Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3116

Entwicklung eines Werkzeugs zur standardisierten Verarbeitung von Prozessdaten und operativen Daten

Bernhard Maier

Studiengang:	Informatik
Prüfer:	Prof. Dr. Bernhard Mitschang
Betreuer:	M.Sc. Florian Niedermann
begonnen am:	27. Oktober 2010
beendet am:	14. April 2011
CR-Klassifikation:	H.4.1, H.2.8, H.5.2

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

1 Abstract / Kurzfassung

1.1 Abstract

In many lines of business, the execution of formally described business processes called workflows on Workflow Management Systems is already established and is gaining in importance. As many of these workflows describe the core processes, their performance is of key importance to the respective business. Consequently, businesses seek to optimize them with respect to their process goals (e.g. cost, benefit or customer satisfaction).
The deep Business Optimization Platform (dBOP [NRM10]) provides the basis for the optimization of workflows spanning their whole lifecycle including the design-, execution- and the analysis stage. The platform consists of the layers "Data Integration", "Process Analytics" und "Process Optimization". In the Data Integration layer, relevant sources of data are extracted and integrated. In the Process Analytics layer, this data is analyzed and data mining models as well as other insights are generated. These insights can be used in the Process Optimization layer in order to optimize workflow models.
This thesis elaborates on the design and the implementation of the Data Integration layer and the Process Analytics layer. These layers are the basis of the dBOP. The implemented software supports the whole process from extraction of relevant data to its analysis, and it provides its output as XML file for a later use in a webservice. The system was modeled as a pipes-filters [LL07] architecture. This led to a division into 5 steps of processing for the data. These "filters" were designed and implemented in the software.
The user is supported in configuring and running the filters by graphical and text-based widgets. With it, principles of human machine interaction are taken under consideration.
For the exchange of the data mining models and other data with existing dBOP software, respective XML schemas were created. A sample scenario from the insurance business domain demonstrates the application of the thesis results as well as showing their relevance.

1.2 Kurzfassung

In vielen Branchen ist die Unterstützung wichtiger Geschäftsprozesse durch Workflow Management Systeme bereits etabliert und gewinnt weiter an Bedeutung. Oftmals kann durch Optimierung eines Workflows dessen Qualität in Bezug auf Kosten, Nutzen, Kundenzufriedenheit und Ausführungsdauer verbessert werden. Die deep Business Optimization Platform (dBOP [NRM10]) bildet die Basis zur Optimierung von Workflows in allen Phasen ihres Lebenszykluses: Entwurfs-, Ausführungs- und Analysephase.
Die Plattform besteht aus den Schichten Data Integration, Process Analytics und Process

Optimization. In der Data Integration Schicht werden relevante Datenquellen, also Prozessdaten und sonstige operative Daten, zusammengeführt. In der Process Analytics Schicht werden diese Daten analysiert und Modelle und sonstige Erkenntnisse gewonnen, die dann in der Process Optimization Schicht zur Optimierung bestehender Prozessmodelle genutzt werden können.
In dieser Diplomarbeit wird auf den Entwurf und die Implementierung der Data Integration und Process Analytics Schichten eingegangen, die die Basis der Platform bilden.
Die entwickelte Software unterstützt den gesamten Ablauf bis zur Analyse der Daten und stellt die Analyse-Ergebnisse für eine spätere Verwendung in einem Webservice bereit.
Die Systemsicht der Software wurde als Pipes-Filters Architektur [LL07] modelliert. Daraus ergab sich eine Einteilung in 5 Verarbeitungsschritte, die die relevanten Daten durchlaufen müssen. Sie wurden im Rahmen der Diplomarbeit entworfen und implementiert.
Der Benutzer wird während des Ablaufs bis hin zur Modellerstellung mit entsprechenden graphischen und textbasierten Widgets, unter Berücksichtigung von Gesichtspunkten der Mensch-Maschine-Interaktion (HMI) [NW87] unterstützt. Für den Datenaustausch mit bestehender dBOP-Software auf Basis von XML, also z.B. zur Ausgabe der gewonnen Modelle, wurden geeignete XML Schemata erstellt. Ein Anwendungsszenario aus der Versicherungsbranche unterstreicht die Relevanz der Arbeit.

2 Einleitung

2.1 Einführung

In vielen Branchen ist es heutzutage Standard, dass wichtige Geschäftsprozesse durch Workflow Management Systeme unterstützt werden. Unter einem **Geschäftsprozess** (Business Process) versteht man einen Ablauf in einem Unternehmen, der in Aktivitäten untergliedert ist, die von verschiedenen Sachbearbeitern oder Ressourcen des Unternehmens ausgeführt werden, und der eine bestimmte Aufgabe erledigt. Zum Beispiel wird in einer Bank ein Geschäftsprozess angestoßen, wenn ein Kunde einen Kredit beantragt. Den Computergestützten Teil eines Geschäftsprozesses bezeichnet man als **Workflow-Modell** [LR00].
Workflow Management Systeme können Workflow-Modelle interpretieren und die darin vorgesehenen Abläufe anstoßen. Die Instanz des Workflow-Modells die dabei zur Ausführung kommt ist eine Workflow-Instanz oder kurz ein Workflow.
In größeren Unternehmen gibt es eine Vielzahl von Geschäftsprozessen, die für eine Unterstützung durch Workflow Management Systeme in Frage kommen. In [Har91] sind z.B. die typischen Geschäftsprozesse der Firma IBM zu finden. Bei IBM gab es 1991 allein 20 verschiedene Geschäftsprozesse der Sparte Kapitalbilanz, und daneben in der Sparte Finanzielle Planung 8 verschiedene Geschäftsprozesse (siehe Tabelle 2.1). Außerdem gab es

Kapitalbilanz	Ledger control (Buchführung)
	Payroll (Gehaltsliste)
	Taxes (Steuern)
	Accounts receivable (Buchforderungen)
	Accounts payable (Verbindlichkeiten)
	usw...
Finanz-Planung	Budget control (Budget Kontrolle)
	Cost estimating (Kostenschätzung)
	Business planning (Geschäftsplanung)
	Contract management (Vetragsverwaltung)
	usw...

Tabelle 2.1: Geschäftsprozesse von IBM (1991) in den Bereichen Kapitalbilanz und Finanzplanung [Har91]

Geschäftsprozesse in den Sparten: Entwicklung, Vertrieb, Informationssysteme, Produktionskontrolle, Einkauf, Personal, Programmierung, Qualität, usw.

2.2 Motivation

Während früher der Fokus bei den Workflow Management Systemen [1] auf dem Entwurf und der Ausführung von Workflows lag, so gibt es seit einigen Jahren die Bestrebung solche Systeme um Fähigkeiten der Prozessdiagnose zu erweitern. Das Business Process Management (BPM) ist eine Management Disziplin die diese Idee propagiert. In [ATHW03] wird Business Process Management definiert als:„Supporting business processes using methods, techniques, and software to design, enact, control, and analyse operational processes involving humans, organisations, applications, documents and other sources of information.“
Jedoch wird der Analyseschritt im Lebenszyklus von Workflows oft nicht ausreichend unterstützt. Die Analyse von Workflows, die die Basis zu ihrer Verbesserung ist, hat eine hohe Bedeutung, gerade wenn sich Unternehmen in einem vom Wandel geprägten Umfeld bewegen, in dem sie flexibel und schnell auf Änderungen reagieren können müssen. An der Universität Stuttgart wird derzeit die deep Business Optimization Platform entwickelt, die eine (automatisierte) Optimierung von Workflows während aller Phasen des Workflow-Lebenszyklus erlauben soll.
In [NRM10] wird die Optimierung in der Analyse Phase besprochen. Abbildung 2.1 zeigt die Architektur der Software. Sie besteht aus den 3 Schichten „Data Integration“, „Process Analytics“ und „Process Optimzation“. In der Data Integration Schicht wird der Audit Trail des WFMS ausgelesen und mit operativen Daten zusammengeführt, die z.B. in Human Resource Systemen vorliegen. Diese Daten werden in einem Data Warehouse abgelegt, durchlaufen dann in der Process Analytics Schicht einen Preprocessing Schritt und werden mit Metriken angereichert und schließlich analysiert. Die daraus gewonnenen Erkenntnisse/Modelle werden im Process Insight Repository (PIR) gespeichert. Die Process Optimzation Schicht nutzt diese Erkenntnisse im Rahmen einer Muster-basierten Optimierung. Dabei werden in einem Pattern Catalogue Muster vorgehalten, die in einem zu optimierenden Workflow erkannt werden können.

[1] Mittlerweile ist der Name „Business Process Management System“ (BPMS) gebräuchlicher, als Workflow Management System. Die Funktionalität ist aber im Großen und Ganzen dieselbe.

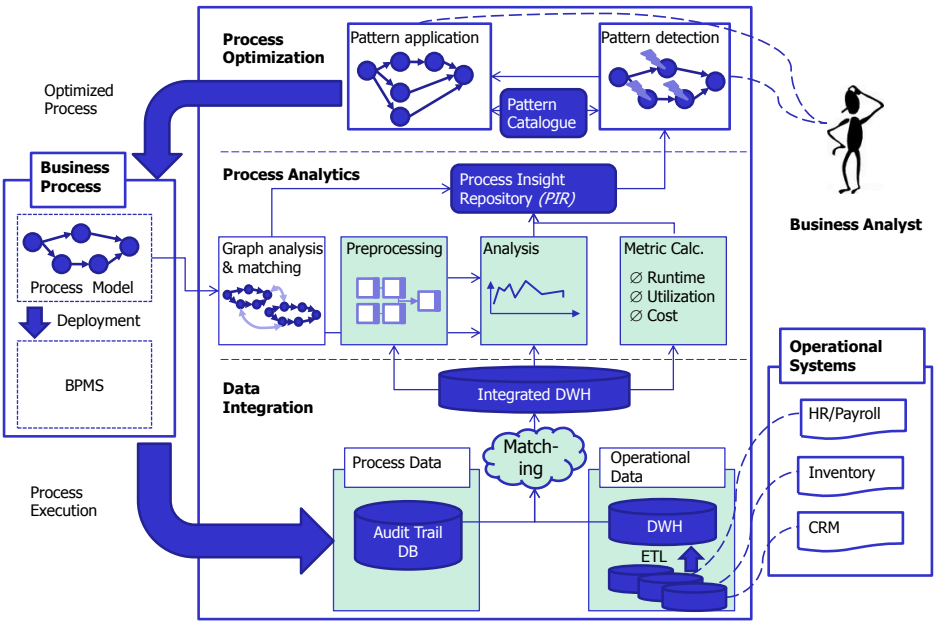


Abbildung 2.1: Die Architektur der deep Business Optimization Platform. Die hellgrün hervorgehobenen Teile wurden in der Diplomarbeit bearbeitet.

2.3 Aufgabenstellung

In dieser Diplomarbeit soll ein Teil der deep Business Optimization Platform implementiert werden.
Dazu ist eine Software zu entwickeln, die Auditdaten über Workflows von einem IBM Websphere Process Server abrufen kann. Da die Auditdaten (auch „Prozessdaten“ genannt) sich auf BPEL, also auf eine ausführungsorientierte Workflow-Notation beziehen, die Workflows aber in BPMN modelliert werden, ist eine entsprechende Transformation auf Basis von Mappings durchzuführen. Die Mappings werden als Ausgabe eines bereits verfügbaren Workflow Modellierungswerkzeugs („dBOP Designer“) in Form einer XML Datei bereitgestellt.
Die Software soll außerdem folgende funktionale Anforderungen erfüllen:

- Für die BPMN Aktivitäten sollen standardisierte Prozessmetriken berechnet werden können. Diese Metriken sollen an einer Schnittstelle zum dBOP Designer verfügbar gemacht werden.
- Für das Matching im Data Integration Layer der dBOP Architektur soll der Anwender die Inhalte und den Umfang der Extraktion der operativen Daten steuern können. Die Auditdaten und operativen Daten sollen durch die Software konsolidiert werden, dafür sollen die Matchings, die mit dem sogenannten BIA Matching Editor [RDoo]

erzeugt werden können, verarbeitet werden. Matchings sind semantische Äquivalenzen zwischen Prozessattributen und operativen Attributen.

- Der Anwender soll die Konsolidierung über ein geeignetes Interface konfigurieren können.
- Die Software soll die Daten für das Data Mining denormalisieren und eine Reihe von Preprocessing-Techniken zur Aufbereitung für das Data Mining anbieten.
- Die Ergebnisse sollen in einer einzelnen Tabelle zur Verfügung gestellt werden und ein entsprechendes Data Mining Verfahren soll prototypisch implementiert werden.

Ergänzend dazu ist ein Szenario zu beschreiben, bei dem der Data Mining Algorithmus die Optimierung sinnvoll unterstützt. Für die Ausgabedateien des Werkzeugs sollen XML Schemata erstellt werden.
Der Bericht soll die angewendeten Verfahren beschreiben, und darüber hinaus verschiedene Preprocessing Techniken aus entsprechender Fachliteratur behandeln.

2.4 Verwandte Arbeiten

In diesem Kapitel sollen Arbeiten vorgestellt werden die ähnlich wie in dieser Diplomarbeit Analyseverfahren auf Workflows und Audit Daten anwenden. Van der Aalst führt in [Aal] den Begriff des „Process Mining“ ein und unterscheidet 5 Typen des Process Minings:

1. Grundlegende Prozessmetriken bestimmen. Wenn das Prozessmodell für einen betrieblichen Ablauf bekannt ist, können solche Metriken bestimmt werden, z.B. mittlere Dauer oder Ausführungshäufigkeit einer einzelnen Aktivität.
2. Ein Prozessmodell erstellen.
3. Modell einer Organisation erstellen. Dabei handelt es sich um Aufgaben, die auf der Grundlage von Audit Daten betrieblicher Systeme, wie z.B. ERP (Enterprise Resource Planning) und SCM (Supply Chain Management) Systemen durchgeführt werden.
4. Ein soziales Netzwerk analysieren (Beziehungen zwischen Sachbearbeitern und zwischen Ressourcen)
5. Leistungseigenschaften analysieren (Regeln ableiten die z.B. besonders gute Leistungen von Ressourcen erklären)

In [ADH+03] beschreiben Van der Aalst et al. „Workflow Mining“ also die Erstellung eines Prozessmodells anhand von Auditdaten betrieblicher Informationssysteme (Vgl. Punkt 2 oben). Das Ziel ist die Gewinnung eines Prozessmodells als Petri Netz unter Verwendung möglichst elementarer Audit Daten wie z.B. Auflistungen davon, welche Aktivitäten nacheinander aktiviert wurden.
Dabei ist den Autoren wichtig auch dann ein korrektes Prozessmodell ermitteln zu können wenn die Auditdaten „verrauscht“ sind, wenn also in einigen Prozessinstanzen Aktivitäten ausnahmsweise nicht zur Ausführung gekommen sind oder z.B. vertauscht wurden.

Workflow Mining kann für eine „Delta Analyse“ wichtig sein: Wenn ein Workflow Designer prüfen möchte, welche Abweichungen es vom vorgegebenen Workflow gegeben hat. Wenn eine Abweichung immer öfter erfolgt, und zur Regel wird, muss möglicherweise der Workflow überarbeitet werden. Die Autoren nennen jedoch als primäres Ziel des Workflow Minings, zu verstehen was in einem Unternehmen tatsächlich abläuft [ADH+03]. Es ist somit kein Tool für den Entwurf oder die Anpassung von Prozessen. Ein wichtiges Teilproblem beim Workflow Mining ist das „Conformance Testing“ [Roz06]. Dabei wird berechnet, wie gut ein potentielles Prozessmodell zu den aufgezeichneten Auditdaten passt. Die Autoren unterscheiden als Maße dafür die „Fitness“ und die „Appropriateness“. Diese Maße erfassen, inwieweit es sich bei einem gegebenen Prozessmodellkandidaten um eine minimale Struktur handelt und inwieweit es dem, in den Auditdaten protokollierten, Verhalten entspricht. Diese Kriterien müssen geprüft werden da es zum einen leicht ist, ein Prozessmodell zu generieren, das jedem beliebigen Aktivitäts-Protokoll entspricht. Andererseits ist es genauso einfach ein Prozessmodell zu erstellen, das nur genau die Prozessabläufe erlaubt, die im Aktivitäts-Protokoll vorkommen. Beide Fälle gilt es zu verhindern, denn solche Prozessmodelle liefern keine über die Auditdaten hinausgehenden Informationen.
Zur Mühlen und Shapiro beschreiben in [MS09] „Business Process Analytics“. Dieser Begriff fasst Methoden und Werkzeuge zusammen, die auf Audit Trails angewandt werden können um Entscheidungen in Unternehmen zu unterstützen [MS09]. Das geschieht z.B. indem die Menschen, die mit dem Prozess zu tun haben mit Informationen zur Effizienz und Effektivität des Prozesses versorgt werden. Es gibt dabei 3 Sparten:

- Process Controlling ist die Analyse von Workflow Audit Trail Daten, die auf bereits abgeschlossenen Prozessinstanzen erfolgt. Das Ziel von Process Controlling ist die Verbesserung zukünftiger Ausführungen eines Workflows. Process Controlling wird in [Mue01] genauer besprochen. In diese Sparte ist auch das Process Mining nach Van der Aalst et al. einzuordnen.
- Business Activity Monitoring [MS09][GCC+04] wertet Prozessinstanzen aus, die sich gerade in der Ausführung befinden. Hier kann auch das Process Monitoring [Mue01] eingeordnet werden. Das ist die Bereitstellung von Informationen über gerade laufende Prozessinstanzen, die dem Workflow Administrator einen Überblick über die laufenden Prozesse ermöglichen oder weitergehende Analysemöglichkeiten eröffnen.
- Process Intelligence umfasst solche Methoden und Werkzeuge, die zur Vorhersage des Verhaltens eines Unternehmens in der Zukunft dienen [MS09][GRC04]. Hier unterscheiden zur Mühlen et al. noch zwischen Simulation, Data Mining und Process Optimization. Simulation kann z.B. Fragen beantworten die Änderungen des Prozessmodells betreffen.

Das in dieser Diplomarbeit entwickelte Werkzeug kann in die Sparte der Process Intelligence eingeordnet werden, da seine Ausgabe verwendet werden kann, um Vorhersagen bezüglich Prozessfluss zu treffen. Um das zu ermöglichen, werden historische Prozessdaten ausgewertet, was eine Einordnung in die Sparte Process Controlling ebenso gut rechtfertigt.

2.5 Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Einleitung: Stellt die Aufgabenstellung vor und ordnet Sie in ihren Kontext, die deep Business Optimization Platform (dBOP), ein. Außerdem werden verwandte Arbeiten vorgestellt.

Kapitel 3 – Grundlagen: Hier werden die Grundlagen besprochen, die zum Verständnis der Arbeit notwendig sind. Dazu gehören Grundbegriffe und relevante Standards wie BPEL und BPMN. Anschließend wird ein Überblick über gängige Preprocessing und Data Mining Verfahren gegeben.

Kapitel 4 – Lösungsansatz: Hier werden die Ideen, Methoden und Konzepte vorgestellt, die bei der Realisierung des Werkzeugs, das Teile der Data Integration und Process Analytics Schicht von dBOP implementiert, eine wichtige Rolle gespielt haben.

Kapitel 5 – Entwurf des Werkzeugs: Hier werden die genauen Anforderungen an das Werkzeug und die Entwurfsphase des Werkzeugs besprochen. Das Kapitel übernimmt die Struktur des Werkzeugs und ist in Abschnitte über GUI, Extraktion, Berechnung der Metriken, Konsolidierung, Preprocessing, Data Mining unterteilt. Der Abschnitt 5.7 beschreibt, wie die gewonnenen Modelle eingesetzt werden können.

Kapitel 6 – Implementierung des Werkzeugs: Hier wird die praktische Umsetzung des Werkzeugs in derselben Struktur wie im Entwurfskapitel besprochen.

Kapitel 7 – Zusammenfassung und Ausblick: Fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

3 Grundlagen

In diesem Kapitel werden die fachlichen und technischen Grundlagen der Diplomarbeit behandelt. Zunächst werden wichtige Grundbegriffe definiert, dann wird die graphische Notation BPMN (siehe Abschnitt 3.3.1) besprochen, die von der deep Business Optimization Platform genutzt wird: Workflows werden in dieser Notation modelliert und analysiert.
Anschließend erfolgt eine Übersetzung in BPEL. BPEL ist eine XML Sprache für ausführbare (und abstrakte) Workflows. BPEL wird in Abschnitt 3.3.2 besprochen.
Das Werkzeug, das in dieser Diplomarbeit entwickelt wurde, setzt nach der Ausführungsphase eines Workflows auf einem IBM Websphere Process Server an. Abschnitt 3.4 stellt die Websphere Produktreihe von IBM und ihren Hintergrund vor.
Das Werkzeug hat das Ziel, eine Analyse durchzuführen und darauf aufbauend eine Optimierung des Workflows zu ermöglichen. Deshalb wird in diesem Kapitel ein Überblick über gängige Data Mining Verfahren und entsprechende Preprocessing Methoden gegeben.

3.1 Workflows und Geschäftsprozesse

An dieser Stelle sollen einige Grundbegriffe auf der Basis von [LR00] eingeführt werden. Ein Geschäftsprozess ist nach Hammer und Champy, die das Business Process Reengineering, einen Vorläufer des Business Process Management geprägt haben, „eine Kollektion von Aktivitäten, die eine oder mehrere Eingaben entgegennimmt und eine Ausgabe erzeugt, die für den Kunden einen Wert hat.“ ([HC93] S.35). Es gibt aber noch weitere davon abweichende Definitionen, für das Verständnis dieser Arbeit genügt diese Definition jedoch.
Ein **Prozess-Modell** ist eine Beschreibung eines Geschäftsprozesses „in der echten Welt“ [LR00], die Instanzen die von diesem Modell erzeugt werden können, sind **Prozesse**. Den Computer-gestützten Teil eines Geschäftsprozesses bezeichnet man als **Workflow-Modell**. Workflows sind Instanzen davon. Um Workflow-Modelle und damit den formalisierbaren Teil von Geschäftsprozessen aufzuschreiben und zu kommunizieren werden sogenannte **Metamodelle** benötigt. Mit BPMN und BPEL werden zwei solche Metamodelle in den Kapiteln 3.3.1 und 3.3.2 beschrieben.
In [LR00] werden 4 Arten von Workflows unterschieden:

1. **Collaborative Workflows**, sind Workflows die zwar für das Unternehmen ein Ergebnis von hohem Wert erzeugen, aber relativ selten angestoßen werden. Oft lassen sie sich nicht in ein festes Modell abbilden. Z.B. ist jeder Auftrag für die Entwicklung von Software wieder anders, auch wenn es dafür sehr wohl Standard Vorgehensmodelle gibt.

2. Auch **Ad Hoc Workflows** werden relativ selten angestoßen, sie erzeugen aber eher einen geringen Wert für das Unternehmen. Hier sind zum Beispiel Revisionsprozesse einzuordnen.
3. **Administrative Workflows** erzeugen kein Ergebnis von hohem Wert, laufen im Unternehmen aber sehr häufig in derselben Weise ab. Zum Beispiel gehören dazu Workflows wie Reisekostenabrechnung oder Einkaufsbewilligung.
4. **Production Workflows** sind schließlich Workflows, die sehr häufig ausgeführt werden müssen und gleichzeitig ein wertvolles Ergebnis für das Unternehmen erbringen, so z.B. die Abwicklung eines Schadensfalls in einer Versicherung.

3.2 Workflow Management Systeme

Um Workflows auszuführen werden Workflow Management Systeme (WFMS) verwendet. Das ist jedoch nur eine der Aufgaben solcher Systeme: Die Workflow Manangement Coalition definiert ein Workflow Management System folgendermaßen:„A system that completely defines, manages and executes „workflows“ through the execution of software whose order of execution is driven by a computer representation of the workflow logic.“ ([Hol95] S.6) Ein WFMS ist also ein System, das Workflows verwaltet und ausführt, und deren Definition ermöglicht. Die Ausführung erfolgt durch Aufruf von Software in der Reihenfolge, die von der Workflow Logik vorgegeben ist.
Während der Ausführung eines Workflows werden dessen Zustandsübergänge im sogenannten Audit Trail protokolliert. Dementsprechend sind die Daten in [WFM99] (S.51) definiert als: „Audit Data: A historical record of the progress of a process instance from start to completion or termination. Such data normally incorporates information on the state transitions of the process instance.“. Oft werden im Audit Trail auch die Eingabe- und Ausgabedaten einzelner Aktivitäten der Workflows protokolliert.

3.3 Modellierung von Prozessen und Workflows

Die Modellierung von Prozessen erfolgt in der Regel durch Personal das sich im Anwendungsgebiet der Prozesse auskennt. Dabei kommen Flussdiagramme [BPM06] oder ähnliche graphische Notationen zu Einsatz, die anschließend in ein Ausführungsformat überführt werden müssen, um eine Ausführung auf einem Workflow Management System möglich zu machen. Durch diese Überführung entsteht aus dem ursprünglichen Prozess ein Workflow. Als Ausführungsformat kommt zum Beispiel BPEL zum Einsatz. BPEL ist ein XML Workflow-Format, das für den Betrieb und die Zusammenarbeit von Workflow Management Systemen optimiert ist [BPM06]. Es eignet sich daher nur bedingt für den Entwurf, die Verwaltung und die Überwachung von Prozessen bzw. Workflows durch Personal.
Der Umstand, dass bisherige graphische Notationen sich nicht für die Ausführung eignen, wird in [BPM06] als technologische Lücke zwischen Entwurfsformat und Ausführungssprache bezeichnet. Deshalb veröffentlichte die Object Management Group den BPMN Standard

[BPM06]. In der Business Process Modelling Notation können Prozesse graphisch modelliert und verwaltet werden. Die Autoren von [BPM06] streben mit BPMN die Entwicklung einer gemeinsamen Sprache für betriebswirtschaftlich ausgebildetes Fachpersonal und IT-Experten an. Die Prozessnotation soll gut lesbar, flexibel und erweiterbar sein und sich auf ein Ausführungsformat wie BPEL abbilden lassen. In Abschnitt 5.2.1 wird der Ansatz der deep Business Optimzation Platform zur Abbildung von BPMN auf BPEL vorgestellt.

3.3.1 BPMN 1.1

Die Business Process Modelling Notation (BPMN) wurde ursprünglich von der Business Process Management Initiative (BPMI) vorgeschlagen. Die BPMN eignet sich zur Konzeption und Kommunikation von Prozessen. BPMN setzt sich in der Wirtschaft mehr und mehr durch, so gab es 2008 schon über 50 Tools zur Modellierung von Geschäftsprozessen mit BPMN [All08]. Die BPMI ging im Juni 2005 in der Object Management Group (OMG) auf, die derzeit an der Version 2.0 des Standards arbeitet. Ab Version 2.0 steht das Kürzel BPMN für Business Process Model and Notation.

Die graphischen Objekte von BPMN 1.1

In diesem Abschnitt sollen die wichtigsten graphischen Objekte der Notation vorgestellt werden. Die Darstellung orientiert sich dabei an [BPM06] S.17f. Die graphischen Objekte von BPMN lassen sich in 4 Kategorien unterteilen:

1. Flow Objects
2. Connecting Objects
3. Swimlanes
4. Artifacts

Flow Objects können Ereignisse (engl. Events), Aktivitäten (engl. Activities) oder Gateways sein. Sie werden als (zum Teil beschriftete) Kreise, Rechtecke bzw. Rauten dargestellt, und können mit Hilfe der Connecting Objects verknüpft werden. Das kann auf 3 Arten geschehen. Entweder kann ein Kontrollfluss, ein Nachrichtenfluss oder eine Assoziation zwischen 2 Flow Objects deklariert werden. Die Grundform für die Connecting Objects ist ein Pfeil.
Mit Pools können die Flow Objects gruppiert werden, so können die Prozesse verschiedener Organisationen klar voneinander getrennt werden. In jedem Pool kann es mehrere Swimlanes geben, dadurch können Prozesse verschiedener Organisationseinheiten voneinander getrennt dargestellt werden.
Mit den Artifacts sieht der BPMN Standard eine Möglichkeit vor, Prozessmodelle mit weiterer Information zum besseren Verständnis anzureichern. Es gibt 3 Standard Artefakte: Datenobjekte können z.B. die Übergabe eines Dokuments, etwa einer Rechnung, von einer Aktivität zur nächsten darstellen. Gruppen können verwendet werden, um Fluss Objekte einer beliebigen Kategorie graphisch zusammenzufassen. Annotationen sind erläuternde

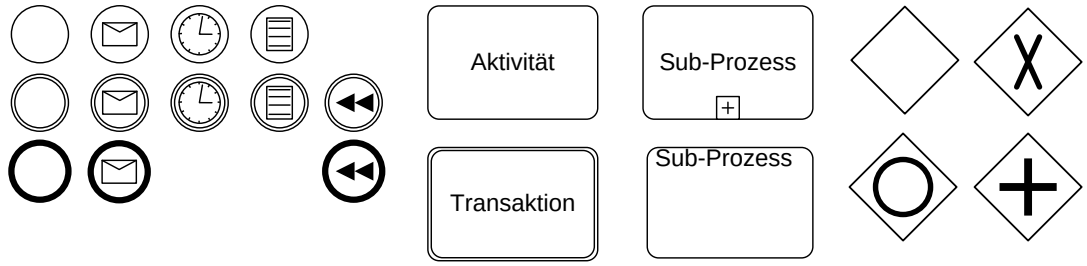


Tabelle 3.1: Beispiele für Flow Objects in BPMN. Links: Ereignisse. Mitte: Aktivitäten. Rechts: Gateways.

Anmerkungen in Textform.
In Tabelle 3.1 sind Beispiele für Flow Objects zu sehen. Die kreisförmigen Symbole sind **Events**, von denen es 3 Kategorien gibt: Start-, Intermediate- und End- Events. Start-Events haben eine einfache Umrandung, sie können verwendet werden um darzustellen, dass ein neuer Prozess beginnt. Das kann z.B. als Reaktion auf eine Nachricht (Briefsymbol) oder zu festgelegtem Zeitpunkt (Uhrensymbol) erfolgen. Intermediate Events haben eine doppelte Umrandung und können in einen Prozess integriert werden, um z.B. darzustellen, dass mit einem Produktionsschritt gewartet werden muss, bis ein benötigtes Bauteil (Nachricht) eingetroffen ist. End Events (Kreise mit dickem Rand) stellen dar, dass bei einem bestimmten Ereignis der Prozess terminiert. So zum Beispiel wenn ein Ausnahmefehler eintritt. Dann besagt der doppelter Rückpfeil, dass alle Änderungen an Daten, die im aktuellen Kompensationsbereich („Compensation Sphere“) gemacht wurden, rückgängig zu machen sind.
Die rechteckigen Kästen in Tabelle 3.1 sind **Aktivitäten**. Aktivitäten können Sub-Prozesse sein, die selbst wieder einen Prozess enthalten können. Ist der innere Prozess nicht dargestellt, so zeichnet man dafür ein Plus. Einfach umrandete Aktivitäten ohne Plus oder inneren Prozessdarstellung sind atomar. Doppelt umrandete Aktivitäten sind Transaktionen.
Rechts in Tabelle 3.1 sind die **Gateways** zu sehen, die verwendet werden um den Kontrollfluss des modellierten Prozess vorzugeben. Der Kontrollfluss kann wie bei Petri Netzen mit Hilfe von Tokens simuliert werden. Ein Token ist wie eine Spielfigur die über das BPMN Prozessmodell verschoben wird, je nachdem welche Aktivität gerade ausgeführt wird. Gateways bestimmen den Weg der Tokens. Gateways können die Tokens, die das Prozessmodell durchlaufen, auf 4 Arten beeinflussen:

- Branching: Auf Basis der Bedingungen die an den n ausgehenden Sequenzpfeilen annotiert sind, werden für die m ($m \leq n$) Sequenzpfeile neue Token erzeugt, deren Bedingung zu true auswertet. Das kann mit dem leeren oder dem „x“-Gateway ausgedrückt werden.
- Forking: An allen ausgehenden Sequenzpfeilen wird ein Token erzeugt. Das bedeutet, es erfolgt eine parallele Ausführung der nachfolgenden Aktivitäten. Das kann mit dem „+“-Gateway ausgedrückt werden.
- Joining: Wenn an allen Eingängen ein Token anliegt wird am Ausgang ein Token erzeugt. Das wird ebenfalls mit dem „+“-Gateway dargestellt.

- Merging: Merging kann durch exklusive oder inklusive Gateways erfolgen. Diese Gateways erzeugen dann ein Token am Ausgang, wenn alle Token des entsprechenden Branchings eingetroffen sind.

Beispiel

Abbildung 4.3 zeigt einen Beispiel Prozess in BPMN. Es handelt sich dabei um ein vereinfachtes Modell der Abläufe in einem Versicherungs-Unternehmen, die ausgeführt werden müssen, wenn ein Versicherter eine Versicherungsleistung anfordert. Der Prozess beginnt immer bei einem Kreis mit dünner Umrandung, im Beispiel also ganz links. Anschließend werden die Daten des Versicherungsfalls entgegengenommen. Dann wird der Versicherungsfall in eine von 3 Risikoklassen bezüglich Betrugswahrscheinlichkeit eingeordnet. Solche Aktivitäten werden bei BPMN als abgerundete Rechtecke dargestellt und durch einfache Pfeile miteinander verbunden, um eine sequentielle Ausführung darzustellen. Wenn der Kontrollfluss in Abhängigkeit des Ergebnisses einer Aktivität unterschiedlich verlaufen soll, kann man z.B. ein rautenförmiges XOR Gateway-Symbol zeichnen. Im Beispiel entspricht es der Entscheidung, wie mit dem Versicherungsfall weiter verfahren wird. Bei einem hohen Betrugsrisiko erfolgt eine eingehende Prüfung, bei einem niederigen Risiko entfällt die Prüfung und es erfolgt eine Leistung an den Versicherten. Der Prozess terminiert bei einem dick umrandeten Kreis.

3.3.2 WS-BPEL 2.0

Die Business Process Execution Language (WS-BPEL) ist eine XML Sprache zur Definition abstrakter und ausführbarer Workflows, die 2007 von der Organization for the Advancement of Structured Information Standards (OASIS) als Standard verabschiedet wurde. An dem Standard haben z.B. Unternehmen wie IBM, Microsoft und SAP mitgewirkt. Abstrakte Workflows sind Workflows, die nur die öffentlichen Aspekte eines Unternehmens- Protokolls („Business Protocol“) definieren [ACD03]. So gibt es viele Szenarien, bei denen Unternehmen mit Kunden-(Unternehmen) oder Partnerunternehmen online interagieren, dabei jedoch ein Austausch aller Details der internen Prozessabläufe der beteiligten Unternehmen nicht sinnvoll und womöglich nicht erwünscht ist. Für einen solches Szenario bietet WS-BPEL die Möglichkeit, die Prozessinformation, die ausgetauscht wird, beliebig abstrakt zu halten. Abstrakte Workflows haben somit in erster Linie das Ziel, einen Workflow grob zu beschreiben. Darüber hinaus können mit WS-BPEL ausführbare Workflows definiert werden, Sie „modellieren das tatsächliche Verhalten eines Teilnehmers in einer Unternehmens-Interaktion“ [WSB]. WS-BPEL Workflows nutzen in ihrer Prozess-Logik Webservices um Aktionen auszuführen. Webservices sind Programme [HB04], die über Netzwerke flexibel interagieren können. Webservices zeichnen sich außerdem dadurch aus, dass ihre Schnittstelle maschinenlesbar in Form eines WSDL Dokuments vorliegt.

Geschichte von WS-BPEL

WS-BPEL setzt die von DeRemer und Kron 1975 [DK75] beschriebene Idee einer „module interconnection language“ um. Das ist eine Sprache die sich für das „Programming in the large“ eignet, also für das Zusammenfügen einfacher Module zu einem komplexeren Gesamtsystem. Dabei entstehen die Module im Rahmen des „Programming in the small“. Die Autoren unterstreichen die Notwendigkeit passender Sprachen für beide Paradigmen. WS-BPEL reiht sich ein in die Kette der Sprachen, die für „Programming in the large “ entworfen wurden, es ist der Nachfolger der der Sprachen BPEL4WS 1.0 und BPEL4WS 1.1, wobei letztere bereits als de-facto Standard galt. In BPEL4WS wurden die Sprachen XLANG von Microsoft und WSFL von IBM zusammengeführt. Während die WSFL auf einem graphentheoretischen Ansatz basiert, handelt es sich bei der XLANG um eine Sprache die ihre formalen Grundlagen im sogenannten Pi-Kalkül hat.

Sprachelemente von WS-BPEL

Hier werden die Sprachelemente von WS-BPEL vorgestellt. Die Informationen wurden der Spezifikation entnommen [ACD03]. Ein typischer Workflow, der in BPEL definiert ist, umfasst folgende Bereiche: Die Partner Links, Correlation Sets, Variablendeklarationen, Fault Handler und den normalen Kontrollfluss („normal behavior“ S.19 in [ACD03]).

Partner Links In diesem Bereich werden alle möglichen Interaktionspartner des Workflows deklariert. Die Interaktionspartner werden dabei nicht statisch, etwa mittels einer URL gesetzt, sondern es wird lediglich angegeben, von welcher Art die Interaktion ist und welche Rolle der Workflow dabei hat, und welche Rolle der Interaktionspartner hat. Dadurch wird dynamisches Binden von Diensten möglich.

Listing 3.1 BPEL Code zur Deklaration von partnerLinks.

```
<partnerLinks>
<partnerLink name="" partnerLinkType="" myRole="" partnerRole="" />
</partnerLinks>
```

Message Exchanges Message Exchanges können definiert werden, um Ambiguitäten in der Zuordnung zwischen Receive und Reply (siehe Paragraph über Kontrollfluss) Aktivitäten aufzulösen. Diese entstehen wenn zwei Receive-Reply Paare denselben Partner Link nutzen und auch die WSDL Operation, der sie zugeordnet sind, dieselbe ist.

Correlation Sets Correlation Sets werden in BPEL verwendet, um Nachrichten Prozessinstanzen zuordnen zu können. Eine Correlation Set hat einen definierten Namen („name“) und eine definierte Menge von Attributen, die für die eindeutige Zuordnung verwendet werden („properties“). Bei einem Online Bestell-Prozess könnte ein solches Attribut z.B. eine Bestellnummer sein, mit der auch der Kunde selbst sich auf seine Bestellung beziehen kann, etwa zwecks Reklamation. Es gibt auch Szenarien, in denen mehrere Correlation Sets für eine Nachricht Sinn ergeben ([WSB] S.75). In allen Nachrichten-Aktivitäten, also z.B. send

Listing 3.2 BPEL Code zur Deklaration von correlationSets.

```
<correlationSets>?
<correlationSet name="CorrelationSetName1" properties="Attribut1" />+
</correlationSets>
```

und receive, können die Correlation Sets genutzt werden (siehe Listing 3.3). Dabei gibt

Listing 3.3 BPEL Code zur Deklaration von correlations.

```
<correlations>
<correlation set="CorrelationSetName1" initiate="yes"/>
</correlations>
```

der boolesche initiate-Wert an, ob die umschließende receive Aktivität auf Empfang einer Nachricht eine neue Instanz des Prozesses erzeugen soll, die dann die ID erhält, die aus den Attributwerten des Correlation Sets besteht.

Variablen In diesem Bereich können Variablen definiert werden die es erlauben, über den Verlauf der Workflow Ausführung hinweg, beliebige Informationen in Form von XML Schema Datentypen und Elementen oder WSDL Nachrichten-Typen vorzuhalten.

Listing 3.4 BPEL Code zur Deklaration von Variablen.

```
<variables>
<variable name="" messageType="" />
</variables>
```

Fault Handler In diesem Bereich wird definiert, was bei Fehlern in dem Workflow oder bei Fehlern in Diensten, die der Workflow aufruft, zu tun ist. So wird hier zum Beispiel auch jeder WSDL-Fehler behandelt, der in der Schnittstellen-Definition eines aufzurufenden Webservice deklariert ist.

Kontrollfluss Der normale Kontrollfluss eines Prozesses kann durch Verkettung und Verschachtelung von WS-BPEL Aktivitäten definiert werden.
Mit der Aktivität <receive> kann auf Aufrufe des Prozesses gewartet werden. Erfolgt ein Aufruf z.B. durch einen anderen Webservice so wird eine neue Instanz des Prozess erstellt.

Listing 3.5 BPEL Code zur Deklaration von faultHandlern.

```
<faultHandlers>
<catch faultName="" faultVariable="" faultMessageType="">
...Aktivität (z.B. reply)...
</catch>
</faultHandlers>
```

Anschließend wird begonnen, die nachfolgende Aktivitäten auszuführen.
Um den Kontrollfluss zu programmieren stehen dabei nicht nur die üblichen Konstrukte wie <if>, <while>, <repeatUntil>, <forEach>, <throw> zur Verfügung, sondern es kann auch gewählt werden, ob Aktivitäten sequentiell (<sequence>) oder parallel (<flow>) ausführbar sind. Außerdem besteht die Möglichkeit, Aktivitäten auf Eintreten eines Ereignisses hin auszuführen:<pick> wartet auf mehrere Ereignisse. Wenn eines davon eintritt, wird die entsprechende Aktivität ausgeführt und keine weiteren Ereignisse mehr entgegengenommen.
Mit <invoke> kann auf einfache Weise ein externer Webservice aufgerufen werden.
Mit <assign> können Werte zwischen Variablen ausgetauscht werden, z.B. um die Eingabedaten eines aufzurufenden Webservice festzulegen. Mit <validate> können Inhalte von Variablen auf Übereinstimmung mit einer XML Schema oder WSDL Datendefinition geprüft werden. Der <wait>-Befehl drückt aus, dass entweder eine bestimmte Zeit gewartet werden soll bis die innere Aktivität ausgeführt wird, oder dass diese zu einem bestimmten Zeitpunkt auszuführen ist.
<empty> dient als Platzhalter für Stellen im Code, wo zwar eine Aktivität stehen muss, aber nichts ausgeführt werden soll, so zum Beispiel bei einem Fehler der zwar abgefangen wird, aber keine Fehlerbehandlung erfordert [ACD03].
Mit <scope> kann ein vom Hauptprogramm getrennter Gültigkeitsbereich für Variablen, partnerLinks usw. definiert werden. Mit <compensateScope> können die Aktionen, die in einem inneren Gültigkeitsbereich bereits (erfolgreich) stattgefunden haben, wieder rückgängig gemacht werden. <compensate> macht dasselbe für alle inneren Scopes.
<extensionActivity>s können verwendet werden um einen neue Aktivitätstypen, die der WS-BPEL Namespace nicht enthält, anzusprechen. In einem Fault Handler besteht außerdem die Möglichkeit mit <rethrow> den Fehler erneut zu werfen, sodass ein weiterer Fault Handler darauf reagieren kann.
Der Kontrollfluss des Prozess kann mit <exit> beendet werden, mit <reply> können die Ergebnisse an den aufrufenden Prozess zurückgegeben werden.

3.4 IBM WebSphere Process Server

In diesem Kapitel wird der IBM WebSphere Process Server vorgestellt und es wird auf verwandte Produkte eingegangen. Es ist eine Zusammenfassung der, für die vorliegende Diplomarbeit relevanten Informationen aus [PGG05].
In der Diplomarbeit wurde der Websphere Process Server als Workflow Management System genutzt: Das Werkzeug ruft am Anfang Daten des Audit Trails von einem WebSphere Process Server ab.

Der Grund für die Entwicklung des WebSphere Process Servers und verwandter Produkte war, dass sich viele Unternehmen mit einem immer stärkeren Wandel ihrer Umgebung konfrontiert sahen. Bei IBM entwickelte sich deshalb die Idee des On Demand Business. Ein Unternehmen das diese Idee umsetzt, zeichnet sich nach [PGG05] durch folgende Eigenschaften aus:

- Es ist auf seine Kernkompetenzen fokussiert. Für alle anderen Aufgaben gibt es Partnerschaften mit Fremdunternehmen. Es kann schnell auf veränderte „Kundenwünsche, Marktbedingungen und externe Bedrohungen“ [PGG05] reagieren.
- Es ist außerdem flexibel was seine Prozesse betrifft.
- Darüber hinaus ist es robust gegenüber Änderungen seines wirtschaftlichen und technischen Umfelds.

Um ein On Demand Business zu realisieren ist es laut [PGG05] z.B. wichtig, offene Standards einzuführen („Open Standards“), Systeme zu Integrieren („Integration“) und bestimmte Abläufe zu automatisieren („Automation“) [PGG05].
Als passende Architektur für ein On Demand Business propagiert IBM die Service-orientierten Architekturen (SOA).
Den Service orientierten Architekturen liegen die Ideen Objekt-orientierten Designs, des Komponenten-basierten Designs und der Enterprise Application Integration zugrunde.
SOAs nutzen Konzepte wie lose Kopplung und Kapselung um eine hohe Flexibilität bei der Integration auf Unternehmensebene zu erreichen.
Die zentrale Rolle spielt dabei der „Dienst“, der eine wiederverwendbare Unternehmensfunktion realisiert, und durch eine explizite Schnittstelle definiert ist [Mel10]. Es gibt Dienstanbieter, Dienstnutzer und Dienstverzeichnisse. Dienste können andere Dienste in den Dienstverzeichnissen nachschlagen, sie dynamisch binden und nutzen.
Damit möglichst alle bestehende Funktionalität in einem Unternehmen weitergenutzt werden kann, wurde das Konzept des Enterprise Service Bus entwickelt, der eine Vermittlerrolle zwischen heterogenen Diensten einnimmt, und mit Message orientierter Middleware verglichen werden kann.
Die WebSphere Produkte von IBM ermöglichen den Aufbau einer SOA in Unternehmen. Um das Dienstkonzept von SOAs zu realisieren, bietet der WebSphere Process Server die Webservice Technik an (siehe Abschnitt 3.3.2). Webservices bauen auf den Standards WSDL und SOAP und UDDI auf. SOAP ermöglicht den Aufruf von Webservices über HTTP. UDDI spezifiziert die technischen Details einer Registry in der Dienstanbieter ihre Webservices bereitstellen können, und Dienstnutzer diese finden können.
Der Process Server unterstützt zusammen mit den anderen WebSphere Produkten den gesamten Lebenszyklus von Applikationen. Mit Hilfe des WebSphere Business Modellers können Workflows modelliert werden. Mit dem Rational Application Developer können einzelne Dienste z.B. in Form von Webservices realisiert werden. Diese können im Rahmen von BPEL Workflows mit dem WebSphere Integration Developer orchestriert werden, die dann auf einem WebSphere Process Server ausgeführt werden können. Der WebSphere Process Server unterstützt neben BPEL Workflows auch Human Tasks, Business State Machines und Business Rules.
Human Tasks sind Komponenten, die verwendet werden können um z.B. Sachbearbeitern

Aufgaben zuzuweisen [PGG05]. Business State Machines erlauben die Implementierung von Geschäftsprozessen auf Basis von Zuständen und Ereignissen. Business Rules sind Regeln der Form IF ... THEN ... . Sie können in natürlicher Sprache formuliert werden und erlauben so einen schnellen Zugriff auf die Workflow-Logik eines Unternehmens, auch für Personal ohne IT Hintergrund. Zum Beispiel könnte ein solche Regel sein, dass bei einem Autovermieter bei besonders großer Nachfrage die Rabatte angepasst werden sollen [IBM]. Ausgeführte Workflow-Instanzen können mit dem WebSphere BusinessMonitor überwacht werden.

3.5 Data Mining

In diesem Abschnitt soll geklärt werden, was unter Data Mining verstanden wird. Data Mining ist der Prozess der „Extraktion von interessantem Wissen aus großen Datenbeständen“ [HK06]. Bei solchen Datenbeständen kann es sich um Datenbanken, Data Warehouses oder z.B. einfache Dateien handeln. Ein häufig verwendetes Synonym ist Knowledge Discovery in Databases (KDD). Es gibt aber auch eine Reihe sinnverwandter Begriffe, wie z.B. Knowledge Mining from Databases.
Data Mining im weiteren Sinne umfasst eine Reihe von Aktivitäten an dessen Ende der Benutzer Informationen erhält, die für seine Aufgabe(n) von Nutzen sind.
Han und Kamber grenzen Data Mining Systeme von ähnlichen Systemen ab, die für große Datenbestände nicht ausgelegt sind, wie. z.B. Machine Learning Software oder statistische Datenanalyse-Tools. Online Analytical Processing (OLAP) ist auch davon zu unterscheiden. Das sind „Technologien und Werkzeuge die eine (ad-hoc) Analyse multidimensional aggregierter Daten ermöglichen“ [Mit10]. Eine typische OLAP Operation ist z.B. Roll up zum Zusammenfassen durch Aufstieg in einer Hierarchie (z.B. von Ort zu Bundesland, oder von Tag zu Monat) oder durch Reduzierung der Dimensionen. Wenn z.B. die Einnahmen einer Kaufhauskette tagesweise für verschiedene Standorte vorliegen, könnte man die Zeitdimension durch die Summe über alle Tage ersetzen und hätte somit die Gesamteinnahmen pro Standort.
Ein wichtiger Unterschied zwischen OLAP und Data Mining ist, dass OLAP im Gegensatz zu Data Mining keine Modelle über Datenbestände generieren kann. OLAP kann jedoch dazu benutzt werden um bestehende Modelle oder Annahmen zu verfizieren. In der Fachliteratur gibt es verschiedene Herangehensweisen, den Data Mining Prozess zu beschreiben. Petersohn [Pet05] bezieht in den Prozess den Anwender mit ein, der zunächst das Ziel des Prozesses definiert, und am Ende die Ergebnisse interpretiert.
Han et al. beschreiben das was dazwischen abläuft. Zunächst erfolgt ein Preprocessing wie es in Kapitel 3.6 vorgestellt wird, dazu gehört insbesondere das Zusammenführen von relevanten Daten (Data Integration). Danach werden im zentralen Data Mining-Schritt durch intelligente Methoden relevante Muster gesucht, die dannach durch Relevanzmaße als interessant oder nicht interessant eingestuft werden. In der Regel wird in diesem Schritt ein Data Mining Modell der Trainingsdaten erstellt. Am Ende werden die Ergebnisse visualisiert. In [Pet05] gehört darüber hinaus der Schritt der Evaluierung des Data Mining Modells und

dessen Anwendung zum Gesamtprozess.

3.5.1 Überblick über gängige Data Mining Verfahren

Im zentralen Data Mining Schritt kommen je nach Zielsetzung verschiedene Verfahrensklassen zum Einsatz, die hier auf Basis von [HK06] und [Pet05] vorgestellt werden. Man unterscheidet dabei zwischen „supervised“ und „unsupervised“ Verfahren [HK06]. Zu den „supervised“ Verfahren gehört die Klassifizierung und die Regression. Mit solchen Verfahren können anhand von Trainingsbeispielen Modelle erstellt werden, die zur Vorhersage eines, für die Trainingsbeispiele bekannten, Attributes benutzt werden können. Daneben gibt es Verfahren bei denen diese Attribute nicht vor der Modellerstellung bekannt sein müssen, dazu zählt in erster Linie die Klassenbildung (Clustering). Daneben gibt es z.B. die Assoziationsanalyse, die im folgenden Abschnitt vorgestellt wird. Auf „Outlier-Detection“ wird im Preprocessing Kapitel kurz eingegangen (siehe Abschnitt 3.6.1) . Eine genaue Besprechung von „Outlier-Detection“ und weiterer Verfahrensklassen wie z.B. „Evolution Analysis“ geht über den Fokus dieser Arbeit hinaus und kann in [HK06] nachgelesen werden.

Assoziationsanalyse

Assoziationsanalyse-Verfahren kommen in erster Linie in Anwendungsgebieten zum Einsatz, bei denen Zusammenhänge auf Basis von nominalen bzw. diskreten Quelldaten ermittelt werden sollen. Das bekannteste Beispiel dafür ist die Warenkorbanalyse. Dabei stellt sich das Problem, aus einer großen Anzahl von protokollierten Verkaufsvorgängen („Transaktionen“) Informationen z.B. für den Katalog Entwurf zu gewinnen [AS98]. Durch die Assoziationsanalyse-Verfahren können Aussagen gemacht werden wie, „wenn ein Kunde Produkt X und Y kauft, dann kauft er mit hoher Wahrscheinlichkeit auch Produkt Z“.
Assoziationsregeln haben allgemein die Form $A \rightarrow C$. Dabei steht das A für Antecedent und das C für Consequent. Bei Antecedent und Consequent handelt es sich formal um Itemsets, das sind Mengen von sogenannten Items. Items ihrerseits sind Literale [AS98], die im Fall der Warenkorbanalyse einzelne Produkte repräsentieren.
Die Itemsets sind im Beispiel also Mengen von gekauften Produkten. Das Beispiel lässt sich als Assoziationsregel $XY \rightarrow Z$ notieren.
Um dem Anwender eine Einschätzung der Bedeutsamkeit einer konkreten Assoziationsregel zu ermöglichen, gibt ein Algorithmus für Assoziationsregeln zu jeder gefundenen Regel einen Support und einen Confidence-Wert aus. Der Support ist ein Maß dafür, wie hoch der Anteil der Transaktionen, die sowohl die Items des Antecedent als auch die des Consequent enthalten, an der Gesamtmenge aller Transaktionen ist. Die Angabe des Support kann wahlweise auch in absoluter Form erfolgen. Die Confidence gibt dagegen an, wie hoch die Wahrscheinlichkeit ist, dass eine Transaktion das Consequent enthält, sofern es das Antezedent enthält. Die Algorithmen zum Auffinden von Assoziationsregeln erlauben zur Einschränkung der Ergebnismenge dem Anwender die Angabe, wie hoch Support und Confidence mindestens sein müssen, damit eine Regel ausgegeben wird.

Mit dem AIS Algorithmus wurde im Jahr 1993 von Agrawal et al. erstmals ein Verfahren zur Assoziationsanalyse veröffentlicht [AIS93]. Der AIS Algorithmus ist ein Algorithmus für boolesche Assoziationsregeln, wie es im Warenkorb-Beispiel erklärt wurde. Der Algorithmus findet nur solche Regeln deren Consequent aus einem einzigen Item besteht [AS98]. Der bekannteste Vertreter der Assoziationsregel-Algorithmen ist der Apriori Algorithmus [AS98]. Dieser Algorithmus hebt die Einschränkung des AIS bezüglich des Consequent auf.
Seit 1993 wurden viele weitere Assoziationsanalyse-Algorithmen veröffentlicht, darunter auch ein Algorithmus für die Ermittlung quantitativer Assoziationsregeln [SA96]. Das sind Assoziationsregeln deren Antecedent und Consequent Terme der Form $X \leq Attribut \leq Y$ enthalten können. Dadurch wird es möglich, Aussagen zu treffen wie „10% aller Verheirateten im Alter von 50 bis 60 haben mindestens 2 Autos“ [SA96]. Boolesche Assoziationsregeln sind nicht mächtig genug um solche Aussagen zu treffen, es sei denn die quantitativen Attribute des Antecedent und des Consequent werden auf die richtigen Intervalle abgebildet und anschließend als Items interpretiert.

Klassifizierung

Klassifizierung ist die Zuweisung einer Klasse zu einem gegebenen Objekt auf Basis eines Analysemodells das aus Beispielen erlernt wurde. Wichtige Vertreter der Klassifizierungsverfahren sind der Naive Baysesian Classifier, Bayessian Belief Networks, Neuronale Netze und Entscheidungsbauminduktion. Diese Verfahren werden auf Basis von einzelnen Relationen mit Trainingsbeispielen trainiert.
Die Naive Bayessian Classifier basieren auf dem Bayes'schen Theorem. Das Verfahren arbeitet unter der Annahme, dass die Eingabe-Attribute des Classifiers statistisch unabhängig sind. Es kann auf effiziente Art und Weise die Klasse eines Objekts mit maximaler a posteriori Wahrscheinlichkeit bezüglich der Eingabe-Attribute zu bestimmen. Aufgrund ihrer Effizienz eignen sich solche Classifier auch dann, wenn sehr viele Attribute vorliegen.
Bayessian Belief Networks können im Gegensatz zu den Naive Bayessian Classifiern, Abhängigkeiten zwischen Attributen berücksichtigen.
Neuronale Netzwerke können ebenfalls zur Klassifizierung verwendet werden. Das sind gerichtete azyklische Graphen, deren Knoten in Input Layer, Hidden Layer und Output Layer eingeteilt sind. Das Training besteht in der Anpassung von Kantengewichten durch ein Gradientenabstiegsverfahren („Backpropagation“). Neuronale Netzwerke kommen gut mit Objekten zurecht, die nicht in der Trainingsmenge enthalten waren und sind tolerant gegenüber verrauschten Daten. Allerdings haben sie eine relativ lange Trainingsdauer und sind als Modelle nur schwer interpretierbar, jedoch gibt es Verfahren um aussagekräftige Regeln aus trainierten Neuronalen Netzwerken zu gewinnen.
Entscheidungsbauminduktion gilt als eines der einfachsten Lernverfahren und ist leicht zu implementieren [RN04]. Ein Entscheidungsbaum stellt dabei eine einfache hierarchische Anordnung von inneren Knoten und Blättern dar. Die inneren Knoten stehen für Attribute, die Kanten zu deren Kindern stehen für Werte oder Wertbereiche dieser Attribute. Ein Knoten mit nominalem Attribut A zusammen mit einer Kante zu einem Kindknoten mit Wert W bildet eine Bedingung $(A = W)$. Die Blätter enthalten Klassenzuweisungen.
Um mit einem solchen Analysemodell Objekte zu klassifizieren, wird der Baum so bis zu

einem Blatt durchlaufen, dass die Bedingungen entlang des Pfades genau auf das Objekt zutreffen. Die Klassenzuweisung erfolgt dann durch Auslesen des Inhalts des Blattknotens. Die gängigen Entscheidungsbaumalgorithmen versuchen einen möglichst „kleinen“ Entscheidungsbaum herzuleiten, da ein großer Entscheidungsbaum zur Klassifizierung nicht in der Trainingsmenge enthaltener Objekte wenig nützt. Üblich ist dabei die Anwendung einer Greedy Strategie: Dabei wird der Baum von der Wurzel ausgehend aufgebaut. Das Attribut des aktuellen Knotens wird mit Hilfe einer Metrik ausgewählt, die über alle zur Auswahl stehenden Attribute entweder zu maximieren oder zu minimieren ist. Übliche Metriken sind Entropie (ID3 und C4.5 Algorithmus - mehr dazu in Abschnitt 5.6.2), Gini Index (CART Algorithmus) und Korrelation (ChAID Algorithmus) [Mit10].

Regression

Die im letzten Abschnitt vorgestellten Entscheidungsbäume haben das Defizit, sich nur bedingt für stetige Attribute zu eignen. D.h. dann, wenn der Wert eines stetigen Attributes vorhergesagt werden soll, haben diese Verfahren Probleme. Es wurde zwar versucht das Problem der Vorhersage stetiger Attribute mit Hilfe von Standard-Klassifizierungsverfahren zu lösen, indem die numerischen Attributwerte der Beispiele der Trainingsmenge auf Intervalle abgebildet wurden, jedoch scheitert ein solcher Ansatz oft deshalb, [Qui92] weil Entscheidungsbaumverfahren nicht die implizite Ordnung der Intervalle berücksichtigen.
Der erste Entscheidungsbaum-Algorithmus, der zur Vorhersage stetiger Attributwerte geeignet war und darüber hinaus auch stetige Attribute von Objekten zur Vorhersage nutzen konnte war der M5 Algorithmus. Die Vorhersage stetiger Attributwerte wird dabei dadurch erzielt, dass die Blattknoten nicht mehr eine einfache Klassifizierung enthalten, sondern lineare Regressions-Modelle. Das sind Funktionen, die mit der Fehlerquadratmethode [BSMM01] an die Trainingsbeispiele angepasst werden.
Der M5 wird in [Qui92] vorgestellt und genauer in [Pet05] beschrieben. In Abschnitt 5.6.2 wird eine Variante davon besprochen, die in dieser Diplomarbeit zum Einsatz kam. Weitere Regressionsverfahren sind z.B. in [Pet05] im Kapitel Zeitreihenanalyse zu finden.

Klassenbildung

Klassenbildung ist in der englischsprachigen Fachliteratur als Clustering bekannt und umfasst alle Algorithmen, die auf eine nicht überwachte Art und Weise Objekte einer gegebenen Objektmenge, auf Basis ihrer Attribute, in vorher nicht näher beschriebene Klassen einordnet. Dabei werden immer solche Objekte ein und derselben Klasse zugeordnet, die sich möglichst „ähnlich“ sind, während je zwei Objekte, die aus unterschiedlichen Klassen stammen, möglichst „verschieden“ sind. Dabei gibt es verschiedene Ähnlichkeitsmaße, wie z.B. die Manhattan Distanz oder die euklidische Distanz, die zur Anwendung kommen können.
In [HK06] werden folgende Typen von Klassenbildungs-Algorithmen unterschieden: Methoden die mit Partitionierung arbeiten, hierarchische Methoden, Dichte-basierte Methoden, Grid-basierte Methoden und Modell-basierte Methoden.
Einer der bekanntesten Klassenbildungs-Algorithmen ist der k-Means Algorithmus, der

mit Partitionierung folgendermaßen funktioniert: Der Algorithmus bekommt die Anzahl k der zu findenden Klassen als Eingabe. Er wählt per Zufallsprinzip k Objekte als vorläufige Klassenzentren aus. Dann ordnet er jedes Objekt jeweils der Klasse zu, dessen „Mittelwert“ es am nächsten ist. Der Mittelwert ist dabei ein gedachtes Objekt, dessen Attribute genau den Mittelwerten aller Objekte in der Klasse entsprechen. Nach der Zuordnung aller Objekte zu einer Klasse werden die Mittelwerte neu berechnet. Und der Algorithmus ordnet alle Objekte erneut einer Klasse auf Basis der Mittelwerte zu. Das wird solange gemacht bis sich keine Änderung bei der Klassenzugehörigkeit von Objekten mehr ergibt.
Bei den hierarchischen Clustering Algorithmen unterscheidet man Verfahren die agglomerativ und solchen die divisiv vorgehen. Agglomerative Verfahren ordnen zunächst jedem Objekt eine eigene Klasse zu und verschmelzen anschließend schrittweise möglichst ähnliche Paare von Klassen miteinander. Divisive Verfahren arbeiten top-down und unterteilen die Gesamtmenge aller Objekte schrittweise.
Ein Clustering Verfahren kann entweder auf einer Datenmatrix oder auf einer „Dissimilarity Matrix“ [HK06] arbeiten. Die Datenmatrix enthält pro Objekt alle Attributwerte in einer Zeile, während die Dissimilarity Matrix für jedes Paar von Objekten einen Abstandswert enthält.

3.6 Preprocessing

Vor der Anwendung von Data Mining Verfahren ist in der Regel eine Vorverarbeitung („Preprocessing“) notwendig, da häufig die zu untersuchenden Daten Lücken und Inkonsistenzen aufweisen, oder verrauscht sind [HK06]. Außerdem kann das Data Mining Verfahren weitergehende Anforderungen an die Eingabedaten (z.B. in Bezug auf das Eingabeformat) stellen. Han und Kamber unterteilen die Preprocessing Techniken für Data Mining Verfahren in 3 Kategorien [HK06]:

- Data Cleaning Verfahren. Sie beseitigen Inkonsistenzen aus den Daten und ermitteln Ersatzwerte für unbelegte Attribute.
- Data Integration and Transformation Verfahren. Diese Verfahren führen für das Data Mining relevante Daten zusammen.
- Data Reduction Verfahren. Sie ermitteln reduzierte Repräsentationen der ursprünglichen Daten. Dabei ist in der Regel das Ziel, das anschließende Data Mining zu beschleunigen.

In dieser Arbeit wird diese Unterteilung übernommen und es wird in Abschnitt 3.6.5 auf spezifisches Preprocessing für Clustering, Assoziationsanalyse und Regression eingegangen. Ein Großteil der Informationen über die Preprocessing-Verfahren stammt ebenso aus [HK06].

3.6.1 Data Cleaning

In [HK06] werden unter Data Cleaning solche Verfahren zusammengefasst, die fehlende Werte ersetzen, inkonsistente Daten behandeln oder Rauschen bzw. Ausreißer entfernen. In [RD00] wird Data Cleaning für Data Warehousing besprochen. Die Autoren zählen zum Data Cleaning auch das Entfernen von Fehlern aus den Daten. Die Begriffe Data Cleansing und Data Scrubbing werden häufig synonym für Data Cleaning verwendet.
Sowohl beim Data Mining als auch beim Data Warehousing ist es entscheidend, korrekte Daten zu verwenden, denn qualitativ schlechte Daten führen zu „falschen Schlussfolgerungen" [RD00] und somit letztendlich zu falschen Entscheidungen in Unternehmen. Daher spielt Data Cleaning eine wichtige Rolle in beiden Szenarien.
Rahm et al. [RD00] ordnen die Probleme, die sich auf die Datenqualität beziehen in eine 2-stufige Hierarchie ein. Sie unterscheiden Probleme, die bei Integrationsszenarien vorliegen können von solchen Qualitätsproblemen die sich auf nur 1 Datenquelle beziehen. Die 2.Stufe unterscheidet zwischen der Schema-Ebene und der Instanz-Ebene.
In diesem Abschnitt sollen Verfahren vorgestellt werden, die bei Problemen mit nur 1 Datenquelle angewendet werden können. Dabei wird davon ausgegangen, dass die Daten in einer Tabelle einer relationalen Datenbank vorliegen. Die Probleme bei Integrationsszenarien („Multi-source problems") werden im Abschnitt Data Integration behandelt.
Wenn eine Datenbank fehlende Werte aufweist, können verschiedene Strategien zur Anwendung kommen [HK06]: Eine Möglichkeit ist, die entsprechenden Tupel zu entfernen. Oder man könnte den Mittelwert aller Tupel für das Attribut einsetzen, oder den Mittelwert aller Tupel mit derselben Klasse.
Rauschen ist „ein zufälliger Fehler oder eine zufällige Varianz in einer gemessenen Variable" [HK06]. Es kann z.B. durch verschiedene Arten des Binnings entfernt werden.
Umfassen die Messungen mehr als eine Variable, so kann versucht werden, die Ausreißer durch Clustering zu identifizieren. Oder es können spezielle „Outlier-Detection" Verfahren zur Anwendung kommen. [KKZ09] fasst einige solche Verfahren zusammen, darunter auch die Dichte-basierten Verfahren. Breunig et al. [BKNS99] stellen fest, dass die herkömmliche Definition eines Ausreißers bei Clustern unterschiedlicher Dichte an ihre Grenzen stößt und stellen einen Algorithmus vor, der auch in solchen Situationen funktioniert. Kriegel et al. [Kri08] schlagen einen Algorithmus für hochdimensionale Problemklassen vor, der auf die Auswertung euklidischer Distanzen verzichtet und stattdessen Winkel-basiert arbeitet. Ein weiteres relevantes Verfahren zur Entfernung von Rauschen ist (multiple) linear Regression [HK06].
Inkonsistente Daten entstehen oft bei der Dateneingabe [RD00], z.B. durch Rechtschreibfehler, Verwendung von nicht eindeutigen Abkürzungen oder dadurch, dass eine Information in das falsche Feld einer Eingabemaske eingegeben wird. Eine wichtige Voraussetzung für die Konsistenz von Daten sind daher feste Standards für die Dateneingabe. Diese können in vielen Fällen durch ein passendes Schema forciert werden, z.B. durch Fremdschlüsselbeziehungen, uniqueness-Constraints oder Wahl des richtigen Datentyps für eine Eingabevariable.

3.6.2 Data Integration

Data Integration Techniken sind geeignet, um Daten aus mehreren, auch heterogenen Quellen zusammenzuführen, zum Beispiel aus einfachen Dateien und aus Datenbanken.
Rahm et al.[RD00] schlagen ein generisches 5-stufiges Vorgehen vor, das (neben dem reinen Data Cleaning) den Fall der Datenintegration mit einschließt:

1. Data Analysis: Der erste Schritt besteht darin, Fehler und Inkonsistenzen in den Datenquellen aufzudecken.
2. Definition of transformation and mapping rules: In diesem Schritt wird definiert, wie die einzelnen Quellen bereinigt werden sollen (siehe Data Cleaning) und wie die Schemas integriert werden sollen. Handelt es sich um einen ETL Prozess, so macht es Sinn, diesen als Workflow zu spezifizieren.
3. Verification: Test und Evaluation der definierten Workflows und Transformationen.
4. Transformation: Ausführung der Workflows und Transformationen
5. Backflow of cleaned data: Die bereinigten Daten sollen in ihre ursprünglichen Quellen zurückgeschrieben werden.

Data Integration ist wie das Data Cleaning ein bedeutender Schritt bei vielen Data Warehouse Szenarien. Es existieren dabei in der Regel bereits eine Reihe operativer Datenbanken, die sich nicht zur effizienten Auswertung komplexer Anfragen eignen, wie sie z.B. auf der Management-Ebene eines Unternehmens von Interesse sind. Deshalb integriert man diese Daten in ein Data Warehouse.
Die Anfragen werden dann mit dem Data Warehouse abgearbeitet.
Bei der Integration mehrerer Datenquellen, die Informationen über dieselben Entities speichern, tritt häufig das Problem auf, dass semantisch gleiche Attribute verschieden benannt, unterschiedlich codiert oder in anderer Weise nicht einheitlich vorliegen. Oder aber, dass die beiden Datenquellen dieselben Entities mit unterschiedlichen Attributen beschreiben. Das Problem das dann entsteht, wenn die Daten zusammengeführt werden sollen, wird in der Fachliteratur „entity identification problem“ genannt.
Ganesh et. al. [GSR96] verwenden einen semi-automatischen Ansatz um Entities aus zwei Quellen einander zuzuordnen. Dabei annotiert der Anwender eine Reihe von Tupeln aus beiden Datenquellen und vergibt für zusammengehörige Tupel eine eindeutige Entity ID. Dann wird eine Tabelle aufgebaut mit Distanzen (z.B. Edit distance) zwischen Tupeln und der Information, ob diese dieselbe Entity identifizieren (Match vs. NoMatch). Danach kommt der C4.5 Algorithmus (siehe Kapitel 5.6.2) zum Einsatz, um einen Entscheidungsbaum aufzubauen, der noch nicht gesehene Paare von Tupeln in die Klassen Match und NoMatch einordnen kann.
Ein weiteres Problem tritt auf, wenn für Attribute die Semantik nicht bekannt ist, deshalb ist es bei größeren Datenbanken sehr wichtig Metadaten vorzuhalten. Häufig kommt es bei der Datenintegration vor, dass manche Daten redundant vorliegen. Um Redundanzen

aufzudecken [HK06], helfen Korrelationstests wie z.B. der Chi-Square Unabhängigkeitstest für nominale Attribute. Für numerische Attribute eignet sich diese Formel:

$$r_{A,B} = \frac{\sum(A - \overline{A})(B - \overline{B})}{(n-1)\sigma_A \sigma_B}$$

[1] Liefert die Formel den Wert 0, so sind die Attribute A und B unabhängig, bei einem Wert größer 0 liegt eine (positive) Korrelation vor.
Über die hier vorgestellten Probleme im Bereich Datenintegration hinaus, sind z.B. noch Duplikate, Wertkonflikte und semantische Heterogenitäten erwähnenswert [HK06], die in dieser Arbeit nicht besprochen werden.

3.6.3 Data Transformation

Die Data Transformation Verfahren überführen die Daten in eine für das Data Mining passende Repräsentation [HK06]. Dazu zählt z.B. das Smoothing zur Entfernung von Rauschen und die Aggregation zur Anpassung der Granularitätsstufe von Daten. In manchen Fällen kann es sinnvoll sein, Objekte zu generalisieren. Dabei wird eine Hierarchie von Konzepten herangezogen, und Attributwerte die in der Hierarchie unten angeordnet sind durch solche Konzepte ersetzt, die darüber angeordnet sind. Zum Beispiel könnte man in einer Tabelle mit Zitaten die Seitenangaben durch Kapitelangaben ersetzen.
Normalisierung, also die Transformation eines stetigen Attributs auf einen einheitlichen Wertebereich (z.B. $[0..1]$) wird aus Effizienzgründen als Preprocessing bei Anwendung neuronaler Netze durchgeführt, und für Data Mining Verfahren die auf Distanzmaßen basieren dient es dazu, dass alle Dimensionen/Attribute gleich stark gewichtet werden. Gängige Verfahren sind die Min-Max Normalisierung und die Z-Score Transformation (siehe Abschnitt 3.6.5).
Darüber hinaus kann es in einigen Fällen sinnvoll sein, aus vorhandenen Attributen neue Attribute zu berechnen um die Daten damit für das Data Mining anzureichern.
In der Regel unterscheiden Data Mining Werkzeuge zwischen nominalen Daten und numerischen Daten, jedoch kann ein Werkzeug nicht immer anhand der syntaktischen Eigenschaften der Daten entscheiden, welcher dieser beiden Kategorien das Attribut zuzuordnen ist. Deshalb muss in diesem Fall der Anwender das Attribut annotieren. Bei WEKA gibt es dafür z.B. den Filter NumericToNominal.

3.6.4 Data Reduction

Data Reduction hat in erster Linie das Ziel, die Mining Dauer zu reduzieren [HK06].
In realen Datenbanken ist es keine Seltenheit dass die Relationen sehr viele Tupel bzw. Attribute haben. Dadurch steigt der Rechenaufwand für die Data Mining Algorithmen.

[1] n ist die Anzahl der Tupel. $\overline{A}$,$\overline{B}$ sind die Mittelwerte von A und B. σ_A,σ_B sind die Standardabweichungen von A und B.

Eine Möglichkeit große Datenmengen auf das Data Mining vorzubereiten ist die Aggregation, dadurch sinkt die Anzahl der Tupel. Der SQL Standard [SQL92] bietet dazu z.B. die GROUP-BY und HAVING Klauseln an.
„Attribute subset selection“ Verfahren begegnen dem Problem durch eine Vorauswahl relevanter Attribute. Bei solchen Verfahren unterscheidet man zwischen Filtern und Wrappern [HH03]: Wrapper nutzen im Gegensatz zu Filtern den Lernalgorithmus zur Bewertung der Relevanz von Attributen, der hinterher auf den Vorverarbeiteten Daten zum Einsatz kommen soll. Eine weitere Unterscheidung ist die zwischen forward-selection und backward-elimination Ansätzen. Erstere starten mit der leeren Menge und fügen zu der Menge relevanter Attribute nach dem Greedy Prinzip jeweils das Attribut hinzu, das nach einem Relevanzmaß am wichtigsten ist.
Zwei Beispiele solcher Relevanzmaße sind die statistische Signifikanz und der Informationsgewinn.
Mit Heuristiken wie dem Informationsgewinn umgeht man den Aufwand des naiven Verfahrens, einfach jede der 2 hoch D (bei D Dimensionen) Teilmengen aller Attribute auf ihre Eignung zu testen.
Aufgrund des niedrigen Rechenaufwands eignet sich die Methode, die den Informationsgewinn nutzt auch sehr gut für Textklassifikation [HH03], wo hochdimensonale Vektoren verarbeitet werden müssen.
Backward elimination Ansätze gehen von der Gesamtmenge verfügbarer Attribute aus und eliminieren schrittweise die irrelevanten Attribute.
Kompression zählt auch zu den Datenreduktionserfahren. Die Diskrete Wavelet Transformation (DWT) [HK06] ist eine in Linearzeit durchführbare Transformation, sie wandelt einen Eingabevektor in einen Vektor von Wavelet Koeffizienten um. Für eine verlustbehaftete ("lossy") Kompression genügt es, nur die stärksten Koeffizienten zu speichern. Der Originalvektor kann hinterher wieder aus diesen Koeffizienten angenähert werden.
Die Principal Component Analysis (PCA) betrachtet die Tupel in einer Relation als Vektoren in einem k-dimensionalen Raum, und findet dazu $c \leq k$ orthonormale Basisvektoren die sich zur Darstellung dieser Vektoren besonders gut eignen. Alle Tupel werden anschließend auf diese Basis umgerechnet. Natürlich haben solche Kompressions-Verfahren immer den Nachteil dass die Lesbarkeit durch den Menschen dadurch verlorengeht. Jedoch ist es in manchen Szenarien notwendig entsprechend Speicherplatz einzusparen.
Als weitere Möglichkeit zur Datenreduktion wird in [HK06] „Numerosity Reduction“ angeführt.
Dazu gehört Regression, die im linearen Fall aus den Trainingsdaten eine Geradengleichung für die sogenannte „Response Variable“ in Abhängigkeit von der „Predictor Variable“ aufstellt. „Multiple Regression“ leistet das gleiche für vektorielle Prädiktionsvariablen. Leider skalieren Regressionsverfahren nicht gut mit der Anzahl der Dimensionen. Zu „Numerosity Reduction“ gehören auch Histogramme, Sie können die Werteverteilung eines oder mehrerer Attribute erfassen, im letzteren Fall spricht man von multidimensionalen Histogrammen.
Darüber hinaus kann „Numerosity Reduction“ mit Klassenbildung (Clustering) erzielt werden.
Wenn anhand von n Attributen geclustert wird, so können hinterher die n Attribute durch einen Klassenindex ersetzt werden.

3.6.5 Spezifisches Preprocessing für Klassenbildung, Assoziationsanalyse und Regression

Die bisher vorgestellten Verfahren kommen allgemein bei der Vorbereitung von Daten auf Data Mining Vorgänge zur Anwendung. Clustering, Assoziationsanalyse und Regression benötigen jedoch ein spezifisches Preprocessing.

Preprocessing für Klassenbildungs-Methoden (Clustering)

Klassenbildungs-Methoden bestimmen die Cluster auf Basis von Distanzmaßen. Einige Klassenbildungs-Algorithmen benötigen deshalb als Eingabe (anstatt einer „Data Matrix“) eine „Dissimilarity Matrix“ mit je einem vorberechneten Distanzwert pro Objekt-Paar. Im Folgenden wird beschrieben, wie die Distanzwerte der Matrix berechnet werden können. Es handelt sich dabei um eine Zusammenfassung des Kapitels 8.2 aus [HK06].
In [HK06] wird zwischen 5 verschiedenen Datentypen unterschieden:

- Interval-Scaled Variables
- Binary Variables
- Nominal Variables
- Ordinal Variables
- Ratio-Scaled Variables

Eine Variable ist „Interval-scaled“, wenn Sie stetig ist und einen Wert auf einer linearen Skala angibt. Han und Kamber weisen darauf hin, dass die Maßeinheit, die verwendet wird, das Ergebnis der Klassenbildung beeinflussen kann. Um das zu verhindern sollte vor der Anwendung von Klassenbildungsverfahren standardisiert werden. Eine Möglichkeit das zu tun ist die z-score Transformation: Der Wert v eines Attributes A wird durch Abziehen des Mittelwerts von A und anschließendes Teilen durch die Standardabweichung von A standardisiert. $v' = \frac{v-\overline{A}}{\sigma_A}$ [HK06]. Nach der Standardisierung, die in manchen Fällen auch entfallen kann, können zwischen Objekten die nur „Interval-scaled“ Attribute haben, mit dem Manhattan-Distanzmaß oder dem Euklidischen Distanzmaß oder dem Minkowski-Distanzmaß, Abstände berechnet werden.
„Binary“ Variablen sind Variablen die nur entweder den Wert 0 oder 1 annehmen können. Für Objekte mit solchen binären Variablen kann zur Berechnung der Distanz eine Kontingenztafel aufgestellt werden. Die Distanz ergibt sich dann als Anzahl aller Variablen deren Wert sich bei den beiden betrachteten Objekten unterscheidet dividiert durch die Gesamtzahl der binären Variablen. Einen Sonderfall stellen asymmetrische binäre Variablen dar, bei denen die beiden Binärwerte unterschiedlich wichtig eingestuft werden. In diesem Fall eignet sich z.B. der Jaccard Koeffizient zur Bestimmung der Distanz; dabei verwendet man dieselbe Formel, nur dass man die binären Variablen, die im Wert 0 übereinstimmen nicht zur Gesamtzahl aller binären Variablen rechnet.
Nominale Variablen ähneln binären Variablen, jedoch können Sie mehr als 2 verschiedene

Werte annehmen, auf denen keine Ordnung definiert ist. Die Ähnlichkeit zweier Objekte kann bestimmt werden, indem die Anzahl m nominaler Variablen gezählt wird, die für beide Objekte denselben Wert haben. Dann ergibt sich für p nominale Variablen: $d(i,j) = (p-m)/p$. Alternativ kann man nominale Variablen auch binarisieren.
Ordinale Variablen sind nominale Variablen auf deren Wertebereich eine Ordnung definiert ist. Ordinale Variablen entstehen z.B. durch Diskretisierung von „Interval-scaled" Variablen [HK06]. Han und Kamber schlagen ein dreistufiges Vorgehen zur Berechnung von Ähnlichkeiten zwischen Objekten mit ordinalen Variablen vor.

1. Nummeriere die möglichen Variablenwerte durch $(1, 2, ..., N)$. Ersetze jeden Wert durch seine Nummerierung.
2. Damit alle ordinalen Variablen bei der Klassenbildung gleichberechtigt behandelt werden, muss der Wertebereich nach der Nummerierung auf den Intervall $[0, 1]$ normalisiert werden.
3. Verwende eines der Distanzmaße für „Interval-scaled" Variablen.

„Ratio-Scaled Variables" sind Variablen deren Werte auf einer non-linearen Skala definiert sind, das heißt ihr tatsächlicher Wert ergibt sich durch Berechnung einer nicht-linearen Funktion. Solche Variablen können nicht mit einfachen Distanzmaßen verarbeitet werden, sondern sie sollten vorher logarithmiert werden oder wie stetige, ordinale Variablen vorverarbeitet werden.

Preprocessing für Assoziationsregel-Methoden: Diskretisierung

Data Mining Algorithmen wie z.B. die Assoziationsregelverfahren können numerische (stetige) Daten nur indirekt verarbeiten, nämlich indem die Datenwerte zunächst auf sinnvolle repräsentative Intervalle abgebildet werden, und somit nominalisiert werden. Nominale Werte können relativ einfach auf Binäre Werte abgebildet werden, womit Assoziationsregelverfahren eigentlich arbeiten.
In der Regel ist eine Auswahl von Intervallgrenzen nicht trivial, da bei falscher Wahl der Intervallgrenzen interessante Assoziationsregeln aufgrund eines zu niedrigen Supports möglicherweise nicht aufgefunden werden. Folglich scheitert der einfache Ansatz häufig, äquidistante Intervallgrenzen zu wählen, insbesondere dann wenn die Wertebereiche der numerischen Attribute in zu viele Intervalle aufgeteilt werden.
In [Bay00] wird ein Diskretisierungs-Verfahren für multivariate numerische Daten vorgestellt. Die Idee des MVD Algorithmus ist, zunächst eine feine Unterteilung des Datenraumes in Zellen vorzunehmen und anschließend (nur) solche benachbarten Intervalle zu verschmelzen, deren Instanzen dieselbe oder zumindest eine ähnliche Verteilung aufweisen. Getestet wird diese Bedingung mit dem STUCCO Verfahren [BP99]. Der STUCCO Algorithmus nimmt als Eingabe zwei Gruppen von Instanzen entgegen und findet darin solche „Contrast Sets" also Konjunktionen von Attribut-Wert Paaren, die in den beiden Gruppen eine signifikant unterschiedliche Auftritts-Wahrscheinlichkeit haben und auch in ihrem jeweiligen Support eine gewisse Abweichung haben. Werden solche Contrast Sets für zwei benachbarte Intervalle gefunden, werden sie beim MVD Algorithmus nicht verschmolzen.

In [HH03] wird ein weiteres, Entropie-basiertes Diskretisierungs-Verfahren [FI93] beschrieben, das einzelne Attribute rekursiv aufsplittet.

Preprocessing für Regression

Wenn Regressionsverfahren auf sehr große Datensätze angewendet werden, (Datensätze mit vielen Trainingsbeispielen oder vielen Attributen) ist es sinnvoll vorher Data Reduction Verfahren anzuwenden (z.B. Principal Component Analysis). In [JMDX06] wird ein Ansatz beschrieben, der nach bestimmten Regeln einzelne Trainingsbeispiele aussondert bzw. in der Trainingsmenge belässt. Z.B. besagt eine Regel, dass Ausreisser entfernt werden sollten. Die Regeln werden auf der Grundlage eines Clusterings der Daten geprüft.

4 Lösungsansatz

Dieses Kapitel liefert einen Überblick über die Herausforderungen, die bei der Entwicklung des Werkzeugs zu bewältigen waren, sowie über die neu entwickelten Konzepte. Zunächst wird in Abschnitt 4.1 die Aufgabenstellung nochmals kurz vorgestellt und anschließend analysiert. Daraus ergeben sich 5 wichtige Komponenten der zu entwickelnden Software. Diese werden im Abschnitt 4.3 einzeln anhand eines Beispiel-Szenarios besprochen. Zum Schluss wird in Abschnitt 4.3.9 ein Blick auf die Umsetzung der graphischen Benutzeroberfläche geworfen.

4.1 Analyse der Aufgabenstellung

In der vorliegenden Diplomarbeit soll eine geeignete Benutzeroberfläche entwickelt werden, mit der Workflow Auditdaten mit operativen Daten konsolidiert werden können. Die Daten sollen sich mit Metriken anreichern lassen und am Ende für Data Mining genutzt werden können. Ein entsprechendes Szenario soll beschrieben werden, in dem mit Hilfe von Data Mining ein Workflow optimiert wird.
Aus den Anforderungen der Aufgabenstellung wurde die Datenpipeline aus Abbildung 4.1 gewonnen. Die Datenpipeline ist in der üblichen Notation des Pipes-Filters Architekturmodells [LL07] dargestellt. Die Notation bietet eine Systemsicht kombiniert mit einer dynamischen Sicht auf die Architektur eines Systems. Eine solche Architektur wird für Software zur Verarbeitung von Datenströmen verwendet.
In der Notation unterscheidet man zwischen Datenquelle, Datensenke, Kanal und Filter. Die zu verarbeitenden Daten durchlaufen die Filter entlang der Kanäle von der Datenquelle zur Senke. Das dargestellte Modell beschreibt den Normalablauf bei der Verwendung des Werkzeugs. Das Modell zeigt 5 voneinander zu unterscheidende Schritte beim Normalablauf, in Klammern wird jeweils angegeben, in welchem Abschnitt der Schritt besprochen wird:

1. Extraktion und Transformation (siehe Abschnitt 4.3.4)
2. Berechnung der Metriken (siehe Abschnitt 4.3.5)
3. Konsolidierung (siehe Abschnitt 4.3.6)
4. Preprocessing (siehe Abschnitt 4.3.7)
5. Data Mining (siehe Abschnitt 4.3.8)

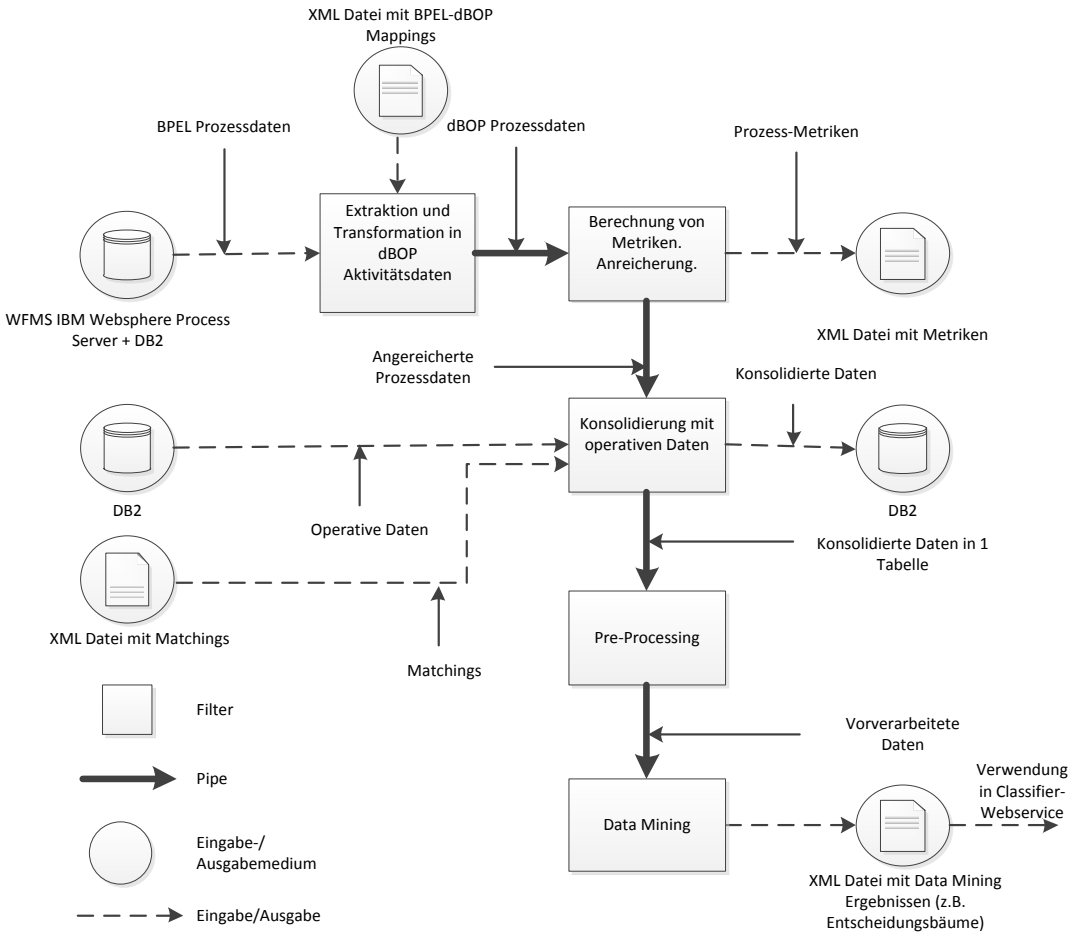


Abbildung 4.1: Datenpipeline

Das Modell zeigt als Datenquellen einen IBM Process Server, eine DB2 Datenbank und verschiedene XML Eingabedateien, die für Schritt 1 und 3 benötigt werden. Dazu gehört einerseits eine Abbildung von BPEL- Attributen und Aktivitäten auf deren dBOP Pendants in XML Form („BPEL-dBOP Mappings“). Andererseits gehören dazu die Zuordnungen zwischen Auditdaten und operativen Daten („Matchings“). Als Datensenken sind die DB2 Datenbank für die konsolidierten Daten, sowie XML Dateien für Metriken und Data Mining Ergebnisse vorgesehen.
Dieser generische Ablauf zur Verarbeitung der Prozessdaten und operativen Daten, wurde für das Optimierungsmuster Task Automation [RLM04] realisiert, da sich eine automatische Optimierung von Workflows nach diesem Muster nach Auffassung des Autors besonders gut umsetzen lässt.

4.2 Die Architektur des Werkzeugs (statische Sicht)

Die oben genannten Arbeitsschritte der Pipeline werden durchgehend von dem Werkzeug unterstützt. Eine der ersten Herausforderungen war es, einen grundsätzlichen Aufbau des Werkzeugs zu bestimmen. Die Architektur des Werkzeugs ist in Abbildung 4.2 zu sehen.
Es besteht aus einem Java Client und einem Webservice, der aufgrund seiner Aufgaben als

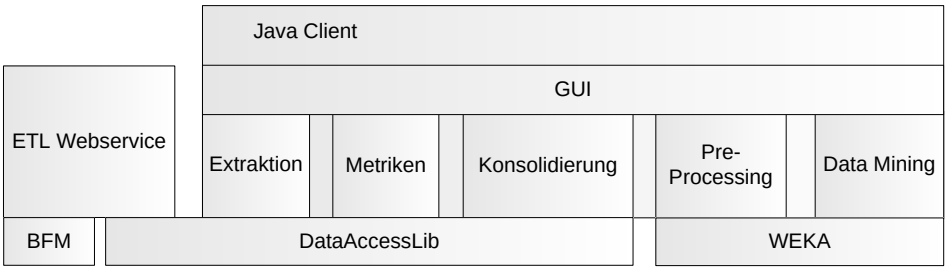


Abbildung 4.2: Architektur des Werkzeugs.

„ETL-Webservice“ bezeichnet wird. Der Benutzer interagiert nur mit dem Java Client: Er hat in der GUI die Möglichkeit, die einzelnen Schritte der Pipeline zu aktivieren und zu konfigurieren. Dementsprechend sind die Schritte in dem Schichtenmodell als tragende Säulen der GUI eingezeichnet.
Die Extraktion nutzt den ETL Webservice, der als solcher nicht zum Java Client gehört, sondern nur von ihm genutzt wird. Jedoch gibt es mit der DataAccessLib eine gemeinsame Basis, über die der Java Client und der Webservice auf Eingabedateien und die Datenbank zugreifen können.
Über die Datenbank erfolgt der Datenaustausch zwischen Java Client und ETL Webservice. Deshalb ist es sinnvoll den Zugriff beider Komponenten über die DataAccessLib als gemeinsame Zugriffskomponente abzuwickeln: Das bringt Vorteile für die Wartbarkeit, denn lesende und schreibende Funktionen der gleichen Daten können so in einer einzigen Java Klasse untergebracht werden. Ab dem Preprocessing Schritt stützt sich das Werkzeug auf die WEKA Data Mining Bibliothek [HFH+09].

4.3 Die Filter der Datenpipeline

In diesem Abschnitt werden die einzelnen Filter anhand eines durchgehenden Anwendungs-Szenarios erklärt. Das Szenario spielt sich in einem Versicherungs-Unternehmen ab.
Der Fokus liegt im Folgenden auf den besonderen Herausforderungen und neu entwickelten Konzepten. Eine ausführliche Besprechung kann im Entwurfs- und Implementierungskapitel (Kapitel 5 bzw. 6) nachgelesen werden.

4.3.1 Szenario: Bewertung von Versicherungsfällen („Fraud Detection“)

Große Versicherungen sind mit einer hohen Zahl an Versicherungsfällen konfrontiert, die täglich zu bearbeiten sind. Um einen Versicherungsbetrug unwahrscheinlich zu machen, ist es notwendig, die eingehenden Fälle näher zu untersuchen.
Ein Prozess zur Abwicklung könnte in 3 Schritten ablaufen:

1. Aufnahme des Versicherungsfalls
2. Einordnung durch einen Sachbearbeiter in Risikogruppen.
3. Getrennte Behandlung der Fälle nach Risikogruppe: Eingehende Prüfung, Normale Prüfung, Keine Prüfung.

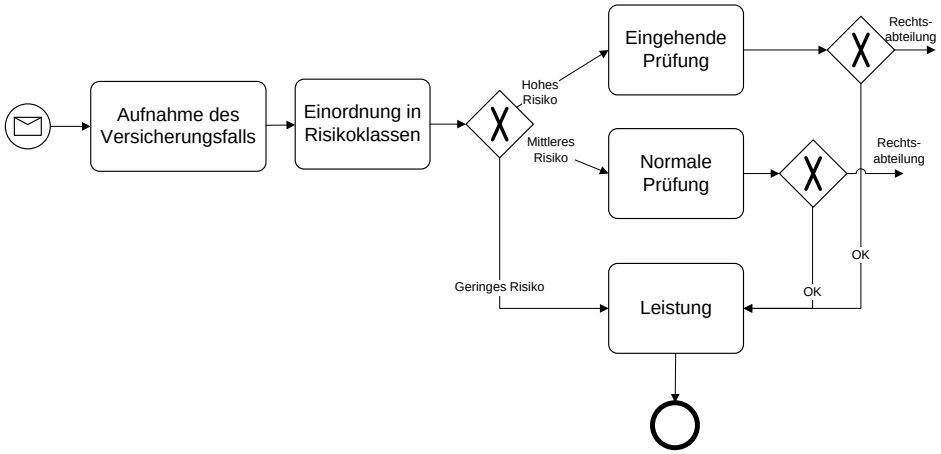


Abbildung 4.3: Geschäftsprozess zur Verarbeitung von Versicherungsfällen.

4.3.2 Ziel des Werkzeugs

Das Werkzeug hat das Ziel, eine Optimierung von Prozessen wie dem hier beschriebenen Versicherungsfall-Prozess, zu unterstützen. Dabei wird das Task Automation Optimierungsmuster [RLM04] verwendet. Bezogen auf einen in BPMN modellierten Prozess heißt das, dass eine manuelle Aktivität durch eine Computer-gestützte Aktivität ersetzt wird. Im Beispiel ist damit die Aktivität 2 „Einordnung in Risikoklassen“ gemeint. Diese Automatisierung kann erzielt werden, indem viele manuelle Risiko-Klassifikationen protokolliert und dann in ein Entscheidungsbaum-Modell umgesetzt werden. Das Entscheidungsbaum-Modell kann im letzten Schritt in einem Klassifikations-Webservice benutzt werden, der die Aktivität 2 ersetzt.

4.3.3 Verwendung des Werkzeugs

Um das Werkzeug einsetzen zu können ist es erforderlich, den IBM Websphere Process Server als Workflow Management System einzuführen. Dann muss der oben beschriebene Prozess mit dem dBOP Editor in BPMN entworfen und auf dem Process Server als BPEL Workflow deployt werden. Anschließend protokolliert der Process Server die Bewertungen der Sachbearbeiter in Form von Workflow Auditdaten.
Es folgt die Beschreibung der Filter des Werkzeugs, die anschließend auf den Audit Trail angewandt werden können.

4.3.4 Der Filter für die Extraktion und Transformation der Audit Daten

In diesem Filter werden, bezogen auf das Fraud Detection Beispiel, Auditdaten über den Versicherungs-Workflow abgerufen und auf BPMN Ebene transformiert.
Das Werkzeug ruft Aktivitäts- und Prozesszentrierte Daten ab, transformiert sie und speichert sie in einer, dem Java Client zugänglichen, Datenbank.
Der Abruf dieser Daten erfolgt mit Hilfe der Business Flow Manager API. Dieser Vorgang und die Transformation ist komplett im ETL-Webservice implementiert. Nach Ausführung dieses Filters im Beispielszenario existieren in der Datenbank Aktivitätstabellen und eine Prozesstabelle. Die Aktivitätstabellen enthalten Daten wie Uhrzeit zu Beginn und Ende der Aktivitäten, Eingabedaten wie Versichertennummer, Sachbearbeiter-ID, einen Text der den Versicherungsfall beschreibt und die Information welche Aktivitäten danach zur Ausführung gekommen sind. Bei Aktivität 2 ist außerdem als Ausgabedatum die Risikogruppe vermerkt, die der Sachbearbeiter dem Versicherungsfall zugewiesen hat.
Um zu diesem Ergebnis zu kommen, werden die Daten, die an der Schnittstelle zum Business Flow Manager abgerufen werden können und sich noch auf den BPEL Workflow beziehen in BPMN transformiert.
Dazu wird pro BPEL Aktivität über die DataAccessLib mit einem XPath in der XML Eingabe nachgesehen, welcher BPMN Aktivität sie entspricht. Anschließend werden die Eingabedaten und Ausgabedaten in Form von DataObjects abgerufen und rekursiv traversiert. Gefundene Attribute werden ebenfalls in der XML Eingabe per XPath ihrem dBOP Pendant zugeordnet.
Der Extraktionsalgorithmus geht eine Prozessinstanz nach der anderen durch und ruft für jede Prozessinstanz die protokollierten Aktivitätsinstanzen ab. Jede davon vermerkt er im ActivityActivationLog der mit einer HashMap realisiert wurde. Sind alle Aktivitätsinstanzen abgerufen, wird für jede davon der ActivityActivationLog ausgelesen. Im Fraud Detection Beispiel würden für die Tabelle der Aktivität „Einordnung in Risikoklassen“ die Spalten $AKTIVITÄT_{EingehendePrüfung}$, $AKTIVITÄT_{NormalePrüfung}$ und $AKTIVITÄT_{Leistung}$ erzeugt und gefüllt. So kann später im Data Mining Schritt (auch unabhängig von dem Attribut Risikogruppe) der Prozessfluss analysiert werden. Abbildung 4.4 zeigt den Ablauf schematisch.

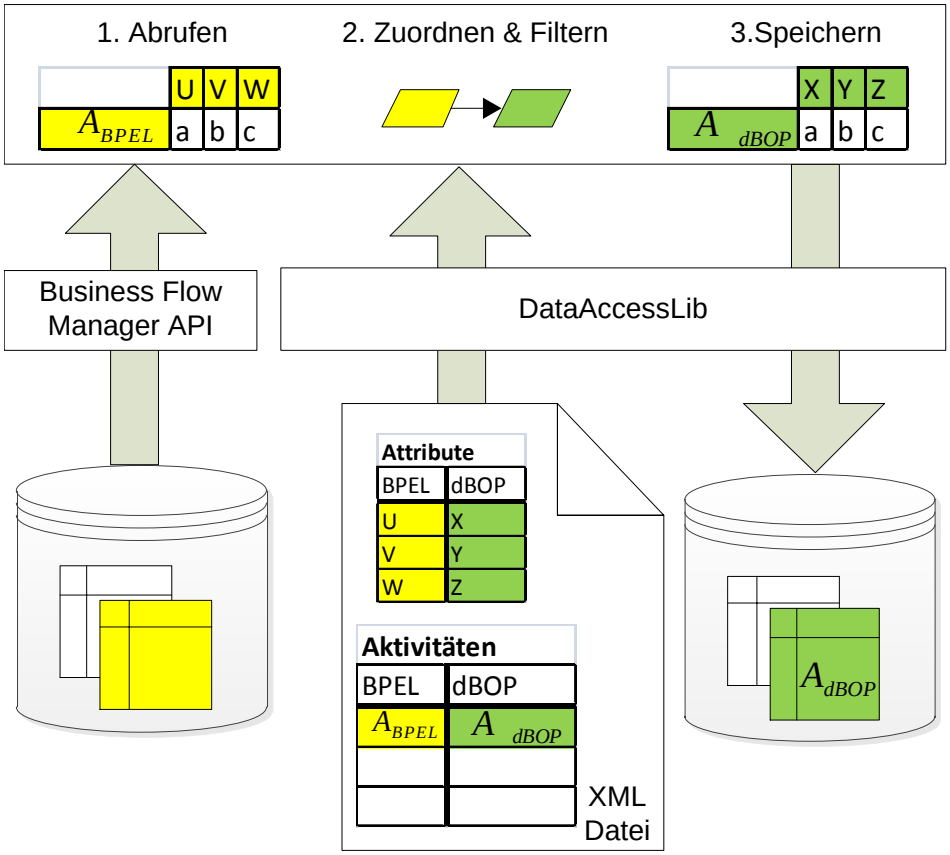


Abbildung 4.4: Der 3-stufige Extraktionsvorgang.

4.3.5 Der Filter für die Berechnung der Metriken

Dieser Filter reichert die Auditdaten mit weiteren Attributen (Metriken) an, die sich ausschließlich aus Audit-Attributen berechnen lassen. Im Fraud Detection Beispiel könnte eine solche Metrik die Zeit sein, die der Sachbearbeiter benötigt um die Versicherungsfälle einzuordnen. Da zur Entwicklungszeit des Werkzeugs nicht alle Metriken, die einmal benötigt werden könnten bekannt sind, spielt hier Erweiterbarkeit eine wichtige Rolle. Die Metriken sollten darüber hinaus im XML Format für den dBOP Editor bereitgestellt werden.
Die Metriken sind in der Implementierung auf Klassen abgebildet. Dabei wird zwischen Prozessmetriken und Aktivitätsmetriken unterschieden.
Die Gemeinsamkeiten, nämlich Berechenbarkeit pro Instanz, Aggregierbarkeit und Serialisierbarkeit dieser Metrik-Klassen wurden in einer gemeinsamen Vaterklasse der Klassen deklariert. Serialisierbarkeit und Aggregierbarkeit zu Minima, Maxima, Mittelwert und Median wurde per SQL bereits in dieser Vaterklasse implementiert, sodass neue Metriken durch Vererbung diese Fähigkeiten automatisch mitbringen, sofern Sie sich um die Berechnung der

Werte pro Aktivitäts- oder Prozessinstanz kümmern.
Da es Metriken gibt, die andere Metriken nutzen, wie zum Beispiel die Aufruffrequenz einer Aktivität, wurde ein Repository implementiert in dem beliebige Metriken in Form von Java Objekten unter einem String Schlüssel zwischengespeichert werden können. Der Programmierer einer Metrik hat außerdem die Möglichkeit, seine Metrik in die Reihenfolge der Berechnung bestehender Metriken beliebig einzuordnen.

4.3.6 Der Filter für die Konsolidierung

Bei der Konsolidierung werden die dBOP Aktivitätsdaten mit Hilfe der BIA Matchings [RNB10] mit passenden operativen Daten zusammengeführt. Im Fraud Detection Beispiel handelt es sich bei solchen Daten z.B. um die Gesamtzahl von Versicherungsfällen eines Versicherten, die Summe der erbrachten Leistungen für den Versicherten und möglicherweise sein Alter. In Bezug auf den Sachbearbeiter können z.B. Daten wie absolvierte Fortbildungen und Erfahrung aus operativen Datenbanken ausgelesen werden. Alle diese Attribute werden als neue Spalten der Aktivitätstabelle der Aktivität „Einordnung in Risikoklassen“ hinzugefügt. Eine besondere Herausforderung in diesem Schritt waren der Entwurf und die Implementierung eines passenden Bedienelements. Die operativen Daten spezifiziert der An-

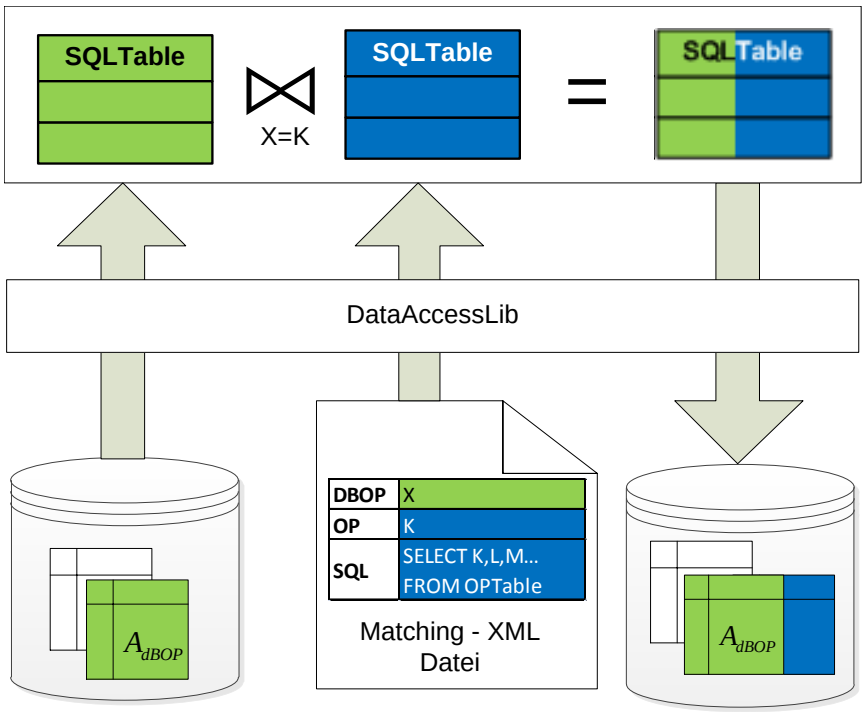


Abbildung 4.5: Konsolidierung der Prozessdaten und operativen Daten.

wender vor Ausführung des Werkzeugs in einer Matching Datei im XML Format. Für jedes Matching gibt er einen SQL Befehl zur Extraktion (Dieser kann in der GUI noch korrigiert

werden.) an, sowie die Spalten der Aktivitätstabelle und der extrahierten operativen Tabelle über die der Join der beiden Tabellen erfolgen soll.
Da diese Tabellen nun verwaltet werden müssen, wurde die DataAccessLib um eine entsprechende Klasse SqlTable erweitert, die zum Beispiel den Name einer Tabelle speichert und ihre Spaltennamen. Bevor der Join durchgeführt wird, wird per SQL getestet, ob zwischen der Aktivitäts-Entity und der operativen Entity eine 1:N Beziehung besteht. Kann das aus den Daten erschlossen werden, so wird die Spalte, die für eine Aktivitätsinstanz mehrere Werte aufweist, umcodiert in „N“ binäre Spalten. Das ist für die Korrektheit des Data Mining Schritts erforderlich, der mit Verfahren arbeitet, die mit mehrwertigen Attributen nicht funktionieren.
Abbildung 4.5 zeigt schematisch den Vorgang der Konsolidierung (ohne Binarisierung). In Abbildung 4.6 sind die Bedienelemente für die Konsolidierung zu sehen. Der Screenshot zeigt, wie ein SQL Befehl zur Extraktion operativer Daten in der GUI editiert werden kann. Am linken Rand ist die Visualisierung der Datenpipeline zu sehen. Rechts oben befindet sich ein Fenster mit einem Hilfetext und darunter das Protokoll.

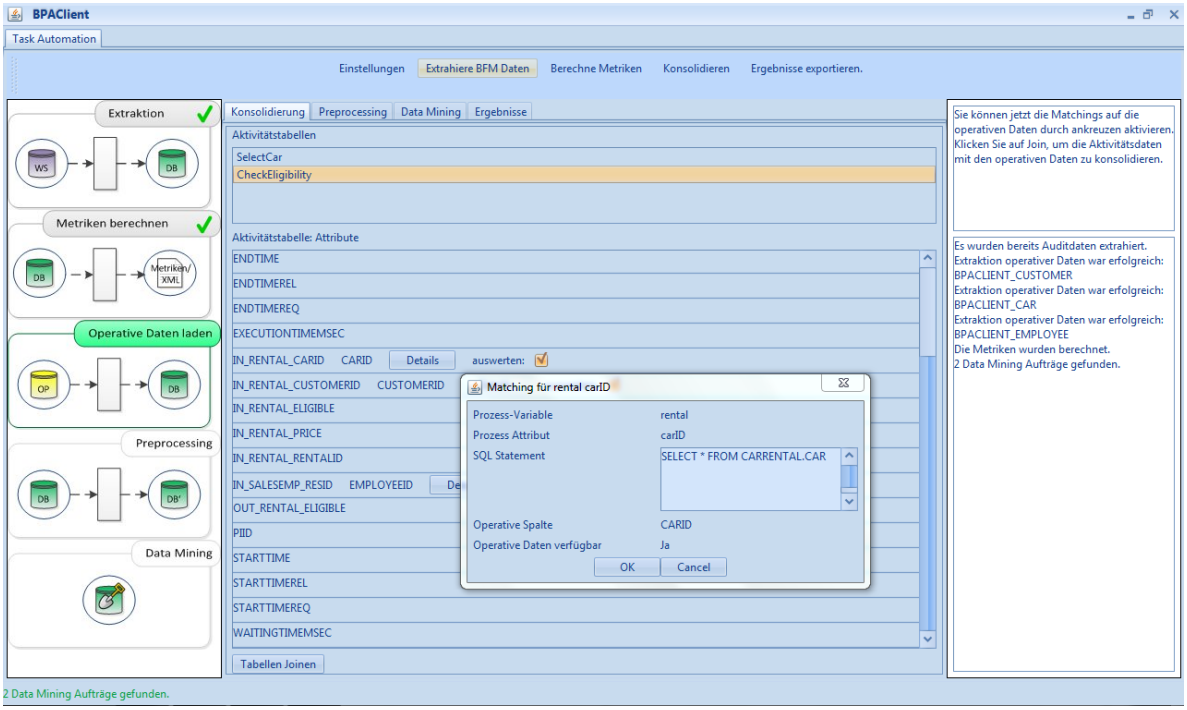


Abbildung 4.6: Ansicht für den Filter Konsolidierung.

4.3.7 Preprocessing

Im Preprocessing Schritt können die konsolidierten Daten für das Data Mining vorbereitet werden. Dazu wird in dem Werkzeug das Preprocessing Panel von WEKA wiederverwendet.

Da WEKA den JDBC Treiber der DB2 Datenbank von Haus aus nicht enthält, überschreibt das Werkzeug die Methode, die die Daten von der Datenbank in den Hauptspeicher holt. Das Preprocessing Panel erlaubt unter anderem folgende Aktionen:

- Nicht benötigte Attribute der Aktivitätstabelle der zu automatisierenden Aktivität können entfernt werden. Im Fraud Detection Beispiel macht es z.B. keinen Sinn die AktivitätsID in das Mining einzubeziehen.
- Fälschlicherweise numerisch interpretierte Attribute können nominalisiert werden: Eine Versichertennummer ist zwar vom Typ Integer, ist jedoch nominal zu interpretieren.
- Das Weka Preprocessing Panel bietet noch viele weitere Möglichkeiten, zum Beispiel Diskretisierung und eine automatische Auswahl wichtiger Attribute.

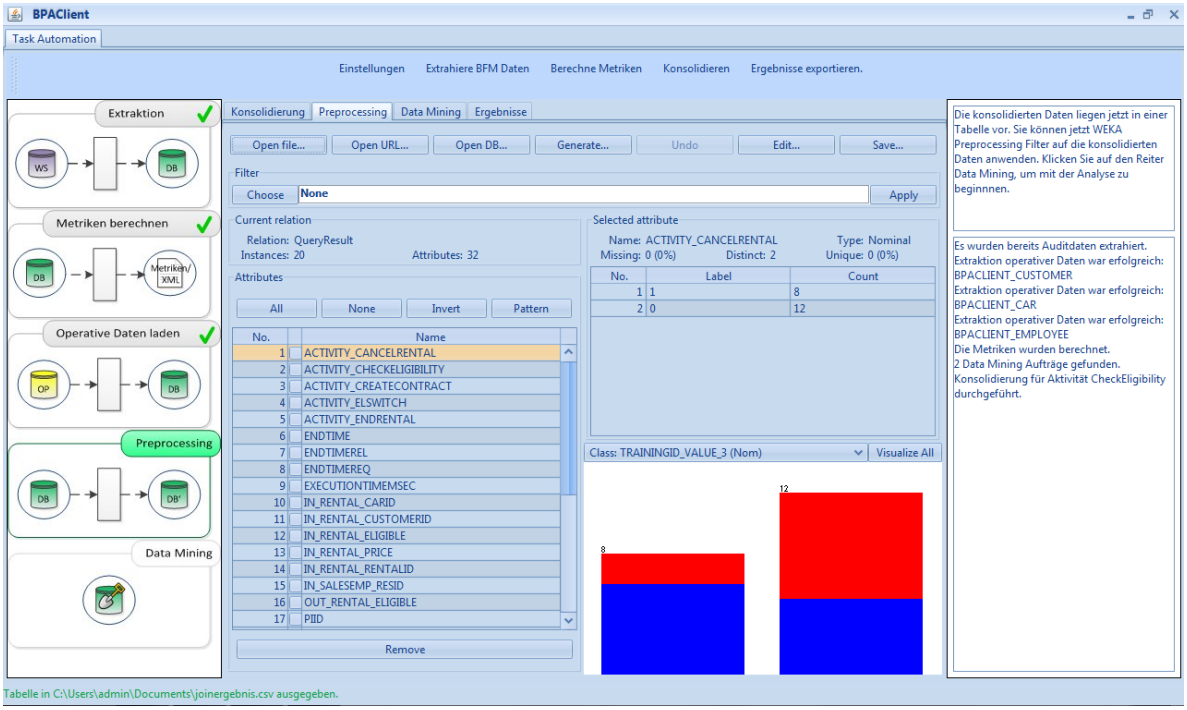


Abbildung 4.7: Ansicht für den Filter Preprocessing. Dargestellt ist eine Liste aller Attribute der Aktivitätstabelle die der Benutzer bearbeiten kann. Außerdem werden statistische Eigenschaften ausgewählter Attribute visualisiert (Balkendiagramm)

4.3.8 Data Mining

Im Data Mining Schritt werden für die Ausgabedaten der untersuchten Aktivität Entscheidungsbäume und Regressionsbäume ermittelt. Im Fraud Detection Beispiel muss für die Risikogruppe ein Entscheidungsbaum ermittelt werden. Dieser Entscheidungsbaum kann

anschließend von einem Classifier-Webservice verwendet werden, um die Aktivität „Einordnung in Risikoklassen“ automatisch durchzuführen. Das Grundgerüst eines solchen Webservice wurde im Rahmen der Diplomarbeit entworfen. Für die Erstellung des Entscheidungsbaums kommt die WEKA Implementierung des C4.5 [Qui93] zum Einsatz. Die Auswahl der Testdaten für den Klassifikator kann der Anwender über das WEKA Data Mining Panel beeinflussen.
In Abbildung 4.8 ist ein Beispiel zu sehen, wie im Fraud Detection Beispiel ein Entscheidungsbaum-Klassifikator aussehen könnte. Der Klassifikator wird von dem Werkzeug als serialisiertes Java Objekt in einer XML Datei ausgegeben, zusammen mit der Genauigkeit des Klassifikators auf den Testdaten und mit weitergehenden Informationen über die Konfiguration der tatsächlich angewandten Filter (z.B. über im Konsolidierungsschritt tatsächlich genutzte Matchings).

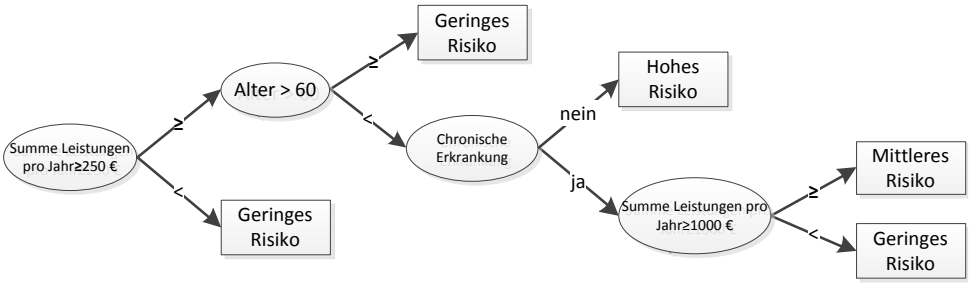


Abbildung 4.8: Ein Entscheidungsbaum zur Kategorisierung von Versicherungsfällen. Dieser Classifier kann zur Automatisierung der Aktivität „Einordnung in Risikoklassen“ (siehe Abbildung 4.3) mittels eines Classifier-Webservice verwendet werden.

4.3.9 Die graphische Benutzerschnittstelle

Eine weitere Herausforderung war die Konzeption und Implementierung einer aufgabengerechten und verständlichen Bedienoberfläche. Hier kam das Sites Modes Trails HMI-Modell [NW87] zum Einsatz. Es wurde durch Visualisierung der Datenpipeline sowie durch ein Hilfetextfenster und ein Aktionsprotokoll umgesetzt (siehe Abbildung 4.6).
Fehlbedienungen werden durch einen entsprechenden DEA erkannt. Sonstige Fehler werden in Form von Exceptions abgefangen und dem Benutzer über die Informationsleisten mitgeteilt. Das Erscheinungsbild der GUI ist in Abbildung 4.6 zu sehen.

5 Entwurf des Werkzeugs

In diesem Kapitel werden die genauen Anforderungen, die an das Werkzeug gestellt wurden, besprochen. Es wird außerdem darauf eingegangen, wie die Anforderungen bezüglich der GUI und bezüglich der Filter umgesetzt wurden. Die entsprechenden Abschnitte haben einen einheitlichen Aufbau: Unter der Überschrift „Ziel“ werden die Anforderungen in der Regel an einem Anwendungsfall (Use Case) verdeutlicht. Die eigentlichen Entwurfsentscheidungen sind jeweils unter „Entwurf“ zu finden.

5.1 Entwurf der graphischen Benutzeroberfläche

5.1.1 Ziel

Für die Steuerung des Werkzeugs sollte ein geeignetes Benutzer-Interface bereitgestellt werden. Über dieses Interface sollte festgelegt werden können, welche Inhalte in welchem Umfang aus der Websphere Process Server Umgebung extrahiert werden sollen. Darüber hinaus sollte über das Benutzer-Interface der Konsolidierungsvorgang konfigurierbar sein.

5.1.2 Entwurf der graphischen Benutzeroberfläche

Das Benutzerinterface des Werkzeugs teilt sich auf in eine graphische Benutzeroberfläche und eine Schnittstelle für Konfigurationsdaten, die der Benutzer dem Werkzeug in Form einer XML Datei bereitstellt.
In der XML Datei können die Extraktionsinhalte spezifiziert werden, und es können Matchings für die Konsolidierung darin definiert werden. Die restlichen Eingaben, die für die Konsolidierung erforderlich sind, können über die graphische Benutzeroberfläche erfolgen (siehe Abbildung 4.6). Die graphische Benutzeroberfläche wurde auf Grundlage des Sites-Modes-Trails Modells von Nievergelt and Weydert [NW87] entworfen. Es sagt aus, dass der Benutzer einer graphischen Benutzeroberfläche zu jedem Zeitpunkt die Fragen beantworten können sollte,

1. wo er sich befindet,
2. was er dort tun kann,
3. wie er dorthin gekommen ist,
4. und wie er andere „Orte“ erreichen kann.

Dementsprechend sollte bei der Verwendung des Werkzeugs stets klar sein, an welcher Stelle der Datenpipeline die Verarbeitung angelangt ist. Es sollte klar sein wie der aktuelle „Filter“ konfiguriert werden kann, und wie die „Filterung“ angestoßen wird. Wenn dazu noch die Datenpipeline selbst visualisiert wird, sind die Kriterien des Sites-Modes-Trails Modells für Benutzerfreundlichkeit erfüllt.
Da es sich bei dem Programmverlauf um eine einfache Daten-Pipeline, also eine Sequenz von Arbeitsschritten/Filtern handelt, besteht eine gute Lösung zur Beantwortung der Fragen 1,3 und 4 darin, die Pipeline zu visualisieren. Das erfolgt in einer vertikalen Leiste, die links neben dem Arbeitsbereich, also dem Hauptbildschirm platziert ist.
Die Visualisierung zeigt für jeden Arbeitsschritt ein Symbol an, und ob er gerade aktiv, bereits ausgeführt, noch anstehend ist oder einen Fehler gemeldet hat. Der Status eines Arbeitschritts wird mit einer Farbcodierung gezeigt. Ein grünes Symbol bedeutet, dass der Arbeitsschritt ausgeführt werden kann. Rot bedeutet, dass ein Fehler aufgetreten ist. Ein weißes Symbol bedeutet, dass der Arbeitsschritt noch nicht aktiviert werden kann. Ein grüner Haken bedeutet, dass der Arbeitsschritt ausgeführt wurde.
Zusätzlich gibt es rechts neben dem Arbeitsbereich ein Protokoll, das alle Arbeitsschritte, die ausgeführt wurden, schriftlich aufführt. Der Benutzer erhält darüber hinaus Hilfestellungen zu dem, was er in der aktuellen Ansicht tun kann (Frage 2) über ein Fenster, das rechts über dem Protokoll platziert ist. Über dem Hauptbildschirm befindet sich eine beschriftete Toolbar in der die Arbeitsschritte angestoßen werden können.
Die Benutzerinteraktion wird durch einen DEA überwacht. So wird sichergestellt, dass der Benutzer nicht einen Pipelinefilter anstößt, an dem noch keine Eingabedaten verfügbar sind. Der Automat ist in Abbildung 5.1 zu sehen. Zusätzlich zu den gezeigten Kanten gibt es Kanten, die von Zustand Z_n zu Zustand Z_m führen mit $m < n$. D.h. wenn die Datenverarbeitung an einem bestimmten Filter angelangt ist, kann ein beliebiger vorheriger Filter erneut angestoßen werden.

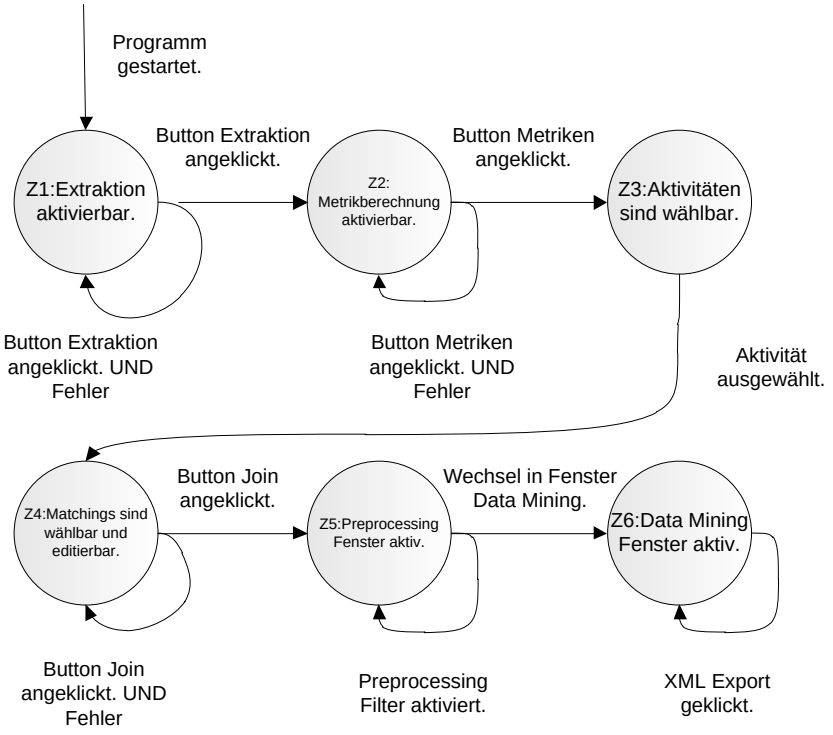


Abbildung 5.1: Vereinfachter deterministischer endlicher Automat zur Kontrolle der Benutzerinteraktion. Es sind nicht alle Zustände und Kanten dargestellt.

5.2 Extraktion der Audit Daten

5.2.1 Ziel

Das Ziel des ersten Abschnittes der Entwicklung des Werkzeugs ist die Extraktion von Audit-Daten eines Websphere Process Servers. Die erforderlichen Audit-Daten, die extrahiert werden sollen, beziehen sich grundsätzlich auf die BPMN-Prozessebene und nicht auf die tatsächliche Form in der der Prozess auf dem Process Server vorliegt, also in Form von BPEL Code.
Die Audit-Daten umfassen pro (BPMN) Aktivität den Startzeitpunkt, Endzeitpunkt und die Wartezeit, also die Zeit, in der die Aktivität Ressourcen anfordert, auf die Freigabe von Ressourcen wartet, oder sie selbst wieder freigibt. Der weitere Prozessfluss nach der Ausführung einer Aktivität gehört ebenfalls zu den Audit Daten. Darüber hinaus sollen die Eingabe- und Ausgabedaten aller Aktivitäten erfasst und zur weiteren Verarbeitung bereitgestellt werden.

Name	Prozessdaten-Extraktion
Ziel	Extraktion von Prozessdaten für einen BPEL-Prozess. Transformation in dBOP Prozessinformationen.
Vorbedingung	Das Tool wurde mit validen Eingabedaten aufgerufen.
Nachbedingung	Pro dBOP-Aktivität existiert eine Tabelle mit Informationen über einzelne Aktivitätsinstanzen.
Akteure	Anwender, Tool
Normalablauf	1. Der Anwender klickt auf die Option "BFM Daten extrahieren." 2. Das Tool ermittelt aus den Eingabedaten den zu analysierenden Prozess. Es ruft dann alle relevanten BPEL-Prozessdaten vom Websphere Process Server ab. Mit Hilfe der Eingabedatei werden einzelne BPEL Aktivitäten den dBOP Aktivitäten zugeordnet, genauso werden BPEL Attribute dBOP Prozessattributen zugeordnet. Das Tool transformiert unter Verwendung dieser Mappings die BPEL Prozessdaten in dBOP Prozessdaten, und speichert Sie in einem DBMS.

Tabelle 5.1: Use Case: Audit-Daten extrahieren.

Alle Prozesse, die das in dieser Arbeit vorgestellte Werkzeug verarbeiten kann, müssen mit dem in der Aufgabenstellung bereits erwähnten dBOP-Designer in Form von BPMN-Prozessdiagrammen erstellt worden sein. Der dBOP-Designer übersetzt die BPMN-Prozessdiagramme in BPEL Code, der anschließend auf einem IBM Websphere Process Server deployt werden kann. Zur Übersetzung von BPMN in BPEL generiert der dBOP-Designer pro Aktivität ein dreistufiges Schema in BPEL:
1. Stufe: Ressourcen anfordern.
2. Stufe: Aufruf des Webservice, der die Aktivität realisiert.
3. Stufe: Ressourcen freigeben.
Dabei wird die erste und dritte Stufe vom sogenannten Resource Manager durchgeführt, dessen Entwicklung nicht Teil der vorliegenden Arbeit ist. Dieser fordert zunächst Ressourcen an, die die Aktivität verwendet. Anschließend erfolgt die Ausführung der eigentlichen Aktivität in Form eines Webservice Aufrufs. Am Ende gibt der ResourceManager die Ressourcen wieder frei.
Folglich ergibt sich die Startzeit als die Startzeit der 1.Stufe und der Endzeitpunkt als der Endzeitpunkt der dritten Stufe. Die Wartezeit ist die Ausführungsdauer der ersten und dritten Stufe zusammen und die Eingabedaten bzw. Ausgabedaten sind genau die des Webservice der in Stufe 2 aufgerufen wird.
Nach dem Deployment und mindestens einer Ausführung des Prozesses, kann das Werkzeug die Prozessdaten analysieren. Der IBM Websphere Process Server erlaubt den Zugriff auf Prozessdaten auf seiner Betrachtungsebene, der BPEL Ebene. Dazu gibt es zwei Möglichkeiten: Den Zugriff über SQL und den Zugriff über die Business Flow Manager Programmierschnittstelle (BFM API). Der Zugriff über die Business Flow Manager API ist besser dokumentiert

und bietet einen direkten Weg um auf Ein- und Ausgabedaten von BPEL-Aktivitäten zuzugreifen.
Das Werkzeug erhält als Eingabe eine XML Datei, die für jede Aktivität die Information enthält, welche Prozessvariablen und welche Attribute davon aus dem Audit Trail ausgelesen werden sollen. Die XML Datei kann vom dBOP Designer auf Wunsch generiert werden, ein Beispiel ist in Listing 5.1 zu sehen.

5.2.2 Entwurf der Extraktion

In einer Vorstudie der vorliegenden Diplomarbeit hat sich herausgestellt, dass sich die Business Flow Manager API am Besten aus dem Kontext eines Websphere Process Servers nutzen lässt. Deshalb ist für die eigentliche Extraktion der Audit-Daten ein Webservice vorgesehen, der auf demselben Process Server deployt und ausgeführt wird, wie der zu analysierende Prozess. Der Webservice verfügt über 2 konzeptionelle Schnittstellen: Eine zur Entgegennahme der XML Eingabedatei, der er die Zuordnungen zwischen BPEL und BPMN Aktivitäten, sowie die zu extrahierenden Prozessvariablen entnimmt und eine weitere Schnittstelle zum Hauptprogramm, um die Prozessdaten zurückgeben zu können.
Die Extraktion läuft in 5 Stufen ab:

1. Das Hauptprogramm liest die XML Eingabedatei (Beispiel siehe Listing 5.1) ein und stellt eine Verbindung zu einer DB2 Datenbank her, auf die auch von dem Websphere Process Server aus zugegriffen werden kann.

2. Das Hauptprogramm erzeugt in der Datenbank jeweils eine Tabelle $Activity_X$ pro BPMN Aktivität. Abbildung 5.2 zeigt die XML Datenquelle und eine entsprechende Tabelle. Die Aktivitätstabellen haben also folgende Struktur:
   $Activity_X$($AIID$:INTEGER,$Start$:LONG,End:LONG,$Start_{Request}$:LONG,$End_{Request}$:LONG,
   $Start_{Release}$:LONG,$End_{Release}$:LONG,$Input_1$:D_{Input_1},...,$Input_m$:D_{Input_m},
   $Output_1$:D_{Output_1},...,$Output_n$:D_{Output_n},
   $Activated_{Activity_1}$:BOOLEAN,...,$Activated_{Activity_o}$:BOOLEAN)
   Dabei werden auch die Spalten für die Eingabe ($Input_1$:D_{Input_1},...,$Input_m$:D_{Input_m}) und Ausgabedaten ($Output_1$:D_{Output_1},...,$Output_n$:D_{Output_n}) erzeugt, ihr Typ ist der SQL Typ, der dem XML Typ der Prozessvariable entspricht. Z.B. wird für xs:decimal eine BIGINT Spalte erstellt. Außerdem die Spalten ($Activated_{Activity_1}$:BOOLEAN,...,$Activated_{Activity_o}$:BOOLEAN) die für alle nachfolgenden Aktivitäten die Information enthalten, ob die entsprechende Aktivität nach Aktivität X zur Ausführung kam oder nicht. Diese Informationen können später genutzt werden um Prozessflussinformationen in das Data Mining einzubeziehen. Die Wartezeit wird nicht durch den Webservice berechnet sondern im Hauptprogramm als Metrik (siehe nächster Abschnitt).

3. Das Hauptprogramm ruft den Webservice auf und übergibt dabei die XML Eingabe als Argument.

4. Der Webservice ruft die Prozessdaten mit Hilfe der BFM API ab und füllt die Aktivitätstabellen.

5. Der Webservice meldet an das Hauptprogramm zurück, ob die Extraktion erfolgreich ausgeführt werden konnte.

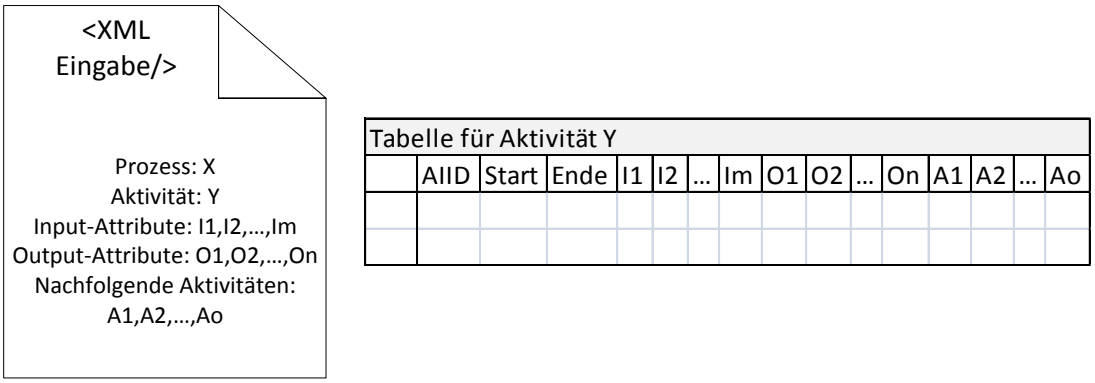


Abbildung 5.2: Erstellung der Aktivitätstabellen. Links: XML Datei mit Informationen über Aktivität X. Rechts: Aktivitätstabelle.

Das Szenario ist in Abbildung 5.3 zu sehen. Dabei ist es unerheblich, ob sich der Process Server und die DB2 Datenbank auf ein und demselben Server befinden oder auf getrennten.

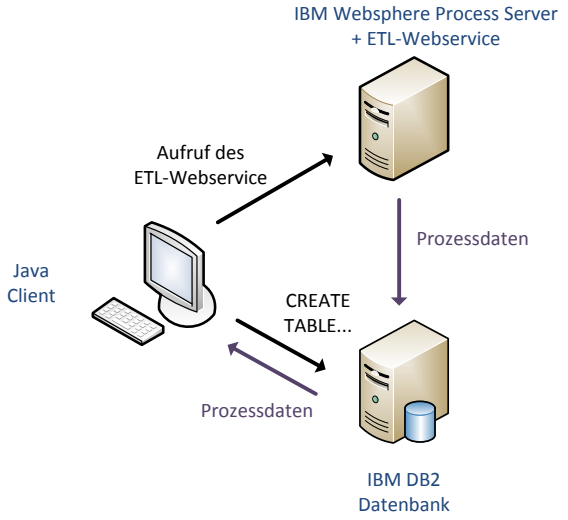


Abbildung 5.3: Daten- und Kontrollfluss bei der Extraktion der Prozessdaten.

Listing 5.1 Beispiel Datei für eine XML Eingabedatei, die vom dBOP Designer geliefert wird.

```
<process bpelName="testprozess">
  <activity dbopName="CheckEligibility" dbopType="Activity">
    <invokeMapping>
      <implementationMapping bpelName="CheckEligibility" dbopName="CheckEligibility" />
      <resourceAcquisitionMapping>
        <mapping bpelName="CheckEligibilityAcqResSalesEmp" dbopName="CheckEligibility" />
      </resourceAcquisitionMapping>
      <resourceReleaseMapping>
        <mapping bpelName="CheckEligibilityRelResSalesEmp" dbopName="CheckEligibility" />
      </resourceReleaseMapping>
    </invokeMapping>
    <inputMapping>
      <mapping processVariable="rental" processAttribute="rentalID"
        wsVariable="" wsPart="checkEligibilityParameters"
        wsExpression="checkEligibility/rental/rentalID"
        mappingType="inputData" dataType="xs:decimal" />
      <mapping processVariable="rental" processAttribute="customerID" dataType="xs:decimal"
          wsVariable="" wsPart="checkEligibilityParameters"
          wsExpression="checkEligibility/rental/customerID" mappingType="inputData"
          ></mapping>
      <mapping processVariable="SalesEmp" processAttribute="resId" dataType="xs:decimal"
          wsVariable="" wsPart="checkEligibilityParameters"
          wsExpression="checkEligibility/rental/employeeID" mappingType="inputResource"
          ></mapping>
    </inputMapping>
    <outputMapping>
      <mapping processVariable="rental" processAttribute="eligible" dataType="xs:string"
          wsVariable="" wsPart="checkEligibilityResult"
          wsExpression="checkEligibilityResponse/rental/eligible" mappingType="outputData" />
    </outputMapping>
    <requiredAnalysis>
      <metrics basicMetrics="true" extendedMetrics="true" />
      <mining>
        <taskAutomationMining enabled="true"/>
      </mining>
    </requiredAnalysis>
    <graphInformation>
      <predecessors>
       ...
      </predecessors>
      <successors>
       ...
      </successors>
      <exclusives />
      <parallels>
        ...
      </parallels>
    </graphInformation>
  </activity>
 ... weitere Aktivitäten mit der gleichen Syntax...
 </process>
```

5.3 Berechnung der Metriken

5.3.1 Ziel

Nachdem die Extraktion beendet ist, liegen alle benötigten Rohdaten vor, um Prozess- und Aktivitätsmetriken berechnen zu können. Aktivitätsmetriken machen eine Aussage über eine einzelne Aktivität oder eine Instanz einer Aktivität. Prozessmetriken beziehen sich hingegen auf den Gesamtworkflow oder einzelne Instanzen davon. Diese Metriken sollen einerseits dem dBOP Designer an einer Schnittstelle zugänglich gemacht werden, andererseits sollen sie in der Data Mining Phase als Zusatzinformationen zur Verfügung stehen.
Der Use Case „Metriken berechnen“ (Tabelle 5.2) beschreibt die funktionalen Anforderungen des Anwendungsfalls „Metriken berechnen“. Zusätzlich zu den in der Aufgabenstellung genannten, machen im Hinblick auf die Metriken folgende funktionale Anforderungen Sinn: Es soll möglich sein, dass eine Metrik das Ergebnis einer anderen Metrik nutzen kann. Für neue Metriken, die pro Aktivitätsinstanz berechnet werden, soll es eine einfache Aggregationsmöglichkeit geben, mit Aufnahme der aggregierten Metrik in die Ausgabe des Werkzeugs.

Name	Berechnung der Metriken
Ziel	Berechnung der Metriken auf Basis der im Extraktionsschritt gewonnenen Prozessdaten
Vorbedingung	Die Extraktion wurde durchgeführt, und die Aktivitätstabellen sind gefüllt.
Nachbedingung	Die Aktivitätstabellen sind um Spalten für die Metriken erweitert und entsprechende Einträge wurden vorgenommen. Die aggregierten Metriken liegen in Form einer XML Datei vor.
Akteure	Anwender, Tool
Normalablauf	1. Der Anwender klickt auf die Option "Metriken Berechnen." 2. Das Tool berechnet die Metriken mit Hilfe aller im Quellcode vermerkten Metrik-Klassen. Das Interface zu den Metrik-Klassen erlaubt die Definition eines neuen Attributs pro Aktivität, und die Berechnung eines Werts des Attributs für jede Aktivitätsinstanz. Es erlaubt außerdem die Definition eines passenden XML Tags und die Berechnung eines entsprechenden aggregierten Wertes für alle Aktivitätsinstanzen. Dieser Wert wird unter Verwendung des Tags in der XML Ausgabedatei protokolliert.

Tabelle 5.2: Use Case: Metriken berechnen.

5.3.2 Entwurf zur Berechnung der Metriken

Da es Metriken gibt, die andere Metriken nutzen, wird ein „MetricRepository“ eingeführt, in dem Metriken beliebige bereits berechnete Ergebnisse in Form von Java Objekten unter einem wohldefinierten, menschenlesbaren Schlüssel zwischenspeichern können.
Die Metrik-Klassen haben alle dieselbe abstrakte Vaterklasse, sodass sie in Form einer Liste abgearbeitet werden können. Dabei definiert der Programmierer anwendungsspezifischer Metriken die Reihenfolge in der die Metriken abgearbeitet werden sollen. Zuerst werden die Prozessmetriken berechnet, da die Aktivitätsmetrik „Frequenz“ z.B. die Prozess- Metrik „InvocationCount“ benötigt. Im zweiten Schritt folgen dann die Aktivitätsmetriken.
Diese Vorgehensweise hat den Nachteil, dass Prozessmetriken keine Ergebnisse von Aktivitätsmetriken nutzen können, jedoch ist dies durch eine einfache Anpassung bei Bedarf leicht zu beheben.
Für jede Aktivitätsmetrik wird zunächst pro Aktivität eine passende Spalte für ihre Werte erstellt (siehe Abbildung 5.4). Die Aktivitätstabellen haben danach bei einer Anzahl von p

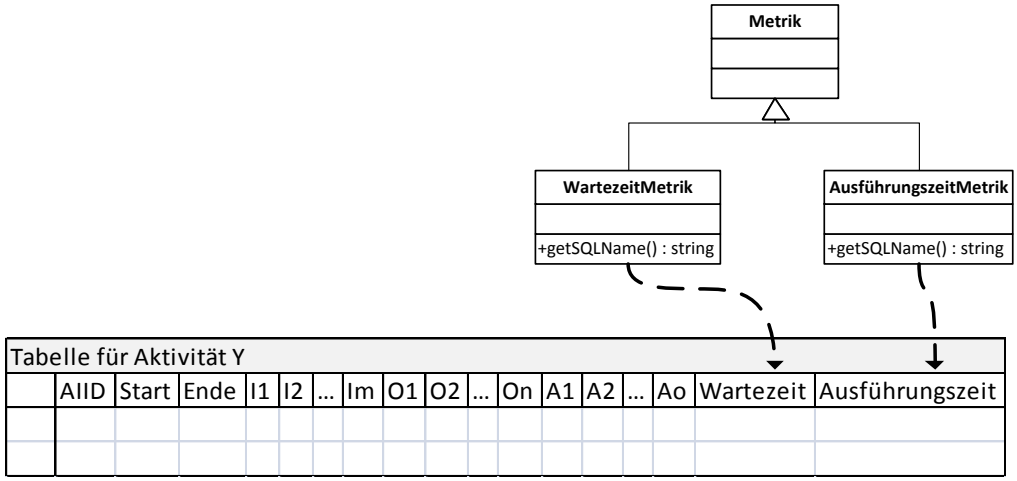


Abbildung 5.4: Erweiterung der Aktivitätstabellen. Oben: Metrik-Klassen die die Funktion getSQLName definieren. Unten: Erweiterte Aktivitätstabelle.

Metriken dieses Schema:
$Activity_X$($AIID$:INTEGER,$Start$:LONG,End:LONG,$Start_{Request}$:LONG,$End_{Request}$:LONG,
$Start_{Release}$:LONG,$End_{Release}$:LONG,$Input_1$:D_{Input_1},...,$Input_m$:D_{Input_m},
$Output_1$:D_{Output_1},...,$Output_n$:D_{Output_n},
$Activated_{Activity_1}$:BOOLEAN,...,$Activated_{Activity_o}$:BOOLEAN,
$Metrik_1$:D_{Metrik_1},...,$Metrik_p$:D_{Metrik_p})
Die Metriken werden nach Erstellung der Spalte einmal pro Aktivitätsinstanz aufgerufen, um die die Spalte zu befüllen. Numerische Metriken wie Durchschnitt, Minimalwert, Ma-

ximalwert und Median, können hinterher automatisch aggregiert werden. Jede Metrik hat danach noch die Möglichkeit, pro Aktivität beliebige weitere Aggregationen durchzuführen, mit freiem Zugriff auf alle Daten.
Am Ende kann die Metrik-Klasse diese Aggregationen in der XML Ausgabe vermerken. Für die XML Ausgabe wird nicht die bereits besprochene DataAccessLib verwendet, da erstens ein schreibender Zugriff auf eine XML Datei erfolgt, und zweitens dieser schreibende Zugriff vom ETL-Webservice nicht benötigt wird. Eine Umsetzung dieser Funktionalität in der DataAccessLib hätte zur Folge, dass sich der ETL-Webservice aufgrund seiner Abhängigkeit von der DataAccessLib im Laufe der Implementierungsphase ständig ändern würde und unter Umständen jedes Mal neu deployt werden müsste.
Listing 5.2 zeigt das XML Schema der Ausgabedatei für die Metriken. Das Schema sieht für die Aktivitäten bisher nur numerische Metriken vor (Typ MetricType).

Listing 5.2 XML Schema für die Ausgabe der Metriken.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.BusinessProcessAnalytics.org/Metrics"
    xmlns:tns="http://www.BusinessProcessAnalytics.org/Metrics"
    elementFormDefault="qualified">

<xs:complexType name="MetricType">
       <xs:sequence minOccurs="0" maxOccurs="1">
       <xs:element name="Avg" type="xs:float"></xs:element>
       <xs:element name="Min" type="xs:float"></xs:element>
       <xs:element name="Max" type="xs:float"></xs:element>
       <xs:element name="Median" type="xs:float"></xs:element>
       </xs:sequence>
<xs:attribute name="MetricName" type="xs:string"></xs:attribute>
</xs:complexType>

<xs:complexType name="ActivityType">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element name="Metric" type="tns:MetricType"/>
</xs:sequence>
<xs:element name="InvocationCount" type="xs:decimal"></xs:element>
<xs:element name="Frequency" type="xs:float"></xs:element>
</xs:sequence>
<xs:attribute name="ActivityName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="ProcessType">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element name="Activity" type="tns:ActivityType"/>
</xs:sequence>
<xs:element name="InvocationCount" type="xs:decimal"></xs:element>
</xs:sequence>
<xs:attribute name="ProcessName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MetricsType">
<xs:sequence>
       <xs:element name="Process" type="tns:ProcessType"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ResultsType">
<xs:sequence>
       <xs:element name="Metrics" type="tns:MetricsType"/>
</xs:sequence>
</xs:complexType>

<xs:element name="Results" type="tns:ResultsType"></xs:element>

</xs:schema>
```

5.4 Konsolidierung

5.4.1 Ziel

Die durch Metriken angereicherten Aktivitäts- bzw. Prozessdaten sollen im 3. Schritt der Datenpipeline 4.1 mit operativen Daten konsolidiert werden. Bei dieser Zusammenführung sollen alle vorgegebenen Matchings, also semantische Äquivalenzen zwischen Prozessdaten und operativen Daten, genutzt werden können, um damit für das Data Mining wichtige operative Daten auszulesen. Matchings können vor der Ausführung des Werkzeugs im sogenannten BIA-Editor [RNB10] definiert werden.
Das Ergebnis der Konsolidierung ist für jeden Data Mining Auftrag genau eine Tabelle, die für jede Aktivitätsinstanz der betrachteten Aktivität eine Zeile mit Informationen enthält. Bei den Informationen handelt es sich um Prozessdaten und operative Daten.

5.4.2 Entwurf

Benutzer-Schnittstelle

Anwendungsfall 5.3 zeigt die funktionalen Anforderungen an den Konsolidierungsteil der graphischen Benutzeroberfläche. Zur Realisierung dieses Anwendungsfalls ist eine einfache, graphische Benutzerschnittstelle vorgesehen (siehe Abbildung 4.6). Sie erlaubt zunächst die Auswahl einer Aktivitätstabelle. Eine Aktivitätstabelle ist für alle Aktivitäten verfügbar, für die in der dBOP-BPEL Mapping Datei, der Data Mining Auftrag Task Automation aktiviert ist.
Wird eine Aktivitätstabelle angeklickt, so erscheinen alle Attribute der Tabelle, inklusive der Metriken in Form einer Liste. Der Benutzer kann dann Matchings auf die operativen Daten aktivieren oder deaktivieren. Die Matchings können mit Klick auf „Details“ geprüft und editiert werden. Der Benutzer kann nach Auswahl der benötigten Matchings die Prozesstabellen mit den operativen Tabellen joinen, indem er „Tabellen joinen.“ anklickt.

Konsolidierungsvorgang

Um die Konsolidierung durchführen zu können, werden Informationen über die Aktivitätstabelle und die Matchings benötigt. Ein Aktivitätstabelle wird beschrieben durch folgende Angaben:

- Name
- Namen der Attribute

Ein Matching definiert diese Angaben:

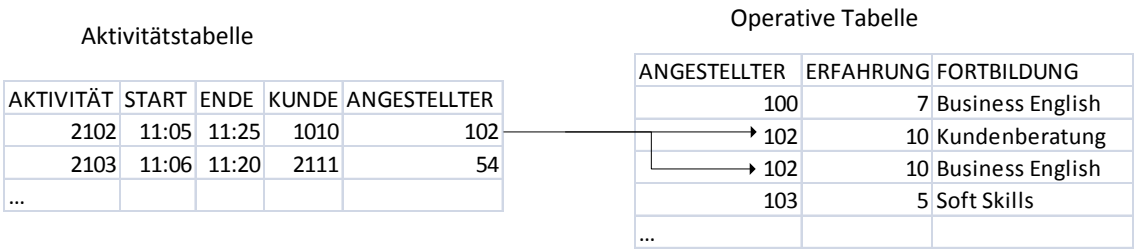
- SQL Statement, um die operativen Daten zu extrahieren.
- Name der Tabelle in die die Extraktion erfolgen soll.

Name	Konsolidierung
Ziel	Überführen aller für das Data Mining relevanten Daten in eine Tabelle.
Vorbedingung	Das Tool wurde mit validen Eingabedaten aufgerufen: Eingabedaten sind die dBOP-BPEL Abbildung und die Matchings. Aktivitätszentrierte Daten sind extrahiert. Operative Daten auf die sich die Matchings beziehen liegen in Form von SQL Tabellen vor.
Nachbedingung	Alle für das Data Mining relevanten Daten liegen in einer einzigen Tabelle vor.
Akteure	Anwender, Tool
Normalablauf	1. Der Anwender wählt die Option "Konsolidieren". 2. Das Tool wechselt in die Ansicht Konsolidieren. Es zeigt alle Aktivitäten, für die der aktuelle Data Mining Auftrag aktiviert ist. 3. Der Anwender wählt eine Aktivität. 4. Das Tool zeigt eine Liste aller Spalten der entsprechenden Aktivitätstabelle. Wenn eine für Spalte ein Matching auf die operativen Daten existiert, zeigt das Tool einen Info-Knopf und eine Checkbox mit der das Matching aktiviert werden kann. 5. Der Anwender drückt den Knopf "Join". 6. Das Tool prüft ob zwischen den Prozessdaten und den operativen Tabellen eine 1:N oder eine M:N Beziehung besteht. Und "verschmelzt"gegebenenfalls n Tupel in 1 Tupel. Anschließend erfolgt ein Join der Prozessdaten mit den operativen Tabellen. 7. Das Tool wechselt in die Ansicht Preprocessing und zeigt die Spalten der konsolidierten Daten.

Tabelle 5.3: Use Case: Konsolidierung.

- Name der Spalte dieser Tabelle, die zu einem Prozess-Attribut korrespondiert.
- Name der Prozess-Variable
- Name des Prozess-Attributes

Die Matchings werden aus einer XML Datei (Beispiel: siehe Listing 5.3) gelesen, die vorher mit dem BIA Editor erstellt werden muss. Das Werkzeug extrahiert zunächst für alle in dieser Datei definierten Matchings die operativen Daten in separate Tabellen die wie im Matching definiert benannt werden.
Wenn der Benutzer seine Auswahl getroffen hat und „Join“ klickt, werden die aktiven Matchingtabellen daraufhin untersucht, ob zu den Prozessdaten eine 1:N Beziehung besteht. Das Kriterium dafür ist die Existenz von Dubletten in der Spalte, die zu dem Prozess-Attribut korrespondiert. Ist dieses Kriterium erfüllt, werden die Spalten der operativen Tabelle „binarisiert“ die für eine Aktivitätsinstanz mehrere Werte aufweisen, sofern es sich



AKTIVITÄT	START	ENDE	KUNDE	ANGESTELLTER
2102	11:05	11:25	1010	102
2103	11:06	11:20	2111	54
...				

ANGESTELLTER	ERFAHRUNG	FORTBILDUNG
100	7	Business English
102	10	Kundenberatung
102	10	Business English
103	5	Soft Skills
...		

Abbildung 5.5: Beispiel für eine 1:N Beziehung zwischen einer Aktivitätstabelle und einer entsprechenden operativen Tabelle. Die Aktivitätsinstanz mit der ID 2102 hat 2 Join-Partner in der operativen Tabelle.

dabei um nominale Spalten handelt. Bei numerischen Spalten erfolgt keine „Binarisierung“. Bei dem Vorgang wird für jeden Wert der Spalte, der in den operativen Daten auftaucht, in der konsolidierten Tabelle eine binäre Spalte angelegt. Diese erhält für ein Tupel genau dann den Wert true, wenn die ursprüngliche Spalte den entsprechenden nominalen Wert enthalten hat.
Die Binarisierung ist erforderlich, da gängige Data Mining Verfahren Trainingsbeispiele mit mehrwertigen Attributen nicht verarbeiten können. Ein Beispiel ist in Abbildung 5.5 zu sehen. Nach der Binarisierung erfolgt ein Join der operativen Tabellen mit der Aktivitätstabelle: Sei $Activity_X$ die Aktivitätstabelle nach der Berechnung der Metriken. Seien Op_1,...,Op_i die operativen Tabellen die durch die Matchings spezifiziert sind, dann wird die Tabelle $Activity_X \bowtie_{ProcessVariable1=OperativeColumn1} Op_1 \bowtie ... \bowtie_{ProcessVariableI=OperativeColumnI} Op_i$ an den Preprocessing Schritt weitergereicht. Der Benutzer gelangt in die Ansicht „Preprocessing“.

Listing 5.3 Beispiel Datei für Matchings.

```
<Matchings xmlns="http://www.BusinessProcessAnalytics.org/Matchings"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.BusinessProcessAnalytics.org/Matchings matchings.xsd">
       <Matching id="1">
              <Process>
              <Variable>rental</Variable>
              <Attribute>customerID</Attribute>
              </Process>
              <Operational>
              <Table>BPACLIENT_CUSTOMER</Table>
              <Col>CUSTOMERID</Col>
              <SQLStatement>SELECT c.CUSTOMERID, c.AGE AS
                  CUSTOMERAGE,c.NUMBERRENTALS,d.AVERAGEINCOME FROM CARRENTAL.CUSTOMER c,
                  CARRENTAL.OCCUPATION d WHERE c.OCCUPATIONID=d.OCCUPATIONID
              </SQLStatement>
              </Operational>
       </Matching>
       <Matching>...
       </Matching>
</Matchings>
```

5.5 Preprocessing

5.5.1 Ziel

Die konsolidierten Daten werden im Preprocessing-Schritt für das Data Mining vorbereitet. Da diese Daten schon in einer einzigen Tabelle vorliegen, fehlt als Voraussetzung für das Data Mining nur noch die Information, ob es sich bei den Spalten um nominale oder numerische Daten handelt.

5.5.2 Entwurf

Für die Umsetzung des Data Mining Schritts wurde die Open Source Data Mining Bibliothek WEKA [HFH+09] eingesetzt. WEKA bietet bereits umfangreiche Preprocessing Filter an. Deshalb integriert das Werkzeug das Preprocessing Panel von WEKA und lädt automatisch die konsolidierten Daten in das Panel. Alle Metriken, die vom SQL-Typ CHAR() oder VARCHAR() sind, werden dann automatisch als nominal interpretiert. Alle SQL-Zahlentypen werden als numerisch interpretiert. Da letzteres manchmal nicht erwünscht ist, kann der Benutzer z.B. für eine Kunden-ID den „NumericToNominal“ Filter von WEKA verwenden. Darüber hinaus besteht die Möglichkeit nicht benötigte Prozess- oder operative Attribute per „Remove“ zu entfernen, und es können nach Bedarf beliebige weitere WEKA Filter angewandt werden.

5.6 Anwendung von Data Mining Verfahren zur Optimierung von Workflows

Abschließend soll ein Szenario spezifiziert werden in dem Data Mining die Optimierung von Workflows sinnvoll unterstützt. Ein passender Data Mining Algorithmus soll beschrieben und prototypisch implementiert werden.

5.6.1 Unterstützung von der Task Automation Best Practice [RLM04] durch Data Mining auf konsolidierten Prozessdaten und operativen Daten

In diesem Kapitel wird ein Szenario beschrieben, in dem Data Mining die Workflow Optimierung unterstützt. In [RLM04] sind Optimierungsmuster für Geschäftsprozesse zu finden. Einige davon lassen sich gut auf die Workflow-Ebene übertragen, um automatisiert Optimierungsvorschläge zu generieren. Als besonders gut dafür geeignet haben sich auf Basis ihrer Beschreibung in [RLM04] die Optimierungsmuster „Task Automation“ und „Triage“ herausgestellt.
Task Automation bedeutet, eine Aufgabe, die sonst ein Sachbearbeiter durchführen müsste, ganz oder teilweise mit Hilfe einer Software zu automatisieren. Eine teilweise Automatisierung könnte zum Beispiel durch Vorbelegung von Webformularen, die der Sachbearbeiter

dann zur Überprüfung erhalten würde, erfolgen. Oder dass unter bestimmten Voraussetzungen eine Software alleine agiert und in schwierigen Fällen ein Sachbearbeiter.
Eine auf das Muster des Task Automation passende Aktivität zeichnet sich also dadurch aus, dass sie vollständig durch einen Sachbearbeiter oder z.B. durch einen Kunden ausgeführt wird, etwa durch Ausfüllen eines Webformulars unter Berücksichtigung bestimmter Angaben. Letztere sind genau die Eingabedaten der Aktivität. Das was in das Formular eingegeben wurde, stellt die Menge der Ausgabedaten dar. Die Wirkung einer passenden Aktivität muss sich auf den Prozess selbst beschränken: Eine Aktivität in der eine Warenauslieferung erfolgt kann nicht automatisiert werden.
Um Task Automation durch Data Mining auf den konsolidierten Prozess- und operativen Daten zu unterstützen, müssen die Ausgabedaten vollständig ermittelt werden. In manchen Fällen kann es auch ausreichend sein, nur die für die zu erwartende Folgeaktivität erforderlichen Ausgabedaten zu ermitteln, inklusive der für die Auswertung der Gateway-Bedingung erforderlichen Ausgabedaten.
Dafür gibt es mehrere Alternativen. Erstens könnte man alle Eingabe- und Ausgabedaten binarisieren und anschließend ein Assoziationsregelverfahren darauf anwenden. Numerische Attribute müssen vor der Binarisierung in Intervalle diskretisiert werden. Wenn beim Assoziationsregel-Mining Regeln mit genügend hohem Support und Confidence-Wert auftauchen, deren Antecedent eine oder mehrere Eingabeattribute darstellt und deren Consequent eine vollständige Aussage über die erforderlichen Ausgabedaten macht, hätte man für einen Teil aller Ausführungen genügend Informationen, um die Aktivität z.B. durch ein Java Programm zu automatisieren.
Die Alternative, die in dieser Diplomarbeit vorgestellt wird, ist, für jede nominale Ausgabevariable einen Entscheidungsbaum-Klassifikator zu erstellen und für jede numerische Variable einen Modellbaum. Diese Alternative hat den Vorteil, dass Entscheidungsbaum-Modelle nicht nur einfach verständlich sind, sondern auch direkt zur Prädiktion verwendet werden können. Passende Assoziationsregeln müssten dagegen erst aus der Ergebnismenge herausgefiltert werden.
Die Modellbäume sind gegenüber normaler (multipler) linearer Regression im Vorteil, da sie die Menge der aller möglichen Objekte hierarchisch aufteilen in Mengen, die in den Trainingsbeispielen eine möglichst geringe Standardabweichung bezüglich des vorherzusagenden Attributs aufweisen. Für jede Menge dieser Aufteilung wird ein passendes lineares Regressionsmodell gespeichert. Gegenüber neuronaler Netzwerke weisen Modellbäume eine bessere Interpretierbarkeit auf.
Listing 5.4 zeigt das XML Schema für die Ausgabe der Modelle. Neben dem serialisierten WEKA Classifier wird auch die Genauigkeit der Modelle ausgegeben. Für alle Entscheidungsbäume wird auch der Java Code ausgegeben.

5.6.2 Entscheidungsbäume und Modellbäume: Die Algorithmen C4.5 und M5P

C4.5

Der von Ross Quinlan vorgeschlagene C4.5 Klassifizierungs-Algorithmus [Qui93] ist ein Entscheidungsbaum-Verfahren, das sich für diskrete Klassen eignet. Es handelt sich dabei

um einen divide-and-conquer Algorithmus der sich das Greedy Prinzip zur Optimierung der Höhe des Entscheidungsbaumes, also dem Resultat einer Ausführung des Trainings-Algorithmus, zu eigen macht. Die Informationen über den C4.5 Algorithmus wurden dem Buch [Qui93] entnommen.

Aufbau des Entscheidungsbaums Um für eine gegebene Trainingsmenge T von Beispielen einen Entscheidungsbaum zu erstellen, wird zunächst für jede Klasse C_i die Häufigkeit $frequ(C_i, T)$ mit der sie in T auftritt bestimmt. Kommt nur eine Klasse C_j in der Trainingsmenge vor, wird ein Blattknoten mit der Entscheidung C_j erzeugt und zurückgegeben. Ist $|T|$ sehr klein, so wird ebenfalls ein Blattknoten zurückgegeben, wobei die mehrheitlich auftretende Klasse die Entscheidung an diesem Blatt darstellt.
Gilt keine der beiden Abbruchbedingungen wird ein neuer Entscheidungsknoten N erstellt und für jedes Attribut A des Datensatzes ermittelt, welchen Informationsgewinn (Gain) eine Aufteilung von T anhand von A einbringt (siehe Abschnitt 5.6.2). Hier können diskrete und stetige Attribute berücksichtigt werden. Das Attribut mit dem höchsten Gain wird zur Auftrennung von T herangezogen. Bei stetigen Attributen wird vor der Auftrennung noch die Trennschwelle bestimmt (siehe Abschnitt 5.6.2).
Anschließend werden die Trainingsteilmengen T_k ($T_k \in T_{Split}$), die durch eine Aufteilung von T entstehen, daraufhin untersucht, ob sie leer sind. Wenn ja, wird dafür ein Blattknoten erzeugt und als Kind des aktuellen Knotens N angehängt, ansonsten erfolgt ein rekursiver Aufruf auf T_k, der dann einen Kindknoten oder Unterbaum für N zurückgibt.
Im Anschluss daran wird der Fehler des aktuellen Knotens N bestimmt und N zurückgegeben. Algorithmus 5.1 auf der nächsten Seite zeigt den groben Ablauf des C4.5 Algorithmus.

Auswahl von Attributtests Der C4.5 kennt 3 Arten von Attributtests:

1. Der Test vergleicht pro Zweig j das diskrete Attribut A_i mit einem festen Wert a_j: $A_i = a_j$.
2. Der Test vergleicht pro Zweig j das diskrete Attribut A_i mit mehreren festen Werten a_x: $A_i = a_j \vee A_i = a_k \vee ....$
3. Der Test hat zwei Zweige und vergleicht ein stetiges Attribut mit einem Schwellwert s : $A_i < s$, $A_i \geq s$

In dieser Diplomarbeit werden die Typen 1 und 3 besprochen. Typ 2 wird nur optional verwendet.
Ziel ist es, einen Attributtest zu finden, der die aktuelle Trainingsmenge so aufspaltet, dass ein möglichst kleiner Entscheidungsbaum entsteht.
C4.5 nutzt, wie einige andere Klassifizierungsalgorithmen (CLS, ID3) das Gain Kriterium als Grundlage für die Attributauswahl. Mit dem Gain Kriterium versucht man auf Basis des Shannon'schen Informationsbegriffs den Informationszuwachs beim Abstieg im Entscheidungsbaum zu maximieren. Ein hoher Informationszuwachs kann z.B. bedeuten: Wenn ein Tupel alle Tests X auf einem Pfad P ($X \in Tests_P$) mit $X = true$ durchläuft, dass es sich dann

Algorithmus 5.1 Pseudocode des C4.5. Der Pseudocode stammt aus [Qui93] und wurde in seiner Darstellung an die Diplomarbeit angepasst.

```
procedure FormTree(T)
    ComputeClassFrequency(T)
    if OneClass or FewCases then
        return a leaf
    end if
    create a decision node N
    for all A ∈ Attributes do
        ComputeGain(A)
    end for
    N.test ← AttributeWithBestGain
    if N.test is continuous then
        find Treshold
    end if
    for all T_i ∈ T_Split do
        if T_i is Empty then
            N.newChild ← leaf
        else
            N.newChild ← FormTree(T_i)
        end if
    end for
    ComputeErrors(N)
    return N
end procedure
```

bei der Klassenzuweisung um eine ganz bestimmte Klasse C_i handeln muss, während ohne die Aussagen, die durch die Tests gemacht werden, eine solche Zuordnung nicht möglich ist und es stattdessen viele mögliche Klassenzuweisungen für das Tupel gibt, die alle gleich wahrscheinlich sind.
Der Informationszuwachs ist dabei definiert als die Differenz zwischen Informationsgehalt der Trainingsmenge vor der Aufteilung und dem mittleren Informationsgehalt der Trainingsteilmengen die man durch die Aufteilung erhält:

$$gain = info(T) - info_X(T)$$

Der mittlere Informationsgehalt der Aufteilung der Trainingsmenge T anhand von Test X ist:

$$info_X(T) = -\sum_{i=1}^{s} \frac{|T_i|}{|T|} \times info(|T_i|)$$

Dabei ist s die Anzahl der möglichen Ergebnisse von Test X. Die Entropie $info(T)$ einer gegebene Trainingsmenge wird berechnet durch:

$$info(T) = -\sum_{j=1}^{NClass} \frac{frequ(C_j, T)}{|T|} \times log_2(\frac{frequ(C_j, T)}{|T|})$$

Dabei ist NClass die Anzahl der Klassen. Konkrete Trainingsmengen werden also als Nachrichten im Sinne der Informationstheorie [Wero9] angesehen. Die Tupel, die die Trainingsmengen enthalten, werden als Zeichen interpretiert. Die Zeichen haben dann für Tupel $t_i \in T$ mit Klassenzuweisung C_i die reduzierte Form „C_i“. Aus den Wahrscheinlichkeiten dieser Zeichen ergibt sich deren Informationsgehalt. Aus den Informationsgehalten dieser j Zeichen ergibt sich der mittlere Informationsgehalt der Nachricht. Es wird davon ausgegangen, dass genügend Trainingsdaten vorliegen. Deshalb werden die statistischen Eigenschaften der Nachrichten mit denen der Quelle gleichgesetzt. Somit kann der Entscheidungsbaum, der als Modell die Quelle repräsentieren soll, zur Prädiktion nicht klassifizierter Tupel herangezogen werden.
Das Gain Kriterium liefert deshalb brauchbare Entscheidungsbäume, da insbesondere Attributtests X, die homogene Trainingsteilmengen T_i erzeugen, begünstigt werden, denn der Gain ist maximal, wenn die Entropie von allen T_i 0 ist. Und die Entropie einer Trainingsteilmenge T_i ist dann 0, wenn sie homogen ist, d.h. nur Tupel einer einzigen Klasse C_i enthält. Jedoch wird das Gain Kriterium nicht in Reinform angewandt, da es dazu neigt, Attribute mit besonders vielen verschiedenen möglichen Werten zu bevorzugen. Soll also z.B. eine Kundendatenbank untersucht werden, besteht die Gefahr, dass die Kunden ID als erstes zur Auftrennung der Trainingsmenge verwendet wird, was zu einem extrem breiten Entscheidungsbaum, ohne Nutzen führt. Deshalb wird bei der Maximierung des Gain, derselbe noch durch einen „Penalty Term“ geteilt, der Attribute mit extrem vielen möglichen Werten bestraft. Das Ergebnis ist die „Gain Ratio“

$$splitInfo(X) = -\sum_{i=1}^{s} \frac{|T_i|}{|T|} \times log_2(\frac{|T_i|}{|T|})$$

Dabei ist s wieder die Anzahl verschiedener möglicher Ergebnisse von Test X.

Behandlung fehlender Werte Wenn bei einem Trainingsbeispiel der Attributwert fehlt, anhand dessen eine Auftrennung erfolgen soll, so gelangt das Trainingsbeispiel in alle Unterbäume des aktuellen Knotens und wird mit der Werteverteilung des Attributs an diesem Knoten gewichtet [MAT10].

Trennschwellen für stetige Attribute finden Um den Informationsgewinn eines stetigen Attributes zu berechnen, muss dessen Trennschwelle s berechnet werden. Eine Trennschwelle s wird aus einer Trainingsmenge T folgendermaßen berechnet: Erst wird T nach den Werten des stetigen Attributes sortiert. T dann wird für alle Mittelwerte aus je 2 aufeinanderfolgenden Tupeln $t \in T$ in zwei Trainingsteilmengen T_1 und T_2 aufgetrennt. Anschließend wird wie gehabt die Gain Ratio für diese Trennstelle ermittelt. Die Trennstelle s_i mit der höchsten

Gain Ratio wird nicht direkt für das Attribut verwendet, sondern es wird die gesamte Trainingsmenge nach einem Wert s_i' für das aktuelle Attribut durchsucht, der möglichst nahe an s_i herankommt. s_i' ist dann die Trennschwelle des stetigen Attributes.

Pruning Beim Pruning wird ein Entscheidungsbaum gekürzt, um die Genauigkeit auf noch nicht gesehenen Tupeln zu verbessern. Dabei verringert sich die Genauigkeit des Entscheidungsbaumes für die Trainingsdaten automatisch.
Um nun testen zu können, ob es besser ist, den gegebenen Entscheidungsbaum an einem Knoten zu kürzen oder nicht, werden Trainingstupel benötigt, die nicht in den Trainingsdaten enthalten sind, die für den Aufbau des Entscheidungsbaumes verwendet wurden. Um die verfügbaren Trainingsdaten möglichst gut zu nutzen, folgt der C4.5 einem cross-validation Ansatz.
D.h. es wird nicht nur 1 Entscheidungsbaum generiert, sondern n Entscheidungsbäume, wobei jeweils 1 von n Stücken der Trainingsmenge, nur für das Pruning verwendet werden. Am Ende wird der beste Entscheidungsbaum als Ergebnis zurückgegeben.
Das Pruning läuft so ab, dass unten beginnend alle nicht-Blätter des Baumes untersucht werden, ob sie gekürzt werden sollten oder nicht, was mit einer Heuristik entschieden wird.

M5P

Der M5P Algorithmus ist ein Entscheidungsbauminduktion-Verfahren für Modellbäume. Dieses Kapitel stellt den Algorithmus vor und orientiert sich dabei stark an [WW97].
Modellbäume sind Entscheidungsbäume die sowohl diskrete als auch stetige Attribute verarbeiten können und stetige Klassenzuweisungen vornehmen können. Das wird erreicht indem ein normaler Entscheidungsbaum an den Blättern durch lineare Regressionsmodelle erweitert wird. Es ist eine Variante von Quinlans M5 Algorithmus, die 1996 von Yong Wang und Ian H. Witten vorgeschlagen wurde und besser dokumentiert ist als der Originalalgorithmus. M5P wurde außerdem für das Open Source Data Mining Tool Weka [HFH+09] implementiert. Deshalb eignet es sich gut für die Einbindung in das Werkzeug. Wie der M5 Algorithmus von Quinlan basiert M5P auf dem CART (Classification and Regression Tree) Algorithmus.
Der Algorithmus nimmt als Eingabe eine Tabelle mit Trainingstupeln entgegen. Für jedes Trainingstupel enthält die Tabelle einen numerischen Wert, den es mit dem Modellbaum, der Ausgabe des Algorithmus, später für neue Tupel vorherzusagen gilt. Das ist der Wert des sogenannten Klassenattributes der Tabelle. Die anderen Attribute können ebenfalls numerischen Typs sein oder nominal. Wenn nominale Attribute in der Trainingstabelle auftauchen, werden sie jedoch umkodiert:

Preprocessing Für jeden der n möglichen nominalen Werte des Attributes werden die Tupel selektiert, die diesen Wert aufweisen und es wird deren Klassenzuweisung gemittelt. Anschließend werden die nominalen Werte nach ihren Mittelwerten sortiert. Es werden dann

$n-1$ neue binäre Spalten angelegt. Das i-te binäre Attribut eines Tupels hat dann den Wert 0, wenn das nominale Attribut des Tupels in der Ordnung an einer Stelle $j \leq i$ angeordnet ist, andernfalls hat das i-te Attribut den Wert 1.

Nominales Attribut 1	Klassenzuweisung (Mittelwert)	Codierung
$Wert_1$	1.1	00
$Wert_2$	1.5	01
$Wert_3$	10.6	11

Tabelle 5.4: Beispiel Vorverarbeitung : Sortierung der nominalen Werte bezüglich Klassenmittelwert und Neukodierung der nominalen Werte

Aufbau des vorläufigen Modellbaumes Der Algorithmus arbeitet nach dieser Vorverarbeitung auf der Trainingsmenge mit einer „Divivde and Conquer“ Strategie, dabei wird die Trainingsmenge T rekursiv aufgespaltet. Sei am Anfang T die gesamte Trainingsmenge. Der Algorithmus sucht ein Attribut und einen Test auf diesem Attribut, sodass die Mengen T_1 und T_2, die bei einer Auftrennung anhand dieses Tests entstehen, eine möglichst geringe Standardabweichung aufweisen, was das Klassenattribut betrifft. Die Standardabweichung wird dabei als Fehlermaß angesehen, das es durch eine richtige Auswahl von Tests zu minimieren gilt. Die Minimierung der Standardabweichung muss für alle Mengen T_i die bei der Auftrennung von T entstehen, minimiert werden. Dafür wird die Fehlerreduktion als Differenz zwischen Standardabweichung von T und mittlerer Standardabweichung in den Mengen T_1 und T_2 berechnet:

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

Dieser Wert wird bei nominalen Attributen mit einem Faktor

$$\beta = e^{7 \times \frac{2-k}{n}}$$

korrigiert, um zu verhindern, dass Attribute mit sehr vielen verschiedenen Werten zu gut bewertet werden, etwa ID-Attribute, die bei jedem Tupel einen eindeutigen Wert haben und somit keinen Informationswert haben. Dabei ist n die Anzahl der Trainingsbeispiele und k die Anzahl verschiedener Werte des nominalen Attributes. β ist maximal 1 und fällt mit k exponentiell ab. Der Attributtest mit der maximalen Fehlerreduktion wird dann zur 1. Auftrennung am Wurzelknoten des Modellbaums verwendet. Bei stetigen Attributen wird zur Bestimmung der Trennschwelle genauso wie bei C4.5 (siehe Abschnitt C4.5 in 5.6.2) vorgegangen, nur dass s_i direkt als Trennschwelle verwendet wird und somit auf eine globale Suche einer guten Trennschwelle verzichtet wird.
Anschließend erfolgt eine Auftrennung in die Mengen T_i. Dann wird für diese Mengen derselbe Algorithmus rekursiv ausgeführt. Die Rekursion endet mit einem Blattknoten, sobald ein Minimalwert an Trainingsbeispielen erreicht wird, oder die Standardabweichung der Klassenzuweisung klein genug ist.

Es gibt noch einen weiteren Korrekturfaktor, der die Fehlerreduktion bei vielen fehlenden Werten eines Attributes verringert: $\frac{m}{|T|}$. Das ist der Anteil der Tupel, die den Knoten erreichen, bei denen der Wert vorliegt.

Behandlung fehlender Werte Fehlt beim Training des Modellbaums bei Tupeln der Wert eines Attributes, so werden erst alle Tupel genutzt, bei denen der Wert nicht fehlt. Es wird die beste Trennschwelle durch Sortieren und Berechnung der SDRs bestimmt. Anschließend werden die Trainingsbeispiele in die Mengen L und R aufgetrennt, je nachdem ob die Trennschwelle unter- (Menge L) oder überschritten (Menge R) wird.
Dann werden bei nominalen Attributen die mittleren Klassenwerte $\overline{L}$ und $\overline{R}$ der Tupel in L und von R ermittelt. Trainingsbeispiele mit fehlenden Werten werden dann entweder in L oder R eingeordnet je nachdem ob ihr Klassenwert näher an $\overline{L}$ oder an $\overline{R}$ herankommt. Bei stetigen Attributen wird genauso vorgegangen jedoch kann hier $\overline{L}$ und $\overline{R}$ einfacher berechnet werden.

Pruning Nachdem der vorläufige Modellbaum aufgebaut ist, erfolgt eine Kürzung („Pruning“). An jedem Knoten des Baumes wird die mittlere Abweichung der Trainingsbeispiele von der vorhergesagten numerischen Klassenzuweisung berechnet, und mit einem Korrekturfaktor multipliziert, der notwendig ist, da der Fehler des Regressions-Modells bei unbekannten Tupeln größer sein wird.

$$\overline{error_{Tree}}' = \overline{error_{Tree}} * \frac{n+\nu}{n-\nu}$$

Dabei ist n die Anzahl der Trainingsbeispiele die den Knoten erreichen und ν ist die Anzahl der Parameter des Regressionsmodells des Knotens. Außerdem wird ein lineares Regressionsmodell für die Trainingsbeispiele erstellt, das genau die Attribute als Parameter verwendet, die auch in Entscheidungsknoten des Teilbaums vorkommen, dessen Wurzel der aktuelle Knoten ist. Anschließend wird für die zum aktuellen Knoten gehörenden Trainingsbeispiele der mittlere Regressions-Fehler $\overline{error_{Regression}}$ ermittelt. Gilt $\overline{error_{Regression}} < \overline{error_{Tree}}'$, so wird der aktuelle Knoten zu einem Blattknoten.

Prädiktion Wenn der Modell Baum aufgebaut ist, kann das Klassenattribut für neue Tupel geschätzt werden. M5P geht dafür, wie auch der Vorgänger M5, so vor, dass der Entscheidungsbaum bis zu einem Blattknoten durchlaufen wird und anschließend entlang dieses Pfades das Klassenattribut durch Filterung berechnet wird. Dazu wird an jedem Knoten der Schätzwert des Regressionsmodells q des Knotens linear kombiniert mit dem Schätzwert p des Sohnknotens.

$$\frac{np+kq}{n+k}$$

Der Koeffizient n für p ist die Anzahl der Trainingsbeispiele, die den Sohnknoten erreichen, der Koeffizient k für q ist ein Parameter, der frei wählbar ist. Desto größer er ist, desto mehr

gehen auf dem Pfad höher gelegene Regressionsmodelle in die Prädiktion ein. Der Wert der Wurzel entspricht schließlich der Prädiktion. Fehlt bei einem Tupel, dessen Klassenwert vorhergesagt werden soll, ein Attributwert, so wird der Mittelwert des Attributes aller Trainingsbeispiele, die den entsprechenden Knoten im Modellbaum erreichen, dafür eingesetzt [WW97].

Listing 5.4 XML Schema für die Ausgabe der Data Mining Ergebnisse.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.BusinessProcessAnalytics.org/DataMining"
    xmlns:tns="http://www.BusinessProcessAnalytics.org/DataMining"
    elementFormDefault="qualified">

<xs:complexType name="ClassifierType">
<xs:sequence minOccurs="1" maxOccurs="1"><xs:element name="ModelObject"
    type="xs:base64Binary"/><xs:element name="JavaCode" type="xs:string"/></xs:sequence>
<xs:attribute name="CorrectlyClassifiedInstances" type="xs:float"/>
<xs:attribute name="IncorrectlyClassifiedInstances" type="xs:float"/>
<xs:attribute name="KappaStatistic" type="xs:float"/>
<xs:attribute name="MeanAbsoluteError" type="xs:float"/>
...
<xs:attribute name="TotalNumberOfInstances" type="xs:float"/>
<xs:attribute name="dBOPVariableName" type="xs:string"/>
<xs:attribute name="dBOPAttributeName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="RegressionTreeType">
<xs:sequence minOccurs="1" maxOccurs="1"><xs:element name="ModelObject"
    type="xs:base64Binary"/></xs:sequence>
<xs:attribute name="CorrelationCoefficient" type="xs:float"/>
<xs:attribute name="MeanAbsoluteError" type="xs:float"/>
....
<xs:attribute name="dBOPVariableName" type="xs:string"/>
<xs:attribute name="dBOPAttributeName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="ActivityType">
<xs:sequence minOccurs="1" maxOccurs="unbounded">
<xs:choice>
<xs:element name="DecisionTree" type="tns:ClassifierType"></xs:element>
<xs:element name="ModelTree" type="tns:RegressionTreeType"></xs:element>
</xs:choice>
</xs:sequence>
<xs:attribute name="ActivityName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="ProcessType">
<xs:sequence minOccurs="0" maxOccurs="unbounded">
       <xs:element name="Activity" type="tns:ActivityType"/>
</xs:sequence>
<xs:attribute name="ProcessName" type="xs:string"/>
</xs:complexType>

...

<xs:element name="Results" type="tns:ResultsType"></xs:element>

</xs:schema>
```

5.7 Entwurf des Classifier-Webservice

In diesem Abschnitt wird die allgemeine Struktur eines Webservice vorgestellt, der die Modelle, die das Werkzeug ausgibt, verwenden kann, um nach der Task Automation Best Practice [RLM04] eine manuelle Aktivität zu automatisieren.
Ein solcher Webservice kann als Plain Old Java Object (POJO) implementiert werden, in das die WEKA Bibliothek importiert wird. Für den Webservice kann das gleiche WSDL-File verwendet werden, das auch die entsprechende manuelle Aktivität nutzt. Der Webservice muss die Datenpipeline (siehe Abbildung 4.1), ausgenommen der Schritte „Berechnung der Metriken“ und „Data Mining“, für ein einzelne Aktivitätsinstanzen ausführen können.
Für jedes Modell sind somit folgende Schritte auszuführen:

- Extraktion, d.h. die aktuellen Eingabewerte der Aktivität müssen als Parameter entgegengenommen werden. Diese Parameter müssen mit Hilfe der XML Datei mit den Mappings auf ihre dBOP-Pendants abgebildet werden. Dazu muss eine WEKA Instance erstellt werden. Der Datensatz der Instance wird auf den Trainingsdatensatz des Modells gesetzt. Das Klassenattribut des Datensatzes wird mit setClass() gesetzt. In die Instance werden die Eingabewerte unter ihrem SQL-Spaltennamen eingetragen. Dieser Name ergibt sich durch Konkatenation ein Ein-/Ausgabepräfix („IN“,“OUT“) mit dem dBOP-Variablenname und dem dBOP-Attributname. Z.B. ist „IN_RENTAL_ELIGIBLE“ der SQL-Spaltenname für einen Eingabeparameter mit Variablenname „Rental“ und Attributname „Eligible“ (siehe Konstruktor der Klasse AVPair der DataAccessLib).
- Konsolidierung, d.h. die operativen Daten der Matchings, die beim Data Mining aktiviert waren, müssen (z.B. als View) verfügbar sein. Die im Extraktionsschritt gewonnenen Eingabeparameter müssen gematcht werden und mit den operativen Daten angereichert werden. Hier wird, wie bei der Erstellung der Modelle, wieder die BIA Matching Datei verwendet.
- „Binarisierung“ (siehe Kapitel 5.4.2 Abschnitt Konsolidierungsvorgang), d.h. nominale Attribute, die bei der Erstellung der Data Mining Modelle auf binäre Spalten abgebildet wurden, muss auch der Webservice entsprechend umwandeln.
- Preprocessing, d.h. alle Schritte, die als Preprocessing vor der Modellerstellung stattgefunden haben, müssen für die aktuellen Prozessdaten nachvollzogen werden. Das dazu notwendige Protokoll des Preprocessings kann aus der Arff Datei des Trainingsdatensatzes ausgelesen werden.
- Anwendung des Modells (Entscheidungsbaum oder Modellbaum) zur Bestimmung der Ausgabe der Aktivität. Dazu muss zunächst der Classifier geladen werden (siehe Kapitel 6.7). Dann kann dessen Methode classifyInstance verwendet werden, um die WEKA Instance auf den Ausgabewert der Aktivität abzubilden.

Wenn alle Ausgabewerte ermittelt wurden, müssen sie den entsprechenden BPEL-Variablen zugeordnet werden und von dem Webservice zurückgegeben werden.

6 Implementierung des Werkzeugs

In diesem Kapitel wird die Implementierung des Werkzeugs mit Java in derselben Struktur wie im Entwurfskapitel besprochen. Es wird zunächst wieder auf die graphische Benutzeroberfläche eingegangen, dann folgen wieder in der gleichen Reihenfolge die einzelnen Filter der Datenpipeline.

6.1 Implementierung der graphischen Benutzerschnittstelle

Der im Entwurfskapitel 5.1 vorgestellte grundsätzliche Aufbau des Programmfensters mit Pipeline-Visualisierung, Hilfetext, Protokoll und Toolbar wurde in Form einer Klasse DataMiningJobPanel umgesetzt. Von dieser Klasse erbt das TaskAutomationPanel, also die Ansicht die vom Anwender zur Prozessanalyse verwendet werden kann, um Entscheidungsbaum-Klassifikatoren zur Automatisierung einzelner Aktivitäten zu gewinnen (siehe Kapitel 5.6.1). Neue Data Mining Typen können je nach Ausprägung von TaskAutomationPanel oder DataMiningJobPanel erben. Wenn sich das Mining auf Aktivitäten bezieht, bietet sich eine Ableitung von TaskAutomationPanel an.

6.1.1 Die Pipeline-Visualisierung

Die Pipeline-Visualisierung ist intern repräsentiert als eine Array Liste vom Typ ArrayList<PicLabel>, wobei ein PicLabel ein JLabel ist, das ein Symbol für eine Pipeline-Stufe anzeigt und einen Zustand hat. Je nach Zustand (aktiviert,deaktiviert,beendet,fehlgeschlagen) wird ein passendes Symbol angezeigt.

6.1.2 Protokoll und Hilfetextfenster

Das Protokoll und das Hilfetextfenster (Klasse RecordPanel) sind 2 JTextAreas, in die über die Methoden addHelpHint() und addDoneHint() Nachrichten an den Benutzer eingefügt werden können. addHelpHint() ersetzt den gesamten Text des Textfelds durch den Hilfetext, während addDoneHint() einen Protokolleintrag einfügt. Abbildung 6.1 zeigt die Klassen der Bedienungsoberfläche.

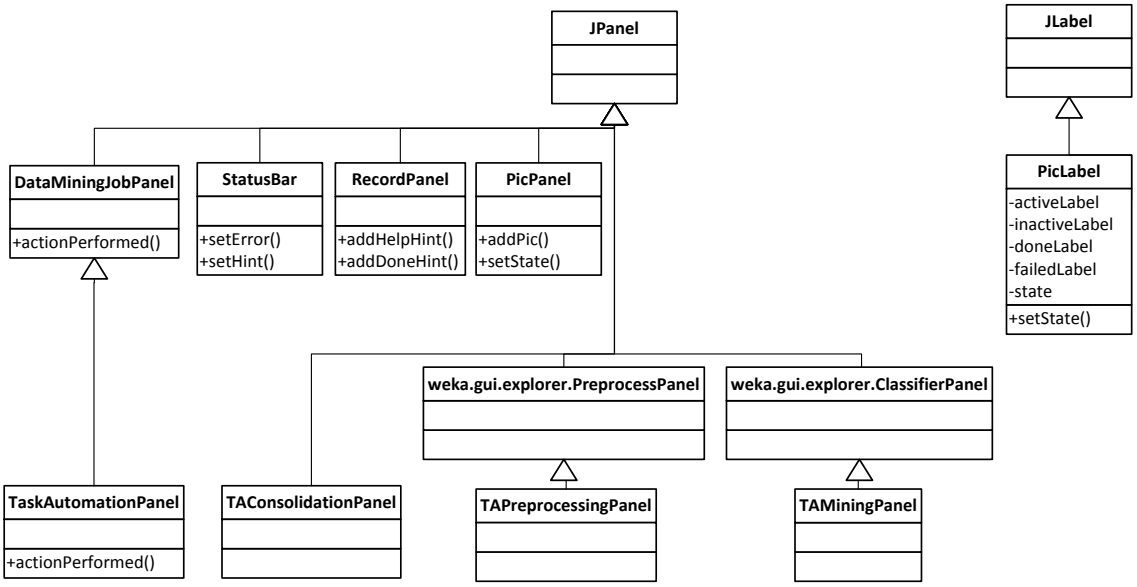


Abbildung 6.1: Klassendiagramm der GUI Klassen.

6.1.3 DEA

Der im Entwurf angesprochene DEA (siehe Abbildung 5.1) wurde als eigene Klasse implementiert, die die Zustände und Eingabezeichen (Benutzeraktionen) in Form von Aufzählungsvariablen (Enumeratoren) speichert. Die möglichen Zustandsübergänge samt Quell- und Zielzuständen werden in einer ArrayList gespeichert.
Immer wenn der Benutzer eine Aktion, z.B. den Export der Data Mining Ergebnisse, anstößt wird zuerst mit der Methode isallowed() des DEA Objekts geprüft, ob die Aktion im aktuellen Zustand überhaupt ausführbar ist.
Wenn ja, dann kann die ActionListener Methode des entsprechenden Buttons die Aktion ausführen. Dann ruft sie die Methode doDEAAction(Aktion) des DEA Objekts auf. Dadurch wird der Automat in den entsprechenden Zielzustand überführt.
Wenn die Aktion nicht ausführbar ist, so gibt isallowed() false zurück und der ActionListener kann eine entsprechende Fehlermeldung ausgeben. Die Abbildung 6.2 zeigt die beteiligten Klassen.

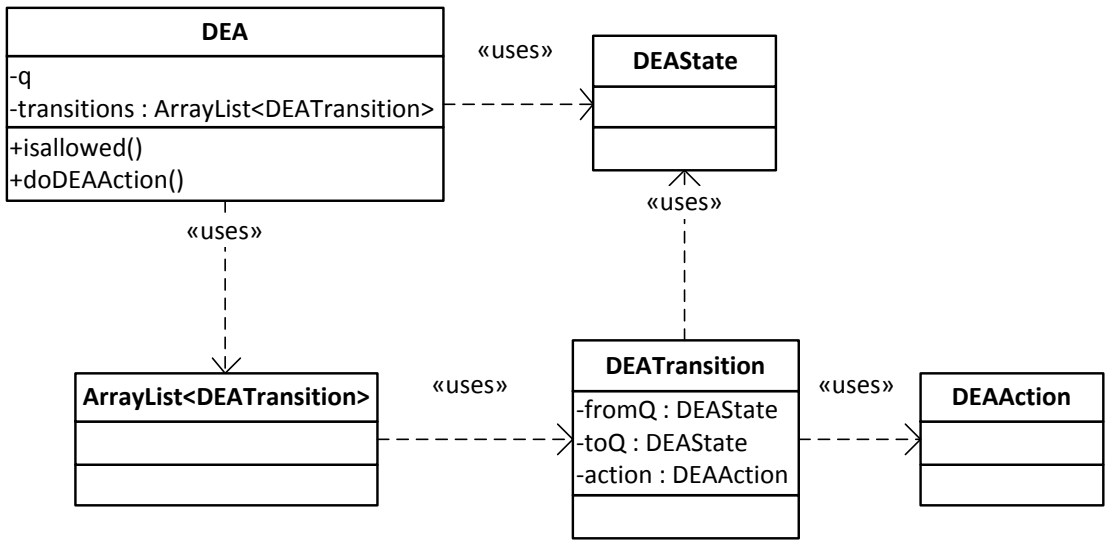


Abbildung 6.2: Klassendiagramm der DEA Klassen.

6.2 Implementierung der Extraktion

Die Extraktion wird in einem normalen Java Client vom Benutzer angestoßen. Daraufhin erstellt der Java Client für jede Aktivität, die in der Eingabedatei des dBOP Designers vermerkt ist eine Datenbank-Tabelle mit einer Spalte pro zu protokollierendem Prozessattribut und einer Spalte pro Aktivität die noch folgt für die Prozessflussinformationen. Diese Tabellen dienen anschließend als Schnittstelle zu dem oben erwähnten Webservice, der nun aufgerufen wird und die Tabellen mit den Audit-Daten füllt.
Die XML Datei mit den Eingaben des dBOP Designers wird als ein Argument binär in einer CDATA Sektion an den Webservice übergeben.
Der Zugriff auf die Datenbank und auf die XML Daten erfolgt im Hauptprogramm und im Webservice über die gleichen Klassen (BPAXMLModule und BPADBModule). Abbildung 6.3 zeigt das entsprechende Klassendiagramm.
Der Webservice führt nach seinem Aufruf folgenden Algorithmus aus: Er übergibt einen SQL Befehl an die Methode query() der Business Flow Manager API, um alle Prozessinstanzen des zu analysierenden Prozesses in einer QueryResultSet festzuhalten. Für jede Instanz werden

Listing 6.1 Beispiel für die Verwendung der Business Flow Manager API zum Auslesen von Auditdaten.

```
public void readAuditData() {
QueryResultSet processes = bfm.query( "DISTINCT PROCESS_INSTANCE.PIID",
    "PROCESS_TEMPLATE.NAME='Testprozess'", "", null, 500, null);
while (processes.next()) {
   PIID piid = (PIID)processes.getOID(1);
   QueryResultSet activities = bfm.getAllActivities(piid, null);
   while(activities.next()){
    AIID activityID = (AIID)activities.getOID(1);
    ActivityInstanceData activity = bfm.getActivityInstance(activityID);
    System.out.println("Name: " + activity.getName());
    System.out.println("Start: " + activity.getActivationTime());
    System.out.println("Ende: " + activity.getCompletionTime());
    int[] allowedActions = activity.getAvailableActions();
     //Hat die Aktivität Ein/Ausgabenachrichten ?
    for (int i : allowedActions){
      if (i == ActivityInstanceActions.GETINPUTMESSAGE){
        cangetinput = true;}
      if (i == ActivityInstanceActions.GETOUTPUTMESSAGE){
       cangetoutput = true;}
    }
    if (cangetinput){
      msgWrapper = bfm.getInputMessage(activityID);
      DataObject inputMessage = (DataObject)msgWrapper.getObject();
      readDataObject(inputMessage,activity.getName());
    }
    ...
  }
}

public void readDataObject(DataObject data, String wsExpression) {
List properties = data.getInstanceProperties();
for(int i = 0; i < properties.size(); i++)
{
commonj.sdo.Property property_i = (commonj.sdo.Property)properties.get(i);
if (property_i.isContainment()) { //kein Blattknoten
  readInput(data.getDataObject(property_i,wsExpression + "/" + property_i.getName());
}
else { //Blattknoten
 System.out.println("Attribut gefunden: " + wsExpression + "/" + property_i.getName());
 System.out.println("Wert von Attribut: " + d.get(property_i).toString());
}
}
```

dann alle Aktivitätsinstanzen abgerufen. Für diese können nun die gewünschten Attribute wie Startzeitpunkt und Endzeitpunkt sowie die Eingabe und Ausgabedaten abgerufen werden. Dazu traversiert der Algorithmus den vom Business Flow Manager zurückgegebenen Baum vom Typ DataObject, der die BPEL Attribute der Eingabe- bzw. der Ausgabedaten enthält (Schema: siehe Methode readDataObject() in Listing 6.1). Immer wenn er auf einen Blattknoten trifft, also z.B. einen String-Wert, wird dieser in eine Liste vom Typ PrintableAVList eingefügt. PrintableAVListen sind einfach verkettete Listen, die pro Knoten ein

Attribut-Wert-Paar enthalten und sich gut zur Verwendung in SQL Statements eignen. Zum Beispiel genügt ein Aufruf von PrintableAVList.getCommaSeparatedValueList() um alle Prozessattribute getrennt durch Kommas in einer Liste als String auszugeben.
Zum Einfügen in die PrintableAVList wird zunächst aus der, an den Webservice übergebenen, XML Datei der Name der entsprechenden Prozessvariable und des Prozessattributs ausgelesen und daraus der Name der entsprechenden Tabellenspalte abgeleitet. Dazu wird dieselbe Bibliothek und Methode verwendet, die auch im Hauptprogramm beim Erstellen der Tabelle verwendet wird, um die Spalte zu benennen.
Listing 6.1 zeigt beispielhaft die Verwendung der Business Flow Manager API, ohne die Details der Abbildung von BPEL auf dBOP. Um den Prozessfluss für jede Prozessinstanz

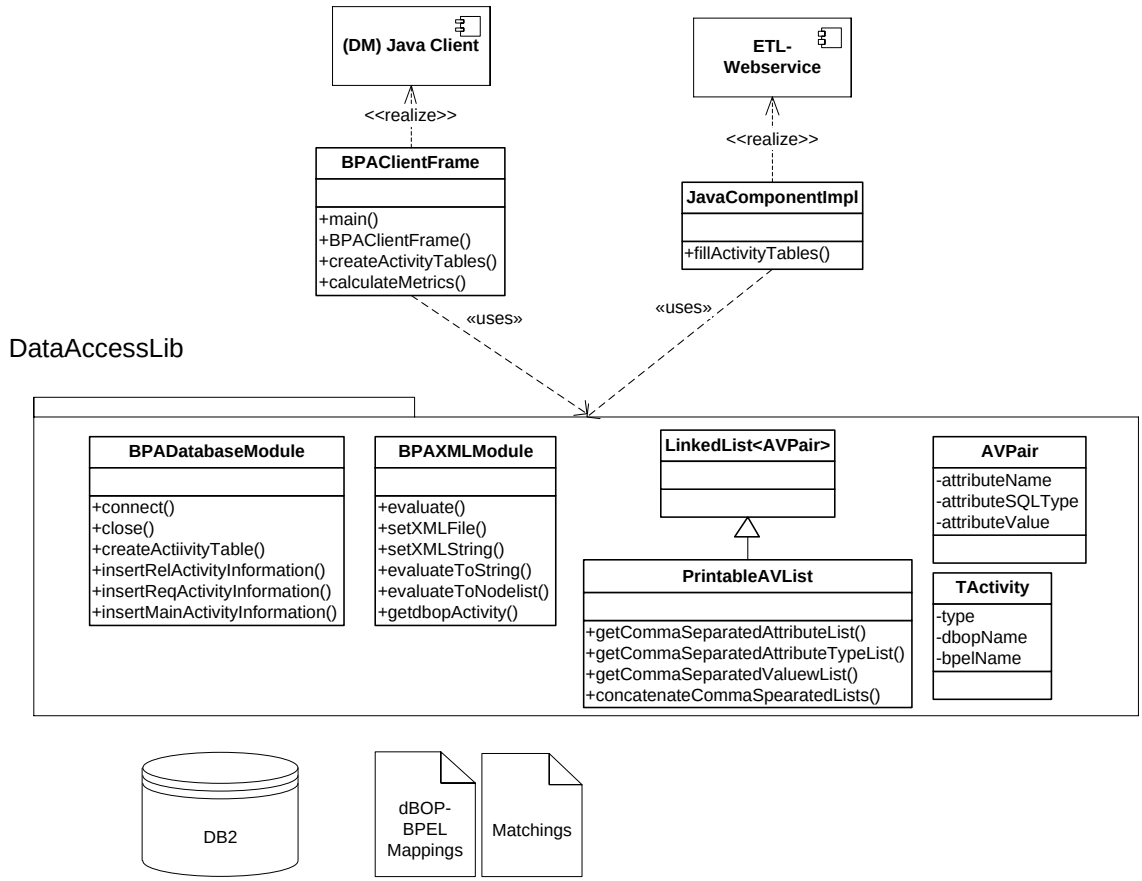


Abbildung 6.3: Klassendiagramm Extraktion

zu protokollieren gibt es eine Klasse „ActivityActivationLog“ die sich beim Auslesen jeder Aktivität aus der BFM API merkt, dass sie in der aktuellen Prozessinstanz ausgeführt wurde.

Ist das Auslesen der Prozessinstanz beendet, wird für jede Aktivität der Prozessfluss als passender Binärvektor (0 bedeutet die Aktivität wurde nicht ausgeführt, 1 heißt sie wurde ausgeführt) aus dem „ActivityActivationLog“ ausgelesen und in die Aktivitätstabelle eingefügt, wodurch die Anzahl der Datenbankzugriffe für das Prozessflussprotokoll genau der Anzahl der Aktivitäten entspricht.
Alternativ zu der oben beschriebenen Vorgehensweise hätte der Transformationsschritt, also die Übersetzung von BPEL Daten in dBOP Daten auch erst auf dem Client erfolgen können. Dies hätte jedoch zur Folge dass unter Umständen die Datenbank unübersichtlich würde, da für jede BPEL Aktivität eine eigene Tabelle vorgesehen werden müsste und nach dem Extraktionsschritt durch den Webservice der Client die Transformation in entsprechende dBOP Tabellen durchführen müsste. Folglich wird bei der implementierten Vorgehensweise eine Zwischenspeicherung eingespart. Der Client kann nach dem Webservice Aufruf direkt mit der Berechnung der Metriken starten (siehe nächster Abschnitt).

6.3 Implementierung der Metriken

Um Implementierungsaufwand einzusparen, wurden alle Gemeinsamkeiten von Prozess- und Aktivitätsmetriken in einer gemeinsamen Oberklasse gekapselt. Dazu gehören Getter-Methoden für Metrikname, Metriktyp (Java), Metriktyp (SQL) sowie Standard-Aggregationsmethoden für numerische Metriken wie z.B. Mittelwert und Median. Letztere wurden mit Standard SQL Konstrukten realisiert. Der Median wird als mittleres Element der Sortierung aller instanzbasierten Metrikwerte ermittelt (siehe Listing 6.2).
Da Aktivitätsmetriken nicht immer auf jede Aktivität angewandt werden sollen, es sei denn, es handelt sich tatsächlich um generische Metriken wie z.B. Ausführungsdauer oder Ausführungshäufigkeit, wurde für Aktivitätsmetriken eine eigene Unterklasse AbstractCustomActivityMetric vorgesehen. Sie erlaubt mit entsprechenden Methoden einerseits die Definition aller Aktivitäten, auf die die Metrik angewandt werden soll, sowie andererseits die Abfrage, ob die Metrik auf eine gegebene Aktivität angewandt werden soll.
Da es Metriken gibt, die auf andere Metriken zugreifen müssen, ist mit dem oben bereits angesprochenen MetricRepository ein entsprechender Datenaustausch vorgesehen. Dazu enthalten alle Unterklassen von AbstractCustomMetric einen Verweis auf ein MetricRepository Objekt. Es enthält eine statische Hashtabelle in der bereits berechnete Metriken unter einem Schlüssel hinterlegt werden können.
Z.B. ist es sinnvoll, hier die Aufrufhäufigkeit des Gesamtprozesses zu speichern, um später die Frequenzen der einzelnen Aktivitäten ausrechnen zu können. Für die Erzeugung der

Listing 6.2 Beispiel für SQL Code zur Bestimmung des Medians. Hier wird der Median von WAITINGTIME berechnet

```
SELECT Tabelle.WAITINGTIME
      FROM (SELECT WAITINGTIME, ROWNUMBER() OVER() As Item
                          FROM CHECKELIGIBILITY
                    ORDER BY WAITINGTIME) Tabelle,
      WHERE Tabelle.Item=(SELECT COUNT(*) FROM CHECKELIGIBILITY)/2;
```

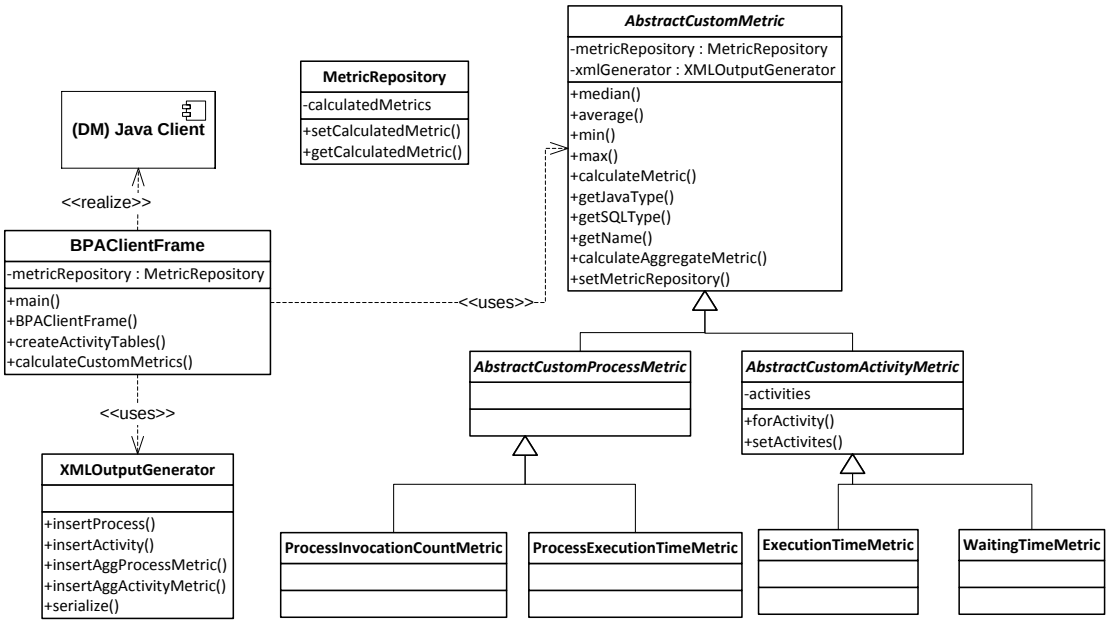


Abbildung 6.4: Klassendiagramm Metriken

XML Ausgabe wird folgendermaßen vorgegangen. Es gibt eine Klasse XMLOutputGenerator, die für jedes XML-Tag eine Methode definiert, die es in das XML Dokument, das zunächst als DOM Baum vorliegt, einfügen kann.
Zum Beispiel gibt es die Methoden insertProcess(), insertActivity(), sowie Methoden die für einen anzugebenden Prozess oder eine anzugebende Aktivität eine (unter Umständen benutzerdefinierte) Metrik in den aktuellen DOM Baum eintragen können.
Bevor die Metriken berechnet werden, wird zunächst ein Prozessknoten erzeugt und darunter für jede dBOP Aktivität ein Knoten. Jeder Metrik Klasse wird ein Verweis auf den für die Ausgabe verwendeten XMLOutputGenerator übergeben, sodass während der Berechnung der Metriken beliebige Einträge in den DOM Baum erfolgen können. Am Ende wird der DOM Baum noch serialisiert. Abbildung 6.4 zeigt die an der Metrikberechnung beteiligten Klassen.

6.3.1 Implementierung Benutzer-spezifischer Metriken

Der Aufbau des Werkzeugs erlaubt die Definition beliebiger benutzerspezifischer Metriken. Je nachdem, ob die Metrik ein Attribut darstellt, das zu einer Aktivität oder einem ganzen

Listing 6.3 Registrierung der Standardmetriken (z.B. Ausführungsdauer von Aktivitäten) in der Methode BPAClientFrame.calculateCustomMetrics().

```
String[] activityMetricNames =
    {"Metrics.WaitingTimeMetric","Metrics.ExecutionTimeMetric",
"Metrics.InvocationCountMetric","Metrics.FrequencyMetric"};
String[] processMetricNames = {"Metrics.ProcessInvocationCountMetric"};
```

Prozess gehört, wird als Vaterklasse AbstractCustomActivityMetric oder AbstractCustomProcessMetric gewählt. Die Metrik muss folgende Methoden implementieren:

- getName(). Gibt den Spaltenname der Aktivitäts- oder Prozesstabelle zurück, in die die Metrik eingetragen werden soll, sofern es sich um eine Aktivität handelt, die pro Instanz zu berechnen ist.
- getSQLType() und getJavaType. Gibt den SQL- bzw. Java Typ der Metrikwerte zurück.
- setActivities(). Trägt in die ArrayList activities die Namen der Aktivitäten als Strings ein, für die die Metrik berechnet werden soll. Handelt es sich um eine allgemeingültige Metrik, so genügt der Eintrag „All“.
- calculateMetric(). Trägt in ein übergebenes JDBC-ResultSet den Wert der Metrik ein (pro Aktivitätsinstanz bzw. pro Prozessinstanz).
- aggregate(). Gibt als booleschen Wert zurück, ob die Metrik aggregierbar ist. Das ist nur dann der Fall wenn die Metrik numerisch ist und pro Instanz berechnet werden kann.
- calculateAggregateMetric(). Berechnet auf Basis beliebiger Datenbankzugriffe eine Metrik die genau einer Aktivität oder einem Prozess zuzuordnen ist. Die Metrik kann per Zugriff auf das Objekt xmlGen vom Typ XMLOutputGenerator den Wert in der XML Ausgabe protokollieren.

Bei der Berechnung der Metriken in calculateMetric() und calculateAggregateMetric() können im MetricRepository neue Werte abgelegt werden (setCalculatedMetric()) und es kann auf bereits berechnete Werte zurückgegriffen werden (getCalculatedMetric()). Die Metrik muss abschließend in der Methode BPAClientFrame.calculateCustomMetrics() wie in Listing 6.3 registriert werden.

6.4 Implementierung der Konsolidierung

Zur Steuerung des Konsolidierungsvorgangs wurde eine passende Ansicht (Klasse TAConsolidationPanel) implementiert. Die Ansicht besteht aus einer JList, in der die zu untersuchende Aktivitätstabelle ausgewählt werden kann.
Jede Aktivitätstabelle ist durch ein Objekt vom Typ SqlTable in einem Objekt vom Typ TAConsolidationData, der Zustandsvariable für die Konsolidierung, repräsentiert.
SqlTable erlaubt es, z.B. die Attribute der Tabelle auszulesen.

Die Klasse AVPair, die bei der Extraktion eingeführt wurde, wird hier wiederverwendet, um die Attribute der Aktivitätstabellen samt dBOP Variablenname und dBOP Attributname darzustellen.
Unter der JList ist eine weitere Liste angeordnet, in der die Matchings auf die operativen Daten eingesehen und editiert werden können. Sie ist vom Typ AttributeTable.

6.4.1 AttributeTable: Ein Bedienelement zur Steuerung der Konsolidierung

Das Bedienelement zur Steuerung der Konsolidierung hat das Erscheinungsbild einer Liste, in der die Attribute der aktuell gewählten Aktivitätstabelle angezeigt werden. Die Aktivitäten, für die ein Matching existiert, haben in ihrer Zeile einen Button „Details“ und eine JCheckBox zur Aktivierung des Matchings.
AttributeTable ist von JTable abgeleitet und verwendet als Renderer für die Zellen AttributeTableCellPanels. Das sind JPanels, die alle Informationen einer Zeile der Tabelle anzeigen können:

- Name eines Attributs der Aktivitätstabelle.
- Name eines matchenden operativen Attributes.
- Einen Button zum Anzeigen von Details über das Matching.
- Eine Checkbox zum Aktivieren des Matchings.

Wird auf „Details“ geklickt, öffnet sich ein Dialog vom Typ MatchingDialog der von JDialog erbt, er zeigt die Matching Informationen wie in Kapitel 5.4 vorgestellt.
Zur Speicherung der Daten des Bedienelements wird ein dafür angepasstes Tabellenmodell verwendet. Es wird durch die Klasse AttributeTableModell realisiert, das von AbstractTableModel erbt und die Daten in Form einer ArrayList von AttributeTableCell Objekten vorhält.
AttributeTable hat eine Methode setModel(), die eine Liste mit Attributinformationen der aktuellen Aktivitätstabelle (Von Typ PrintableAVList. Siehe Kapitel über Extraktion.) entgegennimmt, sowie eine ArrayListe von Matchings und das Tabellenmodell mit den Daten füllt, die letztendlich von AttributeTableCellPanel angezeigt werden. setModel() geht für jedes Attribut der Aktivitätstabelle alle Matchings durch und prüft, ob eines davon zu dem Attribut passt.

6.4.2 Binarisierung

Die Binarisierung, die vor dem Join der operativen Tabellen mit der Aktivitätstabelle erfolgt, wird nur dann angestoßen, wenn das operative Attribut, das für den Join verwendet werden soll, Werte mehrfach enthält. Das kann per SQL geprüft werden, indem die Anzahl der Zeilen der Tabelle mit der Anzahl verschiedener Werte des operativen Attributs verglichen wird. Bei Gleichheit muss kein Attribut binarisiert werden. Bei Ungleichheit werden alle operativen

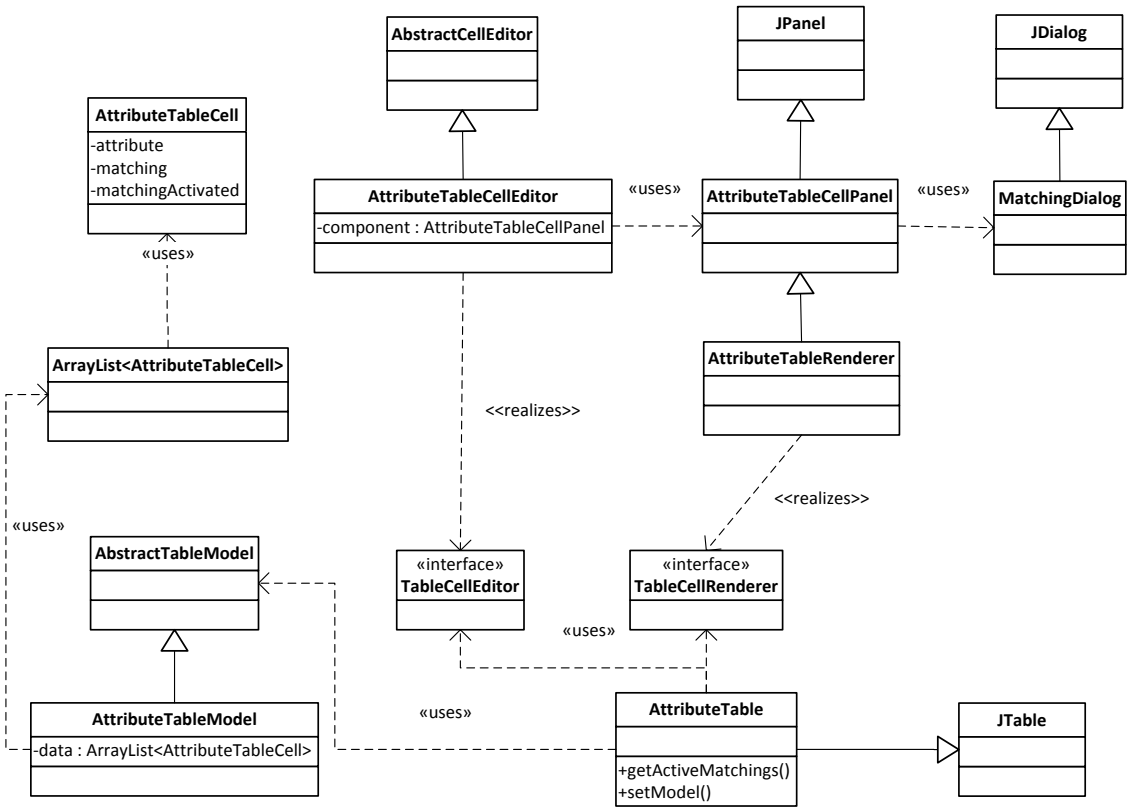


Abbildung 6.5: Klassendiagramm Attribute Table.

Attribute herausgesucht, die für Mehrfach-Einträge der Join-Spalte (im Beispiel 5.5 ist es die Spalte ANGESTELLTER) mehrere verschiedene Wert aufweisen. Das kann per SQL für Attribut ID (das ist die Spalte für Fortbildungs-IDs) wie in Listing 6.4 geprüft werden. Für die gefundenen Attribute wird vor der Ausführung des Joins der Prozesstabelle mit den operativen Tabellen pro nominalem Wert eine neue binäre Spalte in der entsprechenden operativen Tabelle angelegt.
Dann wird für jedes Tupel in den operativen Tabellen, die zu binarisierende Spalten enthalten, der Wert dieser Spalte nachgesehen und als binäres Attribut in die entsprechende binäre Spalte geschrieben.
Danach wird über die neuen binären Spalten eine Aggregation zur Verschmelzung gleicher Entities (z.B. Angestellte) durchgeführt (siehe Listing 6.5). Zum Schluss erfolgt der Join wie im Entwurfskapitel 5.4 beschrieben.

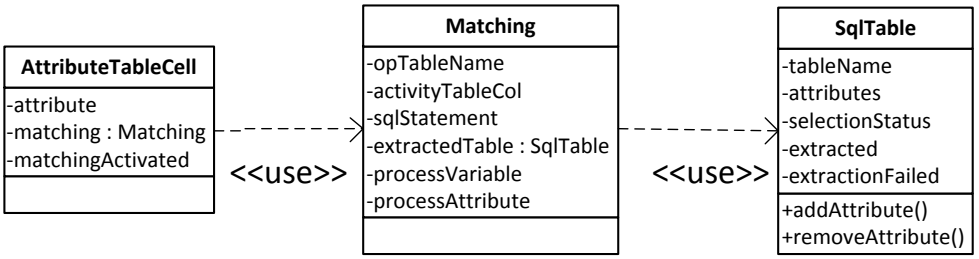


Abbildung 6.6: Klassendiagramm Matchings.

Listing 6.4 Beispiel: SQL Code den das Werkzeug generiert und ausführt, um zu prüfen, ob Spalte ID der Tabelle FORTBILDUNGEN binarisiert werden muss.

```
SELECT fortbildungen.ID
FROM FORTBILDUNGEN fortbildungen,
       (SELECT DISTINCT ANGESTELLTER AS id
       FROM FORTBILDUNGEN f
       WHERE 1 < (SELECT COUNT(ANGESTELLTER)
              FROM FORTBILDUNGEN
              WHERE ANGESTELLTER=f.ANGESTELLTER)) dubletten
WHERE dubletten.ID=fortbildungen.ANGESTELLTER AND 1 < (SELECT COUNT(DISTINCT
    fortbildungen2.ID)
       FROM FORTBILDUNGEN fortbildungen2
       WHERE fortbildungen2.ANGESTELLTER=dubletten.ID)
```

Listing 6.5 Beispiel: SQL Code den das Werkzeug generiert und ausführt, um Mehrfacheinträge für dieselben Entities (hier:Angestellte) miteinander zu verschmelzen.

```
SELECT ANGESTELLTER,MAX(BUSINESSENGLISH),MAX(KUNDENBERATUNG),MAX(SOFTSKILLS)
FROM FORTBILDUNGEN
GROUPBY ANGESTELLTER
```

6.5 Implementierung des Preprocessing

Das Preprocessing Panel von WEKA kann in beliebige Java-Swing Anwendungen integriert werden. Der nächste Schritt besteht dann im Füllen der Datenstruktur für die Trainingsbeispiele (die konsolidierten Daten), da WEKA nicht direkt auf der Datenbank arbeitet, sondern die Trainingsdaten im Hauptspeicher vorhält.
Die Datenstruktur ist vom Typ Instances, sie wird normalerweise bei WEKA durch das SQLViewerPanel, wo der Benutzer eine Datenbank-Verbindung herstellen kann, gefüllt. Allerdings kennt WEKA von Haus aus den SQL JDBC Treiber von DB2 nicht, deshalb bietet es sich an, die Methode setInstancesFromDBQ() des WEKA Preprocessing-Panels in einer neuen Klasse entsprechend zu überschreiben.
setInstancesFromDBQ() nutzt ein Objekt vom Typ InstanceQuery um eine Verbindung zur Datenbank herzustellen und ruft dazu die Methode connectToDatabase() auf. Also wurde auch diese Methode in der Ersatzklasse BPAInstanceQuery so überschrieben, dass der DB2 JDBC Treiber geladen wird.

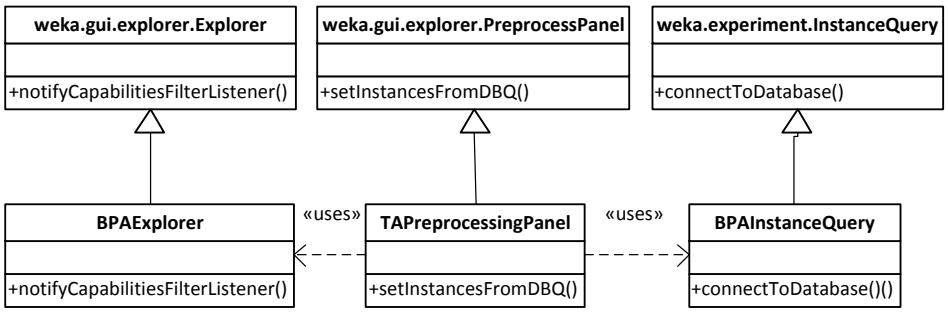


Abbildung 6.7: Klassendiagramm für den Preprocessing Schritt. Das TAPreprocessingPanel erbt von dem entsprechenden WEKA Panel.

6.6 Implementierung des Data Mining

Für die Implementierung des Data Minings wird, wie im Preprocessing Schritt, das entsprechende Panel des WEKA Explorers genutzt (das ClassifierPanel). Dieses Panel erlaubt die Anwendung vieler gängiger Classifier auf die konsolidierten Daten. Darunter auch die Classifier J48 und M5P, also den Algorithmen, die bereits im Entwurfskapitel besprochen wurden.
Das Panel protokolliert jeden Start eines Classifiers in einer JList. Die Einträge in der JList können mit Rechtsklick mit einem PopupMenu näher untersucht werden. Z.B. kann ein bereits erstellter Classifier jederzeit in Form einer Model-Datei gespeichert werden.
Dieses PopupMenu wurde um den Eintrag „In XML Ausgabe aufnehmen.“ erweitert. Klickt der Benutzer diese Option, so wird der Classifier und die entsprechende Auswertung (vom Typ Evaluation) in einer ArrayListe gespeichert. Der Benutzer hat dann die Möglichkeit mit der Option „Ergebnisse exportieren.“ in der Toolbar die Ergebnisse in XML Form zu serialisieren. Dazu wurde die Klasse DMXMLGenerator implementiert, die je eine Methode zum Einfügen von Aktivitäten, Entscheidungsbäumen und Modellbäumen in die XML Datei anbietet. Abbildung 6.8 zeigt die implementierten Klassen im Überblick.

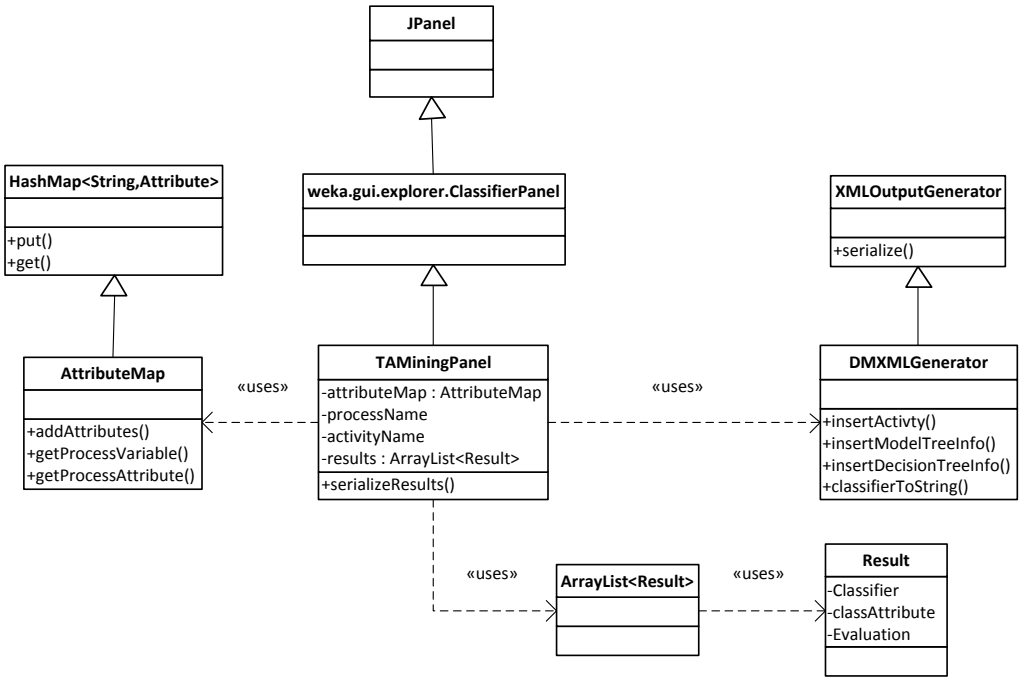


Abbildung 6.8: Klassendiagramm für den Data Mining Schritt. Das TAMiningPanel erbt von dem entsprechenden WEKA Panel.

6.7 Implementierung des Classifier-Webservice

Die Modell- und Entscheidungsbäume, die mit dem Werkzeug erstellt und in Form von XML gespeichert wurden, können von anderen Programmen genutzt werden. Z.B. könnte im beschriebenen Anwendungsszenario (siehe Kapitel 4) ein Webservice die Einordnung von Versicherungsfällen in Risikoklassen übernehmen.
Um die Modelle zu nutzen ist es erforderlich, die WEKA Bibliothek einzubinden. Um das Classifier Objekt zu laden, wird zunächst per XPath das serialisierte Objekt in einen String aus der XML Datei ausgelesen.
Danach erfolgt eine Dekodierung durch einen Base64-Dekoder. Base64 ist eine Klasse der Apache Commons Bibliothek (siehe http:\\commons.apache.org), die eine Kodierung von Bytearrays in US-ASCII Text und eine entsprechende Dekodierung, wie in RFC 2045 beschrieben, implementiert.
Der ByteArray, den der Decoder zurückgibt, wird dann in einen ByteArrayInputStream überführt. Mit diesem Stream wird ein ObjectInputStream initialisiert, auf dem dann die Methode readObject() ausgeführt werden kann. readObject() gibt den Classifier zurück, der dann nur noch als solcher gecastet werden muss. Listing 6.6 zeigt den Ablauf.
Um den Classifier zu nutzen, muss zunächst ein passende Instanz (Typ weka.core.Instance) angelegt werden. Dann kann die Methode classifyInstance() des Classifier Objekts auf die Instanz angewandt werden.

Listing 6.6 Methode zum Auslesen eines Classifiers (z.B. Entscheidungsbaum) aus der XML Ausgabe des Werkzeugs.

```
public Classifier getClassifier(String file, String dBOPVariableName, String
    dBOPAttributeName) {
            DocumentBuilder builder;
            Document document;
            builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
            try {
                   document = builder.parse(new File(file));
                   XPath xPath = XPathFactory.newInstance().newXPath();
                   String serializedClassifier = (String)
                       xPath.evaluate("//ModelTree[@dBOPVariableName='"+dBOPVariableName+"'
                       and
                       @dBOPAttributeName='"+dBOPAttributeName+"']/ModelObject/text()",
                       document,XPathConstants.STRING);

                   byte[] serializedClassifierBytes =
                       Base64.decodeBase64(serializedClassifier);
                   ByteArrayInputStream stream = new
                       ByteArrayInputStream(serializedClassifierBytes);
                   ObjectInputStream oistream = new ObjectInputStream(stream);
                   Classifier c = (Classifier)oistream.readObject();
                   return c;
            } catch...
            }
            return null;
      }
```

7 Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse der Arbeit diskutiert und es wird ein Ausblick auf weitere Arbeiten gegeben.

7.1 Zusammenfassung

Mit dem in dieser Arbeit entwickelten Werkzeug wurde die Basis für die deep Business Optimization Platform geschaffen.
Es ermöglicht eine aktivitätsorientierte Integration von Workflowdaten und operativen Daten. Das heißt, das Werkzeug erstellt pro Aktivität eines Workflows eine Tabelle für alle Instanzen der Aktivität, und erweitert diese Tabelle anschließend um weitere (operative) Spalten. Zur Anreicherung der Daten bietet es eine generische und erweiterbare Metrikbibliothek und eine entsprechende Schnittstelle. Zur Analyse der Daten kann das WEKA Classifier Panel und Preprocessing Panel genutzt werden.
Die Systemarchitektur wurde als Pipes-Filters Architektur [LL07] modelliert. Dabei wurden 5 Verarbeitungsschritte („Filter“) identifiziert, die zusammen als Pipeline die gewünschten Ausgaben (Metriken und Modelle) liefern. Entlang dieser Datenpipeline wurde die Software entworfen und implementiert.
Die Implementierung des Werkzeugs besteht aus einem Hauptprogramm und einem Webservice. Es erlaubt den Abruf von BPEL-Auditdaten von einem IBM Websphere Process Server und deren Zuordnung zu den entsprechenden BPMN Aktivitäten und Prozessattributen. Beide Aufgaben werden mit dem Webservice gelöst, der im gleichen Applikationserver-Kontext läuft, wie die Workflows deren Auditdaten abgerufen werden sollen. Dadurch kann auf einfache Weise die Business Flow Manager API des Websphere Process Servers zum Abruf genutzt werden.
Die Zuordnung der BPEL-Aktivitäten zu BPMN Aktivitäten und von BPEL-Attributen zu dBOP Attributen erfolgt durch Nachschlagen von Mappings im XML Mapping Inputfile.
Die Auditdaten werden vom Webservice in entsprechenden Datenbanktabellen gespeichert, die als Schnittstelle zum Hauptprogramm dienen. Für den Anwender wurde das Hauptprogramm mit graphischer Benutzeroberfläche entwickelt. Der Datenzugriff (auf Datenbank und XML Inputfiles) erfolgt im Hauptprogramm und im Webservice über dieselbe Bibliothek, das hat den Vorteil einer besseren Wartbarkeit, da lesende und schreibende Methoden derselben Daten nicht verstreut sind. Der Anwender hat mit der GUI, neben den XML Inputfiles, die Möglichkeit, die Filter der Datenpipeline zu starten und zu konfigurieren. Nach der Extraktion besteht die Möglichkeit, die Auditdaten mit Metriken anzureichern. Es wurden entsprechende Basisklassen entwickelt, die die Neuentwicklung

benutzerspezifischer Metriken vereinfachen (z.B. durch Vererbung der Aggregationsfähigkeit und XML-Serialisierbarkeit).
Der Anwender kann über die GUI einstellen, wie viele Auditdaten extrahiert werden sollen. Er kann durch Anpassung der Eingabedateien auch steuern, für welche BPMN Aktivitäten Auditdaten extrahiert werden sollen.
Nach der Berechnung der Metriken können die Auditdaten mit operativen Daten über ein passendes Steuerelement konsolidiert werden. Dabei können vorher definierte Matchings genutzt werden. Die Denormalisierung der operativen Daten erfolgt manuell, indem der Anwender einen SQL Befehl eingibt, der dann zur Extraktion der operativen Daten verwendet wird. Das Werkzeug kann im Falle einer 1:N Beziehung zwischen Prozess-Entities und operativen Entities durch Binarisierung die Daten für das Data Mining vorbereiten. Der Benutzer hat die Möglichkeit über ein Preprocessing Panel weitergehende Vorbereitungen, wie z.B. Nominalisierung von numerischen Werten zu treffen.
Er kann über das Data Mining Panel C4.5 und M5P Classifier erzeugen und als XML exportieren.
Der Benutzer wird über den gesamten Ablauf der Datenpipeline durch graphische und textuelle Widgets unterstützt, wobei die Idee des Sites-Modes-Trails HMI Modells [NW87] eine wichtige Rolle gespielt hat.
Es wurde ein Szenario beschrieben, in dem das Werkzeug sinnvoll angewandt werden kann. Die im Data Mining Schritt verwendeten C4.5 [Qui93] und M5P [WW97] Modell-/Entscheidungsbaumalgorithmen werden in Kapitel 5.6 ausführlich besprochen. Neben wichtigen Standards (BPMN und BPEL) wurde in den Grundlagen auch auf wichtige Preprocessing und Data Mining Verfahren eingegangen.

7.2 Ausblick

Das entwickelte Werkzeug ermöglicht es, durch seine Fähigkeit der Extraktion von Auditdaten, der Fähigkeit der Zuordnung dieser Daten zu BPMN Aktivitäten und BPMN Prozessattributen, der Fähigkeit zur Metrikberechnung und der Fähigkeit zur Konsolidierung, die gewonnenen Daten zur Analyse und Optimierung von Workflows zu nutzen. Die im Data Mining Panel trainierten Classifier können exportiert und z.B. in einen Classifier Webservice integriert werden. Die Classifier können auch in WEKA wiederverwendet werden.
Erste Auswertungen [NMRM11] zeigen, dass C4.5 Entscheidungsbäume, wie in Kapitel 5.6 vorgeschlagen, eine gute Wahl sind, wenn der weitere Prozessfluss nach einer Aktivität vorhergesagt werden soll. Es bleibt zu klären, wie gut die Kombination aus C4.5 für nominale vorherzusagende Attribute und M5P für stetige vorherzusagende Attribute gegenüber anderen Klassifizierungs- und Regressionsverfahren bei Task Automation Anwendungsszenarien abschneidet. Für numerische Attribute würden sich als Alternative Neuronale Netze anbieten.
Die Einbindung weiterer Datenquellen im Konsolidierungsschritt, z.B. von XML Datenquel-

len, stellt ebenfalls ein interessantes Themengebiet für weiterführende Arbeiten dar.

Literaturverzeichnis

[Aal] W. M. P. van der Aalst. Process Mining: The next step in Business Process Management. URL http://prom.win.tue.nl/research/wiki/_media/presentations/mining_au_process_days.ppt. (Zitiert auf Seite 13)

[ACD03] T. Andrews, F. Curbera, H. Dholakia. Business Process Execution Language for Web Services, Version 1.1. Specification,, 2003. (Zitiert auf den Seiten 20, 21 und 23)

[ADH+03] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47:237–267, 2003. doi:10.1016/S0169-023X(03)00066-1. URL http://portal.acm.org/citation.cfm?id=961808.961812. (Zitiert auf den Seiten 13 und 14)

[AIS93] R. Agrawal, T. Imieliński, A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22:207–216, 1993. doi:http://doi.acm.org/10.1145/170036.170072. URL http://doi.acm.org/10.1145/170036.170072. (Zitiert auf Seite 27)

[All08] T. Allweyer. *BPMN Business Process Modeling Notation*. Books on Demand GmbH, 2008. (Zitiert auf Seite 18)

[AS98] R. Agrawal, R. Srikant. Fast algorithms for mining association rules. In *Readings in database systems (3rd ed.)*, chapter Fast algorithms for mining association rules, pp. 580–592. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998. URL http://portal.acm.org/citation.cfm?id=302090.302153. (Zitiert auf den Seiten 26 und 27)

[ATHW03] W. M. P. van der Aalst, A. H. M. Ter Hofstede, M. Weske. Business Process Management: A Survey. In *Proceedings of the 1st International Conference on Business Process Management, volume 2678 of LNCS*, pp. 1–12. Springer-Verlag, 2003. (Zitiert auf Seite 11)

[Bay00] S. D. Bay. Multivariate discretization of continuous variables for set mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pp. 315–319. ACM, New York, NY, USA, 2000. doi:http://doi.acm.org/10.1145/347090.347159. URL http://doi.acm.org/10.1145/347090.347159. (Zitiert auf Seite 35)

[BKNS99] M. Breunig, H. Kriegel, R. Ng, J. Sander. Optics-of: Identifying local outliers. *Principles of Data Mining and Knowledge Discovery*, pp. 262–270, 1999. (Zitiert auf Seite 30)

[BP99] S. D. Bay, M. J. Pazzani. Detecting change in categorical data: mining contrast sets. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pp. 302–306. ACM, New York, NY, USA, 1999. doi:http://doi.acm.org/10.1145/312129.312263. URL http://doi.acm.org/10.1145/312129.312263. (Zitiert auf Seite 35)

[BPM06] Business Process Modeling Notation Specification, 2006. (Zitiert auf den Seiten 17 und 18)

[BSMM01] I. N. Bronstein, K. A. Semendjajew, G. Musiol, H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt/Main, 5 edition, 2001. (Zitiert auf Seite 28)

[DK75] F. DeRemer, H. Kron. Programming-in-the large versus programming-in-the-small. *SIGPLAN Not.*, 10:114–121, 1975. doi:http://doi.acm.org/10.1145/390016.808431. URL http://doi.acm.org/10.1145/390016.808431. (Zitiert auf Seite 21)

[FI93] U. M. Fayyad, K. B. Irani. Multi-interval discretization of continuous-valued attributes. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022–1027. Morgan Kaufmann, 1993. (Zitiert auf Seite 36)

[GCC+04] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, M. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004. (Zitiert auf Seite 14)

[GRC04] M. Golfarelli, S. Rizzi, I. Cella. Beyond data warehousing: what's next in business intelligence? In *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, pp. 1–6. ACM, 2004. (Zitiert auf Seite 14)

[GSR96] M. Ganesh, J. Srivastava, T. Richardson. Mining entity-identification rules for database integration. In *Proceedings of the Second International Conference on Data Mining and Knowledge Discovery*, pp. 291–294. 1996. (Zitiert auf Seite 31)

[Har91] H. Harrington. *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness*. McGraw-Hill, 1991. (Zitiert auf den Seiten 6 und 10)

[HB04] H. Haas, A. Brown. Web Services Glossary, 2004. URL http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/. (Zitiert auf Seite 20)

[HC93] M. Hammer, J. Champy. *Reengineering the corporation*. Harper Collins, 1993. (Zitiert auf Seite 16)

[HFH+09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009. doi:10.1145/1656274.1656278. URL http://dx.doi.org/10.1145/1656274.1656278. (Zitiert auf den Seiten 39, 61 und 66)

[HH03] M. Hall, G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data engineering*, pp. 1437–1447, 2003. (Zitiert auf den Seiten 33 und 36)

[HK06] J. Han, M. Kamber. Data Mining: Concepts and Techniques, 2006. (Zitiert auf den Seiten 25, 26, 28, 29, 30, 32, 33, 34 und 35)

[Hol95] D. Hollingsworth. Workflow Management Coalition: Workflow Reference Model, 1995. URL http://www.wfmc.org/reference-model.html. (Zitiert auf Seite 17)

[IBM] IBM Education Assistant. URL http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wpi_v6/wpswid/6.0.2/BusinessRules.html. (Zitiert auf Seite 25)

[JMDX06] W.-F. Jing, D.-Y. Meng, M.-W. Dai, Z. Xu. A New Pre-processing Method for Regression. In J. Wang, Z. Yi, J. Zurada, B.-L. Lu, H. Yin, editors, *Advances in Neural Networks - ISNN 2006*, volume 3972 of *Lecture Notes in Computer Science*, pp. 765–770. Springer Berlin / Heidelberg, 2006. URL http://dx.doi.org/10.1007/11760023_113. (Zitiert auf Seite 36)

[KKZ09] H. Kriegel, P. Kröger, A. Zimek. Outlier detection techniques. In *Tutorial at the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Citeseer, 2009. (Zitiert auf Seite 30)

[Kri08] H. Kriegel. Angle-based outlier detection in high-dimensional data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 444–452. ACM, 2008. (Zitiert auf Seite 30)

[LL07] J. Ludewig, H. Lichter. *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken.* dpunkt.verlag, Heidelberg, 2007. (Zitiert auf den Seiten 8, 9, 37 und 86)

[LR00] F. Leymann, D. Roller. *Production workflow: concepts and techniques*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000. (Zitiert auf den Seiten 10 und 16)

[MAT10] M. M. Mazid, A. B. M. S. Ali, K. S. Tickle. Improved C4.5 algorithm for rule based classification. In *Proceedings of the 9th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases*, AIKED'10, pp. 296–301. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2010. URL http://portal.acm.org/citation.cfm?id=1808036.1808087. (Zitiert auf Seite 65)

[Mel10] I. Melzer. *Service-orientierte Architekturen mit Web Services. Konzepte - Standards - Praxis. 4.Auflage*. Spektrum Akademischer Verlag, Heidelberg, 2010. (Zitiert auf Seite 24)

[Mit10] B. Mitschang. Data-Warehouse-, Data-Mining- und OLAP-Technologien, 2009/2010. (Zitiert auf den Seiten 25 und 28)

[MS09] M. zur Muehlen, R. Shapiro. *Handbook on Business Process Management*, chapter Business Process Analytics. Springer-Verlag, 2009. (Zitiert auf Seite 14)

[Mue01] M. zur Muehlen. Process-driven Management Information Systems - Combining Data Warehouses and Workflow Technology, presented at. In *Proceedings of the 4th International Conference on Electronic Commerce Research (ICECR-4), Dallas (TX*. 2001. (Zitiert auf Seite 14)

[NMRM11] F. Niedermann, B. Maier, S. Radeschütz, B. Mitschang. Automated Process Decision Making based on Integrated Source Data. In *Lecture Notes in Business Information Processing*. 2011. (Zitiert auf Seite 87)

[NRM10] F. Niedermann, S. Radeschütz, B. Mitschang. Deep Business Optimization: A Platform for Automated Process Optimization. In W. Abramowicz, R. Alt, K.-P. Fändrich, B. Franczyk, L. A. Maciaszek, editors, *Business Process and Service Science - Proceedings of ISSS and BPSC: BPSC'10; Leipzig, Germany, September 27th - October 1st, 2010*, volume P177 of *Lecture Notes in Informatics*, pp. 168–180. Gesellschaft für Informatik e.V. (GI), 2010. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2010-89&engl=0. (Zitiert auf den Seiten 8 und 11)

[NW87] J. Nievergelt, J. Weydert. Sites, modes, and trails: Telling the user of an interactive system where he is, what he can do, and how to get to places (excerpt). In R. M. Baecker, editor, *Human-computer interaction*, chapter Sites, modes, and trails: Telling the user of an interactive system where he is, what he can do, and how to get to places (excerpt), pp. 438–441. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. URL http://portal.acm.org/citation.cfm?id=58076.58111. (Zitiert auf den Seiten 9, 46, 47 und 87)

[Pet05] H. Petersohn. *Data Mining - Verfahren, Prozesse, Anwendungsarchitektur*. Oldenburg, 2005. (Zitiert auf den Seiten 25, 26 und 28)

[PGG05] V. de Putte, L. G. Gavin. *Technical Overview of Websphere Process Server and Websphere Integration Developer*. IBM, 2005. URL http://www.redbooks.ibm.com/redpapers/pdfs/redp4041.pdf. (Zitiert auf den Seiten 23, 24 und 25)

[Qui92] J. Quinlan. Learning with continuous classes. In *In Proceedings AI'92*. 1992. (Zitiert auf Seite 28)

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning*. Morgan Kaufmann, 1 edition, 1993. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1558602380. (Zitiert auf den Seiten 7, 46, 62, 63, 64 und 87)

[RD00] E. Rahm, H. Do. Data cleaning: Problems and current approaches. *Bulletin of the Technical Committee on Data Engineering*, 23:3–13, 2000. (Zitiert auf den Seiten 12, 30 und 31)

[RLM04] H. Reijers, S. Liman Mansar. Best Practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega - The International Journal of Management Science*, 2004. (Zitiert auf den Seiten 3, 38, 40, 61 und 71)

[RN04] S. Russel, P. Norvig. *Künstliche Intelligenz. Ein moderner Ansatz.* Pearson Studium, 2004. (Zitiert auf Seite 27)

[RNB10] S. Radeschütz, F. Niedermann, W. Bischoff. BIAEditor - Matching Process and Operational Data for a Business Impact Analysis. *Proceedings EDBT*, 2010. (Zitiert auf den Seiten 43 und 58)

[Roz06] A. Rozinat. Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In *BPM 2005 Workshops (Workshop on Business Process Intelligence), volume 3812 of Lecture Notes in Computer Science*, pp. 163–176. Springer-Verlag, 2006. (Zitiert auf Seite 14)

[SA96] R. Srikant, R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. *SIGMOD*, 1996. (Zitiert auf Seite 27)

[SQL92] Information Technology - Database Language SQL, 1992. URL http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt. (Zitiert auf Seite 33)

[Wer09] M. Werner. *Information und Codierung Grundlagen und Anwendungen*. Vieweg+Teubner, 2009. doi:10.1007/978-3-8348-9550-9. (Zitiert auf Seite 65)

[WFM99] Workflow Management Coalition: Terminology & Glossary, 1999. URL http://www.wfmc.org/reference-model.html. (Zitiert auf Seite 17)

[WSB] Web Services Business Process Execution Language Version 2.0. URL http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf. (Zitiert auf den Seiten 20 und 22)

[WW97] Y. Wang, I. H. Witten. Inducing Model Trees for Continuous Classes. In *In Proc. of the 9th European Conf. on Machine Learning Poster Papers*, pp. 128–137. 1997. (Zitiert auf den Seiten 66, 69 und 87)

Alle URLs wurden zuletzt am 11.04.2011 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Bernhard Maier)