=1972989.1972995. (Zitiert auf Seite 86)

[LGG11] B. Linke, R. Giegerich, A. Goesmann. Conveyor: a workflow engine for bioinformatic analyses. *Bioinformatics*, 27(7):903–911, 2011. doi:10.1093/bioinformatics/btr040. (Zitiert auf Seite 15)

[mqt] MQTT: Frequently Asked Questions. URL http://mqtt.org/faq. (Zitiert auf Seite 66)

[Mü06] J. Müller. *Workflow-based Integration - Grundlagen, Technologien, Management*. Springer-Verlag, Berlin Heidelberg New York, 1. aufl. Auflage, 2006. (Zitiert auf Seite 21)

[nod] Node-Red Documentation. URL http://nodered.org/docs/getting-started/running.html. (Zitiert auf den Seiten 42 und 44)

[NRD06] C. Nagl, F. Rosenberg, S. Dustdar. VIDRE–A Distributed Service-Oriented Business Rule Engine based on RuleML. In *Enterprise Distributed Object Computing Conference, 2006. EDOC '06. 10th IEEE International*, S. 35–44. 2006. doi:10.1109/EDOC.2006.67. (Zitiert auf Seite 18)

[OMG05] OMG. Unified Modeling Language SpecificationVersion 1.4.2. http://www.omg.org/spec/UML/ISO/19501/PDF/, 2005. URL http://www.omg.org/spec/UML/ISO/19501/PDF/. (Zitiert auf Seite 21)

[Org] W. H. Organization. Chronic obstructive pulmonary disease facts. URL http://www.who.int/respiratory/copd/en/. (Zitiert auf Seite 9)

[Qui14] J. R. Quinlan. *C4.5 - Programs for Machine Learning*. Elsevier, Amsterdam, 1. aufl. Auflage, 2014. (Zitiert auf Seite 72)

[RR14] W. Raghupathi, V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1):3, 2014. doi:10.1186/2047-2501-2-3. URL http://dx.doi.org/10.1186/2047-2501-2-3. (Zitiert auf Seite 13)

[SMFGJ15] D. Sanchez-Morillo, M. Fernandez-Granero, A. Jiménez. Detecting COPD exacerbations early using daily telemonitoring of symptoms and k-means clustering: a pilot study. *Medical & Biological Engineering & Computing*, 53(5):441–451, 2015. doi:10.1007/s11517-015-1252-4. URL http://dx.doi.org/10.1007/s11517-015-1252-4. (Zitiert auf Seite 16)

[Ste14] F. Steimle. *Umsetzung eines sicheren Systems zur Verwaltung und Bereitstellung von Gesundheitsdaten.* Diplomarbeit, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2014. URL http://elib.uni-stuttgart.de/opus/volltexte/2014/9696. (Zitiert auf den Seiten 9, 10, 14 und 41)

[Sue02] S. Suehring. *MySQL Bible.* Wiley, New York, 1. auflage Auflage, 2002. (Zitiert auf Seite 73)

[Tei12] P. Teixeira. *Professional Node.Js: Building Javascript Based Scalable Software.* Wrox Press Ltd., Birmingham, UK, UK, 1st Auflage, 2012. (Zitiert auf Seite 42)

Alle URLs wurden zuletzt am 09. 12. 2015 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift