ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Process Mining und regelbasierte Analysen in einem prozessorientierten Fragebogensystem.

Masterarbeit an der Universität Ulm

Vorgelegt von:
Wolfgang Blocherer
wolfgang.blocherer@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Vera Künzle

Betreuer:
Johannes Schobel

2015

Fassung 6. Januar 2016

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Bei der Durchführung von Studien, wird in Anwendungsdomänen wie der Medizin vorwiegend auf papiergebundene Fragebögen gesetzt. Diese verursachen jedoch einen hohen Aufwand, besonders im Hinblick auf die spätere Auswertung der erhobenen Daten. Für eine Auswertung, bspw. in einem Tabellenkalkulationsprogramm, ist es erforderlich, dass die Daten digital vorliegen. Hierzu müssen die Papierbögen jedoch zunächst von einem Mitarbeiter übertragen werden. Neben den hohen Kosten die dadurch entstehen, können durch Tippfehler zudem die Auswertungsergebnisse verfälscht werden. Digitale Fragebogensysteme lösen diese Problematik, indem Fragebögen bspw. über eine Web-Oberfläche erfasst werden.

Nachdem die Fragebögen somit bereits digital durch den Patienten erfasst werden, kann eine automatisierte Auswertung erfolgen. Dies steht im Fokus der vorliegenden Ausarbeitung. Insbesondere werden hierfür regelbasierte Analysen betrachtet. Vom Anwender sollen auf einfache Weise (auch komplexe) Regeln erstellt werden können, die zur Auswertung der Fragebögen herangezogen werden können. Für weitere Analysen, sollen die Daten zudem in verschiedene Formate transformiert werden können. Speziell sollen Formate für Process Mining Analysen bereitgestellt werden. Im Rahmen dieser Ausarbeitung wird hierfür eine Analysekomponente für ein digitales Fragebogensystem entwickelt.

Inhaltsverzeichnis

1

Einleitung

Die Durchführung von medizinischen Studien mit Fragebögen findet vorwiegend in papiergebundener Form statt [1]. Stapelweise Papier wird gedruckt, geheftet und an Patienten bzw. Studienteilnehmer ausgehändigt. Nachdem die Studienteilnehmer die Fragebögen ausgefüllt haben, werden die Papiermappen wieder von einem Mitarbeiter des Studien-Teams eingesammelt. Anschließend, wenn alle ausgefüllten Fragebögen vorliegen, müssen diese zunächst in ein digitales Dokument übertragen werden. Bspw. mit einem Tabellenkalkulationsprogramm. Ein sehr zeitaufwendiger Prozess, da die Fragebögen von einem Mitarbeiter abgetippt werden müssen. Erst wenn die Fragebögen digitalisiert vorliegen, kann mit der Auswertung begonnen werden.

Da die Fragebögen abgetippt werden, können hierbei jedoch Übertragungsfehler passieren. Neben dem hohen Arbeitsaufwand, stellt das Abtippen somit zudem eine Fehlerquelle dar. Beides ist für die Durchführung von Studien ein Problem. Mit der nötigen

Arbeitszeit werden hohe Kosten verursacht und durch die Einbußen in der Datenqualität können die Auswertungsergebnisse verfälscht werden.

Um den dargestellten Ablauf zu verbessern, wird vom Institut für Datenbanken und Informationssysteme der Universität Ulm (DBIS) ein digitales Fragebogensystem entwickelt. Der Einsatz von mobilen Geräten wie Tablet-Computern zur flexiblen Datenerfassung stellte sich dabei als wichtige Anforderung an das System heraus. [1]

Digitale Fragebogensysteme, bspw. web-basierte Online Services, existieren bereits [2]. Diese lassen sich auch auf mobilen Geräten ausführen. Aufgrund der Sensibilität und Vertraulichkeit von Patientendaten, ist es jedoch oft nicht zulässig die Daten über das Internet zu transportieren. Zudem werden die Daten auf diese Weise nicht lokal im Verantwortungsbereich des Studien-Teams gespeichert, sondern beim Anbieter des Fragebogensystems.

Systeme die lokal installiert und betrieben werden können, existieren ebenfalls bereits [3]. Allerdings decken auch diese Systeme nicht alle Anforderungen an eine mobile und flexible Datenerfassung ab [1]. So gibt es bspw. für die mobilen Web-Clients keinen Offline-Modus. Die mobilen Geräte müssen für die Dauer der Befragung über eine Netzwerkverbindung verfügen.

1.1 Problemstellung

Im Gegensatz zu papiergebunden Fragebögen, liegen die Daten mit dem neuen System nun von Beginn an digital vor. Neben der reinen Datenerfassung mit Fragebögen, soll jedoch darüber hinaus der gesamte Zyklus eines Fragebogensystems verbessert werden. Der nächste Schritt in diesem Zyklus ist die Analyse der Fragebögen. [4]

Die Analyseanforderungen an das Fragebogensystem stehen im Fokus dieser Ausarbeitung. So wird eine Verbesserung der Auswertbarkeit angestrebt, in dem die Daten mit dem neuen System direkt in geeignete Auswertungsformate transformiert werden können. Zudem soll das System selbst Analysemöglichkeiten bieten. Insbesondere soll das System eine regelbasierte Analyse der Fragebögen ermöglichen.

Eine einfache Regel zur Auswertung von Fragebögen könnte lauten: *„Die Regel ist erfüllt, falls der Patient jünger als 50 Jahre ist und bisher zwei Vorsorgeuntersuchungen besucht hat“*. Eine Anforderung an die Regel ist, dass diese als mathematischer Ausdruck formalisiert werden kann. Bspw. mit booleschen Operatoren (und, oder, nicht) oder Vergleichsoperatoren (größer, kleiner, usw.). Die Komponente des Fragebogensystems mit der Regeln formuliert werden können, wurde bereits entwickelt [5]. Für die in dieser Ausarbeitung behandelten Analysen, ist die Formalisierung der Regeln somit bereits gegeben.

Regeln können jedoch komplexer sein, als das genannte Beispiel. Eine Regel kann neben einfachen Operatoren auch Funktionen enthalten. Dies können Standardfunktionen wie *Minimum*, *Maximum* und *Absolutbetrag* sein, sowie selbst-definierte Funktionen. Bspw. könnte eine selbst-definierte Funktion den Eingabewert mit einer Tabelle abgleichen. Die Tabelle könnte eine Einstufung in Risikogruppen enthalten oder Vergleichswerte von gesunden Personen.

Neben integrierten Analysemöglichkeiten, wie die vorgestellte Analyse mit Regeln, sollen durch Transformation der Daten weitere Analysen ermöglicht werden. Speziell sollen Process Mining [6] Analysen ermöglicht werden. Process Mining wird im (Geschäfts-) Prozessmanagement [7] verwendet um Geschäftsprozesse strukturiert zu analysieren. Diese Analyse-Technik kann allerdings auch zur Analyse von Fragebögen verwendet werden. Die Grundidee hierbei ist, dass ein Fragebogen als Geschäftsprozess abgebildet werden kann. Detailliert wird dies in [1] vorgestellt.

Für die Analysemöglichkeiten des Fragebogensystems ist dies besonders relevant, da das vom DBIS entwickelte Fragebogensystem auf Prozessmanagement-Technologie basiert. Im Unterschied zu Fragebögen, gibt es für Geschäftsprozesse bereits ausgereifte Technologien, welche die Modellierung, Ausführung und Analyse unterstützen. Durch das Mapping eines Fragebogens auf einen Geschäftsprozess werden diese Technologien für Fragebögen verfügbar gemacht.

1.2 Zielsetzung

Wie bereits erläutert, wurde mit der Umsetzung des Fragebogensystems begonnen. Im Rahmen dieser Ausarbeitung soll das Fragebogensystem nun um eine Analysekomponente erweitert werden. Die Analysekomponente soll in das bestehende Fragebogensystem integriert werden.

Abbildung 1.1 zeigt die schematische Architektur des Fragebogensystems. Das System besteht aus zwei Managementkomponenten, dem (Fragebogen-) Konfigurator und der Evaluation. Zudem gibt es einen zentralen Server sowie verschiedene Clients. Die Regelkomponente sowie die zu erstellende Analysekomponente sind Teil der Management Komponente Evaluation.

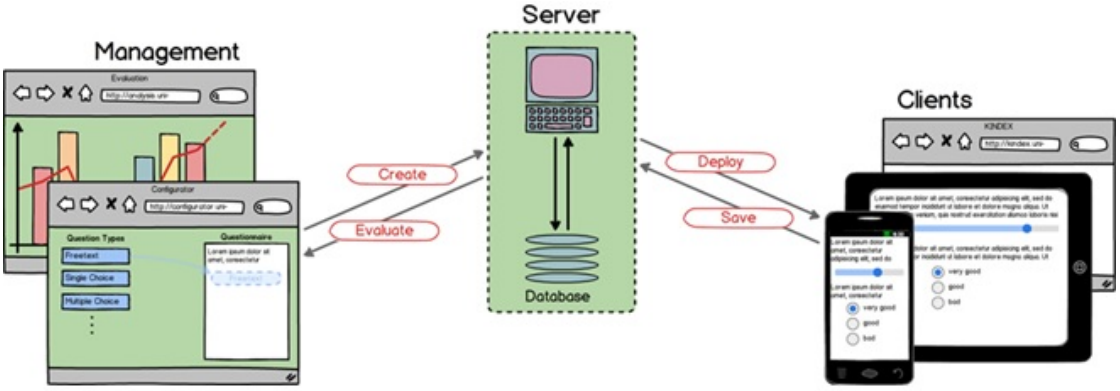

Abbildung 1.1: Architektur des Fragebogensystems [4]

Die zu erstellende Analysekomponente soll nun die auf dem Server gesammelten Fragebögen einlesen und verarbeiten können. Die dargestellten Analyseanforderungen sollen berücksichtigt werden. Hierfür sollen die mit der Regelkomponente erstellten Regeln eingelesen und auf die Fragebögen angewendet werden können. Die Analysekomponente muss daher in der Lage sein, die Regeln auswerten zu können und die Ergebnisse geeignet zu präsentieren. Zudem soll die Analysekomponente den Einsatz von Process Mining Technologie ermöglichen. Hierzu soll die Analysekomponente die gesammelten Daten in dafür geeignete Formate transformieren können.

1.3 Struktur der Arbeit

Zu Beginn werden in Kapitel 2 wichtige Grundlagen vorgestellt, die für das weitere Verständnis der Arbeit relevant sind. Anschließend werden in Kapitel 3 verwandte Arbeiten vorgestellt. In Kapitel 4 werden die Anforderungen an die zu erstellende Analysekomponente detailliert aufgeführt. Danach wird in Kapitel 5 das Analysekonzept und die Architektur der Analysekomponente vorgestellt. In Kapitel 6 wird die erstellte grafische Oberfläche vorgestellt, sowie relevante Aspekte der Implementierung. Abschließend wird in Kapitel 7 die Arbeit zusammengefasst, sowie ein Ausblick auf mögliche Erweiterungen der Analysekomponente gegeben.

2

Grundlagen

Dieses Kapitel fasst wichtige Grundlagen für das weitere Verständnis der Arbeit zusammen. Zunächst wird eine Einführung in das Geschäftsprozessmanagement gegeben. Anschließend wird der XES-Standard sowie die Eclipse 4 Rich Client Platform vorgestellt.

2.1 Geschäftsprozessmanagement

Geschäftsprozessmanagement ist relevant für private und öffentliche Organisationen wie Unternehmen, Krankenhäuser oder Kommunen. Das Ziel welches mit Geschäftsprozessmanagement verfolgt wird, ist die stetige Verbesserung von Geschäftsprozessen. Übergeordnet stehen hierbei Ziele der Organisationen wie die Gewinnung von neuen Kunden oder die Steigerung der Profitabilität. Die optimale Steuerung und kontinuierliche Verbesserung der Prozesse stellt damit eine entscheidende Möglichkeit dar, die

Leistungserstellung insgesamt zu verbessern. [7] definiert einen Geschäftsprozess und Geschäftsprozessmanagement (engl. Business Process Management, BPM) wie folgt:

- „Ein Geschäftsprozess besteht aus einem Set von Aktivitäten die koordiniert in einer Organisationsstruktur und technischen Umgebung ausgeführt werden. Jeder Geschäftsprozess wird von einer einzelnen Organisation initiiert, aber er kann mit Geschäftsprozessen interagieren die von anderen Organisationen ausgeführt werden.“
- „Business Process Management beinhaltet Konzepte, Methoden und Techniken die das Design, Administration, Konfiguration, Umsetzung und Analyse von Geschäftsprozessen unterstützen.“

Ein Beispiel für einen Geschäftsprozess, nachfolgend auch als Prozess bezeichnet, ist ein Bestellprozess. Verschiedene Aktivitäten werden dabei ausgeführt. Bspw. das Prüfen der Kreditwürdigkeit des Kunden. Des weiteren enthält ein Prozess Ereignisse und Entscheidungen. Ein Beispiel für eine zu treffende Entscheidung könnte das Reagieren auf die Prüfung der Kreditwürdigkeit des Kunden sein, ein Beispiel für ein Ereignis der Zahlungseingang des Kunden.

BPM zielt nicht primär darauf ab, einzelne Aktivitäten des Prozesses zu verbessern, sondern zielt auf die Verbesserung der Abfolge des gesamten Prozesses ab. Ziel hierbei ist bspw. die Sicherstellung einer konstant hohen Qualität der Prozessergebnisse. Wesentlich bei der Verbesserung von Prozessen mittels BPM ist die zeitliche Perspektive. Daher liefert BPM einen Ansatz, um Prozesse kontinuierlich zu verbessern. Die Rahmenbedingungen von Prozessen können sich laufend verändern. Es ist daher wichtig den Prozess den geänderten Rahmenbedingungen anzupassen, um dauerhaft eine gute Leistung des Prozesses sicherzustellen. [8]

2.1.1 BPM Lebenszyklus

Um dem Umstand sich verändernder Rahmenbedingungen gerecht zu werden, ordnet der BPM Lebenszyklus die verschieden BPM Teilaspekte in einem Kreislauf an, vgl.

Abbildung 2.1. Dieser wird kontinuierlich zur Steuerung und Optimierung der Prozesse durchlaufen.

1. *Prozess Identifikation* stellt den Einstiegspunkt zur Verbesserung von Geschäftsprozessen dar. Das zu lösende Problem wird benannt und alle hierfür relevanten Prozesse ermittelt. Die ermittelten Prozesse werden in einer Prozesslandkarte dargestellt. Die Prozesse werden voneinander abgegrenzt und Beziehungen zwischen Prozessen dargestellt.

2. Nachdem die Grenzen des Prozesses abgesteckt sind, wird im Schritt *Prozessentdeckung* der tatsächliche Ist-Zustand des Prozesses ermittelt. Hierzu werden z.B. Befragungen von am Prozess beteiligten Mitarbeitern durchgeführt. Als Ergebnis steht typischerweise ein detailliertes Prozessmodell.

3. In der *Prozessanalyse* wird der Ist-Prozess gründlich analysiert um Schwachstellen bzw. Probleme im Prozess zu identifizieren.

4. Um die gefundenen Probleme zu lösen, wird in der *Prozessüberarbeitung* versucht, Lösungen zu finden. Es wird versucht einen Soll-Prozess zu finden, der bekannte Schwachstellen behebt. Die Phase Prozessüberarbeitung wird Hand in Hand mit der Phase Prozessanalyse durchgeführt.

5. In der *Prozessumsetzung* wird der entworfene Prozess umgesetzt. Dies erfordert bspw. Änderungen in der Organisationsstruktur, neue Ressourcen und eine Anpassung der verwendeten Informationssysteme. Die verwendeten Informationssysteme und Sensoren sind die Datenquelle für die nachfolgende Prozessüberwachung.

6. Nachdem der Soll-Prozess umgesetzt wurde, wird in der *Prozessüberwachung* die Ausführung anhand der festgelegten Kriterien überwacht. Kommt es mit der Zeit zu Verschlechterungen der Leistung des Prozesses, wird eine erneute Überarbeitung des Prozesses angestoßen. [8]

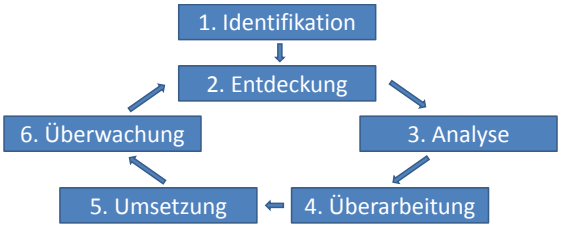

Abbildung 2.1: BPM Lebenszyklus nach [8]

2.1.2 Prozessmanagementsysteme

Die Ausführung von Prozessen wird durch Prozessmanagementsysteme (PMS) unterstützt. PMS nutzen für die Prozessausführung hinterlegte Prozessmodelle. In einem Prozessmodell ist die Ausführungslogik der Aktivitäten eines Prozesses abgebildet. Ein PMS stellt u.a. den Bearbeitern der einzelnen Aktivitäten personalisierte Arbeitslisten zur Verfügung. In diesen können die Bearbeiter die zu erledigenden Aufgaben einsehen. PMS verstecken dabei die Komplexität des Prozesses vor dem jeweiligen Bearbeiter und zeigen nur jene Aufgaben, die von ihm zu erledigen sind. Eine Aufgabe in der Arbeitsliste entspricht dabei einer Aktivität im Prozessmodell. Abhängig von den Werten, die in einer Aktivität erfasst wurden, ermittelt das PMS anhand der Ausführungslogik im Prozessmodell die nächste Aktivität. Bspw. durch Mengenüberschreitungen oder Ergebnisse eines Befundes werden vom PMS unterschiedliche nachfolgende Bearbeiter ermittelt. [7]

Während der Ausführung eines Prozesses generiert ein PMS Log-Daten zu den einzelnen Arbeitsschritten. Ein einzelner Log-Daten Eintrag wird als Ereignislog (engl. Event-Log) bezeichnet. Ein Ereignislog entsteht bspw. durch den Start oder das Abschließen einer Aktivität. Eine Aktivität kann somit in den Log-Daten durch mehrere Ereignislogs repräsentiert sein. Abbildung 2.2 zeigt beispielhafte Log-Daten eines PMS. Die Zeilen (Ereignisse) sind sortiert nach a) Instanz und b) Zeitstempel. Ein Ereignis verfügt über eine Instanz ID, die das Ereignis einer Instanz des Prozesses zuordnet. Zudem verfügt das Ereignis über eine eindeutige Ereignis ID, einen Zeitstempel, den Namen der Aktivität und des Bearbeiters (Ressource). Der Bearbeiter kann sowohl eine

Person als auch ein System sein. Darüber hinaus können Ereignislogs weitere inhaltliche Daten enthalten. [8]

Instanz	Ereignis	Zeitstempel	Aktivität	Ressource
1	Ch-4680555556-1	30.07.2012 11:14	Lagerbestand prüfen	SYS1
1	Re-5972222222-1	30.07.2012 14:20	Ware vom Lager abrufen	Rick
1	Co-6319444444-1	30.07.2012 15:10	Bestellung bestätigen	Chuck
1	Ge-6402777778-1	30.07.2012 15:22	Lieferadresse abrufen	SYS2
1	Em-6555555556-1	30.07.2012 15:44	Rechnung verschicken	SYS2
1	Re-4180555556-1	04.08.2012 10:02	Zahlung erhalten	SYS2
1	Sh-4659722222-1	05.08.2012 11:11	Produkt verschicken	Susi
1	Ar-3833333333-1	06.08.2012 09:12	Bestellung archivieren	DMS
2	Ch-4055555556-2	01.08.2012 09:44	Lagerbestand prüfen	SYS1
2	Ch-4208333333-2	01.08.2012 10:06	Materialbestand prüfen	SYS1
2	Re-4666666667-2	01.08.2012 11:12	Rohmaterial anfordern	Ringe

Abbildung 2.2: Log-Daten nach [8]

2.1.3 Process Mining

Unter Process Mining werden verschiedene Techniken verstanden, um an Informationen über die Ausführung eines Prozesses zu gelangen. Basis hierfür sind die von den Prozessmanagementsystemen erzeugten Log-Daten. In Abgrenzung zum klassischen Data Mining, welches datenorientiert ist, betrachtet Process Mining die Daten (Log-Daten) aus einer prozessorientierten Perspektive. Im BPM Lebenszyklus kann Process Mining besonders in den Phasen Prozessentdeckung und Prozessüberwachung eingesetzt werden. Mittels Process Mining kann aus den Log-Daten ein Prozessmodell rekonstruiert werden. Hierzu gibt es drei verschiedene Typen von Process Mining Verfahren: Erkennung, Konformitätsprüfung und Erweiterung.

Erkennung (engl. Process Discovery) ist das bekannteste der drei Verfahren. Es verwendet die Log-Daten als einzige Eingabe um daraus ein Prozessmodell zu rekonstruieren. Voraussetzung ist hierfür zunächst, dass die Log-Daten entsprechend vollständig von der ersten bis zur letzten Aktivität vorliegen. Mit Hilfe von Algorithmen kann aus den Log-Daten ein Prozessmodell konstruiert werden, bspw. ein Petri-Netz, UML-

Aktivitätsdiagramm, BPMN- oder EPK-Modell. Neben dem Prozessmodell können mit Process Dicovery Algorithmen auch Organisationsmodelle und Soziale Netzwerke ermittelt werden. Soziale Netzwerke zeigen auf, welche Mitarbeiter (Ressourcen) besonders häufig miteinander kommunizieren. [9, 10]

Verfahren zur Konformitätsprüfung (engl. Conformance Checking) überprüfen die Übereinstimmung der Log-Daten mit einem bereits existierenden Prozessmodell. Somit kann die tatsächliche Ausführung des Prozesses überwacht werden. Um die Übereinstimmung zu überprüfen gibt es zwei verschiedene Ansätze: Eine Möglichkeit ist das Nachspielen der Log-Daten im Prozessmodell. Hierbei wird die Ausführung simuliert und Schritt für Schritt anhand des Prozessmodells überprüft, ob der jeweilige Schritt zulässig ist. Die zweite Möglichkeiten ist das Testen der Log-Daten mit expliziten Regeln. Beispiele für Regeln sind Pflicht-, Ausschluss- oder die Reihenfolge von Aktivitäten. Anhand der Regeln kann gezielt nach Verstößen in den Log-Daten gesucht werden. Gibt es Fälle in denen Log-Daten nicht durch das Prozessmodell abgedeckt sind, sind zwei mögliche Ursachen zu betrachten. Die erste Möglichkeit ist, dass das Prozessmodell die Realität nicht korrekt abbildet und nachgebessert werden muss. Die zweite Möglichkeit ist, dass das Prozessmodell korrekt ist und es zu Ausnahmen in der Ausführung des Prozesses kam. [8, 6]

Process Mining Verfahren zur Erweiterung dienen dazu, bestehende Prozessmodelle zu erweitern oder korrigieren. Damit kann ein genaueres und aussagekräftigeres Prozessmodell erstellt werden. Ein Beispiel hierfür ist das Ermitteln von Entscheidungspunkten. Entscheidungspunkte sind Verzweigungen im Prozessmodell. Mit der Verwendung von Data Mining Verfahren, können Geschäftsregeln ermittelt werden, unter welchen Voraussetzungen welcher nachfolgende Ausführungspfad verwendet wird. Ein weiteres Beispiel von Verfahren zur Erweiterung stellt die Ermittlung von Performance Kennzahlen auf Basis der Event-Logs dar. Bspw. können die durchschnittliche Dauer von Aktivitäten oder Wartezeiten ermittelt werden. Performance Kennzahlen können in die Kategorien Zeit, Kosten, Qualität und Flexibilität aufgeteilt werden. Die Qualität eines Produkts oder einer Leistung ist oft nicht direkt anhand von Log-Daten erkennbar. Ein Indikator für Qualitätsmängel können sie dennoch sein. Wenn Teilprozesse einer Instanz wiederholt wurden, kann dies darauf hindeuten, dass es in einem ersten Durchlauf zu Problemen gekommen

ist. Die Flexibilität eines Prozesses kann ebenfalls ein Indikator für die Performance eines Prozesses sein. Messen lässt sich die Flexibilität eines Prozesses bspw. anhand der Anzahl von verschiedenen Ausführungsvarianten. Viele verschiedene Varianten haben oftmals negative Auswirkungen auf die Performance. [11, 8]

2.2 XES Standard

Der XES Standard (eXtensible Event Stream) ist ein XML basiertes Datenformat zur Speicherung von Ereignislogs. Seit 2010 ist der Standard in einer finalen Version verfügbar und ist anerkannter Standard für Ereignislogs der IEEE Task Force on Process Mining. Für Ereignislogs existieren eine Unmenge von verschiedenen Datenformaten. Viele Systeme die einen Log-Mechanismus implementieren, verwenden ein eigens für das System entwickeltes Log-Daten Format. Der XES Standard stellt einen Standard zur Verfügung, um Log-Daten zwischen Systemen und Anwendungsdomänen auszutauschen. Die primäre Zielsetzung des Standards ist die Verwendung für Process Mining Anwendungen. Er wurde jedoch so konzipiert, dass er auch für Data Mining Anwendungen verwendet werden kann. Bspw. zur Analyse von Log-Daten komplexer Maschinen wie Röntgengeräten oder Log-Daten die beim browsen auf einer Website entstehen. [12, 13]

2.2.1 Metamodell

In Abbildung 2.3 ist das Metamodell des XES Standards dargestellt. Das Log Objekt ist das Top-Level Objekt und enthält alle Log-Informationen zu einem spezifischen Prozess. Ein Log Objekt kann mehrere Trace (dt. Spur) Objekte enthalten. Ein Trace Objekt repräsentiert den Ablauf einer Prozessinstanz. Ein Trace Objekt enthält wiederum mehrere Event (dt. Ereignis) Objekte. Ein Event Objekt entspricht der kleinsten Granularität einer Prozessaktivität. Ein Event Objekt hat keine Dauer. Es kann bspw. den Anfang, eine Unterbrechung oder die Fertigstellung einer Aktivität markieren. In der XML Serialisierung des XES Standard entsprechen diese drei Objekte den Elementen <log>, <trace> und <event>. Listing 3.4 zeigt eine Beispiel XES Datei.

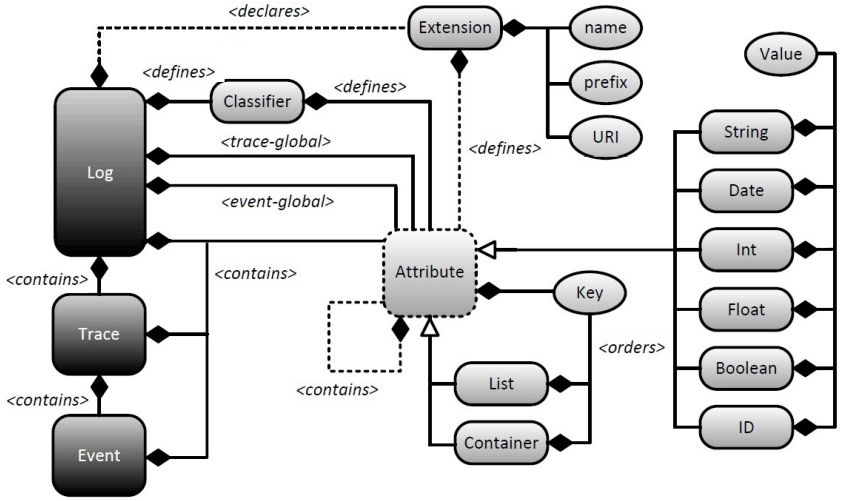

Abbildung 2.3: UML Klassendiagramm XES 2.0 Standard [14]

2.2.2 Attribute

Log, Trace und Event Objekte selbst enthalten keine Daten, sie bilden lediglich die Struktur der Log-Daten ab. Alle Informationen sind in Attributen abgelegt. Ein Attribut Objekt enthält einen eindeutigen Key innerhalb des umschließenden Objekts. Neben den sechs elementaren Attribut-Typen String, Date, Int, Float, Boolean und ID existieren zudem die beiden Attribut-Typen List und Container. Sie enthalten wiederum mehrere Attribute, mit und ohne Reihenfolge. Das Log Objekt enthält zusätzlich zwei globale Attribut-Listen für Trace und Event Objekte. Attribute die hier definiert sind, müssen in jeder Trace bzw. jedem Attribut vorhanden sein.

2.2.3 Event Classifier und Extensions

Ein Event Classifier definiert die Attribute anhand derer ein Event eindeutig einer Aktivität zugeordnet wird. Im einfachsten Fall ist dies ein Attribut wie bspw. der Name oder die ID der Aktivität. Es können jedoch auch mehrere Attribute sein.

Extensions, vgl. Abbildung 2.3, werden verwendet um Attributen eine Semantik zu verleihen. Es existieren die sieben Standard Extensions Concept, Lifecycle, Organizational,

Time, Semantic, ID und Cost. Extensions definieren jeweils verschiedene Keys. Mit Concept Extensions kann ein Objekt mit einem Namen versehen werden. Mit Lifecycle Extensions kann angegeben werden, ob es sich bei dem Event bspw. um den Start oder Stop einer Aktivität handelt. Mit der Extension Organizational kann einem Event eine Resource zugeordnet werden. Die Extension Semantic dient dazu, mehrere Sichten auf einen Prozess zu berücksichtigen. Die ID Extension kann für eindeutige IDs aller Objekte verwendet werden. Die Extension Cost enthält mehrere Keys um Kosten zu markieren. [12, 14]

```
<?xml version="1.0" encoding="UTF-8" ?>
        <log xes.version="2.0" xes.features="arbitrary-depth" xmlns="http://www.xes-standard.org/">
        <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
        <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
        <global scope="trace">
                <string key="concept:name" value=""/>
        </global>
        <global scope="event">
                <string key="concept:name" value=""/>
                <date key="time:timestamp" value="1970-01-01T00:00:00.000+00:00"/>
                <string key="system" value=""/>
        </global>
        <classifier name="Activity" keys="concept:name"/>
        <classifier name="Another" keys="concept:name system"/>
        <float key="log attribute" value="2335.23"/>
        <trace>
                <string key="concept:name" value="Trace number one"/>
                <event>
                        <string key="concept:name" value="Register client"/>
                        <string key="system" value="alpha"/>
                        <date key="time:timestamp" value="2009-11-25T14:12:45:000+02:00"/>
                        <int key="attempt" value="23">
                                <boolean key="tried hard" value="false"/>
                        </int>
                </event>
                <event>
                        <string key="concept:name" value="Mail rejection"/>
                        <string key="system" value="beta"/>
                        <date key="time:timestamp" value="2009-11-28T11:18:45:000+02:00"/>
                </event>
        </trace>
</log>
```

Listing 2.1: XES XML Serialisierung [14]

2.3 Eclipse 4 Rich Client Platform

Die Eclipse 4 Rich Client Platform (RCP) bezeichnet die vierte Generation des Eclipse RCP Framework zur Erstellung von Desktop-Anwendungen mit Java. Eclipse RCP stellt ein Applikationsmodell zur Verfügung, für welches eine Vielzahl von Komponenten zur Verfügung stehen. Bspw. Menüs oder Container zur Anzeige von Inhalten wie Tabellen und Grafiken. Entwicklern die mit der Eclipse Entwicklungsumgebung vertraut sind, sind viele Komponenten bereits bekannt. Die Eclipse IDE (Integrated Development Environment), aktuell in der Version Mars 4.5, wurde mit dem Eclipse RCP Framework erstellt. Das Java SWT (Standard Widget Toolkit) wird von Eclipse RCP als Standard Bibliothek für die Komponenten der Benutzeroberfläche verwendet. Zusätzlich wird die JFace Bibliothek verwendet. [15]

2.3.1 Dependecy Injection

Im Vergleich zur Vorgänger Versionen 3.x ist eine wesentliche Neuerung der Version 4 die Verwendung von Annotationen und Dependency Injection. Mit Annotationen kann in den Lebenszyklus von Objekten eingegriffen werden. In Java beginnen sie mit einem @-Zeichen und können Attributen, Methoden oder Klassen vorangestellt werden.

Eine Klasse die mit der Annotation @Creatable markiert wird, kann via Dependency Injection von anderen Klassen instanziiert werden. Hierzu wird der zu befüllenden Referenz die Annotation @Inject vorangestellt. Somit wird vom Framework die Referenz automatisch mit der entsprechenden Klasse befüllt. Wird die Klasse mit der Annotation @Singleton markiert, wird die Klasse nur einmal instanziiert. Jeder weitere Aufruf mit @Inject greift auf dieselbe Instanz der Klasse zu. Soll die Klasse mehrfach instanziiert werden können, kann mit der Annotation @Named eine Instanz ausgewählt werden. Die in Abschnitt 5.1 vorgestellte Architektur basiert auf dem Dependency Injection Konzept. [15]

2.3.2 Plugin Architektur

Eclipse RCP Anwendungen lassen sich mittels Plugins erweitern. Ein Plugin stellt dabei ebenfalls eine Eclipse RCP Anwendung dar, die wiederum selbst durch Plugins erweitert werden kann. Plugins bzw. RCP Anwendungen lassen sich somit beliebig verschachteln. Die Schnittstelle an welcher ein Plugin an einer Anwendung andocken kann, wird als Extension Point bezeichnet. Das Gegenstück im Plugin wird als Extension bezeichnet. In Abbildung 2.4 ist das RCP Plugin Konzept schematisch dargestellt. Wie zu sehen ist, kann ein Plugin zeitgleich an mehreren andere Plugins andocken. Extensions und Extension Points können mit Hilfe von Wizards definiert werden. [16]

Um das Applikationsmodell einer Anwendung per Plugin zu erweitern, z.B. um einen Menüpunkt, fällt seit der RCP Version 4 die Notwendigkeit der Verwendung eines Extension Points hierfür weg. Im Plugin kann stattdessen ein Model-Fragment angelegt werden. Für dieses Model-Fragment kann direkt die ID des Elements der Ziel-Anwendung angegeben werden. Bspw. die ID des Menüs welches um einen Menüpunkt erweitert werden soll.

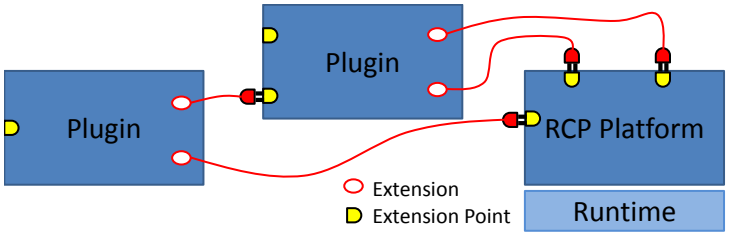

Abbildung 2.4: Eclipse RCP Plugins nach [17]

Seit Einführung der Dependency Injection, kann auch im Plugin per Dependency Injection auf Objekt-Instanzen der Zielanwendung zugegriffen werden. Voraussetzung hierfür ist die Angabe der Klasse in den *exporting packages* der Anwendung. Klassen oder Methoden des Plugins, die vom Plugin selbst initialisiert werden, können somit auch ohne Verwendung des Extension Points mit der Ziel-Anwendung kommunizieren. Dies sind vor allem Klassen des Applikationsmodell, wie bspw. die Handler-Klasse eines Menüpunkts. Plugins die lediglich Klassen losgelöst vom Applikationsmodell enthalten, müssen weiterhin die definierten Extension Points verwenden. Die Anwendung selbst

kann mittels des Extension Managers die Klassen des Plugins initialisieren. [18]

2.3.3 Applikationsmodell

Das Applikationsmodell besteht u.a. aus einem Hauptmenü am oberen Rand der Anwendung. Für dieses stehen Menüs und Menüpunkte zur Verfügung. Jeder Menüpunkt benötigt einen Handler, welcher ebenfalls im Applikationsmodell hinzugefügt wird. Hier kann vom Entwickler eine beliebige Klasse angegeben werden. Eine Methode die mit der Annotation @Execute versehen ist, wird beim Betätigen des Menüpunkts ausgeführt.

Um die Arbeitsfläche der Anwendung zu gestalten, werden vom RCP Framework Parts bereitgestellt. Ein Part der zum Applikationsmodell hinzugefügt wird, wird bereits ohne weitere Angaben als leerer Container angezeigt. Dieser lässt sich verschieben, skalieren, minimieren und maximieren. Um den Part mit Inhalt zu füllen, kann vom Entwickler eine Klasse angegeben werden. Via Dependency Injection kann auf den Part zugegriffen und mit Inhalten wie Tabellen oder Buttons befüllt werden. Dazu wird dem Konstruktor die Annotation @Inject vorangestellt. Das Framework befüllt nun geforderte Parameter wie den zugehörigen Part im Konstruktor automatisch.

Um mehrere Oberflächen zur Verfügung zu stellen, können Perspektiven angelegt werden. Jede Perspektive kann separat mit Parts sowie Layout-Container befüllt werden. Somit ist es möglich, später in der Anwendung zwischen verschiedenen Sichten zu wechseln. [19]

3

Verwandte Arbeiten

Dieses Kapitel stellt mit der Ausarbeitung verwandte Themen vor. Dies sind einerseits Process Mining Anwendungen, andererseits Themen im Bereich regelbasierter Analyse.

3.1 Process Mining Anwendungen

Process Mining Anwendungen setzen die in Kapitel 2.1.3 genannten Verfahren um [10]. Beispiel für Process Mining Anwendungen sind unter anderem ProM 6 [20] und Disco [21]. ProM 6 wird an der Technischen Universität Eindhoven verwaltet und vorwiegend im akademischen Umfeld eingesetzt. Es basiert auf einer Plugin Architektur und lässt sich somit um verschiedene Process Mining Verfahren erweitern. Es ist frei verfügbar und implementiert einen Großteil der bisher entwickelten Verfahren.

Die Anwendung Disco der Firma Fluxicon ist für den professionellen Einsatz durch Endanwender geeignet. Disco wird in Kapitel 5.3.3 dieser Arbeit anhand von Beispiel Log-Daten und Sceenshots vorgestellt. Neben ProM 6 und Disco, implementieren auch zunehmend weitere Anwendungen aus dem Prozessmanagement-Gebiet Process Mining Funktionen.

3.2 Complex Event Processing

Regeln, bzw. Geschäftsregeln, können dazu verwendet werden um die Ausführung eines Prozesses aktiv zu steuern [22] oder um die Ausführung eines Prozesses zu überwachen [23].

Bei der Verwendung von Regeln zur Überwachung, können die Regeln auf abgeschlossene Fälle oder auf die aktuelle Ausführung angewendet werden. Ein Kritikpunkt der Anwendung auf abgeschlossene Fälle ist, dass dabei Fälle entdeckt werden, auf die kein Einfluss mehr genommen werden kann.

Complex Event Processing (CEP) Techniken ermöglichen es, Regeln auf aktive Prozessinstanzen anzuwenden [23]. Mit CEP können Ereignisse verarbeitet werden, während sie passieren. Ereignisse aus unterschiedlichen Quellen werden aggregiert und höherwertige (komplexe) Ereignisse geschaffen. Eine CEP Anwendung ist Business Activity Monitoring (BAM). Anhand von Key-Performance-Indikatoren wird die aktuelle Ausführung des Prozesses überwacht.

Für die Prozessüberwachung mit BAM können Regeln definiert werden. Falls eine Regel durch die auftretenden Ereignisse erfüllt ist, wird ein weiteres Ereignis (bspw. ein Alarm) ausgelöst. CEP Techniken extrahieren aus den auftretenden Ereignissen die entsprechenden Fakten um diese mit den Regeln prüfen zu können. Das Formulieren von Regeln zu diesem Zweck wird auch als Ereignisanfrage bezeichnet.

Weitere CEP Anwendungen finden sich in Sensor-Netzwerken zur Überwachung industrieller Anlagen oder bei der Verarbeitung von Marktdaten wie z.B. Aktien- oder Rohstoffpreise. Regeln sind jedoch nicht die einzige Möglichkeit in CEP-Anwendungen

Ereignisse abzufragen. Ein weiteres Beispiel ist die Ereignisanfrage-Sprache CQL, eine Abwandlung von SQL. Detaillierte Informationen zu CQL und weiteren Ereignisanfrage-Sprachen finden sich in [23].

3.3 Business Rule Engines

Das Anlegen und Auswerten von Geschäftsregeln kann mit Business Rule Engines erfolgen. Durch verschiedene Integrationsmöglichkeiten in Prozessmanagementsysteme decken Business Rule Engines einerseits das Szenario ab, Regeln zur aktiven Steuerung eines Prozesses einzusetzen. Andererseits können sie auch zur Regelauswertungen in Kombination mit CEP-Systemen eingesetzt werden. [22, 24]

Der Einsatz von Business Rule Engines wird empfohlen, falls viele Regeln in einem Prozess (bzw. Anwendung) vorkommen und diese sich häufig ändern [24]. Beispiel Domänen hierfür sind Versicherungen (Bewertungen), Finanzen (Kreditvergabe) der Öffentliche Sektor (Genehmigungen) oder Telekommunikation (personalisierte Werbung).

Neben dem Anlegen und Auswerten von Regeln, bieten Business Rule Engines umfangreiche Möglichkeiten um Regeln zu verwalten. Ein besonderer Aspekt ist, dass Regeln von Business-Experten ohne die Kenntnisse eines Entwicklers erstellt werden können. Folgende Eigenschaften zeichnen Business Rule Engines aus [25, 26]:

Verständliche Ausdruckssprache: Business Rule Engines stellen zur Regelerstellung Ausdruckssprachen zur Verfügung, die von Business Experten und Anwendungsentwickler verstanden werden. Der Experte kann eine Regel formulieren, die vom Entwickler nicht in eine weitere Sprache übersetzt werden muss. Dies beugt Missverständnissen vor, reduziert Fehler und ist flexibler.

Komplexe Logik: Die Regel „Alter > 30“ ist eine einfache Regel. Regeln sind aber in der Praxis selten so einfach. Regeln können komplexe Mustererkennungen und Schlussfolgerungen enthalten. Business Rule Engines sind bestens geeignet um komplexe Logik abzubilden.

Schnelle Änderungen: Business Rule Engines trennen die Regel-Logik vom Anwendungscode. Sie werden in einer separaten Umgebung gepflegt. Sie können von Business Experten geändert werden und von diesen selbstständig aktiviert werden. Somit ermöglichen Business Rule Engines schnelle und flexible Änderungen.

Änderungen durch Business Experten: Mit der Verwendung einer Business Rule Engine kann die Verantwortung für die Regelpflege an die Business-Experten abgegeben werden. Der lange Software-Entwicklungszyklus fällt weg. Die Experten sind – mit den geeigneten Tools – selbst in der Lage die wichtigsten Geschäftsregeln zu ändern.

Anbieter von kommerziellen Business Rule Engines sind Bosch Software Innovations, Corticon Technologies, IBM ILOG, FICO Blaze Advisor und weitere. IBM ILOG ist der weltweite Marktführer kommerzieller Software. In Europa nimmt die Lösung von Bosch Software Innovations die führende Rolle ein. Eine preisgünstige nicht-kommerzielle Alternative ist die Open Source Software JBoss Drools von Red Hat. [25]

3.3.1 Red Hat JBoss Drools

JBoss Drools ist die führende Open Source Business Rules Engine. Das Projekt wurde 2001 gestartet und ist seit 2005 Teil der JBoss Gruppe. Nach der Übernahme von JBoss durch Red Hat gehört es seit 2006 zu Red Hat. [25, 27].

Drools besteht aus mehreren Modulen. Die *Drools Workbech* ist eine Web-Oberfläche um Regeln anzulegen und zu verwalten. *Drools Expert* ist die Business Rule Engine. Zudem verfügt Drools über das CEP Modul *Drools Fusion* und eine Integration in das Prozessmanagementsystem jBPM.

Der Aufbau einer Drools Regel Datei wird in Listing 3.1 gezeigt. Die Regel enthält einen Namen sowie eine Liste der verwendeten Attribute. Nach dem Statement *when* folgt die Bedingung. Ist sie erfüllt, wird der Code nach dem Statement *then* ausgeführt. Im Vergleich mit einer klassischen if-Abfrage fällt auf, dass kein else-Teil vorgesehen ist. Dies erklärt sich dadurch, dass es sich bei der Regel um eine Ereignisabfrage handelt. Es wird nur ein Ereignis ausgelöst, falls die Regel erfüllt ist.

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

Listing 3.1: Drools Regel Syntax [28]

Listing 3.2 zeigt die Deklaration einer Funktion. Funktionen können sowohl im *when* Teil einer Regel als auch im *then* Teil einer Regel verwendet werden. Sie können mit Java programmiert werden.

```
function String hello(String name) {
    return "Hello "+name+"!";
}
```

Listing 3.2: Typische Drools Funktion Deklaration [28]

Ein Beispiel einer einfachen Regel incl. der Verwendung von Funktionen ist in Listing 3.3 zu sehen.

```
rule "using a static function"
when
    $p : Person( name == "Michael" )
then
    System.out.println( hello( $p.name ) );
end
```

Listing 3.3: Drools Regel Beispiel [28]

Mit Drools kann zudem eine domänenspezifische Regelsprache (DSL - Domain Specific Language) definiert werden, die auf die Drools Syntax gemappt werden kann. Für Ausdrücke die häufig verwendet werden, können DSL Sätze definiert werden. DSL Sätze können somit als Vorlage fungieren, für sich wiederholende Ausdrücke mit geringen Abweichungen. Sie unterstützen den Ansatz, dass Business-Experten ohne weitreichende Kenntnisse von logischen Ausdrücken selbst Regeln pflegen können. [27, 28]

```
Registered customer and expense greater than 1000
   ==> $customer : Customer( registered == true, expense > 1000)
Person is at least 42 years old and lives in "Atlanta"
   ==> $person : Person(age >= 42, location="Atlanta")
There is a person with name of "Michael" and Person is at least 30 years old and lives in "Utah"
   ==> $person : Person(name="Michael") and Person(age >= 30, location="Utah")
```

Listing 3.4: Domain Specific Language (DSL) [28]

3.3.2 Bosch Software Innovations Visual Rules

Wie bereits erläutert nimmt Visual Rules BRM (Business Rules Management) von Bosch Software Innovations in Europa die marktführende Stellung ein. Als Referenzkunden werden ThyssenKrupp Steel Europe (verschiedene Stahlqualitäten), REWE Group (mobile Inventuranwendung) und DHL Global Mail (Preiskatalog) genannt [22].

Visual Rules BRM besteht aus einem Server und verschiedenen Client-Komponenten. Zur Regelerstellung existiert ein Desktop- sowie ein Web-Client. Zudem gibt es clientseitig verschiedene administrative Oberflächen für Deployment, Tests, Identity-Management und Regelverwaltung. Die Server-Komponente führt die Regeln aus und hält verschiedene Integrationsmöglichkeiten bereit. Bspw. können externe Anwendungen via WebServices auf Regeln zugreifen.

Wie Abbildung 3.1 illustriert, kann Visual Rules nahtlos in das Prozessmanagementsystem inubit [29], von Bosch Software Innovations, integriert werden. Die erstellten Regeln können als Entscheidungspunkte in Prozessmodellen verwendet werden.

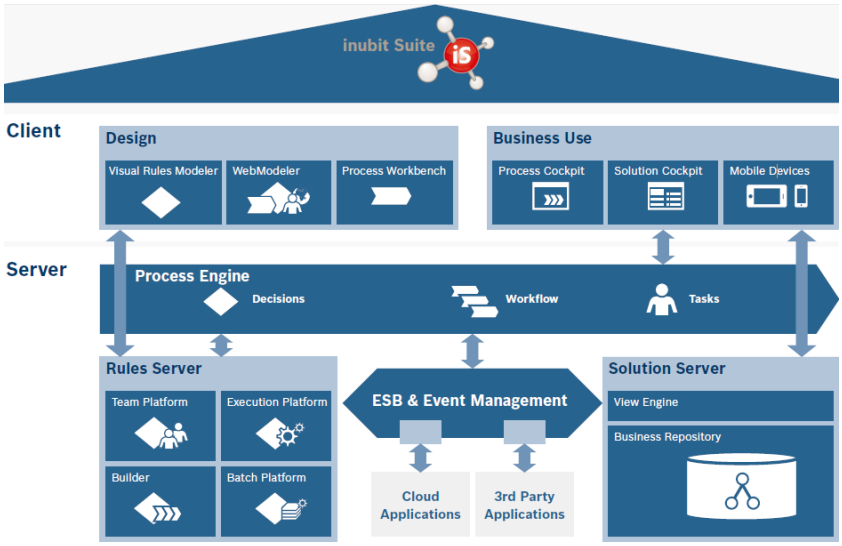

Abbildung 3.1: Visual Rules BRM und inubit BPM [22]

Bei der Erstellung der Regeln bietet Visual Rules drei verschiedene Varianten an. Die Modellierung als Ablaufregel, Entscheidungstabelle oder Zustandsablauf. Für alle drei

Varianten existiert eine grafische Oberfläche. Eine Entscheidungstabelle ermöglicht es mehrere verwandte Regeln übersichtlich darzustellen. Abbildung 3.2 zeigt eine Ablaufregel. Darin enthalten sind zwei weitere Ablaufregeln (Preisberechnung).

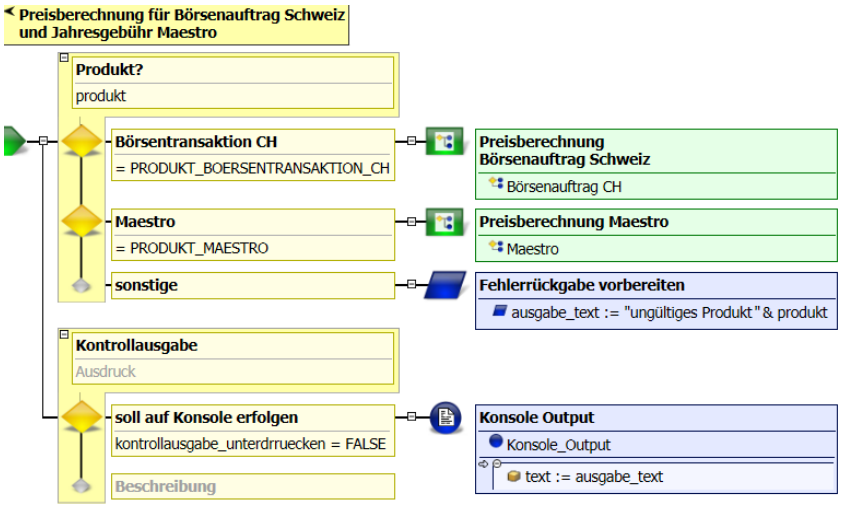

Abbildung 3.2: Visual Rules Ablaufregel [22]

4

Anforderungen

Ziel der Arbeit ist es, das Fragebogensystem QuestionSys um die Komponente QuestionAnalysis zu erweitern. Wie bereits erläutert, soll es die Komponente ermöglichen, Process Mining und regelbasierte Analysen durchführen zu können.

Basis hierfür sind die Ereignislogs, welche QuestionSys als prozessorientiertes Fragebogensystem generiert. Um eine Prozessanalyse durchführen zu können, sollen die Ereignislogs mit der QuestionAnalysis Komponente in dafür geeignete Formate umgewandelt werden können. Die Auswertung der Regeln soll direkt in der QuestionAnalysis Komponente stattfinden. Zudem sollen weitere Auswertungsmöglichkeiten in die QuestionAnalysis Komponente integriert werden. Nachfolgend werden alle Anforderungen an die Komponente aufgelistet.

4.1 Import und Durchsicht von Log-Daten

AF 1 Die Anwendung soll die von den QuestionSys-Clients generierten Ereignislogs importieren können. Aktuell stehen die Ereignislogs in Form von JSON-Dateien zur Verfügung. Die JSON-Dateien sollen in ein geeignetes internes Datenformat umgewandelt werden, um die gewünschten Analysen und Export-Formate umsetzen zu können.

AF 2 Es sollen Ereignislogs im XES Format importiert werden können (siehe Kapitel 2.2).

AF 3 Anhand von Kennzahlen und Diagrammen soll eine Übersicht über die importierten Daten gegeben werden. Zur Übersicht gehören einfache Kennzahlen wie die Anzahl geladener Fragebögen (Instanzen) oder der Erstellungszeitraum der Fragebögen. Zudem soll eine Übersicht über enthaltene Datenelemente und deren Häufigkeiten gegeben werden.

AF 4 Die grafische Oberfläche soll es ermöglichen, auch einzelne Fragebögen durchzusehen. Durch eine einfache Auswahl soll ein Fragebogen ausgewählt werden können, um die Details anzeigen zu können. Hierfür soll es eine Auflistung der Fragebögen geben, welche zu jedem Fragebogen wichtige Übersichtsdaten wie den Namen des Bearbeiters oder das Datum anzeigt.

AF 5 Es soll möglich sein, ein oder mehrere Instanzen zu selektieren. Hierzu soll es zudem eine Suchfunktion geben, um nach einzelnen Instanzen zu suchen.

AF 6 Einzelne oder mehrere Instanzen sollen nach dem Import aus dem Arbeitsbereich der Anwendung gelöscht werden können.

4.2 Regelbasierte Analyse

AF 7 Regeln die mit der Regelkomponente QuestionRule erstellt wurden, sollen importiert werden können. Die Regeln liegen im XML-Format vor.

AF 8 Die Regeln sollen auf die Log-Daten angewendet werden können. Hierfür soll eine geeignete Technologie verwendet werden, welche die Regeln verarbeiten kann.

AF 9 Die Ergebnisse der Regelauswertung sollen geeignet dargestellt werden (Übersicht sowie einzeln pro Instanz).

AF 10 Es soll grundsätzlich möglich sein Instanzen anhand der Regelergebnisse zu selektieren und zu sortieren. Bspw. sollen alle Fragebögen selektiert werden können für welche eine bestimmte Regel zutrifft.

4.3 Exportfunktionen

AF 11 Die Ereignislogs sollen im CSV Format für weitere Analysen in Excel, Business-Warehouse Anwendungen oder anderen Statistik-Programmen exportiert werden können.

AF 12 Die Ereignislogs sollen im XES Format exportiert werden können, um Process Mining Analysen mit Tools wie ProM 6 oder Disco durchführen zu können.

AF 13 Es soll möglich sein, den Export anhand von getätigten Selektionen einzuschränken. Durch eine manuelle Selektion oder die Selektion anhand eines Regelergebnisses.

AF 14 Grundsätzlich sollen alle Export-Formate mit den Ergebnissen der Regelauswertung angereichert werden können, sodass es auch in nachfolgend verwendeten Tools möglich ist, anhand der Regel Auswertungsergebnisse Daten filtern zu können.

4.4 Integration und Erweiterbarkeit

AF 15 Die QuestionAnalysis Komponente soll nicht als separate Anwendung erstellt, sondern in die bereits bestehende Regelkomponente integriert werden. Die Re-

gelkomponente wurde mit dem Eclipse 4 RCP Framework realisiert. Die Analyse-Komponente soll in geeigneter Weise dort integriert werden.

AF 16 Funktionen wie bspw. das Einlesen der Regeldateien, welche bereits in der QuestionRule Komponente implementiert sind, sollen geeignet gekapselt werden. Somit kann die QuestionAnalysis Komponente diese wiederverwenden.

AF 17 Es soll eine Import-Schnittstelle geben, welche es ermöglicht Import Formate im späteren Betrieb auszutauschen oder hinzuzufügen. Dies soll für den Import von Ereignislogs sowie für den Import der Regel-Dateien möglich sein.

AF 18 Eine flexible Export-Schnittstelle soll analog der Import-Schnittstelle hinzugefügt werden, damit später weitere Process Mining Formate hinzugefügt werden können.

AF 19 Die Anwendung soll modular aufgebaut und einfach zu erweitern sein.

AF 20 Die Kompatibilität zwischen der bestehenden Regelkomponente soll sichergestellt werden. Ggf. sind auch Anpassungen an der Regelkomponente durchzuführen.

5

Architektur und Konzept

Dieses Kapitel stellt die Architektur der Analysekomponente vor. Zudem die Struktur der Daten sowie das interne Datenmodell der Anwendung. Anschließend wird darauf basierend das Konzept vorgestellt, wie die Analyse-Anforderungen aus Kapitel 4 umgesetzt werden.

5.1 Architektur

Abbildung 5.1 gibt einen Überblick über die gewählte Architektur zur Umsetzung der QuestionAnalysis Komponente (Analysekomponente). Ausgangspunkt ist die bereits bestehende QuestionRule Komponente [5] (Regelkomponente), welche mit dem Eclipse RCP Framework erstellt wurde. Anforderung AF 15 verlangt eine geeignete Integration der Analysekomponente in die Regelkomponente. Hierzu wird das Eclipse RCP Plugin

Konzept verwendet, welches in Kapitel 2.3 vorgestellt wurde. Die Analysekomponente wird als Plugin zur bestehenden Regelkomponente hinzugefügt.

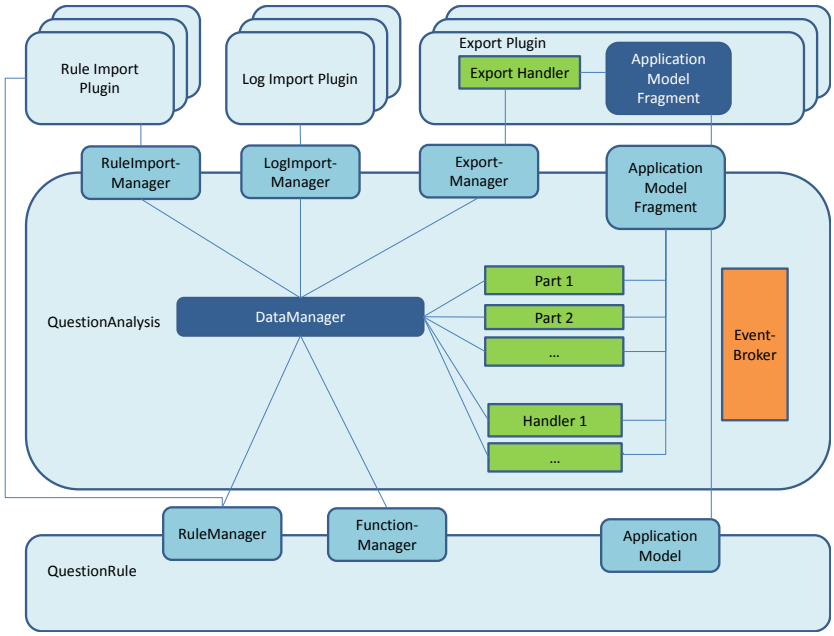

Abbildung 5.1: Architektur QuestionAnalysis

Mittels Fragmenten „dockt“ die Analysekomponente an das Applikationsmodell der Regelkomponente an. Knotenelement der Analysekomponente ist eine Perspektive. Somit sind die beiden Komponenten grafisch getrennt. Über das Hauptmenü der Anwendung kann später zwischen Perspektiven, sprich der Regelkomponente und der Analysekomponente, gewechselt werden.

5.1.1 RuleManager und FunctionManager

Neben dem Applikationsmodell sind weitere Berührungspunkte zwischen Analyse- und Regelkomponente der RuleManager und der FunctionManager. Die beiden Manager gehören zur bereits bestehenden Regelkomponente und stellen der Analysekomponente Funktionalität zur Verarbeitung der Regeln bereit. Via Dependency Injection (vgl. Kapitel 2.3) erfolgt der Zugriff auf diese Manager.

Wie in Anforderung AF 7 verlangt, sollen von der Regelkomponente erstellte Regeln von der Analysekomponente eingelesen werden können. Hierzu gehören die Regeln selbst, sowie die darin verwendeten Funktionen.

Der *RuleManager* stellt der Analysekomponente Methoden bereit, um die Regeldateien einlesen zu können. Alle relevanten Klassen hierfür werden in den *exporting packages* der Regelkomponente bereitgestellt. Die zu importierenden Regeln werden von der Regelkomponente erstellt und im XML-Format abgelegt. Das Format der Regeln ist daher bereits gegeben. Für die XML-Datenbindung zwischen Regeldateien und Java-Klassen wurde JAXB (Java Architecture for XML Binding) [30] verwendet. XML-Regeldateien können somit direkt in Java-Objekte umgewandelt werden.

Der *FunctionManager* stellt der Analysekomponente die Implementierung der Funktionen bereit. Dies ist erforderlich, da die Regeln selbst lediglich den qualifizierten Klassenname der Funktion enthalten. Die Analysekomponente kann somit auf alle verwendeten Funktionen zugreifen um die Regeln auswerten zu können.

5.1.2 Import- und ExportManager

So wie die Regelkomponente die beiden beschriebenen Manager bereitstellt, stellt auch die Analysekomponente entsprechende Manager bereit. Auf diese Manager greifen die Plugins der Analysekomponente zu. Die in Kapitel 4 beschriebenen Anforderungen für Import und Export Schnittstellen werden mittels dieser Manager umgesetzt. Mit den Plugins ist es wie gewünscht möglich, im späteren Betrieb der Anwendung neue Import und Export Formate hinzuzufügen.

Wie in den Anforderungen AF 11 und AF 12 aufgeführt, sollen verschiedene Export Formate zur Verfügung stehen. Diese Anforderungen werden mittels des *ExportManager* umgesetzt. Dieser stellt die Log-Daten für einen Export bereit. Die Log-Daten werden dem ExportManager wiederum vom DataManager bereitgestellt, und können bereits mit den Ergebnissen der Regelauswertung angereichert sein. Das Export Plugin selbst implementiert die Umwandlung in das gewünschte Export Format. Neben dem ExportManager greift das Plugin auf das Applikationsmodell der Analysekomponente zu.

Das Plugin fügt an der vorgesehenen Stelle im Menü einen Menüpunkt für den Export hinzu. Wird der Menüpunkt vom Benutzer ausgewählt, wird der Export Handler des Plugins ausgeführt.

Die Log-Daten welche die Analysekomponente importiert, liegen im JSON-Format vor, vgl. Anforderung AF 1. Gemäß Anforderung AF 2 und AF 17 sollen jedoch später weitere Datenquellen hinzugefügt werden können, wie bspw. das XES-Format. Diese Anforderungen werden mittels des *LogImportManager* umgesetzt. Der LogImportManager stellt analog dem ExportManager die Schnittstelle für Import Plugins dar. Beim Datenformat welches an den LogImportManager übergeben wird, handelt es sich bereits um das in Kapitel 2.2 vorgestellte XES-Format. Die Umwandlung in das XES-Format wird somit im Import Plugin implementiert. Analog den Export Plugins greift ein Import Plugin ebenfalls auf das Applikationsmodell der Analysekomponente zu. Nachdem die Log-Daten an den LogImportManager übergeben wurden, reicht dieser die Daten weiter an den DataManager. Der Data Manager übernimmt die weiteren Verarbeitungsschritte.

Gemäß den Anforderungen AF 7 und AF 17, stellt die Analysekomponente den *RuleImportManager* bereit. Analog dem LogImportManager stellt der RuleImportManager die Schnittstelle für Plugins zum Import von Regeln dar. Wie in Anforderung AF 7 beschrieben, liegen die Regeln im XML-Format vor. Das Einlesen der Regeldateien findet im Plugin statt. Hierzu können Plugins die Funktionalität des RuleManager verwenden. In Abbildung 5.1 hat das Rule Import Plugin daher entsprechend eine Verbindung zu beiden Komponenten, QuestionRule und QuestionAnalysis. Das Rule Import Plugin fügt ebenfalls einen Menüpunkt zum Applikationsmodell der Analysekomponente hinzu. Importierte Regeln reicht der RuleImportManager an den DataManager weiter.

5.1.3 DataManager und EventBroker

Der *DataManager* ist die zentrale Datenverarbeitungskomponente. Daten die von den Import Managern entgegengenommen werden, werden an den DataManager weitergereicht. Der DataManager hält alle Import-Daten sowie die Ergebnisse der Regelauswertung. Die Auswertung der Regeln, wie in Anforderung AF 8 aufgeführt, findet ebenfalls im DataManager statt. Zudem implementiert der DataManager auch die Kennzahlen-

berechnungen zur Log-Übersicht. Nachdem neue Regeln oder Log-Daten importiert wurden, findet automatisch eine Neuberechnung statt. Die Ergebnisse stehen somit auch unmittelbar für den ExportManager zur Verfügung.

Der DataManager steht direkt den Parts der grafischen Oberfläche zur Verfügung. Parts wurden in Kapitel 2.3 vorgestellt und sind Teil des Applikationsmodells. Mittels Tabellen und Diagrammen werden mit Parts die Daten präsentiert. Zudem können Daten in den Parts ausgewählt und durchgesehen werden, wie in den Anforderungen AF 3 und AF 4 beschrieben. Weitere Funktionalitäten (Menüpunkte/Handler) die in der Menüleiste integriert sind, greifen ebenfalls auf den DataManager zu.

Der *EventBroker* verarbeitet alle Ereignisse die in der Anwendung ausgelöst werden. Wurden neue Daten importiert, löst der DataManager ein entsprechendes Ereignis aus. Der EventBroker verarbeitet dieses Ereignis und stößt die Aktualisierung der Parts an, welche vom Datenimport betroffen sind. Umgekehrt stößt der EventBroker die Aktualisierung der Daten an, falls einer der Parts oder Handler ein entsprechendes Ereignis auslöst. Bspw. wenn Datensätze entfernt werden, wie in Anforderung AF 6 gefordert. Der EventBroker dient zudem zur Kommunikation zwischen Parts. Dies ist bspw. bei der Datendurchsicht relevant (Anforderung AF 4). Wird eine Instanz in der Liste ausgewählt, wird der Part mit der Detailansicht entsprechend aktualisiert. Die nötige Kommunikation findet mittels Ereignissen statt.

5.2 Daten

Die Analysekomponente arbeitet mit Ereignislogs, Regeldateien und Funktionen. Das Format der Regeldateien und Funktionen ist durch die bestehende Regelkomponente bereits gegeben. Zudem werden die benötigten Klassen durch die QuestionRule Komponente bereitgestellt, um die Daten intern zu verarbeiten. Für die zu importierenden Ereignislogs wurde der XES Standard als internes Datenformat ausgewählt.

5.2.1 Ereignislogs

In Kapitel 2.2 wurde der Ereignislog Standard XES vorgestellt. Wie aus den Anforderungen AF 2 und AF 12 bekannt, soll die Anwendung Ereignislogs im XES Format importieren und exportieren können. Da das XES Format ein anerkannter Standard ist [6], wurde es als internes Datenformat ausgewählt. Eine Java Implementierung des XES Standards ist OpenXES [14]. OpenXES stellt eine Java-Bibliothek zur Verfügung, um XES Objekte zu erstellen. Zudem können XES Dateien eingelesen werden und XES Objekte im XML Format serialisiert werden.

Wie aus Kapitel 1 bekannt, repräsentieren die Ereignislogs die ausgefüllten Fragebögen. Beim Ausführen des Prozessmodells im Prozessmanagementsystem werden die Ereignislogs generiert. Ein Ereignis repräsentiert dabei die kleinste Granularität einer Aktivität im Prozessmodell. Ein Ereignis markiert bspw. den Start oder das Ende einer Aktivität. Eine Aktivität besteht somit aus mehreren Ereignissen.

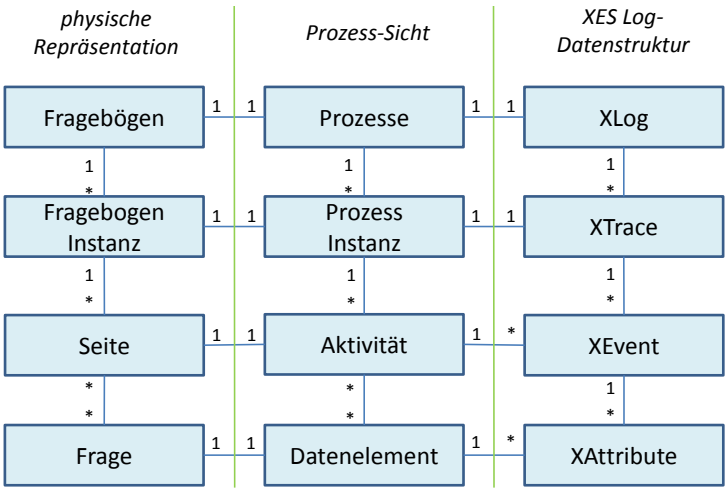

Abbildung 5.2: Log-Mapping

Abbildung 5.2 zeigt das Mapping eines Prozessmodells auf das XES Format, sowie das Mapping des Fragebogenmodells auf ein Prozessmodell. Mehrere Fragebögen werden durch ein XLog Objekt repräsentiert. Eine Fragebogeninstanz ist eins zu eins abgebildet auf eine Prozessinstanz und das XTrace Objekt. Mehrere XEvent Objekte repräsentieren

eine Aktivität bzw. Seite im Fragebogen. Ein XEvent enthält mehrere XAttribute Objekte. Im Unterschied zum Fragebogen- und Prozessmodell ist ein Attribut nun eindeutig einem Event zuzuordnen. In den Modellen ist dies noch nicht möglich, da je nach gewähltem Ausführungspfad eine Frage auf verschiedenen Seiten stehen kann.

5.2.2 Regeln und Funktionen

Eine Regel wie sie von der Regelkomponente erstellt wird zeigt Listing 5.1. Eine XML-Regeldatei enthält genau ein umschließendes Element <rule>. Dieses enthält sechs weitere Elemente. Jeweils genau ein Element <dependencies>, <condition>, <ruleID>, <ruleDescription> sowie genau ein Element <languageTextContainer>. Das Element <condition> enthält die Regel in Form eines Strings. Alle verwendeten Funktionen sind im Element <dependencies> gesammelt. Für jede Funktion existiert ein Element <dependency>. Es folgen mit <ruleName> und <ruleDescription> Name und Beschreibung der Regel. Das Element <languageTextContainer> enthält Texte in verschiedenen Sprachen. Für beliebig viele Sprachen kann hier hinterlegt sein, welcher Text zur Anzeige gebracht wird, falls die Regel zutrifft oder nicht.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <rule>
3    <dependencies>
4      <dependency>general.bmi</dependency>
5    </dependencies>
6    <condition><![CDATA[25 <= general.bmi.evaluate(gewicht,groesse)]]></condition>
7    <ruleID>c61c6539-0a26-49e9-9e3c-445781cc7f5e</ruleID>
8    <ruleName>Übergewicht</ruleName>
9    <ruleDescription>Prüfen ob Patient übergewichtig ist.</ruleDescription>
10   <languageTextContainer>
11     <languageText languageCode="DE">
12       <ruleTrue>Patient hat Übergewicht</ruleTrue>
13       <ruleFalse>Patient hat kein Übergewicht</ruleFalse>
14     </languageText>
15   </languageTextContainer>
16 </rule>
```

Listing 5.1: Regel Beispieldatei

Bei der beschriebenen Datei handelt es sich um eine einzelne Regel. Sie hat die Dateiendung *.rule*. Die Regelkomponente ermöglicht es zudem, mehrere Regeln in einer *.rules* Datei zu exportieren. Die XML-Schema Dateien für .rule und .rules Dateien finden sich in [5].

Funktionen werden wie bereits erläutert mittels des FunctionManager von der Regelkomponente zur Verfügung gestellt. Eine Funktion implementiert das Interface Function, siehe Abbildung 5.3.

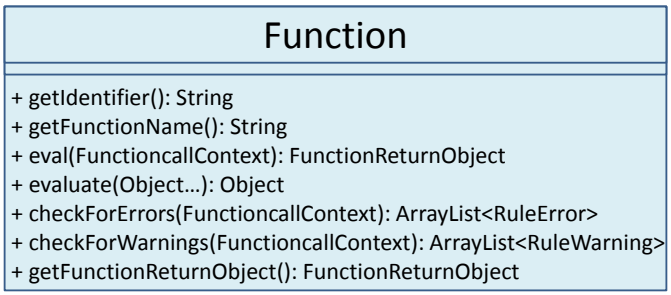

Abbildung 5.3: Function Interface [5]

Für die Analysekomponente sind die beiden Methoden *getFunctionName()* und *evaluate()* relevant. Die Methode *getFunctionName()* liefert den Namen der Funktion, die Methode *evaluate()* implementiert die Auswertung der Funktion. Die übrigen Funktionen dienen der Regelkomponente zur Syntaxprüfung. Sie unterstützen den Anwender beim Erstellen einer Regel. Nähere Informationen zur Erstellung von Funktionen und der Syntaxprüfung von Funktionen können [5] entnommen werden.

5.2.3 Datenmodell

Um die XES-Klassen der verwendeten OpenXES Bibliothek geeignet in die Anwendung zu integrieren, wurde zusätzlich zum DataManager der InstanceDataManager erstellt. Für jede Instanz (XTrace) exisitiert ein InstanceDataManager.

Abbildung 5.4 zeigt einen Ausschnitt aus dem Datenmodell. Auf der linken Seite sind die XES Klassen XLog und XTrace abgebildet, rechts davon der DataManager und der InstanceDataManager. Ein InstanceDataManager verwaltet ein XTrace Objekt. Zugriffe auf das XTrace Objekt finden über den InstanceDataManager statt. Somit können häufige Operationen auf dem Datenmodell gekapselt werden. Bspw. das Auslesen aller Datenelemente der Instanz.

Zudem implementiert der InstanceDataManager die Regelauswertung und enthält das Ergebnis der Regelauswertung der zugehörigen Instanz. Angestoßen wir die Regelauswertung vom DataManager. Dieser iteriert über alle InstanceDataManager.

Der ExportManager (vgl. Kapitel 5.1.2) ist auf der rechten Seite in Abbildung 5.4 dargestellt. Er enthält eine Referenz des DataManagers sowie eine Liste von ExportInstance Objekten. Jedes ExportInstance Objekt enthält eine Referenz auf einen InstanceDataManager.

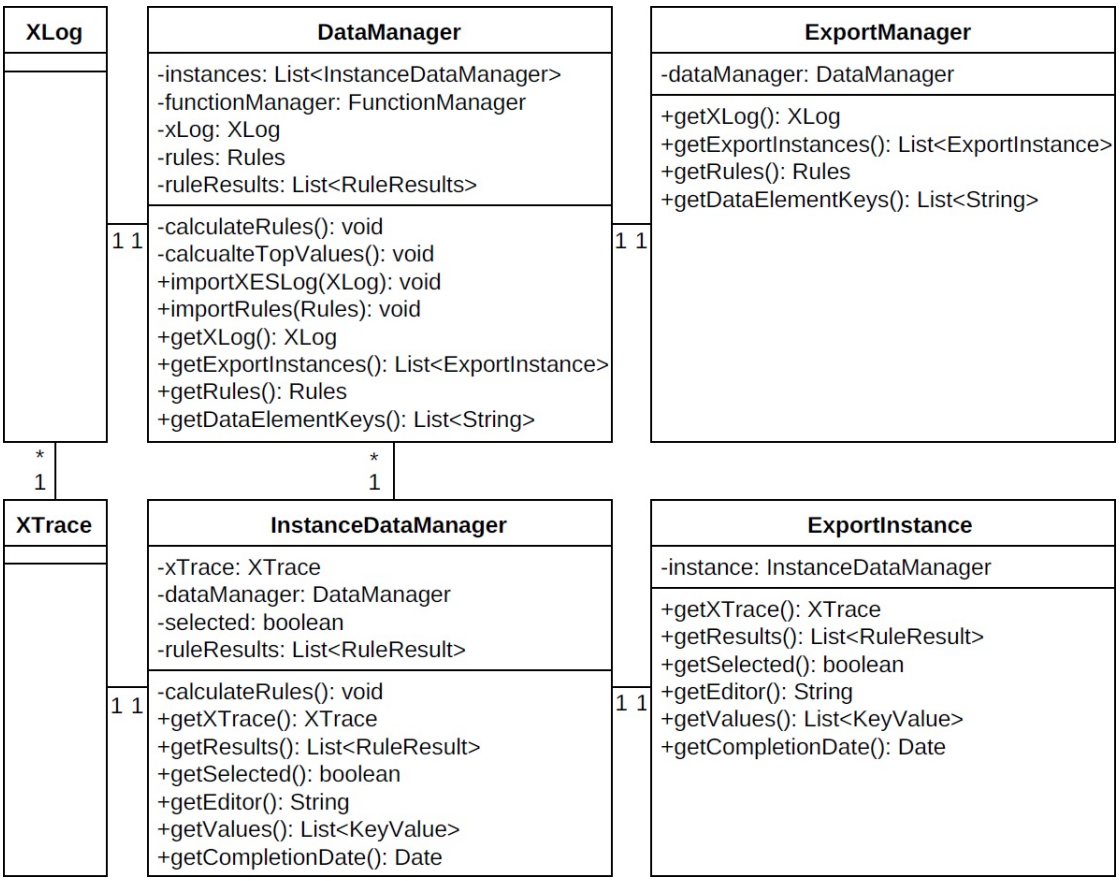

Abbildung 5.4: UML Diagramm QuestionAnalysis

Der ExportManager und die ExportInstance Objekte verfügen lediglich über einfache Getter-Methoden. Alle Methoden die in Abbildung 5.4 in den beiden Export Klassen dargestellt sind, werden im DataManager bzw. InstanceDataManager implementiert.

Der ExportManager stellt lediglich eine Schnittstelle dar, die per Dependency Injection erreicht werden kann. Damit kann die Implementierung des DataManagers und InstanceDataManagers verändert werden, ohne dabei die Plugin-Schnittstelle ändern zu müssen.

5.3 Analysekonzept

Nachfolgend wird das Konzept für die Regelauswertung und die Umsetzung der Process Mining Anforderungen vorgestellt. Zur Demonstration der Ergebnisse wird ein Process Mining Export mit dem Tool Fluxicon Disco analysiert. Anschließend wird das Konzept der grafischen Oberfläche erläutert sowie weitere Export-Formate.

5.3.1 Regelauswertung

Wie bereits erläutert, wird die Regelauswertung in der Klasse InstanceDataManager implementiert. Die Methode calculateRules() des DataManager iteriert über alle InstanceDataManager und ruft die calculateRules() Methode der jeweiligen Instanz auf. Zur Auswertung der Regel ist die Regel selbst erforderlich, die verwendeten Funktionen sowie die Datenelemente der zu prüfenden Instanz.

Abgelegt werden die Ergebnisse in RuleResult und RuleResults Objekten. Die Einzelergebnisse der Regelauswertung werden in einem *RuleResult* Objekt abgelegt, siehe Abbildung 5.5. Es enthält das Ergebnis pro Instanz und Regel. Das RuleResult Objekt besteht aus einer Referenz auf das zugehörige Rule Objekt, sowie zwei Attributen mit dem Ergebnis der Auswertung. Das Attribut *result* kann die Werte TRUE (Regel wird erfüllt), FALSE (Regel wird nicht erfüllt) oder NOTAPPLICABLE (Regel kann nicht angewendet werden) annehmen. Für letzteren Fall gibt es verschiedene Ursachen. Eine mögliche Ursache ist, dass ein für die Auswertung relevantes Datenelement nicht ausgefüllt wurde. Die tatsächliche Ursache wird im Ergebnisfeld *message* festgehalten. Befüllt wird dieses Feld ebenfalls bei der Regelauswertung. Pro angewendeter Regel enthält das InstanceDataManager Objekt einen Eintrag in der Liste ruleResults.

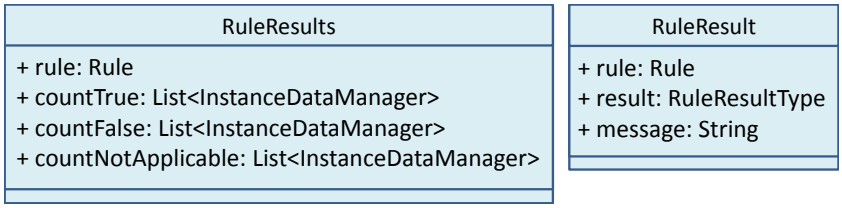

Abbildung 5.5: Klassen RuleResult und RuleResults

Die aggregierten Ergebnisse der Regelauswertung werden in einem *RuleResults* Objekt abgelegt. Für jede Regel gibt es insgesamt ein RuleResults Objekt. Der DataManager enthält die Liste von RuleResults Objekten, siehe Abbildung 5.4. Das RuleResults Objekt besteht aus einer Referenz auf das zugehörige Rule Objekt, sowie den drei Attributen countTrue, countFalse und countNotApplicable. Darin sind die Instanzen nach Ergebnis zusammengefasst. Somit enthält das Attribut countTrue alle Referenzen auf Instanzen bei denen das Ergebnis TRUE lautet.

5.3.2 Process Mining Export

Voraussetzung um Process Mining Tools nutzen zu können, sind Ereignislogs in dafür geeigneten Formaten. Ein Standardformat hierfür ist das bereits in Kapitel 2.2 vorgestellte XES Format. Es wird von Disco und ProM 6 als Import-Format unterstützt.

Bereits die Import Plugins der QuestionAnalyis Komponente wandeln die Ereignislogs in dieses Format um, da das XES Format in der QuestionAnylsis Komponente zur internen Verarbeitung der Daten verwendet wird. Export Plugins können die Daten anschließend im XES Format über den ExportManager abrufen. Weitere Export Formate wie bspw. der MXML Standard (Mining eXtensible Markup Language) [31] oder ein CSV-Format können hinzugefügt werden. Sie werden realisiert, indem das XES Format im Export Plugin entsprechend umgewandelt wird.

Die Ergebnisse der Regelauswertung können im Export-Plugin in die Ereignislogs integriert werden. Sie stehen im oben vorgestellten Format im Export Plugin zur Verfügung. Im bereits realisierten XES Export wurden die Regelergebnisse als Attribute zum Ereignislog hinzugefügt, pro angewendeter Regel ein Attribut. Die Attribute werden dem

ersten XEvent Objekt einer Instanz hinzugefügt. Der Key des Attributs besteht aus dem Namen der Regel, der Wert des Attributs aus dem Ergebnis der Regelauswertung (TRUE, FALSE oder NOTAPPLICABLE).

Die QuestionAnalysis Komponente bietet die Möglichkeit anhand der Regelergebnisse zu selektieren. Somit kann bereits in der QuestionAnalysis Komponente nach den Regelergebnissen selektiert werden, um anschließend nur mit den gewünschten Instanzen im Process Mining Tool weiterzuarbeiten. Abbildung 5.6 zeigt einen Ausschnitt aus der XES Export Maske.

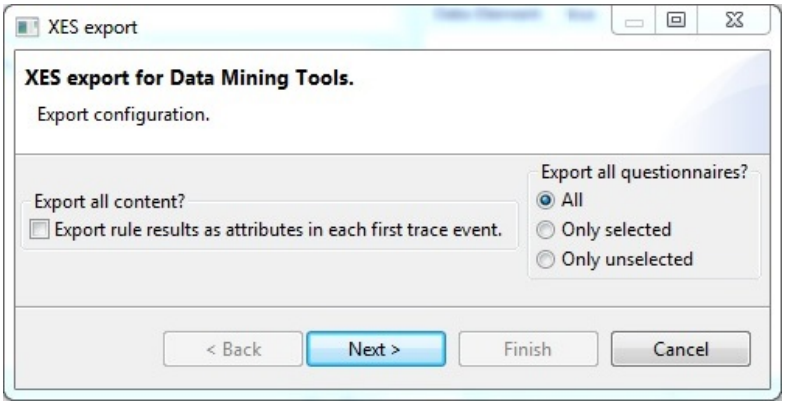

Abbildung 5.6: XES Export Konfiguration

5.3.3 Process Mining mit Fluxicon Disco

Disco wurde bereits mehrfach als Beispiel für ein Process Mining Tool aufgeführt. Eine detaillierte Beschreibung aller Funktionen findet sich in [21]. Nachfolgend werden einige Funktionen anhand des Analysebeispiels vorgestellt. Abbildung 5.7 zeigt einen Screenshot aus Disco. Am oberen Rand bietet Disco eine Schnellnavigation zwischen den drei Ansichten Map, Statistics und Cases.

Die Ansicht Map zeigt sich dem Benutzer zu Beginn, nachdem Log-Daten importiert wurden. Sie enthält das ermittelte Prozessmodell. Abbildung 5.7 zeigt das Prozessmodell welches aus einem unserer Beispiel Fragebögen erstellt wurde. Es sind 35 Fragebogeninstanzen und sieben Aktivitäten (Fragebogenseiten) zu sehen. Das ermit-

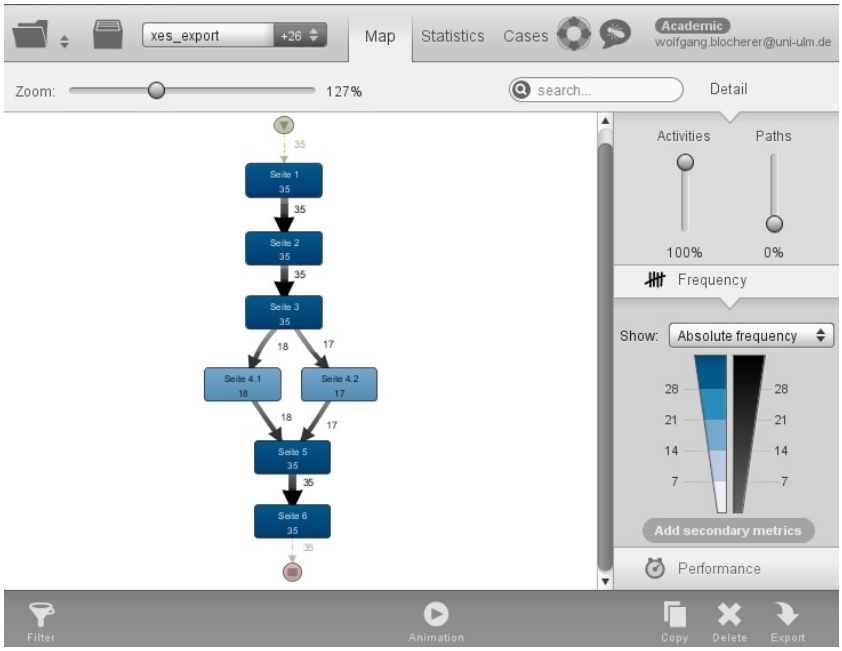

Abbildung 5.7: Disco Map

telte Prozessmodell zeigt neben den Namen der Aktivitäten die Häufigkeit der einzelnen Aktivitäten in den Log-Daten. Anstelle der Häufigkeit lassen sich mit einer einfachen Navigation am rechten Rand auch weitere Kennzahlen einblenden. So bietet die Rubrik Performance zeitliche Werte wie die durchschnittliche Ausführungsdauer oder die maximale Ausführungszeit einer Aktivität. Wie am Prozessmodell zu erkennen ist, hat der Fragebogen einen Abzweig. 18 mal wurde der Zweig über *Seite 4.1* gewählt, 17 mal der Zweig mit der *Seite 4.2*.

Die Ansicht Statistics zeigt mit einem Dashboard Übersichtszahlen oder Details zu Aktivitäten und einzelnen Attributen. Abbildung 5.8 zeigt einen Screenshot der Übersichtskennzahlen. Dies sind bspw. die Gesamtanzahl von Events, Instanzen und Aktivitäten. Die Anzahl an Aktivitäten entspricht der Anzahl verschiedener Fragebogenseiten in den Log-Daten. Zudem enthält die Ansicht ein Diagramm welches die Events der 35 Fragebögen auf einem Zeitstrahl abbildet. Auf der linken Seite der Anwendungen können einzelne Attribute ausgewählt werden.

Wie zu sehen ist, sind die Regelergebnisse ebenfalls als Attribute verfügbar. Wird eines dieser Attribute ausgewählt, ändert sich das Dashboard entsprechend. Angezeigte Werte sind bspw. alle vorkommenden Werte des Attributs sowie deren Häufigkeit. Bezogen auf die Regel kann somit nachvollzogen werden, wie viele Instanzen die jeweilige Regel erfüllen.

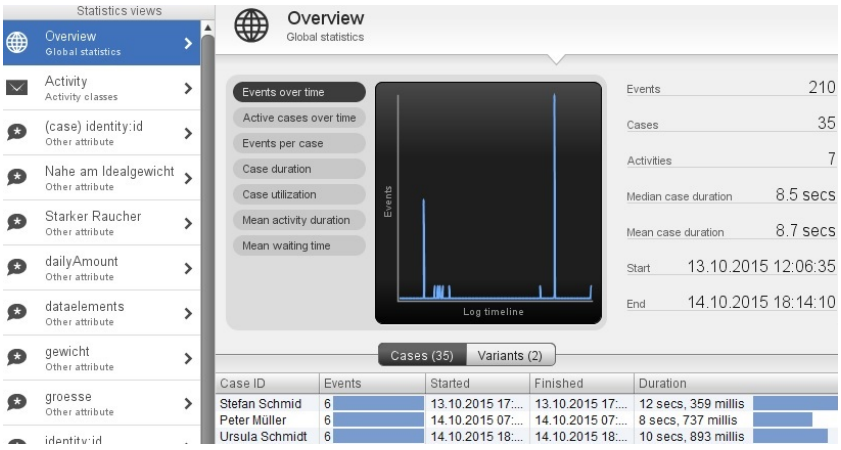

Abbildung 5.8: Disco Statistics

Abbildung 5.9 zeigt einen Screenshot der Ansicht Cases. In dieser Ansicht können einzelne Instanzen durchgesehen werden. Abbildung 5.9 zeigt die tabellarische Übersicht zur Instanz. In der darüber liegenden Zeit-Grafik ist die ausgewählte Instanz rot markiert. Auf der linken Seite der Anwendung kann zwischen den einzelnen Instanzen navigiert werden. Zudem sind dort die verschiedenen Ausführungsvarianten ebenfalls in die Navigation integriert.

Neben den drei Ansichten Map, Statistics und Cases gibt es zusätzlich die Möglichkeit Filter anzuwenden. Die Filter können unten links in der Anwendung hinzugefügt werden, siehe Abbildung 5.7. Somit können anhand der Regelergebnisse Instanzen gefiltert werden.

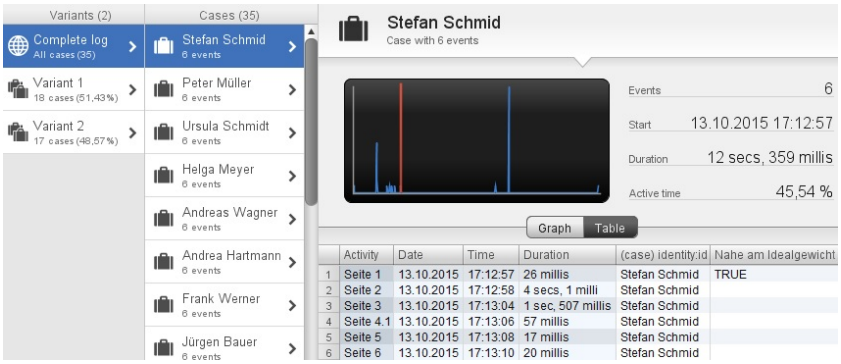

Abbildung 5.9: Disco Cases

5.3.4 Integrierte Analysen und weitere Export-Formate

Neben den vorgestellten Analysemöglichkeiten, die sich durch den Export im XES Standard ergeben, soll gemäß den Anforderungen AF 3 und AF 4 die QuestionAnalysis Komponente selbst Analysen bereitstellen. Insbesondere in Bezug auf die Regelauswertung. Wie im obigen Kapitel zu sehen, werden Ergebnisse von Regelauswertungen in Process Mining Tools nicht grundsätzlich berücksichtigt. Im Disco Beispiel konnten die Ergebnisse lediglich in Form eines Attributs berücksichtigt werden.

Zudem ist es sinnvoll, besonders relevante und häufig durchgeführte Analysen direkt in die QuestionAnalysis Komponente zu integrieren. Das Projekt richtet sich auch an Anwender ohne speziellen Process Mining Hintergrund. Daher sollen relevante Ergebnisse auf einfache Weise dargestellt werden können, ohne dass zwingend weitere spezifische Software Werkzeuge zum Einsatz kommen müssen.

Hierzu wurde eine einfach zu bedienende grafische Oberfläche konzipiert, mit der es möglich ist die Ereignislogs basierend auf den Regeln zu analysieren. Abbildung 5.10 zeigt eine schematische Darstellung der grafischen Oberfläche. Am oberen Rand ist die Menüleiste platziert. In den zur Verfügung stehenden Menüs können Daten importiert, selektiert, gelöscht oder exportiert werden. Zudem kann zwischen den beiden Komponenten QuestionRule und QuestionAnalysis gewechselt werden. Die Arbeitsfläche der Anwendung besteht aus vier Bereichen. Unten rechts werden Übersichtskennzahlen und Diagramme zur Verfügung gestellt. Unten links werden die aggregierten Ergebnisse

der Regelauswertung dargestellt. Darüber befindet sich die Liste der Instanzen. Die Liste enthält zudem die Ergebnisse der Regelauswertung pro Instanz. Durch eine einfache Selektion, können Instanzen anhand der Regelergebnisse ausgewählt werden. Rechts oben befindet sich die Detailansicht einer einzelnen Instanz. Durch die Instanz-Liste ist es möglich, einzelne Instanzen für die Detailansicht auszuwählen.

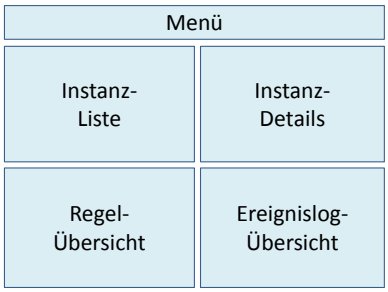

Abbildung 5.10: Schema der grafischen Oberfläche

Zusätzlich zu den Analysen die in der grafischen Oberfläche integriert sind, wurde gemäß Anforderung AF 11 ein erster CSV Export realisiert. Damit sollen weitere Analysen für Anwender ermöglicht werden die mit bspw. Microsoft Excel vertraut sind. In Abbildung 5.11 ist ein Beispiel CSV Export dargestellt.

Name	groesse	gewicht	dailyAmount	Nahe am Idealgewicht	Übergewicht	Starker Raucher
Stefan Schmid	170	70	7	TRUE	FALSE	FALSE
Peter Müller	190	80	11	FALSE	FALSE	TRUE
Ursula Schmidt	180	80	8	TRUE	FALSE	FALSE

Abbildung 5.11: CSV Export

6

Implementierung

Dieses Kapitel stellt die grafische Oberfläche der realisierten Analyseanwendung sowie zwei Implementierungsaspekte vor. Die beiden Implementierungsaspekte betreffen die Regelauswertung sowie die Plugin-Schnittstelle. Die Details der Plugin-Schnittstelle sind für die spätere Benutzung der Anwendung besonders relevant, da die Anwendung mit diesem Mechanismus erweitert werden kann.

6.1 Regelauswertung mit JEXL

Zur Auswertung der Regeln wurde JEXL (Java Expression Language) [32] ausgewählt. JEXL wird im Folgenden vorgestellt sowie die Implementierung im InstanceDataManager.

6.1.1 JEXL

JEXL ist eine Java Bibliothek zur Auswertung von Skripten. Sie basiert auf der JSTL (JSP Standard Tag Library) Expression Language [33].

Zur Auswertung der Regeln, die von der Regelkomponente erstellt wurden, sind einfache mathematische Ausdrücke relevant, welche zudem Java Objekte enthalten können. In den Ausdrücken können alle Vergleichsoperatoren wie $<$, $<=$, $==$, $>$ und $>=$, boolesche Operatoren (&& und ||) sowie einfache Operatoren ($+$, $-$, $/$ und $*$) enthalten sein (siehe [5]). Zudem unterstützt JEXL Methodenaufrufe von Java Objekten. Die Objekte, welche die verwendeten Methoden implementieren, müssen bei Auswertung des Ausdrucks verfügbar sein.

Ein einfaches Beispiel ist in Listing 6.1 zu sehen. Ein auszuwertender Ausdruck liegt in Form eines Strings vor (Zeile 2), mit dem eine JEXL Expression angelegt wird (Zeile 3). Hierzu wird die JexlEngine verwendet (Zeile 1). Um die Expression auszuwerten, werden die enthalten Variablen durch tatsächliche Werte ersetzt. Variablen sind wie bereits erläutert auch verwendete Objekte. Für den Aufruf der Methode *math.min()* benötigt JEXL zur Auswertung eine Instanz der Klasse Math. Zur Bereitstellung der Variablen wird das Objekt JexlContext befüllt (Zeilen 4-7). Hier wird jeweils der Name der Variablen sowie der Wert, bzw. eine Instanz, übergeben. Mit der Expression Methode *evaluate()* wird der Ausdruck ausgewertet (Zeile 6).

```
1 JexlEngine jexl = new JexlEngine();
2 String jexlExp = "math.min(d1,d2)";
3 Expression e = jexl.createExpression( jexlExp );
4 JexlContext jc = new MapContext();
5 jc.set("math", new Math() );
6 jc.set("d1", 5 );
7 jc.set("d2", 7 );
8 Object o = e.evaluate(jc);
```

Listing 6.1: JEXL Beispiel [32]

Die JexlEngine kann im Modus *strict* verwendet werden. Treten bei der Auswertung der Ausdrücke Fehler auf, wird eine Java Exception vom Typ JexlException ausgelöst. Die JexlException enthält eine detaillierte Fehlermeldung. Mögliche Fehler sind bspw. das Fehlen einer notwendigen Variable, oder ein Syntaxfehler des Ausdrucks.

6.1.2 Implementierungsdetails

In der Analysekomponente wird die Regelauswertung durch den InstanceDataManager pro Instanz durchgeführt. Hierzu greift der InstanceDataManager auf die importierten Regeln des DataManager zu. Zudem auf die Funktionen, welche ebenfalls vom DataManager bereitgestellt werden. Der JexlContext wird zu Beginn mit allen zur Verfügung stehenden Funktionen gefüllt (siehe Abbildung 6.1).

Jeder Funktionsaufruf in einer Regel endet mit dem Methodenaufruf *evaluate()*. Die Variablennamen die an den JexlContext übergeben werden, enthalten somit den vollständigen Klassennamen der Funktion, ohne den Methodenaufruf *evaluate()*.

Anders als die Klasse Math, welche in Listing 6.1 verwendet wurde, enthält eine Function Klasse nicht mehrere Methoden wie bspw. *min()*, *max()* und *abs()* der Math Klasse. Eine Function Klasse enthält wie bereits bekannt jeweils nur eine Auswertungs-Methode [5].

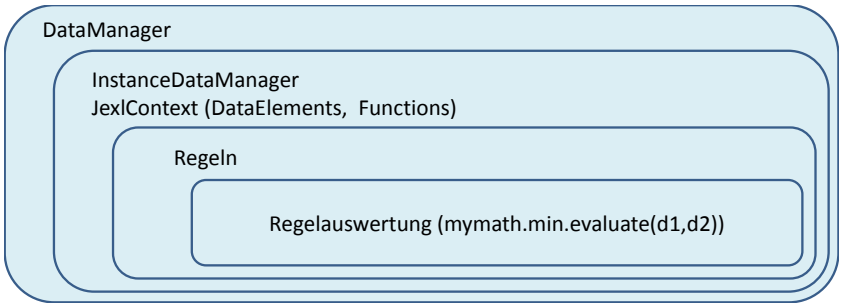

Abbildung 6.1: Regelauswertung

Nachdem alle Funktionen dem JexlContext hinzugefügt wurden, werden anschließend alle Datenelemente (Antworten der Fragen) ebenfalls als Variablen zum JexlContext hinzugefügt. Damit ist der JexlContext vollständig befüllt.

Für jede importierte Regel wird nun eine Expression angelegt. Der zuvor befüllte JexlContext bleibt für jede Regel derselbe. Anschließend werden alle Regeln ausgewertet. Die Ergebnisse der Regelauswertung sowie Fehlermeldungen werden in einem RuleResult Objekt (vgl. Kapitel 5.2) abgelegt.

6.2 Plugins

In Kapitel 2.3 wurde das Eclipse Plugin Konzept als Teil des Eclipse RCP Frameworks eingeführt. Nachfolgend werden Details der konkreten Implementierung in der Analysekomponente behandelt. Zunächst werden die Schritte vorgestellt, welche für das Hinzufügen eines Plugins nötig sind. Zudem die dafür erforderlichen anwendungsspezifischen Angaben.

6.2.1 Plugin Schnittstelle

In [18] findet sich ein detailliertes Tutorial wie Plugins zu einer Anwendung hinzugefügt werden. Die folgenden Erläuterungen sind an dieses Tutorial angelehnt. Zu Beginn wird in der Eclipse Entwicklungsumgebung ein neues Plugin-Projekt angelegt. Anschließend werden in der Rubrik Abhängigkeiten (engl. Dependencies) die benötigten Plugins hinzugefügt. Neben RCP Standard Plugins werden die beiden Plugins hinzugefügt, welche die Regelkomponente und Analysekomponente beinhalten.

- com.questionsys.questionanalysis
- com.questionsys.questionrule

Durch das Hinzufügen der beiden Plugins, stehen dem zu erstellenden Plugin alle relevanten Klassen zur Verfügung. Diese sind in den zu exportierenden Paketen (engl. exporting packages) enthalten. Konkret sind dies die aus Kapitel 5.1 bekannten Manager. In Abbildung 6.2 sind die Manager auf der linken Seite abgebildet.

In den nächsten Schritten wird eine Handler Klasse angelegt sowie das Model-Fragment. Dem Model-Fragment wird ein Menüpunkt hinzugefügt, welcher später im Menü der Analysekomponente zur Verfügung stehen soll. Hierzu wird zunächst im Model-Fragment die ID des Applikationsmodells angegeben: org.eclipse.e4.ide.application.

Zudem die ID des Menüs, zu welchem der Eintrag hinzugefügt werden soll. Folgende IDs sind für die drei Menüs *Fragebögen importieren*, *Regeln importieren* und *Fragebögen exportieren* vergeben:

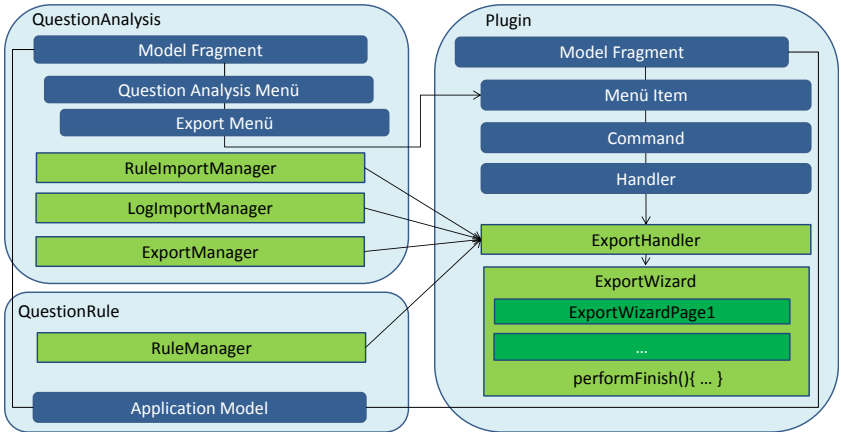

Abbildung 6.2: Plugin Implementierung

- com.questionsys.questionanalysis.menu.importlog
- com.questionsys.questionanalysis.menu.importrule
- com.questionsys.questionanalysis.menu.export

Eine beispielhafte Implementierung einer Handler-Klasse ist in Listing 6.2 zu sehen. Per Dependency Injection wird auf den ExportManager zugegriffen (Zeile 2). Der ExportManager wird an den Wizard übergeben, welcher nach dem Betätigen des Menüs aufgerufen wird (Zeile 6). In der Methode *canExecute()* wird auf die Methode *getExecute()* des ExportManagers zugegriffen (Zeilen 10-13). Diese steuert, ob der Menüeintrag aktiv ist oder nicht. Er ist nur aktiv, solange die Analyse-Perspektive aktiv ist.

```
1  public class XESExportHandler {
2          @Inject ExportManager manager;
3
4          @Execute
5          public void execute(Shell shell) {
6                  WizardDialog dialog = new WizardDialog(shell, new XESExportWizard(manager));
7                  dialog.open();
8          }
9
10         @CanExecute
11         public boolean canExecute( EModelService modelService) {
12                 return manager.getExecute();
13         }
14 }
```

Listing 6.2: Handler

Im letzten Schritt wird das Plugin zur Datei feature.xml hinzugefügt. Diese sammelt die Plugins der Anwendung. Plugins die in der feature.xml Datei aufgelistet sind, werden beim Erstellen der .exe Datei mit exportiert. Zu einer bereits bestehenden .exe Anwendung, können Plugins im Ordner *plugins* hinzugefügt werden.

6.2.2 Wizard

Ein Wizard führt den Benutzer durch eine Aktion mit mehreren Schritten [34]. Im Fall des XES Export ist dies das Konfigurieren des Exports sowie das Auswählen des Speicherorts. Jeder der Schritte wird auf einer Wizard-Seite der Reihe nach ausgeführt. Für die Import und Export Plugins wurde daher der Wizard für die Umsetzung ausgewählt.

Der Wizard ist dabei nicht Teil der Analysekomponente, sondern jedes Plugin implementiert einen Wizard (siehe Abbildung 6.2). Somit kann jedes Plugin eine flexible Konfiguration des Import oder Exports anbieten. Bspw. kann somit konfiguriert werden, ob und an welcher Stelle ein Export Format die Regelergebnisse hinzufügt, vgl. Abbildung 5.6.

In [34] wird die Funktionsweise eines Wizards anhand eines Tutorials erklärt. Ein Wizard besteht aus einer Hauptklasse und mehreren Seiten. Die Hauptklasse erbt von der Klasse Wizard, die Seiten von der Klasse WizardPage. Die Hauptklasse hält jeweils eine Referenz der einzelnen Seiten. Über Methoden die von den Seiten bereitgestellt werden, kann die Hauptklasse die Eingabedaten auslesen. Somit kann sie die Reihenfolge der einzelnen Seiten abhängig von den Eingabedaten steuern. Spätestens nachdem der Wizard mit *Finish* beendet wird, greift die Hauptklasse auf die Eingabewerte der einzelnen Seite zu, um bspw. den Export wie vom Benutzer konfiguriert durchführen zu können.

Nach dem Beenden des Wizard mit *Finish*, wird die *performFinish()* Methode aufgerufen. Eine Beispiel-Implementierung ist in Listing 6.3 zu sehen. Die Variable *page2* (Zeile 3) ist eine Referenz auf eine der Wizard Seiten. Um Listing 6.3 kompakt zu halten, wurde u. a. auf die Anreicherung mit den Ergebnissen der Regelauswertung verzichtet.

```
1  @Override
2  public boolean performFinish() {
3          String location = page2.getTemplateLocation();
4          XLog log = exportManager.getXESLog();
5          XesXmlSerializer serializer = new XesXmlSerializer();
6          File target = new File(location);
7          OutputStream os = null;
8          try {
9                  os = new BufferedOutputStream(new FileOutputStream(target));
10         } catch (FileNotFoundException e) {
11                 e.printStackTrace();
12         }
13         try {
14                 serializer.serialize(log, os);
15                 os.flush();
16                 os.close();
17         } catch (IOException e) {
18                 e.printStackTrace();
19         }
20         return true;
21 }
```

Listing 6.3: performFinish() Methode

6.3 Benutzeroberfläche

Wie bereits erläutert, wird die Analysekomponente als Plugin zur bestehenden Regelkomponente hinzugefügt.

Nach dem Start der Anwendung, wird dem Benutzer zunächst die Regelkomponente (Regel-Perspektive) angezeigt. Über das Menü kann zur Analysekomponente (Analyse-Perspektive) gewechselt werden. Die Benutzeroberfläche der Analysekomponente besteht aus einem separaten Menü sowie insgesamt sechs verschiedenen Parts. Parts sind die in Kapitel 2.3 vorgestellten Oberflächen-Container. Sie lassen sich mittels Drag&Drop verschieben, ein- und ausblenden.

Nach dem Start der Anwendung sind zunächst vier Parts sichtbar. Die beiden weiteren Parts sind lediglich als Register sichtbar. Da der Benutzer die Anordnung der Parts beliebig verschieben kann, können auch alle sechs Parts zeitgleich angezeigt werden.

6.3.1 Menü der Analysekomponente

Der Menübaum der Analysekomponente ist in Abbildung 6.3 dargestellt. Mit dem ersten Menüpunkt *Analyse-Perspektive zeigen*, kann der Benutzer von der Regel-Perspektive zur Analyse-Perspektive wechseln. Der selbe Menüpunkt findet sich analog im Menü der Regelkomponente.

Mit dem Menüpunkt *Fragebögen importieren* können die Import-Plugins aufgerufen werden. Analog können mit den Menüpunkten *Regeln importieren* und *Fragebögen exportieren* die dort verfügbaren Plugins aufgerufen werden. Die bislang vier realisierten Plugins sind in Abbildung 6.3 aufgeführt.

Der Menüpunkt *Fragebögen selektieren* hat die Unterpunkte *Alle selektieren* und *Alle de-selektieren*. Die beiden Menüpunkte beziehen sich auf die Instanz-Liste, welche nachfolgend vorgestellt wird. Der Menüpunkt *Fragebögen entfernen*, die beiden Unterpunkte *Selektierte entfernen* und *Nicht selektierte entfernen* bezieht sich ebenfalls auf die Instanz-Liste.

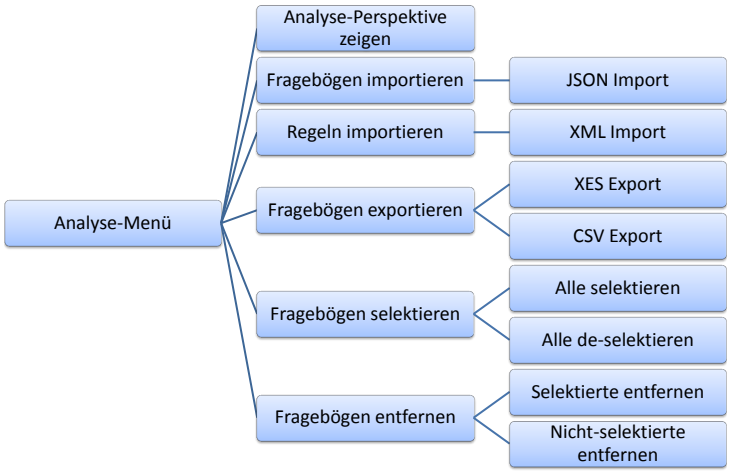

Abbildung 6.3: Menübaum Analysekomponente

6.3.2 Log-Daten Übersicht

Wie in den Anforderungen (siehe Kapitel 4) verlangt, wurden Parts zur Übersicht über die importierten Ereignislogs erstellt. Vier Parts mit Übersichts-Kennzahlen wurden realisiert. Abbildung 6.4 zeigt einen Screenshot der Anwendung, in der alle vier Übersicht Parts eingeblendet sind. Ein Part mit allgemeinen Kennzahlen (oben links), zudem ein Part mit Informationen zu den enthaltenen Datenelementen des Ereignislogs (oben rechts). Zu den Datenelementen wird jeweils in Spalten angezeigt, wie oft das Element vorkommt und wie viele verschiedene Werte es annimmt. Die Grafik (oben mitte) zeigt die Anzahl der Fragebögen nach Erfassungsdatum.

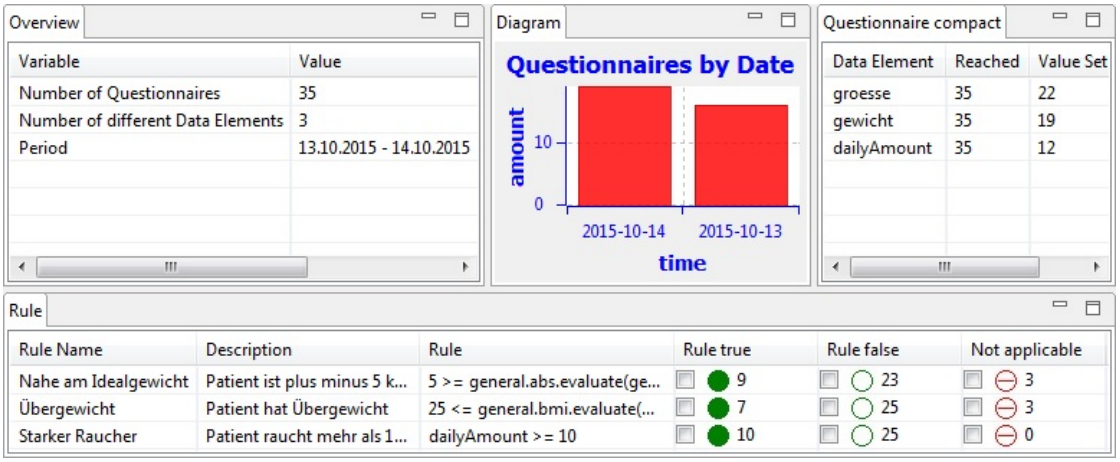

Abbildung 6.4: Log-Daten Übersicht

Der vierte Part (unten) gibt eine Übersicht über die importierten Regeln. Zu jeder Regel wird der Name, die Beschreibung und die Regel selbst angezeigt. Die Spalten *Rule true*, *Rule false* und *Not applicable* enthalten die aggregierten Regelergebnisse. Es wird die Anzahl der Instanzen angezeigt, für welche das Ergebnis den Wert TRUE, FALSE oder NOTAPPLICABLE annimmt.

Um die Anwendung übersichtlich zu gestalten, wurden Symbole hierfür eingeführt. Für TRUE und FALSE wurde jeweils ein grünes Symbol gewählt. Die Farbe grün symbolisiert, dass die Regel angewendet werden konnte. Als Form wurde ein leerer und ein ausgefüllter Kreis gewählt. Die Symbole sollen nicht negativ oder positiv belegt sein,

da TRUE oder FALSE je nach Formulierung der Regel beide negativ oder positiv sein können.

Die Texte welche für TRUE und FALSE hinterlegt sind, bspw. TRUE entspricht „Patient hat Übergewicht“, werden in einem Tooltip angezeigt. Es wird angezeigt, sobald mit dem Mauszeiger über ein Regelergebnis gefahren wird.

Die Checkboxen im Regel-Part können verwendet werden, um Instanzen anhand der Regelergebnisse zu selektieren, wie in Anforderung AF 3 beschrieben. Die Selektion bezieht sich auf die Instanz-Liste, welche nachfolgend als Teil der Datendurchsicht vorgestellt wird.

6.3.3 Durchsicht von Instanzen

Um einzelne Instanzen durchsehen zu können, wurde die Instanz-Liste erstellt. Zudem ein Part mit einer Detail Ansicht einer einzelnen Instanz. In Abbildung 6.5 sind beide Parts eingeblendet. In der Instanz-Liste wird der Name des Bearbeiters angezeigt, das Datum sowie die Ergebnisse der Regelauswertung pro Instanz. Für jede importierte Regel wird in der Instanz-Liste eine Spalte angelegt. Für die Darstellung des Ergebnisses pro Instanz werden die bereits vorgestellten Symbole verwendet.

Konnte die Regel nicht angewendet werden, so wird die Ursache dafür (RuleResult Feld *message*) in einem Tooltip angezeigt. Die Instanz-Liste lässt sich nach allen Spalten sortieren. Zudem wurde ein Suchfeld angelegt, um Instanzen anhand des Namens des Bearbeiters suchen zu können.

Jede Instanz lässt sich durch eine Checkbox am linken Rand einzeln auswählen. Somit ist es möglich nur ausgewählte Instanzen zu exportieren, oder einzelne Instanzen von der Arbeitsfläche zu löschen.

Wird eine Instanz ausgewählt, so werden die Details zur Instanz im Detail-Part angezeigt. Da mehrere Instanzen ausgewählt werden können, werden die Details zur letzten ausgewählten Instanz angezeigt. In der Detailansicht werden alle in der Instanz enthaltenen Datenelemente angezeigt, sowie die Wertbelegung.

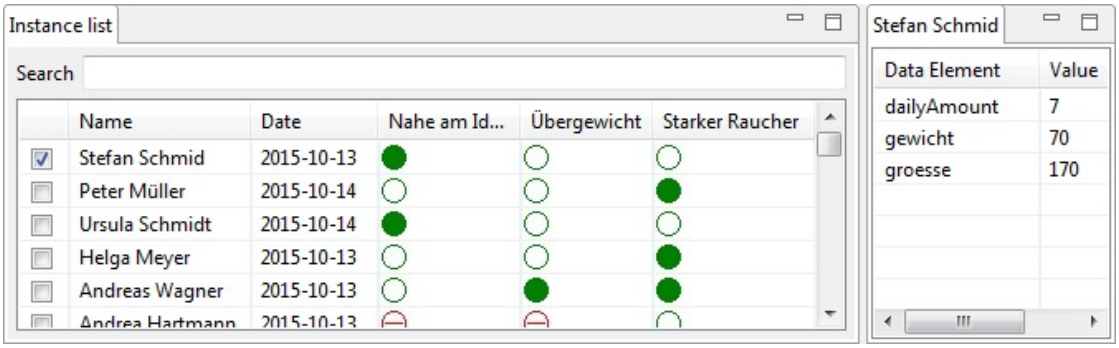

Abbildung 6.5: Fragebögen Durchsicht

Instanzen können auch wie bereits erläutert über die Checkboxen des Regel-Part ausgewählt werden. Werden Instanzen auf diese Weise ausgewählt, werden alle Selektionen aufgehoben, und nur die Instanzen mit dem entsprechenden Regelergebnis sind ausgewählt. Umgekehrt gehen die Selektionen im Regel-Part verloren, sobald eine einzelne Instanz in der Instanz-Liste ausgewählt wird.

Mit der Funktion *Alle de-selektieren* in der Menüleiste können die getätigten Selektionen wieder aufgehoben werden.

7

Zusammenfassung & Ausblick

In diesem Kapitel wird zusammengefasst, welchen Beitrag die Analysekomponente im Fragebogensystem QuestionSys leistet. Zudem wird in einem Ausblick diskutiert, um welche Funktionalitäten die Analysekomponente zukünftig erweitert werden könnte.

7.1 Beitrag der Analysekomponente

Die im Rahmen dieser Ausarbeitung erstellte Komponente erweitert das generische Fragebogensystem um die gewünschten Analysefunktionalitäten. Wie in Kapitel 1 betont, stellt die automatisierte Auswertung von Fragebögen ein zentrales Ziel von Fragebogensystemen dar.

Als Grundlage zur Auswertung der Fragebögen, wurde in [5] bereits die Regelkomponente vorgestellt. Mit dieser Komponente können Regeln basierend auf einem Frage-

bogenmodell erstellt werden. Diese Regeln können zur Auswertung von Fragebögen herangezogen werden. Die Auswertung der Fragebögen mit diesen Regeln wird mit der Analysekomponente durchgeführt.

Die Analysekomponente wurde zur bereits bestehenden Regelkomponente hinzugefügt. Beide Komponenten bilden zusammen eine Anwendung basierend auf dem Eclipse RCP Framework, vgl. Abbildung 7.1. Dies ermöglicht eine einfache Handhabung für den Benutzer. Nachdem der Benutzer mit der Komponente Regeln erstellt hat, kann er in der selben Anwendung zur Analyseansicht wechseln. In der Analysekomponente kann der Benutzer nun auswählen, welche Regeln er für eine Analyse verwenden möchte.

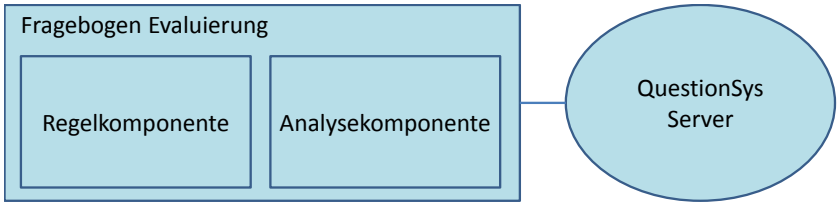

Abbildung 7.1: Fragebogen Evaluierung

Vor oder nach der Auswahl der Regeln, importiert der Benutzer die zu analysierenden Fragebögen. Da das Fragebogensystem QuestionSys auf Prozessmanagement-Technologie basiert, liegen die ausgefüllten Fragebögen in Form von Log-Daten vor. Hierzu wurde ein Import-Plugin erstellt, mit welchem die Log-Daten eingelesen werden können. Nachdem Regeln und Log-Daten in die Analysekomponente importiert wurden, findet automatisch die Auswertung statt.

Die Ergebnisse der Regelauswertung werden auf der grafischen Oberfläche der Analyseanwendung dargestellt. Es werden Übersichtskennzahlen sowie eine Liste mit Details zu einzelnen Fragebögen angezeigt. Die Fragebogen-Liste kann anhand der Ergebnisse gefiltert und sortiert werden.

Die Fragebögen können in verschiedenen Formaten exportiert werden. Dazu wurden bspw. ein CSV- und XES-Export realisiert, die nachträgliche Analysen in anderen Anwendungen erlauben. Wie bereits erläutert, ist XES ein Log-Daten Standard Format für Process Mining Anwendungen. Durch den Export in diesem Format wird es ermög-

licht, die Fragebögen mittels Process Mining Techniken zu analysieren. Zusätzlich zur Analyse mit den Regeln, kann der Benutzer somit auf diese Möglichkeit der Analyse zurückgreifen. Zudem kann jeder Export optional mit den Regelergebnissen angereichert werden. Diese können somit auch bei der Analyse mit Process Mining Anwendungen berücksichtigt werden.

Insgesamt stehen dem Anwender somit schnelle Analysen basierend auf Regeln zur Verfügung, sowie die Möglichkeit von Analysen mittels Process Mining Technik.

7.2 Ausblick

Um die Analysekomponente zukünftig um neue Importquellen und Exportformate erweitern zu können, wurde der Plugin-Mechanismus entwickelt. Als neue Exportformate sind bspw. weitere CSV-Exporte oder neue Process Mining Formate denkbar. Ein formatierter Export im PDF-Format einzelner Fragebögen ist ebenfalls denkbar. Somit könnten Fragebögen mit den Ergebnissen der Regelauswertung ausgedruckt werden. Hierzu ist allerdings ein Import des Fragebogenmodells in die Analysekomponente erforderlich. Denn die Log-Daten alleine enthalten nicht die zugehörigen Fragen im Klartext.

Mit dem bisher umgesetzten Import-Plugin können Log-Daten in Form von Dateien geöffnet werden. Denkbar ist jedoch auch eine Datenbankverbindung zum QuestionSys Server oder eine REST [35] Schnittstelle. Somit könnten Fragebögen selektiert werden, bevor sie mit der Analysekomponente verarbeitet werden.

Neben neuen Import- und Export-Plugins könnte auch die Benutzeroberfläche erweitert werden. Bspw. eine Verbesserung der Selektions- und Sortiermöglichkeiten. Wie bereits erläutert, können Fragebögen anhand der Regelergebnisse selektiert werden. Momentan kann immer genau eine Regel zur Selektion ausgewählt werden. Denkbar wäre auch eine Verknüpfung von Regeln mit UND/ODER Logik, bspw. zur Selektion von Fragebögen die zwei oder mehr Regeln erfüllen.

Neben der Selektion und Sortierung anhand von Regeln, könnte diese Möglichkeit auch für Attribute (Antworten) der Fragebögen hinzugefügt werden. Sodass Fragebögen in

der Anwendung bspw. anhand des Datums selektiert werden können. Dies stellt bisher einen besonderen Nachteil dar, da bislang nur Dateien importiert werden können. Das Handling von großen Log-Dateien ist ohne Attribut-Filtermöglichkeit umständlich, da nicht relevante Fragebögen ggf. einzeln entfernt werden müssen.

Die Realisierung der Selektion und Sortierung nach Attributen könnte gut auf Basis der aktuellen Benutzeroberfläche erfolgen. Der Attribut-Filter könnte zur Fragebogenübersicht hinzugefügt werden, die Sortiermöglichkeit, indem Attribute auch in der Instanz-Liste eingeblendet werden.

Als zukünftige Erweiterungen sind auch neue Übersichts-Kennzahlen und Diagramme denkbar. Relevante Kennzahlen und Diagramme hängen jedoch vom konkreten Einsatzgebiet des Fragebogensystems ab. Um dem gerecht zu werden, könnte wiederum das Eclipse RCP Plugin Konzept in Betracht gezogen werden. Es könnte ein Plugin-Mechanismus zum Hinzufügen von Kennzahlen und Diagrammen entwickelt werden, ähnlich des Mechanismus für Import- und Export-Plugins. Somit könnten flexibel Kennzahlen und Diagramme hinzugefügt werden.

Letzter Punkt in diesem Ausblick ist die Integration zwischen Regel- und Analysekomponente. Diese könnte zukünftig noch ausgebaut werden. Bislang kann bspw. nur über das Hauptmenü von der Regelkomponente zur Analysekomponente gewechselt werden. Denkbar wäre ein Rechtsklick innerhalb der Regelkomponente, mit dem zur Analysekomponente gewechselt werden kann. Mit einem Rechtsklick auf eine Regel oder ein Projekt (vgl. [5]) könnten Regeln direkt als Eingabe für die Analysekomponente ausgewählt werden. Bislang müssen Regeln nach dem Wechsel in die Analysekomponente erneut selektiert werden, dies kann bei häufigem Wechseln zwischen den beiden Komponenten stören.

Literaturverzeichnis

[1] Schobel, J., Schickler, M., Pryss, R., Maier, F., Reichert, M.: Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: 10th Int'l Conf on Web Inf Sys and Technologies (WEBIST 2014), Special Session on Business Apps, Barcelona (2014)

[2] SurveyMonkey: Free online survey software and questionnaire tool. http://www.surveymonkey.com/ (10.12.2015)

[3] Electric Paper Evaluationssysteme GmbH: EvaSys. http://www.evasys.de/startseite.html (10.12.2015)

[4] Schobel, J., Schickler, M., Pryss, R., Reichert, M.: QuestionSys – A Generic and Flexible Questionnaire System Enabling Process-Driven Mobile Data Collection. http://www.uni-ulm.de/en/in/dbis/research/projects/questionsys.html (14.09.2015)

[5] Mertesz, B.: Concept and Implementation of a Rule Component Enabling Automatic Analysis of Process-Aware Questionnaires. Masterarbeit, Universität Ulm. http://dbis.eprints.uni-ulm.de/1096/ (10.09.2015)

[6] van der Aalst, W.M.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)

[7] Weske, M.: Business Process Management - Concepts, Languages, Architectures. Springer, Heidelberg (2012)

[8] Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2013)

[9] IEEE Task Force on Process Mining: Process Mining Manifesto. In: Barkaoui K, Dustdar S (eds) Business Process Management Workshops (1), Vol 99 of Lecture Notes in Business Information Processing., Berlin, Springer (2011) 169–194

[10] Accorsi, R., Ullrich, M., van derAalst, W.M.P.: Aktuelles Schlagwort – Process Mining. In: Informatik Spektrum 35-5-2012, Berlin, Springer (2012) 354–359

[11] Heckl, D., Moormann, J.: Process Performance Management. In: Handbook on Business Process Management 2, Heidelberg (2010) 115–135

[12] Günther, C.W., Verbeek, E.: XES Standard Definition Version 2.0. `http://www.xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf` (28.03.2014)

[13] Technische Universität Eindhoven: Event logs. `http://www.processmining.org/logs/start` (20.10.2015)

[14] Verbeek, E.: Open XES. `http://www.xes-standard.org/openxes/start` (30.09.2015)

[15] Eclipse Foundation: Eclipse4/RCP/Dependency Injection. `https://wiki.eclipse.org/Eclipse4/RCP/Dependency_Injection` (10.09.2015)

[16] Vogel, L.: Eclipse Extension Points and Extensions – Tutorial. `http://www.vogella.com/tutorials/EclipseExtensionPoint/article.html` (10.09.2015)

[17] Meixner, G.: Der Eclipse-Plugin-Mechanismus. `http://www.hs-augsburg.de/~meixner/saj/skript/osgi/eclipseplugins.html#l` (10.09.2015)

[18] Vogel, L.: Eclipse application model modularity with fragments and processors - Tutorial. `http://www.vogella.com/tutorials/Eclipse4Modularity/article.html#contributefragments_extensionpoint` (10.09.2015)

[19] Eclipse Foundation: Eclipse4/RCP. `https://wiki.eclipse.org/Eclipse4/RCP#Model_Elements` (10.09.2015)

[20] Process Mining Group, Technische Universität Eindhoven: ProM Tools. `http://www.promtools.org/doku.php` (3.11.2015)

[21] Fluxicon: Disco. https://fluxicon.com/disco/ (09.10.2015)

[22] Bosch Software Innovations: Visual Rules BRM. https://www.bosch-si.com/media/de/bosch_software_innovations/documents/brochure/products_2/visual_rules_brm/brochure.pdf (03.11.2015)

[23] Eckert, M., Bry, F.: Aktuelles Schlagwort – Complex Event Processing (CEP). In: Informatik Spektrum 32-2-2009, Berlin, Springer (2009) 163–164

[24] Kumar, N., Patil, D.D., Wadhai, V.M.: Rule based programming with Drools. In: International Journal of Computer Science and Information Technologies, Vol. 2 (3), Singapore, Singapore Computer Society (2011) 1121–1126

[25] Rymer, J.R., Gualtieri, M.: Market Overview: Business Rules Platforms 2011. http://isol.fox.proyectiva.com.ar/web/esp/brm/papers/ForresterMarket%20Overview%20Business%202011.pdf (05.07.2011)

[26] Florida Atlantic University: A Comparison Framework for Rule Engines. http://pire.fiu.edu/posters/2009/Petersen_PIRE09_poster.pdf (2009)

[27] De Ley, E., Jacobs, D.: Rule-based analysis with JBoss Drools. In: 13th International Conference on Accelerator and Large Experimental Physics Control Systems, Grenoble, ESRF (2011) 790–793

[28] Red Hat: Drools Expert User Guide. https://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/html_single (29.10.2015)

[29] Bosch Software Innovation: Business Process Management mit inubit BPM. https://www.bosch-si.com/de/produkte/business-process-management-bpm/business-process-management.html (03.11.2015)

[30] Oracle: Project JAXB. https://jaxb.java.net/ (10.12.2015)

[31] Technische Universität Eindhoven: MXML (Mining eXtensible Markup Language). http://www.processmining.org/logs/mxml1 (17.11.2015)

[32] The Apache Software Foundation: Java Expression Language (JEXL). http://commons.apache.org/proper/commons-jexl/ (22.09.2015)

[33] Corporation, O.: JSP Standard Tag Library. `https://jstl.java.net/l` (18.11.2015)

[34] Vogel, L.: Creating Eclipse Wizards – Tutorial. `http://www.vogella.com/tutorials/EclipseWizards/article.html` (09.10.2015)

[35] Fielding, R.T., Taylor, R.N.: Principled Design of the Modern Web Architecture. In: ACM Trans. Internet Technol. 2, ACM, New York (2002) 115–150

Abbildungsverzeichnis

Listings

Name: Wolfgang Blocherer Matrikelnummer: 847007

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Wolfgang Blocherer