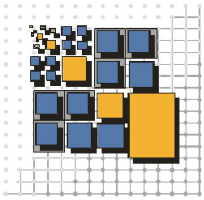

Otto-Friedrich-Universität Bamberg

Lehrstuhl für Praktische Informatik

Bachelorarbeit

im Studiengang Wirtschaftsinformatik
der Fakultät Wirtschaftsinformatik und Angewandte Informatik
der Otto-Friedrich-Universität Bamberg

Zum Thema:

Automatisierung der Datenerfassung für Wissensdatenbanken im technischen Kontext

Vorgelegt von:

Petr Vasilyev

Themensteller:
Prof. Dr. Guido Wirtz

Abgabedatum:
16.05.2017

Inhaltsverzeichnis

1 Einführung **1**

1.1 Motivation 1

1.2 Zielsetzung 1

1.3 Verwandte Arbeiten 2

1.4 Aufbau der Arbeit 3

2 Grundlagen von Expertensystemen **4**

2.1 Begriffsdefinition 4

2.2 Architektur eines Expertensystems 6

2.3 Wissensbasis 8

2.4 Wissenserwerbskomponente 11

3 Methoden der Wissens- und Datenerfassung **13**

3.1 Schnittstelle zur Dateneingabe 13

3.2 Datenerfassung aus dem Web 15

3.3 Maschinelles Lernen beim Wissenserwerb 19

4 Umsetzung der Wissensträgerschnittstelle **21**

4.1 Systembeschreibung 21

4.2 Wissensträgerschnittstelle 23

4.3 Worker für Datenübermittlung 27

5 Ausblick **31**

5.1 Abgrenzung der Datenquellen 31

5.2 Vergleich der Informationsquellen 32

5.3 Future Work 34

6 Fazit **35**

Literaturverzeichnis **36**

Abbildungsverzeichnis

1 Begriffsabgrenzung, [Mat00, S.30] 5

2 Expertensystem nach Haun, [Mat00, S.126] 6

3 Phasen der Expertensystementwicklung, [GD94, S.138] 8

4 Expertensystem nach Beierle und Kern-Isberner, [CG14, S.18] 9

5 Wissenserwerbskomponente 12

6 Die Struktur und die Informationsflüsse im DSS, [SK09, S.98] 14

7 XPath im Dokumentenbaum, [EPGR14, S.304] 16

8 Die Architektur vom SG-WRAM, [XDC03, S.2] 18

9 Die Struktur vom *PaaSfinder* 21

10 JSON-Objekt Spezifikation 22

11 JSON-Array Spezifikation 22

12 Profilaktualisierung mittels Wissensträgerschnittstelle 23

13 Aktivitätsdiagramm der Aktualisierung eines Vendors 24

14 Ausschnitt aus Vendor Datentyp 25

15 Vendor Page 26

16 Update Page 26

17 Review Page 26

18 Kontaktdatenform 27

19 Anwendungsbereich vom Worker 27

20 Pull-Requests Ansicht auf Github 30

21 Heroku Pull Request 30

22 RSS Angebot bei Heroku 34

Tabellenverzeichnis

1 Vergleich von RSS, Twitter, Newsletter und Blogs 33

Listings

1 „/vendor“ Route 28
2 Response beim erfolgreichen Branch-Erstellen 29
3 Response beim erfolgreichen File-Update 29
4 Response beim erfolgreichen Pull-Request-Erstellen 29
5 Ein RSS Beispiel von Heroku 32

Abkürzungsverzeichnis

API Application Programming Interface

DOM Document Object Model

DSS Decision Support System

DTD Document Type Definition

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

MVVM Model View ViewModel

PaaS Platform-as-a-Service

REST Representational State Transfer

URI Uniform Resource Identifier

XML Extensible Markup Language

XPath XML Path Language

1 Einführung

1.1 Motivation

Die Idee von wissensbasierten Systemen entstand aus dem Bedürfnis, ein intelligentes System zu schaffen, das mittels spezifischen Wissens die Fachexperten bei den Problemlösungen unterstützt [RR10, S.18]. Eine Spezialisierung der wissensbasierten Systeme stellen Expertensysteme dar, in denen das Wissen letztendlich von Experten der entsprechenden Domänen stammt. Ein wissensbasiertes System bzw. ein Expertensystem ist durch die Trennung der Wissensdarstellung eines Problembereichs und der Wissensverarbeitung gekennzeichnet [CG14, S.11].
Die Beschreibung des Wissens von einem wissensbasierten bzw. Expertensystem erfolgt in der Wissensbasis, die in Form einer Wissensdatenbank realisiert wird. Die Anfangswissensbasis wird in der Regel mithilfe manueller Datenerfassung aufgebaut [Kar92, S.70]. Hierzu wird beispielsweise häufig die Methode des Interviews zwischen einem Wissensingenieur und einem Wissensträger eingesetzt [GTW90, S.76]. Allerdings sind die manuellen Wissenserwerbsmethoden für die Weiterentwicklung der Wissensbasis hinsichtlich der Erweiterung und Aktualisierung der Wissensdatenbank eher schlecht geeignet, da sie fehleranfällig sowie kosten- und zeitintensiv sind. Aus diesen Gründen gibt es Bestrebungen, den Prozess der Datenerfassung zu automatisieren. Eine Auswahl an bestehenden Ansätzen, die für die vorliegende Arbeit relevant sind, wird im Abschnitt 1.3 gegeben. Trotz der vielen Herangehensweisen gibt es kein einheitliches Konzept, das einen allgemeinen Rahmen für die Automatisierung der Datenerfassung bildet.

1.2 Zielsetzung

Das Ziel der vorliegenden Arbeit ist die Erarbeitung eines allgemeinen Konzeptes zur Automatisierung der Datenerfassung. Da die ausschließlich automatisierte Datenerfassung sehr schwierig umzusetzen ist, liegt der Schwerpunkt dieser Arbeit auf der Kombination zwischen den manuellen und maschinellen Vorgehensweisen.
Die praktische Umsetzung soll am Beispiel von *PaaSfinder*[1] erfolgen. Bei *PaaSfinder* handelt es sich um eine Web-Anwendung, die eine Wissensdatenbank im Bereich Platform-as-a-Service (PaaS) verwaltet. Das Ziel von PaaS besteht in der Erleichterung der Anwendungsentwicklung, indem eine Entwicklungsumgebung von einem PaaS-Anbieter als ein konfigurierbarer Service angeboten wird [Geo08, S.14]. Aufgrund der hohen Anzahl von PaaS-Anbietern, der Vielzahl von Einstellungsmöglichkeiten und potentiellen Inkompatibilitäten zwischen den unterschiedlichen Anbietern gibt es einen realen Bedarf an einem systematischen Marktvergleich, der mithilfe von Daten der PaaS-Anbieter erfolgt. Die im *PaaSfinder* enthaltenen Daten wurden bisher hauptsächlich manuell erfasst, was mühsam, zeit- und kostenintensiv ist. Das Ziel der Arbeit besteht darin, die Methoden zur Automatisierung der Datenerfassung hinsichtlich des Anwendungsfalls *PaaSfinder* zu erforschen und umzusetzen. Hierzu werden folgende Aspekte bearbeitet. Als erstes soll eine Benutzerschnittstelle zur Korrektur der bestehenden Daten entwickelt werden. Momentan erfordert eine Aktualisierung der Daten fachspezifische Vorkenntnisse, was viele Personen daran hindert, ein Update zu erstellen. Als nächstes soll die Erstellung eines

[1] https://paasfinder.org

Pull Requests mit den Daten der Benutzerschnittstelle automatisiert werden. Ferner sollen weitere Möglichkeiten erforscht werden, Daten automatisch zu erfassen. Es können beispielsweise soziale Netzwerke eingesetzt werden. Schließlich sollen die automatischen Tests erweitert werden, um die Konsistenz der Daten sicherzustellen.

1.3 Verwandte Arbeiten

Den Ausgangspunkt dieser Arbeit stellt die Publikation [Ghe92] von G. Tecuci dar, die die Automatisierung der Wissenserfassung als ein Konzept der Erweiterung, Aktualisierung und Verbesserung der Wissensbasis beschreibt [Ghe92, S.1444]. In diesem Zusammenhang wird ein lernendes System vorgestellt, das eine Auswahl an generischen Ansätzen des maschinellen Lernens bei der Wissenserfassung umsetzt [Ghe92, S.1445]. Ferner wird ein Framework entwickelt, das die Wissenserfassung durch maschinelles Lernen automatisiert. Darüber hinaus werden die erlernten Daten von einem Experten auf Korrektheit überprüft. Die Entwicklung der Wissensbasis wird in drei Phasen durchgeführt. In der ersten Phase wird die Anfangswissensbasis aufgebaut, die unvollständig und teilweise widersprüchlich sein kann. Die zweite Phase umfasst die inkrementelle Erweiterung und Verbesserung der Wissensbasis. Schließlich wird in der dritten Phase die Wissensbasis in Bezug auf Effizienz optimiert [Ghe92, S.1444]. Die Kernaussage der Arbeit besteht darin, dass die Kooperation zwischen dem Experten und dem Expertensystem in jeder Phase die Automatisierung der Datenerfassung deutlich erleichert. Beispielsweise können die Daten von einem Algorithmus generiert werden. Daraufhin werden sie vom fachlichen Experten auf formale und semantische Korrekteit überprüft und in die Wissensbasis gespeichert [Ghe92, S.1445]. In [GD94] wird die Entwicklung des Frameworks fortgeführt, wobei der Schwerpunkt im Bereich von multi-strategischem Lernen (multistrategy leraning) liegt [GD94, S.137].
Neben [Ghe92] und [GD94] gibt es eine Reihe weiterer Ansätze, die das maschinelle Lernen bei der Automatisierung der Datenerfassung zur Hilfe nehmen. Einige Beispiele sind [JJJ99], [JJJ01], [Geo96]. Dabei ist die Idee der Zusammenarbeit zwischen dem Experten und dem Lernalgorithmus durchaus verbreitet. Ein Beispiel stellt die Arbeit von Castro et al. [JJJ01] dar. Ihr Ansatz beruht auf die Arbeit von [Ghe92]. Als Startpunkt wird eine unvollständige Anfangswissensbasis betrachtet, die schrittweise verbessert wird, indem der Experte die Fragen vom System beantwortet. Bei der Frageerstellung wird ein Lernalgorithmus eingesetzt, der aus einer Trainingsmenge die Regeln lernt und kontinuierlich die Qualität der Fragen verbessert. Dabei betonen Castro et al. [JJJ01], dass der Lernalgorithmus keineswegs den Wissensingenieur ersetzen kann. Vielmehr soll der Algorithmus dem Wissensingenieur die Routinearbeit abnehmen und bei den schwierigeren Aufgaben unterstützen, indem verschiedene Varianten des Interviews vom Algorithmus vorgeschlagen werden [JJJ01, S.308].
Ein praxisorientierter Ansatz für die Automatisierung der Wissenserfassung wird in [SK09] thematisiert. Allgemein handelt es sich um die Transformation eines datenbasierten Systems in ein wissensbasiertes System, um die Effizienz der Produktion zu steigern. Im Hinblick auf die Automatisierung der Aktualisierung und Erweiterung der unvollständigen Wissensbasis beziehen sich Gebus und Leiviskä auf die Erkenntnisse aus [Ghe92], [Gre92] und [LHCP02]. Bezüglich der Erfassung von Erfahrungswissen nehmen die Autoren Bezug auf die Arbeit von Okamura et al. [IKTH91], die Heuristik bei Problemlösungen einsetzen. Im praktischen Teil wird ein bereits bestehendes datenbasiertes Decision Support System (DSS) betrachtet, das die Unternehmensführung bei den Entscheidungen in Be-

zug auf die Produktionsoptimierung unterstützen soll. Allerdings werden die Störungen in der Produktion von Anlagenbedienern (Experten) mithilfe von Erfahrungswissen intern behoben, ohne dieses Wissen weiterzugeben. Als Folge hat die Unternehmensführung kein umfangreiches Bild der Produktion, was sich langfristig negativ auf die Produktion auswirkt. Aus diesem Grund erweitern Gebus und Leiviskä das System um eine Wissensträgerschnittstelle, um das Expertenwissen in die Datenbank zu integrieren [SK09, S.94]. Gebus und Leiviskä veranschaulichen damit, wie die Idee der Zusammenarbeit zwischen dem Experten und dem wissensbasierten System zur Automatisierung der Wissenserfassung im Kontext eines Unternehmens umgesetzt werden kann.
Mit der rasanten Entwicklung des World Wide Web hat sich eine Forschungsrichtung entwickelt, die sich mit der Daten- und Wissenserfassung aus Online-Ressourcen befasst. Ein fundierter Überblick über die Webdatenerfassung sowie aktuelle Ansätze und Anwendungen in Bereichen wie Business Intelligence, Web-Crawling etc. wurde in [EPGR14] vorgestellt. Dabei werden sowohl theoretische als auch praktische Aspekte umfassend thematisiert. In theoretischer Hinsicht wurden zuerst allgemeine Probleme wie Automatisierungsgrad, Skalierbarkeit, Datenschutz, Instabilität der Ressourcenstruktur und Trainingsmenge angesprochen [EPGR14, S.301-302]. In Bezug auf die praktischen Ansätze wurden das Baumparadigma, Web-Wrapper und hybride Systeme durch die Analyse zahlreicher Publikationen systematisiert. Bei Anwendungen zur Webdatenerfassung wächst der Trend Richtung freier Open-Source Projekte, die mit kommerziellen Anwendungen im Wettbewerb stehen [EPGR14, S.310]. Die Autoren nennen als Beispiel die Pipes[2] von Yahoo [EPGR14, S.315]. Allerdings ist das Beispiel schon veraltet, da die Plattform nicht mehr unterstützt wird. Diese Tatsache bestätigt jedoch die Aussage über die hohe Dynamik im Webbereich. Nichtsdestotrotz gibt es einige Technologien, die bereits über mehrere Jahre hinweg bestehen. Ein Beispiel stellen Feed-Services wie RSS und Atom dar, die Pull-Benachrichtigungen über über Änderungen auf der Webseite, Blogs usw. ermöglichen (mehr dazu in [Ben05] und [SMSO15]). Bezogen auf Anwendungen, die aus dem kommerziellen Bereich stammen, führen die Autoren in [EPGR14] Lixto Web-Wrapper als Beispiel an. Ursprünglich entstand Lixto aus einem Forschungsprojekt und wurde später als kommerzielle Anwendung implementiert [ER11]. Die Idee von Lixto in [ER11] spricht das Problem der Stabilität des Web-Wrappers an. Die Anwendung erkennt automatisch Änderungen einer Webseite und passt sich an die neue Struktur an [EPGR14, S.309].

1.4 Aufbau der Arbeit

Die vorliegende Arbeit ist wie folgt aufgebaut. In Kapitel 2 wird der Begriff und die grundlegende Architektur eines Expertensystems erläutert. Darauffolgend werden die Bestandteile, die für das Konzept relevant sind, näher betrachtet. Anschließend wird schematisch der Kontext der Datenerfassung dargestellt und in Bezug auf Automatisierungsmöglichkeiten behandelt. In Kapitel 3 wird genauer auf die Automatisierungsmethoden bei der Datenerfassung eingegangen. Dabei werden bestehende Forschungsergebnisse vorgestellt und konzeptuell verallgemeinert. Kapitel 4 umfasst die praktische Umsetzung der Wissensträgerschnittstelle am Beispiel von *PaaSfinder*. In Kapitel 5 wird ein Ausblick über weitere Möglichkeiten bei der Automatisierung der Datenerfassung von *PaaSfinder* gegeben. Anschließend wird das Ergebnis der Arbeit im Fazit erläutert.

[2] https://en.wikipedia.org/wiki/Yahoo!_Pipes

2 Grundlagen von Expertensystemen

2.1 Begriffsdefinition

Ursprünglich waren Expertensysteme Anwendungsprogramme, die logische Schlussfolgerungen aus einer Wissensbasis ziehen konnten. Außerdem konnten sie überprüfen, ob eine Aussage aus einer vorhandenen Wissensbasis abgeleitet werden kann [Pet10, S.75]. Daher handelt es sich in der früheren Literatur meist um Anwendungen, die ihr Wissen in Form von logischen Ausdrücken darstellen und in der Lage waren, neue Erkenntnisse von bestehendem Wissen abzuleiten [Ghe92]. Im Laufe der Zeit hat sich das Konzept eines Expertensystems auf andere Anwendungsbereiche ausgeweitet. Aus diesem Grund gibt es mehrere Definitionen, die im Allgemeinen ähnlich sind und im Spezifischen Merkmale des zugehörigen Anwendungsbereichs beinhalten.
Allgemein lässt sich sagen, dass ein Expertensystem ein Computersystem (Hardware und Software) ist, das in einem bestimmten Bereich Wissen und Schlussfolgerungsfähigkeit eines menschlichen Experten nachbildet [CG14, S.12]. Aus Sicht der Wirtschaftsinformatik zielen Expertensysteme darauf ab, das Expertenwissen menschlicher Fachleute in der Wissensbasis eines Computers abzuspeichern und für eine Vielzahl von Problemlösungen zu nutzen [PFW+12, S.59]. Im Weiteren gehen Beierle und Kern-Isberner auf die Eigenschaften ein, die ein Expertensystem aufweisen sollen [CG14, S.12]. Im Rahmen dieser Arbeit sind folgende Eigenschaften besonders relevant:

- Anwendung des Wissens eines oder mehrerer Experten, um Probleme in einem bestimmten Anwendungsbereich zu lösen,
- Leicht lesbare Wissensdarstellung,
- Möglichst anschauliche und intuitive Benutzerschnittstelle,
- Leichte Wartbarkeit und Erweiterbarkeit des Wissens im Expertensystem,
- Unterstützung beim Wissenstransfer vom Experten zum System.

Hier ist es außerdem wichtig anzumerken, dass die Begriffe “Künstliche Intelligenz“, “wissensbasiertes System“ und “Expertensystem“ in einer engen Beziehung zueinander stehen. Haun stellt eine systematische Abgrenzung dieser Begriffe vor, die sich folgendermaßen beschreiben lässt [Mat00, S.30]:

- *Künstliche Intelligenz* stellt den Oberbegriff dar und bildet den theoretischen Rahmen für die Entwicklung von wissensbasierten Systemen und Expertensystemen.
- *Wissensbasierte Systeme* sind eine Teilmenge der Anwendungen innerhalb des Bereichs der künstlichen Intelligenz. Sie wenden die Wissensverarbeitung auf ein konkretes Aufgabengebiet an und verwalten Allgemeinwissen explizit und getrennt vom Rest des Systems.
- *Expertensysteme*, die ein Teilbereich der wissensbasierten Systeme sind, stellen eine Spezialisierung von wissensbasierten Systemen dar. Sie verwalten spezifisches Expertenwissen, das von einem Experten stammt und auf praxisbezogene Probleme angewandt wird.

Graphisch lässt sich die vorliegende Abgrenzung in Abbildung 1 darstellen:

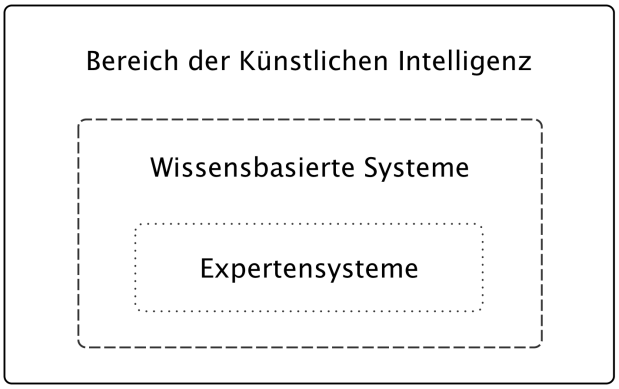

Abbildung 1: Begriffsabgrenzung, [Mat00, S.30]

Nach dieser Abgrenzung lässt sich feststellen, dass der Unterschied zwischen einem wissensbasierten System und einem Expertensystem darin besteht, dass das Wissen im Endeffekt von einem Experten stammt. Allerdings ist dieses Kriterium nicht besonders aussagekräftig. Beierle und Kern-Isberner weisen darauf hin, dass nach diesem Kriterium viele der existierenden wissensbasierten Systeme als Expertensysteme bezeichnet werden könnten [CG14, S.11]. Als Reaktion auf fehlende Kriterien stellen die Autoren die Eigenschaften eines Experten dar, die sich folgendermaßen zusammenfassen lassen:

- Experten sind selten und teuer.
- Experten sind nicht immer verfügbar.
- Leistungsfähigkeit der Experten ist nicht konstant, sondern kann nach Tagesverlauf schwanken.
- Expertenwissen kann oft nicht als solches weitergegeben werden.
- Expertenwissen kann verloren gehen.

Ein gutes Beispiel hinsichtlich der Gefahr, dass Expertenwissen verloren gehen kann, wird in [SK09, S.94] vorgestellt. Gebus nimmt hier Bezug auf die Mitarbeiter der sogenannten Baby-Boomgeneration. Es handelt sich um Experten, die ein umfangreiches Erfahrungswissen besitzen und bald aus Altersgründen das Unternehmen verlassen. Somit geht auch das Erfahrungswissen aus dem Unternehmen verloren.
Zusammenfassend lässt sich sagen, dass die Entwicklung eines Expertensystems ein hohes Potenzial besitzt. Allerdings kann ein Expertensystem nicht als Ersatz für einen menschlichen Experten betrachtet werden. Vielmehr geht es um eine Erfassung, Darstellung und Pflege des Expertenwissens in einem Expertensystem, um die Arbeitsprozesse effizienter zu gestalten und sowohl erfahrene als auch neue Anwender in einem bestimmten Wissensbereich bei der Aufgabenabwicklung zu unterstützen.

2.2 Architektur eines Expertensystems

Beierle und Kern-Isberner betonen, dass die Trennung zwischen der Darstellung des Wissens (Wissensbasis) und der Wissensverarbeitung (Wissensverarbeitungskomponente) der wichtigste Aspekt eines wissensbasierten Systems ist. [CG14, S.11]. Die Wissensbasis kann man sich als eine Art Datenstruktur vorstellen, in der das benötigte Wissen gespeichert wird. Die Wissensverarbeitungskomponente umfasst eine Menge von anwendungsunabhängigen Algorithmen, die mithilfe der Wissensbasis eine Lösung für ein gegebenes Problem erarbeiten. Somit stehen die Wissensbasis und die Wissensverarbeitungskomponente in einer engen Beziehung zueinander [Kar92, S.18].
Allgemein umfasst ein Expertensystem folgende Bestandteile [Pet10, S.75]:

- *Die Wissensbasis*, die Expertenwissen in Form von Fakten in einer bestimmten Sprache speichert sowie Regeln zur Wissensorganisation beinhaltet.
- *Die Inferenzmaschine*, die unter Berücksichtigung des zugrunde liegenden Wissensbedarfs die Wissensbasis durchsucht bis das System einen Problemlösungsvorschlag erarbeitet hat oder herausfindet, dass kein solcher existiert.
- *Die Dialogkomponente*, die eine Schnittstelle zwischen dem Nutzer und dem System darstellt.
- *Die Erklärungskomponente*, die dem Benutzer erläutert, warum und auf welche Weise eine bestimmte Lösung gefunden bzw. nicht gefunden wurde [Mat00, S.126].
- *Die Wissenserwerbskomponente*, die den Entwickler des Expertensystems bei der Erweiterung, Änderung und Wartung der Wissensbasis unterstützt.

Laut Tecuci stellen Wissensbasis und Inferenzmaschine grundlegende Bestandteile eines Expertensystems dar und bilden damit den Kern des Expertensystems [Ghe92, S.1444]. Dialogkomponente, Erklärungskomponente und Wissenserwerbskomponente gehören zur sogenannten Schale und sind für die Kommunikation zwischen dem Systemverwalter und dem Nutzer zuständig (siehe Abbildung 2).

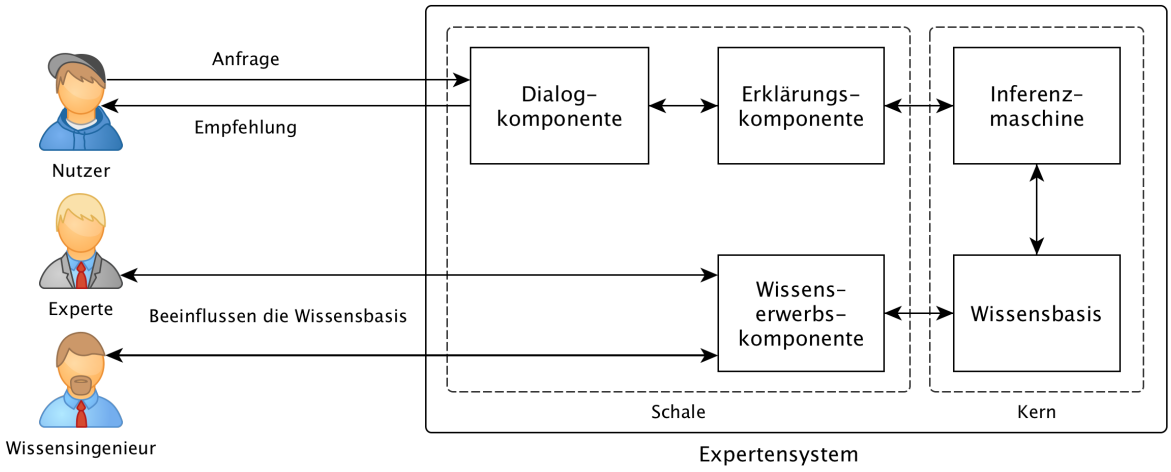

Abbildung 2: Expertensystem nach Haun, [Mat00, S.126]

Im Hinblick auf die Interaktion gibt es drei Gruppen, die mit dem Expertensystem interagieren:

- *Nutzer*, der das Expertensystem zum Lösen eines Problems benutzt und mit der Dialogkomponente kommuniziert. Der Wissensingenieur und der Experte können ebenso als Nutzer auftreten [JFI93, S.758].
- *Wissensingenieur*, der sich mit dem Aufbau und der Wartung der Wissensbasis beschäftigt. Unter anderem ist Wissensmodellierung ein wichtiger Aufgabenbereich eines Wissensingenieurs [JFI93, S.742].
- *Experte*, der über spezifisches Erfahrungswissen verfügt, das für das Expertensystem relevant ist.

Der Ablauf der Kommunikation zwischen dem Nutzer und dem Expertensystem sieht folgendermaßen aus:

- Der Nutzer schickt eine Anfrage an die Dialogkomponente des Expertensystems.
- Die Dialogkomponente übermittelt die Anfrage an die Inferenzmaschine.
- Die Inferenzmaschine erarbeitet eine Lösung für das gegebene Problem mittels der Wissensbasis und gibt das Ergebnis an die Dialogkomponente zurück.
- Anschließend teilt die Dialogkomponente dem Nutzer die Lösung des Problems mit. Falls keine Lösung zum Problem existiert, wird eine entsprechende Fehlermeldung angezeigt.

Auf der anderen Seite können die Inhalte der Wissensbasis von einem Wissensingenieur mithilfe der Wissenserwerbskomponente beeinflusst werden. Der Wissenserwerb durch den Wissensingenieur ist die verbreitetste Vorgehensweise, neue Daten für ein wissensbasiertes System zu erschließen. Meistens handelt es sich um ein Interview zwischen dem Wissensingenieur und dem Experten [Pet10, S.76], [HDNR97, S.210]. Neben dem Interview kann der Wissensingenieur eine Recherche in den verfügbaren Wissensquellen wie Texte, technische Zeichnungen oder Web-Ressourcen durchführen. Anschließend werden die Daten vom Wissensingenieur formalisiert und in die Wissensbasis gespeichert.
Die Wissensbasis kann in einigen Fällen von einem fachlichen Experten beeinflusst werden. Dafür ist eine geeignete Expertenschnittstelle innerhalb der Wissenserwerbskomponente notwendig, die es den Experten ermöglicht, ihr Erfahrungswissen selbst zu formalisieren und in die Wissensbasis einzutragen [JFI93, S.743]. Die Überprüfung des Dateninputs ist ebenfalls die Aufgabe der Wissenserwerbskomponente. Dies kann beispielsweise mittels der Durchführung von automatisierten Tests erfolgen, um die Konsistenz der Wissensbasis zu gewährleisten [JFI93, S.743].
Um ein geeignetes Konzept der automatisierten Datenerfassung zu entwickeln, ist ein grundlegendes Verständnis der Struktur und Funktionsweise der Wissensbasis sowie der Wissenserwerbskomponente erforderlich. Im weiteren Verlauf der Arbeit werden deshalb die Erkenntnisse über die Wissensbasis und die Wissenserwerbskomponente erläutert, die in der Forschung von Expertensystemen entstanden sind.

2.3 Wissensbasis

Neben der Inferenzmaschine stellt die Wissensbasis den zentralen Teil eines Expertensystems dar, der die Daten des gesamten Systems beinhaltet [JFI93, S.754]. Im Folgenden werden der allgemeine Prozess der Wissensbasisentwicklung, der Inhalt der Wissensbasis und die Möglichkeiten der Wissensrepräsentation thematisiert. Gheorghe Tecuci beschreibt folgende Phasen bei der Entwicklung der Wissensbasis [Ghe92, S.1444]:

- Systematische Erfassung vom Expertenwissen
- Verfeinerung der Wissensbasis
- Reorganisation der Wissensbasis

In der ersten Phase werden das Vokabular und die geeignete Wissensrepräsentation festgelegt. Gebus und Leiviskä betonen, dass die Wissensrepräsentation einen entscheidenden Einfluss auf die Generierung und spätere Handhabung der Wissensbasis hat [SK09, S.95]. Die initialen Daten werden meistens im Rahmen eines Interviews mit einem Experten erfasst [Ghe92, S.1444]. Das Ergebnis der ersten Phase ist eine initiale Wissensbasis, die unvollständig und teilweise widersprüchlich sein kann. In der zweiten Phase wird die initiale Wissensbasis mithilfe geeigneter Datenerfassungsmethoden solange erweitert und verbessert, bis sie vollständig und korrekt genug ist, um ein gegebenes Problem richtig zu lösen. In der dritten Phase wird die vollständige und korrekte Wissensbasis reorganisiert, um die Effizienz der Lösungsberechnung zu steigern [Ghe92, S.1445]. Zusammenfassend werden die Phasen in Abbildung 3 dargestellt.

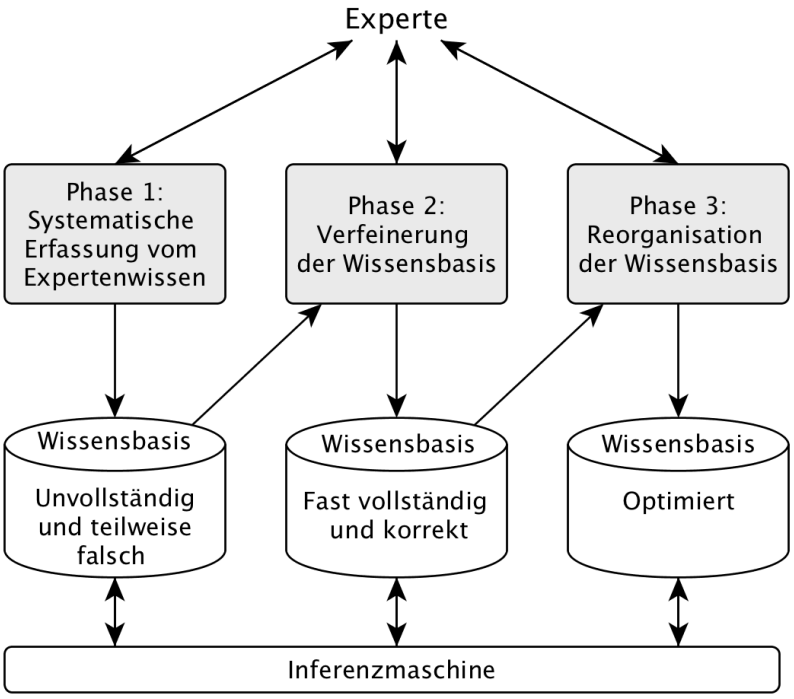

Abbildung 3: Phasen der Expertensystementwicklung, [GD94, S.138]

In Abbildung 3 sieht man, dass Tecuci dem Experten die gesamte Kontrolle über die Entwicklung der Wissensbasis zuweist. Allerdings ist diese Sichtweise nicht vollständig, da im Entwicklungsprozess der Wissensingenieur und der Systementwickler beteiligt sind und dementsprechend berücksichtigt werden müssen.
In Bezug auf den Inhalt der Wissensbasis unterscheiden Beierle und Kern-Isberner folgende Wissensarten [CG14, S.5]:

- *Fachspezifisches Wissen*: Dabei handelt es sich um das spezifische Wissen, das sich nur auf den gerade betrachteten Problemfall bezieht. Das sind z.B. Fakten, die von Beobachtungen oder Untersuchungsergebnissen stammen.
- *Regelhaftes Wissen*: Dieses Wissen stellt den eigentlichen Kern der Wissensbasis dar und kann noch genauer differenziert werden:
  - *Bereichsbezogenes Wissen*, das sich auf den gesamten Problembereich bezieht. Das kann sowohl theoretisches Fachwissen, als auch Erfahrungswissen sein.
  - *Allgemeinwissen*, das z.B. generelle Problemlösungsheuristiken, Optimierungsregeln oder auch allgemeines Wissen über Objekte und Beziehungen in der realen Welt beinhaltet.

Unter Berücksichtigung der Differenzierung der Wissensarten innerhalb der Wissensbasis beschreiben die Autoren in [CG14, S.18] auf eigene Weise die Architektur des Expertensystems, die in Abbildung 4 dargestellt wird.

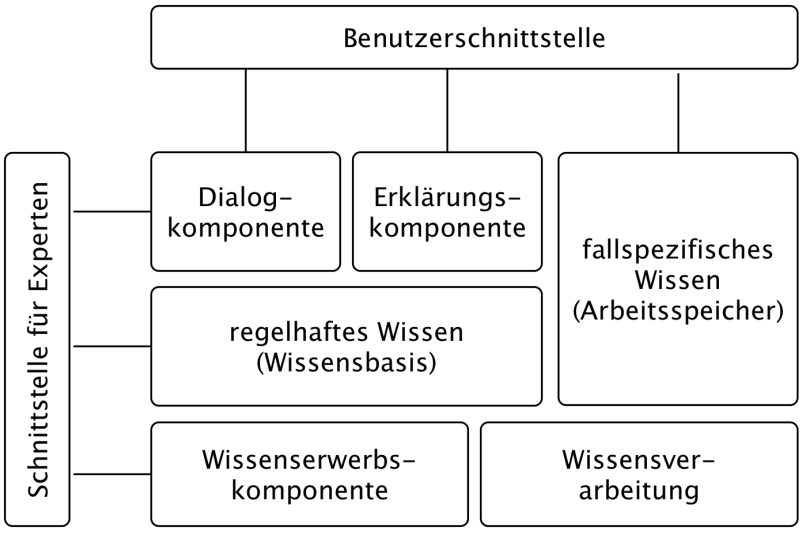

Abbildung 4: Expertensystem nach Beierle und Kern-Isberner, [CG14, S.18]

Laut Beierle und Kern-Isberner können verschiedene Wissensarten in einem wissensbasierten System je nach dem Anwendungsbereich unterschiedlich umfangreich auftreten. Ein hochspezialisiertes System kann beispielsweise über sehr wenig oder gar kein Allgemeinwissen verfügen. Auf der anderen Seite kann ein wissensbasiertes System den Schwerpunkt auf das gewöhnliche Alltagswissen legen [CG14, S.5-6].
Ein weiterer Aspekt beim Aufbau der Wissensbasis ist die Wissensrepräsentation. Die

grundlegende Aufgabe der Wissensrepräsentation ist die Formalisierung von Wissen, um eine maschinelle Verarbeitung zu ermöglichen [Mat00, S.22]. Sinz und Ferstl unterscheiden folgende Formen der Wissensrepräsentation [OE13, S.366]:

- *Regelorientierte Darstellung*, in der das Wissen in Form von WENN-DANN-Regeln beschrieben wird. Diese Darstellungsform wird beispielsweise bei Prolog[3]-Regeln eingesetzt.
- *Objektorientierte Darstellung*, die das Konzept der Objekttypen übernimmt und mit deklarativen Operatorbeschreibungen verbindet.
- *Constraints Darstellung*, die Modellbeschreibungen aus dem Bereich der Operation Research benutzt. Dabei handelt es sich um Lösungsräume durch Nebenbedingungen und Zielvorgaben.

Hinsichtlich der Wissensrepräsentation stellen Ferstl und Sinz imperative und deklarative Paradigmen gegenüber [OE13, S.366]. Ein Programm, das dem imperativen Paradigma folgt, besteht aus einer Folge von Befehlen, die nacheinander ausgeführt werden [OE13, S.341]. Bei einem deklarativen Programm handelt es sich um eine Beschreibung der Aufgabenaußensicht. Ein deklaratives Programm hat keine festgelegten Lösungsverfahren je Aufgabe. Stattdessen wird eine Lösung zum Zeitpunkt der Aufgabendurchführung mittels Inferenzmaschine abgeleitet [OE13, S.361].
Allgemein beziehen sich die Autoren darauf, dass an ein wissensbasiertes System nur geringe Anforderungen bezüglich Vollständigkeit, Widerspruchsfreiheit und Eindeutigkeit gestellt werden können. Aus diesem Grund ist das deklarative Paradigma für die Wissensrepräsentation besser geeignet. Folgende Gründe nennen die Autoren für die deklarative Umsetzung der Wissensbasis [OE13, S.366]:

- *Wissensdarstellung*: Da ein Mensch das Erfahrungswissen durch assoziative Beziehungsmuster aufbaut, ist die deklarative Wissensdarstellung eher geeignet.
- *Wissensauswertung*: Änderungen von Erfahrungswissen werden normalerweise in deklarativer Form erfasst.
- *Wissensverfügbarkeit*: Die Codewartung eines imperativen Programms ist fehleranfällig sowie kosten- und zeitintensiv, da das Erfahrungswissen häufig geändert und aktualisiert werden muss.

Der objektorientierte Ansatz ist eine weitere Möglichkeit, das Wissen zu beschreiben. Ein Beispiel für die objektorientierte Implementierung wird in [KM90] vorgestellt. Die Wissensbasis wird dabei als eine Sammlung von Klassen, Objekten und Methoden definiert [KM90, S.40]. Der große Vorteil einer solchen Umsetzung besteht in der Modularität des Wissens, d.h. das Wissen wird in unabhängige Module aufgeteilt. Da die einzelnen Module voneinander unabhängig sind, können sie getrennt getestet und modifiziert werden, ohne den Rest der Wissensbasis zu beeinträchtigen. Dies ermöglicht eine hohe Flexibilität bei der Wissensbasiserweiterung [KM90, S.43].
Neben der Implementierung der Wissensbasis ist eine geeignete Umsetzung der Wissenserwerbskomponente erforderlich, um die Wissensbasis aktuell, möglichst fehlerfrei und konsistent zu halten. Im Folgenden wird die Wissenserwerbskomponente in Hinsicht auf den allgemeinen Aufbau und ihre Funktionen thematisiert.

[3]Für weitere Informationen siehe z.B. http://www.swi-prolog.org/

2.4 Wissenserwerbskomponente

Bei der Wissenserwerbskomponente handelt es sich um einen Bestandteil des Expertensystems, der den Wissensingenieur oder einen Experten beim Aufbau und späterer Erweiterung der Wissensbasis unterstützt [GTW90, S.18]. Allgemein umfasst die Wissenserwerbskomponente zwei grundsätzliche Aufgaben, nämlich den Wissenserwerb und die Prüfung des Dateninputs auf Konsistenz, Vollständigkeit und Einschränkungen des Expertensystems [JFI93, S.759].
Unter dem Wissenserwerb wird eine Übertragung sowie Eingliederung von Wissen über Problemlösungsverfahren in ein Computerprogramm verstanden [GTW90, S.178]. Es werden folgende Grundarten des Wissenserwerbs unterschieden [JFI93, S.742]:

- *Indirekter Wissenserwerb*: Ein Wissensingenieur führt ein Interview mit einem Experten, oder allgemein mit einem Wissensträger durch. Die Analyse der Dokumente, die für das System relevant sind, gehört ebenso zur Aufgabe des Wissensingenieurs.
- *Direkter Wissenserwerb*: Ein Wissensträger gibt sein Wissen selbst mittels einer Schnittstelle in das Expertensystem ein.
- *Automatisierter Wissenserwerb:* Die Wissensbasis wird entweder mithilfe der automatisierten Datenerschließung aus verfügbaren Dokumenten oder Methoden des maschinellen Lernens erweitert.

Die Methoden des indirekten Wissenserwerbs lassen sich grundsätzlich in unstrukturierte und strukturierte Verfahren unterteilen. Das unstrukturierte Interview ist die am häufigsten verwendete Methode [GTW90, S.76]. Dabei stellt der Wissensingenieur dem Experten problembezogene Zwischenfragen, um ein möglichst vollständiges Bild des zur Problemlösung erforderlichen Wissens zu bekommen. Die Hauptschwierigkeit bei der Wissenserhebung durch ein Interview ist die Formulierung der Fragen. Wenn die Fragen zu spezifisch sind, können wichtige Informationen ausgelassen werden [SK09, S.95]. Eine strukturierte Vorgehensweise der Wissenserhebung ist die Protokollanalyse, wobei der Experte beim Lösen eines Problems aufgezeichnet wird. Um den Lösungsweg nachvollziehbar zu machen, kann der Experte die Aufgabe gezielt langsamer durchführen oder die Aufzeichnung mit Kommentaren versehen. Die aufgabenbezogenen Lösungen werden mithilfe von Induktion generalisiert. Bei der Induktion wird eine allgemeine Regel aus den Einzelfällen abgeleitet [JJJ01, S.308]. Anschließend werden die erzielten Ergebnisse vom Wissensingenieur formalisiert und ins Expertensystem eingetragen.
Beim direkten Wissenserwerb soll eine geeignete Schnittstelle im Rahmen der Wissenserwerbskomponente zur Verfügung gestellt werden, die es dem Wissensträger ermöglicht, sein Wissen ins System einzugeben. Die Schnittstelle soll eine dem Wissensträger bekannte Wissensrepräsentation verwenden und benutzerfreundlich bei der Dateneingabe sein [JFI93, S.743]. Ein Beispiel für Benutzerfreundlichkeit ist die gleichzeitige Validierung der Benutzereingaben sowie eine Rückmeldung bei unzulässigen Aktionen. Der direkte Wissenserwerb hat den Vorteil, dass die Wissensbasis ohne den Wissensingenieur erweitert werden kann. Allerdings betonen die Autoren in [JFI93, S.765], dass dies nur in gut verstandenen und strukturierten Anwendungsbereichen möglich sei.
Bezüglich des automatisierten Wissenserwerbs gibt es wenige Erkenntnisse, die allgemein anwendbar sind. Meistens handelt es sich um Lösungen, die nur innerhalb eines spezifischen Anwendungsbereichs funktionieren. Nichtsdestotrotz lässt sich sagen, dass dass sich

das Wissen entweder aus vorhandenen Daten (maschinelles Lernen) oder aus verfügbaren Dokumenten (automatisierte Dokumentenanalyse) generieren lässt [GTW90, S.78]. Die Implementierung hängt jedoch vom Anwendungsgebiet ab. Ein Beispiel für die Anwendung maschinellen Lernens auf große Datenmengen ist ein Diagnosesystem, das für eine Vielzahl von Fällen mit bestimmten Symptomen Diagnosen enthält. Für die Textanalyse können beispielsweise Bedienungsanleitungen analysiert werden, wobei diese Vorgehensweise gewisse Einschränkungen aufweist. Die Schwierigkeit besteht darin, dass das Erfahrungswissen nicht in den Textdokumenten zu finden ist [GTW90, S.79].
In allen Fällen des Wissenserwerbs werden die Daten an die zentrale Schnittstelle weitergereicht. Diese Schnittstelle beschäftigt sich mit der Zwischenspeicherung und der Prüfung der Datensätze auf Korrektheit, bevor die Daten endgültig in der Wissensbasis gespeichert werden. Zum Teil kann der Wertebereich direkt im einzelnen Modul des Wissenserwerbs eingeschränkt werden. Beispielsweise kann ein Eingabefeld in der Wissensträgerschnittstelle nur positive Zahlen zulassen. Für den restlichen Teil werden automatisierte Tests durchgeführt, die sicherstellen, dass neue Daten keine Inkonsistenzen in Bezug auf die Einschränkungen der Wissensbasis erzeugen [JFI93, S.765].
Zusammenfassend lässt sich die Wissenserwerbskomponente schematisch in Abbildung 5 wie folgt darstellen:

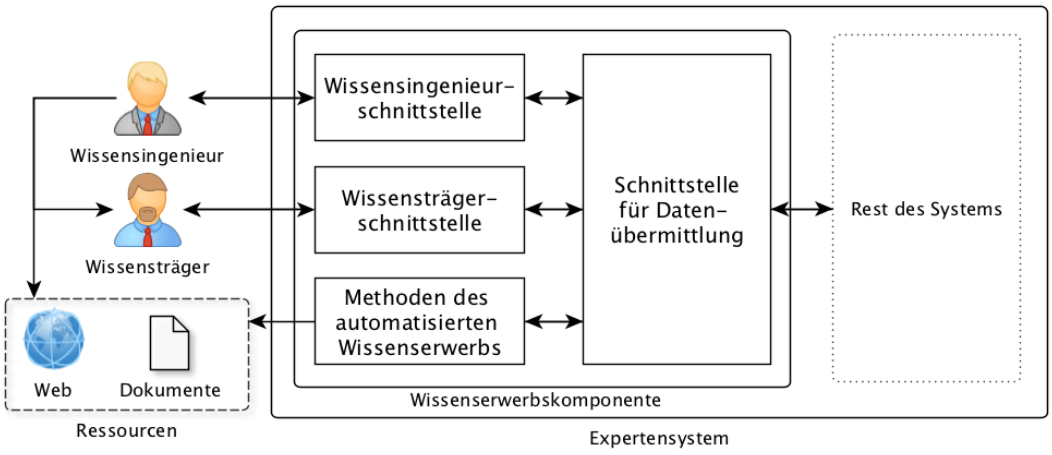

Abbildung 5: Wissenserwerbskomponente

In Abbildung 5 sieht man deutlich, dass die Wissenserwerbskomponente modular aufgebaut ist. Allgemein kann dieses Modell in jedem Expertensystem eingesetzt werden, wobei die konkrete Umsetzung in Bezug auf den Anwendungsbereich spezifiziert wird. Dabei lässt sich der Automatisierungsgrad der Datenerfassung leicht anpassen, indem der Schwerpunkt auf ausgewählte Bestandteile der Wissenserwerbskomponente gelegt wird. Beispielsweise kann sich ein Expertensystem mit umfangreichen Falldaten auf das maschinelle Lernen konzentrieren. Ein System, das bei der Datenerfassung mithilfe der Wissensträgerschnittstelle basiert, wird in [SK09, S.97] vorgestellt.
Im weiteren Verlauf der Arbeit werden die Wissensträgerschnittstelle, die Methoden des automatisierten und halb-automatisierten Wissenserwerbs sowie die Schnittstelle für Datenübermittlung thematisiert, da sie ein großes Potenzial für die Automatisierung der Datenerfassung aufweisen.

3 Methoden der Wissens- und Datenerfassung

Mit der Wissenserwerbskomponente wurde bereits angedeutet, dass die Erweiterung bzw. Aktualisierung der Wissensbasis eines Expertensystemes meistens nur teilweise automatisierbar ist. Die Autoren in [GD94] weisen ebenso darauf hin, dass manuelle und maschinelle Wissenserschließung jeweils eigene Stärken haben, die sich gegenseitig ergänzen [GD94, S.137]. Aus diesem Grund ist bei der Datenerfassung ein hybrides Modell sinnvoll, das die Vorteile manueller und maschineller Verfahren kombiniert. Aufgrund der Fragestellung dieser Arbeit wird sich im weiteren Verlauf auf den direkten sowie automatisierten Wissenserwerb beschränkt.

3.1 Schnittstelle zur Dateneingabe

Die Schnittstelle zur Dateneingabe (auch als Wissensträgerschnittstelle in Kapitel 2.4 beschrieben) ist in der Regel ein Bestandteil eines Tools zum Wissenserwerb. Diese Schnittstelle soll dem Wissensträger ermöglichen, sein Wissen auf eine einfache Weise in das System einzutragen. Da die Daten manuell eingegeben und maschinell verarbeitet werden, wird dieser Ansatz als semi-automatisiert bezeichnet. Im Zusammengang mit der Wissensträgerschnittstelle wurden zahlreiche Tools entwickelt (z.B. in [AHM03] oder [SK09]), die sich in Automatisierungsgrad, Problembereich oder Benutzergruppe unterscheiden [AHM03, S.49]. Im Folgenden wird eine Arbeit genauer betrachtet, die den Einsatz der Wissensträgerschnittstelle beispielhaft darstellt.
Das Praxisbeispiel wird in [SK09] vorgestellt und bezieht sich auf die Wissenserfassung aus der Produktion eines Elektrotechnikunternehmens. In der Fallstudie wird ein bereits bestehendes Decision Support System (DSS) betrachtet, das die Führungskräfte bei der Entscheidung nicht-strukturierter Probleme in der Produktion unterstützt [SK09, S.94]. Das DSS verfügt bereits über eine Datenbank, die allerdings nur Daten von Produkteigenschaften enthält. Die eigentlichen Prozesse werden hierbei von Experten gesteuert, die mithilfe ihres Erfahrungswissens Störungen in der Produktion beseitigen. Dieses Erfahrungswissen über Störungen wird im System nicht erfasst. Als Folge hat die Unternehmensführung nur einen begrenzten Überblick über die Situation in der Produktionsabteilung. Außerdem besteht die Gefahr, dass das spezifische Expertenwissen verloren geht, falls der Wissensträger das Unternehmen verlässt [BZL16, S.467].
Die Zielsetzung von [SK09] ist die Erfassung des Expertenwissens aus der Produktion und die Integration dieses Wissens in die Entscheidungsprozesse auf der Organisationsebene. Um das Wissen aus der Produktionsabteilung zu erschließen, erweitern Gebus und Leiviskä das bestehende System um eine Schnittstelle für Anlagenbediener. Die Umsetzung erfolgt in Form eines Prototyps. Der Entwicklungsprozess lässt sich allgemein wie folgt beschreiben:

1. Die Definition der Zielgruppe, die für die Schnittstelle relevant ist.
2. Ausgehend von der Zielgruppe werden die Wissensrepräsentation sowie funktionale und nicht-funktionale Anforderungen spezifiziert.
3. Die Umsetzung und Evaluation des Prototyps.

Im ersten Schritt definieren Gebus und Leiviskä die bestehenden Nutzergruppen des gesamten Systems [SK09, S.97]:

- Anlagenbediener (Experte), der sein Wissen zu den Störungen mittels der Wissensträgerschnittstelle in die Datenbank eingibt,
- Administrator, der das gesamte System verwaltet,
- Qualitätsabteilung, die die Störungsstatistik analysiert, eine Qualitätsrückmeldung an die Produktion und einen Bericht an die Führungskraft übermittelt,
- Führungskraft, die eine umfangreiche Übersicht von der Qualitätsabteilung erhält und davon ausgehend Entscheidungen zur Prozessoptimierung trifft.

Im Rahmen der Fallstudie ist die Nutzergruppe der Anlagenbediener relevant, wobei das System jeder Nutzergruppe eine geeignete Benutzerschnittstelle zur Verfügung stellt. Schematisch lässt sich die Struktur und die Informationsflüsse im DSS in Abbildung 6 vereinfacht nachbilden.

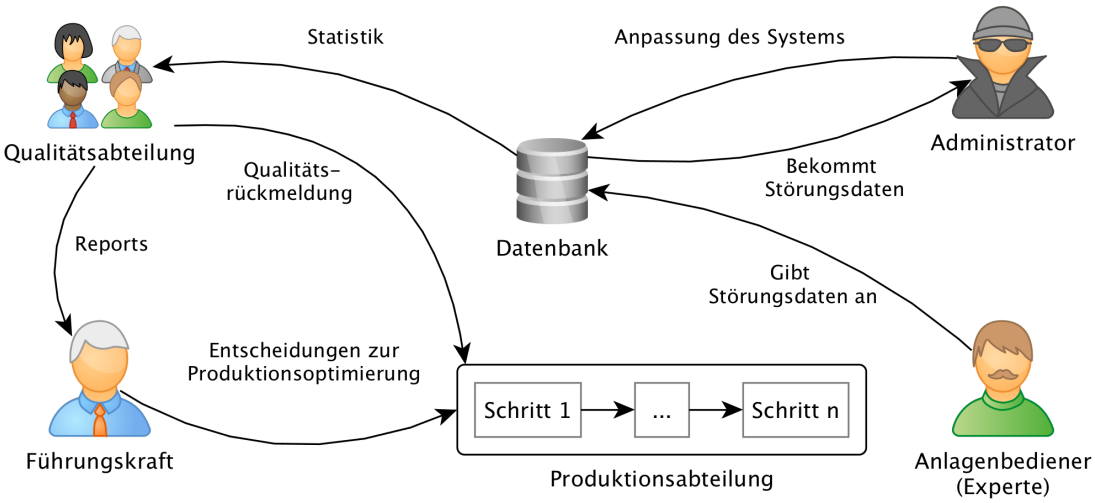

Abbildung 6: Die Struktur und die Informationsflüsse im DSS, [SK09, S.98]

Für die Informationsrepräsentation werden die Anlagenbilder benutzt. Die Autoren betonen, dass die Expertenschnittstelle so einfach und intuitiv wie möglich gehalten wurde, um die Dateneingabe zu einer alltäglichen Tätigkeit zu machen [SK09, S.97]. Im Hinblick auf die nicht-funktionalen Anforderungen werden beispielsweise Einstellungen und Maschinenbilder vom Administrator vorkonfiguriert und automatisch beim Hochfahren des Systems geladen. Die funktionalen Anforderungen umfassen die Erfassung einer Störung durch die Auswahl des passenden Anlagebildes und das Markieren der betroffenen Stelle. Darauffolgend wird eine Vorlage mit möglichen Ursachen zur Auswahl angezeigt. Zusätzlich gibt es ein Feld zur Freitexteingabe, wenn die vorliegende Störung im System noch nicht vorhanden ist [SK09, S.99].
In der Evaluation haben sich folgende benutzerorientierte Kriterien für die weitere Systemverbesserung ergeben [SK09, S.100]:

- *Usability (engl. Gebrauchstauglichkeit)*: Wie einfach und intuitiv ist das System zu verwenden.
- *Usefulness (engl. Nützlichkeit)*: Wie nützlich ist das System für primäre (Anlagenbediener) und sekundäre (z.B. Qualitätsabteilung) Nutzer.
- *Usage (engl. Nutzung)*: Inwiefern wird das System verwendet.

Wenn es um die Benutzerinteraktion mit einem System geht, spielt Usability die zentrale Rolle. Gemäß ISO 9241-11 wird Usability durch folgende Aspekte definiert: Effektivität (wurde das Ziel erreicht), Effizienz (wie schnell wurde das Ziel erreicht) und Zufriedenheit (positive Erfahrung bei der Systemnutzung) [ISO98]. Usefulness wird in [Ste14] behandelt. Steve Krug bezeichnet einen Gegenstand als nützlich, wenn eine Person mit durchschnittlichen Fähigkeiten herausfinden kann, wie dieser Gegenstand zu bedienen ist, um ein Ziel zu erreichen. Dabei soll der Aufwand kleiner als der Wert des Ziels sein [Ste14, S.9].
Statistisch gesehen gab es in der betrachteten Testperiode 183 Anlagenausfälle. Allerdings wurden nur 70 Fälle kommentiert, was eine Nutzungsrate des vorgestellten Informationsaustauschsystems von 38% ergibt. Laut Gebus und Leiviskä lag es daran, dass der Prototyp aus der Usability-Perspektive nicht optimal war. Beispielsweise waren einige Elemente der Schnittstelle eher verwirrend. Nach den notwendigen Anpassungen konnte die Nutzungsrate fast auf 100% gesteigert werden. Hinsichtlich der Nützlichkeit gab es positive Bewertungen. Es wurde festgestellt, dass die Umwandlung des Systems aus einer reinen Datensammlung über Störungen zu einem Informationsaustauschsystem Vorteile für alle Nutzergruppen bringt. In Abbildung 6 sieht man deutlich, dass die Informationen, die in der Datenbank gespeichert werden, von allen Beteiligten genutzt werden. Der Administrator kann mithilfe der Daten das System entsprechend anpassen. Die Qualitätsabteilung setzt die Information zur Berichterstellung für die Führungskräfte und die Produktion ein. Die Unternehmensführung erhält ein umfangreicheres Bild über die Produktionslage und kann aufgrund dessen effizientere Entscheidungen bei der Produktionsoptimierung treffen.
Zusammenfassend lässt sich sagen, dass die Entwicklung der Wissensträgerschnittstelle aus drei Phasen besteht. In der ersten Phase wird die Zielgruppe definiert. Im Hinblick auf die Zielgruppe werden in der zweiten Phase die geeignete Informationsrepräsentation sowie die funktionalen und nicht-funktionalen Anforderungen festgelegt. Die dritte Phase umfasst die Implementierung mit der darauffolgenden Evaluation, in dem die Schnittstelle nach drei Kriterien bewertet wird, nämlich Usability, Usefulness und Usage.

3.2 Datenerfassung aus dem Web

Der Prozess der Webdatenerfassung umfasst die Datenextraktion aus unstrukturierten bwz. semi-strukturierten Webdokumenten (z.B. eine Webseite oder eine E-Mail) und die Transformation der erfassten Daten in eine strukturierte Form für spätere Verwendung [EPGR14, S.301]. Im Folgenden werden die allgemeinen Probleme bei der Erfassung der Webdaten angesprochen. Anschließend werden generelle Paradigmen der Datenerfassung erläutert, nämlich Baumparadigma, Web-Wrapper und hybrides System.
Bei der Erfassung der Webdaten gibt es mehrere Faktoren, die berücksichtigt werden sollen. Ferrara et al. [EPGR14, S.302] identifizieren folgende Herausforderungen:

- *Automatisierungsgrad*: Die Erfassung der Webdaten soll regelmäßig von einem menschlichen Experten überwacht werden, um die Genauigkeit der Daten zu gewährleisten.
- *Skalierbarkeit*: Bei umfangreichen Webressourcen soll innerhalb kurzer Zeit eine große Datenmenge bearbeitet werden.
- *Datenschutz*: Wenn es um die Erfassung der personenbezogenen Daten geht (bei sozialen Netzwerken wie Facebook), soll die Privatsphäre des Individuums nicht beeinträchtigt werden.
- *Änderung der Ressourcenstruktur*: Die Struktur von Webressourcen ändert sich oft. Die Datenerfassungsmethoden für das Web sollen eine gewisse Flexibilität besitzen, um weiterhin korrekt zu funktionieren.
- *Trainingsdaten*: Bei Einsatz des maschinellen Lernens ist eine ausreichende Trainingsmenge an Webseiten erforderlich, die manuell vorbereitet wird. Dies ist eine aufwendige und fehleranfällige Aufgabe.

Das Baumparadigma nutzt die Baumstruktur einer Webseite aus, um die gewünschten Daten zu erfassen. Dabei handelt es sich um Dokumente, die in Hypertext Markup Language (HTML) beschrieben sind. Eine HTML-Seite wird als Document Object Model (DOM)[4] definiert. Die Idee des DOM besteht darin, dass eine HTML-Webseite einen Baum darstellt, der mittels HTML-Tags (z.B. Button-Tag) ausgezeichnet wird. Tags können weitere Tags beinhalten und bilden somit eine hierarchische Struktur. Diese hierarchische Baumstruktur ermöglicht eine effiziente Datensuche in einer HTML-Seite [EPGR14, S.303].
Da HTML Eigenschaften einer Extensible Markup Language (XML) integriert, kann XML Path Language (XPath)[5] für die Navigation im DOM eingesetzt werden. In einem XPath-Ausdruck können beliebige Elemente einer HTML-Webseite ausgewählt werden. In Abbildung 7 werden zwei Beispiele dargestellt. Im ersten Fall (A) wird genau ein Element (die erste Zelle in der ersten Zeile) ausgewählt. Im Beispiel (B) werden mehrere Elemente (alle Zellen der zweiten Zeile) angesprochen [EPGR14, S.303].

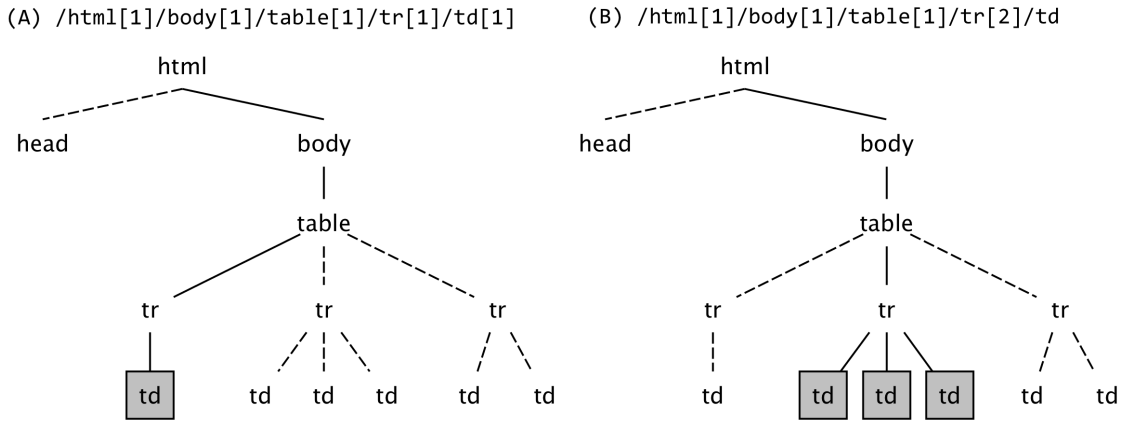

Abbildung 7: XPath im Dokumentenbaum, [EPGR14, S.304]

[4]https://www.w3.org/DOM
[5]https://www.w3.org/TR/xpath

Der Hauptnachteil von XPath besteht darin, dass XPath-Ausdrücke strikt an die DOM-Struktur gebunden sind. Wenn eine Änderung im DOM stattfindet, funktioniert der von der Änderung betroffene Ausdruck nicht mehr. Aus diesem Grund müssen die XPath-Ausdrücke nach jeder Veränderung der HTML-Webseite manuell angepasst werden. Hinsichtlich dieses Problems wurden im letzten Release von XPath[6] relative XPath-Ausdrücke eingeführt [EPGR14, S.304].
Ein weiterer Ansatz, die Webdaten zu erfassen, stellt ein Web-Wrapper dar. Ein Web-Wrapper umfasst in der Regel einen oder mehrere Algorithmen, die zur Datenerfassung aus den Webdokumenten eingesetzt werden. Anschließend werden die erfassten Daten in eine strukturierte Form transformiert und für die weitere Nutzung gespeichert. Ein Web-Wrapper umfasst folgende Schritte [EPGR14, S.305]:

1. *Generierung*: Definition des Wrappers.
2. *Ausführung*: Datenerfassung mithilfe des Wrappers.
3. *Wartung*: Anpassung des Wrappers bei der Änderung der DOM-Struktur.

Nach Ferrara et al. [EPGR14, S.306] kann ein Web-Wrapper mittels folgender Ansätze generiert und ausgeführt werden:

- *Reguläre Ausdrücke*: Daten werden gemäß Regeln (expressions) gewonnen. Im Rahmen dieser Arbeit wird sich auf diesen Ansatz der regulären Ausdrücke beschränkt.
- *Logikbasierter Ansatz*: Zur Datenerfassung wird eine Wrapper-Programmiersprache eingesetzt (wrapper programming language).
- *Baumbasierter Ansatz*: Es wird die Annahme getroffen, dass bestimmte Bereiche im DOM generell für Daten zuständig sind. Die Identifikation und Datenextraktion aus diesen Bereichen ist der Gegenstand des baumbasierten Ansatzes.
- *Maschinelles Lernen*: Daten werden mithilfe eines Lernalgorithmus und einer Trainingsmenge erfasst.

Reguläre Ausdrücke ermöglichen die Erkennung von Mustern in unstrukturierten bzw. semi-strukturierten Dokumenten unter Verwendung von Regeln, die z.B. in Form von Wortgrenzen oder HTML-Tags definiert werden. Der Vorteil der regulären Ausdrücke besteht in der Möglichkeit, ein beliebiges Element auf einer Webseite anzusprechen. Außerdem bieten einige Implementierungen ein grafisches Benutzerinterface, sodass der Benutzer die Elemente auf einfache Weise auswählen kann. Die für die Erkennung benötigten Regeln werden dann automatisch generiert. Eine mögliche Umsetzung des Wrappers wird in [AF99] mit W4F vorgestellt. Dieses Tool verfügt über eine Hilfsmethode, die den Benutzer bei der Auswahl der Elemente unterstützt. Auf Basis der ausgewählten Elemente werden die Regeln erstellt. Allerdings sind die Regeln in Bezug auf DOM-Änderungen nicht flexibel und können sehr schnell verletzt werden [EPGR14, S.306].
In der dritten Phase geht es um die Anpassung des Web-Wrappers im Hinblick auf Veränderungen der DOM-Struktur. Die Anpassung erfolgt entweder manuell oder in gewisser

[6] https://www.w3.org/TR/xpath20

Weise automatisiert. Ferrara et al. [EPGR14, S.308] betonen, dass der Automatisierungsgrad der Wartung besonders kritisch ist. Bei einer kleinen Dokumentenanzahl ist die manuelle Anpassung noch akzeptabel. Allerdings ist die manuelle Wartung bei einer großen Menge von Dokumenten nicht mehr realisierbar.
Ein Beispiel der automatisierten Wrapper-Wartung wird in [XDC03] mit dem System namens SG-WRAM (Schema-Guided Wrapper Maintenance for Web-Data Extraction) vorgestellt. Die Architektur vom SG-WRAM wird in Abbildung 8 dargestellt.

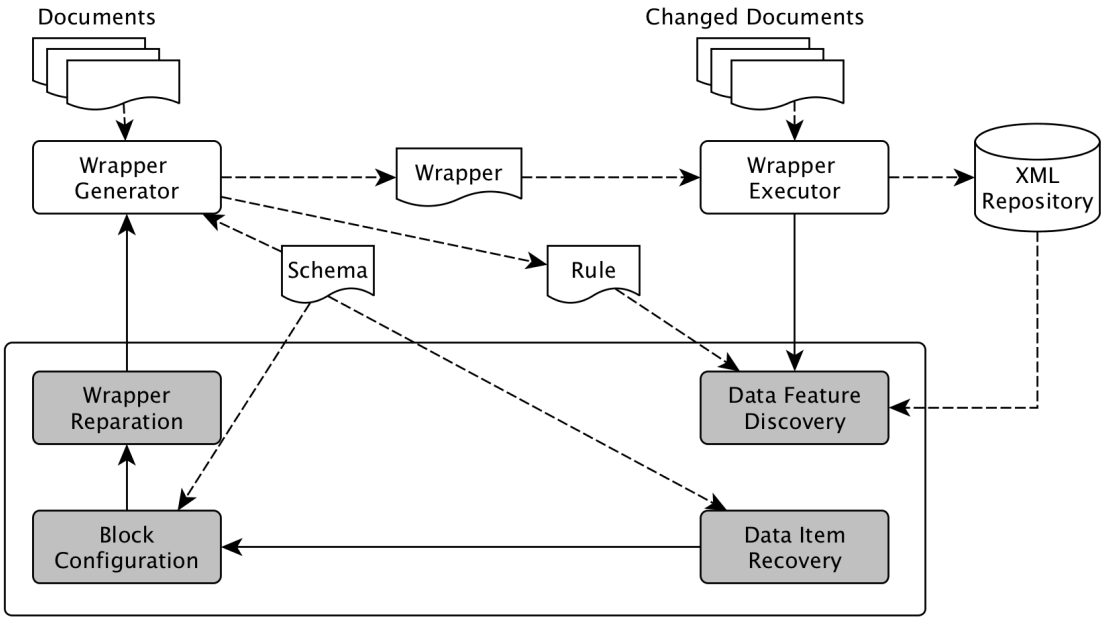

Abbildung 8: Die Architektur vom SG-WRAM, [XDC03, S.2]

Im ersten Schritt werden die HTML-Seite, das XML-Schemata und das Mapping zwischen diesen in den Wrapper-Generator eingegeben. Das XML-Schema wird durch Document Type Definition (DTD) beschrieben. Daraufhin generiert das System die Regeln (Wrapper) für die gegebene Webseite, um die Daten zu erfassen und in einer XML-Datei gemäß der DTD-Datei zu speichern. Neben der Datenextraktion werden zusätzlich die Probleme bei der Extraktion erfasst, um ein Wiederherstellungsprotokoll für die fehlgeschlagenen Regeln zu erzeugen. Wenn das Protokoll die Fehler beseitigt wird die Datenerfassung fortgesetzt. Bei auftretenden Fehlern werden Warnungen und Benachrichtigungen angezeigt, dass die Regeln nicht mehr funktionieren. In diesem Fall müssen die Regeln manuell von einem Experten angepasst werden [EPGR14, S.308].
Als Ausblick wird ein hybrides System der Webdatenerfassung angesprochen. Ein Beispiel des hybriden Ansatzes stellt RoadRunner von [VGP01] dar. RoadRunner kann sowohl mit der Trainingsmenge von einem Nutzer als auch mit selbst erstellten Trainingsdaten ausgeführt werden. Das Verfahren arbeitet gleichzeitig mit zwei HTML-Seiten und analysiert die Gemeinsamkeiten und die Unterschiede zwischen diesen Seiten, um die Muster zu finden. Im Allgemeinen kann das Verfahren die Daten aus beliebigen Quellen erfassen, solange mindestens zwei Seiten mit ähnlicher Struktur gegeben sind. Da Webseiten normalerweise dynamisch auf Basis eines Templates generiert werden, befinden sich relevante Daten in gleichen oder ähnlichen Bereichen [EPGR14, S.309].

3.3 Maschinelles Lernen beim Wissenserwerb

In diesem Abschnitt wird ein Ausblick in weitere Wissenserwerbsmethoden gegeben, und zwar die Anwendung des maschinellen Lernens. In der Literatur gibt es zahlreiche Ansätze, die dieses Themenfeld bearbeiten. Im Rahmen dieser Arbeit wurden die Ansätze aus [JJJ01] und [Geo96] betrachtet. Trotz den Unterschiede in der Umsetzung wird die Zusammenarbeit zwischen dem Experten und dem Lernalgorithmus betont. Im Folgenden wird der Ansatz von [JJJ01] genauer betrachtet.
In der Arbeit von [JJJ01] handelt es sich um die Anwendung des maschinellen Lernens bei der Erfassung von Expertenwissen im Medizinbereich. Die Idee des Lernverfahrens besteht in der Erschließung des naheliegenden Expertenwissens aus den Trainingsdaten der Schlussfolgerungen eines Experten. Das Konzept der Wissensbasiserweiterung orientiert sich an [Ghe92] und besteht darin, dass anfangs eine initiale Wissensbasis aufgebaut und inkrementell verbessert wird, indem der Experte die Fragen vom System beantwortet. Bei der Erstellung der Fragen wird ein Lernverfahren eingesetzt.
Der Kern des Verfahrens besteht aus dem induktiven Lernen und dem generischen Algorithmus von [JJJ99], der auf der Fuzzylogik basiert. Die Fuzzylogik beschäftigt sich mit den graduellen Aussagen, die nicht eindeutig als falsch oder wahr bezeichnet werden können. Diese Aussagen werden mithilfe der vagen Prädikate beschrieben. Ein typisches Beispiel ist das Prädikat „groß“. Eine 1,80 m große Person wird als „groß“ bezeichnet, während eine 1,75 m Person nicht genau so „groß“, aber auch nicht „klein“ ist [CG14, S.27].
Das Lernverfahren umfasst folgenden Schritte [JJJ01, S.316]:

1. Eingabe der Trainingsmenge θ.
2. Entfernen von Noise aus den Trainingsdaten.
3. Ermittlung der Gene, die sich nicht ändern können.
4. Ermittlung der Menge der initialen Regeln mithilfe des Algorithmus von [JJJ99].
5. Anwendung des Algorithmus von [JJJ99] zur Ermittlung der Menge der Regeln, die das naheliegende Expertenwissen aus der Trainingsmenge beschreiben.
6. Ausschluss der Regeln, die ein Bestandteil anderer Regeln sind.

Aufgrund der hohen Komplexität und des spezifischen Anwendungsbereichs des Verfahrens wird sich im Weiteren auf den ersten Schritt beschränkt, der das grundlegende Verständnis zu dem vorliegenden Ansatz liefert.
Als erstes definieren Castro et al.[JJJ01, S.309] die Wissensrepräsentation. Es werden folgende Informationstypen unterschieden:

- *Numerische Information*, die aus einer empirischen Beobachtung stammt und als eine Beispielmenge von Input-Output-Beziehungen erfasst wird.
- *Sprachinformation*, die von einem Experten stammt und in Form von IF-THEN-Regeln beschrieben wird.

Die Trainingsmenge wird folgendermaßen definiert (siehe Formeln 1 und 2):

$$\theta = \{e_1 \dots e_m\} \tag{1}$$

$$e_i = ((x_{i0}, \dots, x_{in}), y_j) \tag{2}$$

wobei:

e_i = eine Schlussfolgerung,

m = Anzahl der vorhandenen Schlussfolgerungen,

x_{in} = Input-Variable,

n = Anzahl der Input-Variablen in der Schlussfolgerung,

y_j = Output-Wert (Expertenschlussfolgerung).

Die Output-Werte nach der Formel 2 stellen die Teilmenge des kartesischen Produkts aller Input- und Output-Werte $X^n \times Y$ dar. Das Ziel ist die Approximation der Funktion $\Omega : X^n \to Y$ durch die Ermittlung von Fuzzy-IF-THEN-Regeln, um das naheliegende Expertenwissen aus der Trainingsmenge zu erschließen. Fuzzy-IF-THEN-Regel R_j wird wie folgt definiert (siehe Formel 3):

$$R_j : \text{ IF } X \text{ is } E \text{ THEN } Y \text{ is } y_i \tag{3}$$

wobei:

X = Menge der Variablen in den Schlussfolgerungen,

E = Teilmengen der Fuzzy-Labels,

Y = Expertenschlussfolgerung.

Castro et al. [JJJ01, S.309] treffen die Annahme, dass die Teilmengen der Fuzzy-Labels E aus einer endlichen Menge der Labels $\mathcal{L}$ entnommen sind (siehe Formel 4):

$$\mathcal{L} = \{L_{i1}, \dots, L_{ik}\} \tag{4}$$

Dabei ist k die Anzahl der Labels und E_i ist eine Expertenschlussfolgerung, die ein Element der Funktionsmenge $\mathcal{P}(\mathcal{L}_i)$ ist, oder zusammenfassend $E_i \in \mathcal{P}(\mathcal{L}_i)$.
Das Ergebnis der Durchführung des Lernverfahrens ist die Menge von Regeln, die bei der Erstellung der Fragen verwendet wird. Dabei kann die Differenzstrategie eingesetzt werden [JJJ01, S.317]. Bei der Differenzstrategie werden die Regeln gesucht, die zu einem Konflikt führen, wenn sie gleichzeitig angewandt werden. Um den Konflikt zu lösen, kann der Experte eine neue Variable zulassen. Alternativ kann der Experte eine neue Meta-Regel definieren, wie z.B wenn R_i und R_j sich widersprechen, soll R_i bevorzugt werden, da es die allgemeinere Klasse repräsentiert [JJJ01, S.318].

4 Umsetzung der Wissensträgerschnittstelle

Nachdem die Grundlagen der wissensbasierten Systeme und Wissenserfassung betrachtet wurden, behandelt dieses Kapitel die Umsetzung der Wissensträgerschnittstelle am Beispiel von *PaaSfinder*. Als erstes wird *PaaSfinder* kurz vorgestellt. Danach wird das Konzept und die Umsetzung der Wissensträgerschnittstelle zum Aktualisieren der bestehenden Daten erläutert. Anschließend wird die Schnittstelle zur Datenübermittlung betrachtet.

4.1 Systembeschreibung

Die Umsetzung der Wissenserwerbskomponente erfolgt am Beispiel von *PaaSfinder*[7], einem Expertensystem und Open-Source-Projekt im Bereich von Platform-as-a-Service (PaaS). Das Ziel des Systems besteht darin, zahlreiche PaaS-Anbieter (im Weiteren Vendors) vergleichbar zu machen. *PaaSfinder* verfügt bereits über eine umfangreiche Wissensdatenbank, die aufgrund der häufigen Änderungen im gegebenen Anwendungsbereich regelmäßig aktualisiert werden muss. Die Struktur von *PaaSfinder* lässt sich wie in Abbildung 9 folgendermaßen beschreiben.

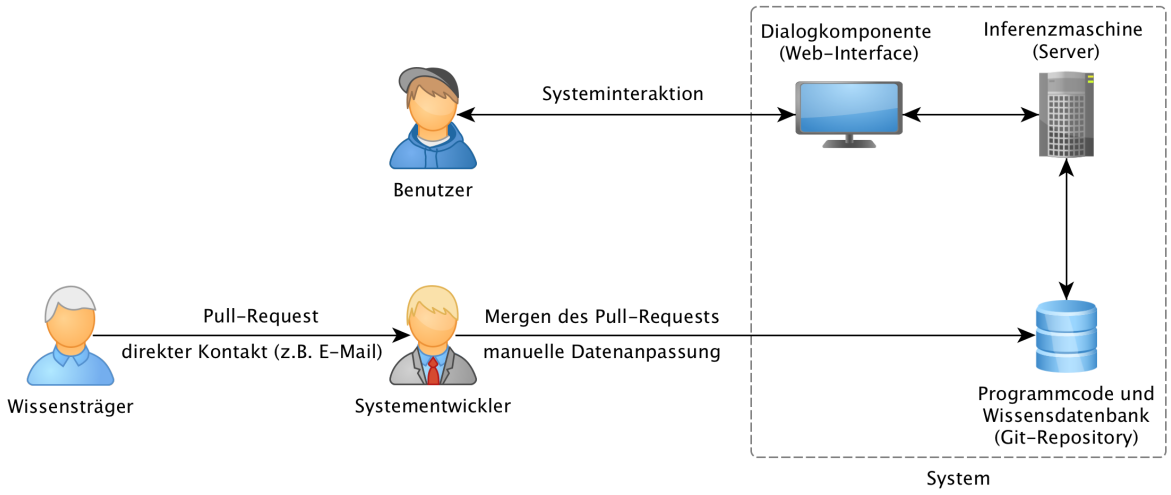

Abbildung 9: Die Struktur vom *PaaSfinder*

PaaSfinder setzt sich aus dem Programmcode der Wissensdatenbank und der Webanwendung zusammen. Die Webanwendung wird wiederum auf einem Server betrieben. Ein Vendor wird in einem Profil[8] beschrieben, das wie folgt aufgebaut ist:

1. **General Properties**: allgemeine Eigenschaften des Vendors (z.B. Name, URL, Status, Typ etc.)
2. **Extensible**: generelle Erweiterungsmöglichkeit nach Kundenbedarf (z.B. spezielle Laufzeitumgebung)

[7] https://paasfinder.org
[8] https://github.com/stefan-kolb/paas-profiles#profile-specification

3. **Pricing**: verfügbare Preismodelle
4. **Quality of Service**: die Servicequalität (z.B. Verfügbarkeit)
5. **Hosting**: Art der Bereitstellung (z.B. privat)
6. **Scaling**: Skalierbarkeit (z.B. Speichererweiterung)
7. **Runtimes**: unterstützte Laufzeitumgebungen (z.B. Java[9])
8. **Middleware** (z.B. Tomcat[10])
9. **Frameworks** (z.B. Play[11])
10. **Services** (z.B. Datenspeicher)
11. **Infrastructures** (z.B. Informationen zum Standort)

Ein Vendor wird als JSON[12]-Eintrag gespeichert. JSON stellt ein Datenaustauschformat dar und basiert auf zwei Datenstrukturen:

- *Name/Wert Paar*, das ein Objekt darstellt (siehe Abbildung 10).
- *Eine geordnete Liste von Werten (Array)*, die kein oder mehrere durch Komma getrennte Objekte enthält (siehe Abbildung 11).

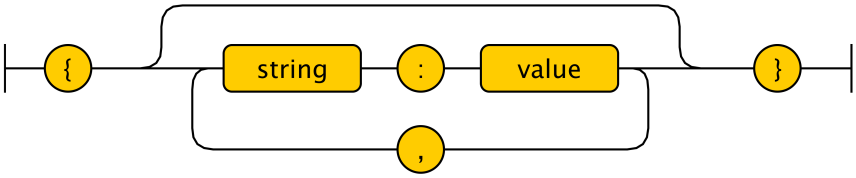

Abbildung 10: JSON-Objekt Spezifikation

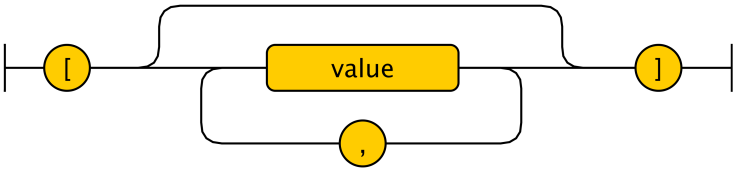

Abbildung 11: JSON-Array Spezifikation

Die Wissensdatenbank und der Programmcode von *PaaSfinder* befinden sich in einem Git-Repository. Bei Git[13] handelt es sich um ein verteiltes System zur Projektverwaltung.

[9] https://java.com
[10] https://tomcat.apache.org
[11] https://www.playframework.com
[12] http://json.org
[13] https://git-scm.com

Da *PaaSfinder* ein Open-Source-Projekt ist, kann im Prinzip jeder die Wissensdatenbank erweitern bzw. modifizieren. Momentan kann das auf zwei Weisen erfolgen (siehe Abbildung 9):

- *Pull Request*[14]
- *Direkter Kontakt mit dem Systementwickler*

Beim Pull Request wird zuerst das gesamte Git-Repository lokal gespeichert. Danach werden Änderungen vorgenommen und anschließend ein Pull Request erstellt. Der Pull Request wird vom Projektmaster überprüft und zugehörige Änderungen übernommen („gemerged"), falls keine Konflikte vorliegen. Wenn der Wissensträger mit dem Git-System nicht vertraut ist, kann direkter Kontakt (beispielsweise per E-Mail) mit dem Systementwickler aufgenommen werden. Die Daten werden dann manuell vom Systemverwalter in die Wissensbasis eingetragen.
Sowohl Pull Request als auch der Kontakt mit dem Systementwickler entsprechen dem direkten Wissenserwerb gemäß der Klassifikation in Kapitel 2.4. Der Vorteil dabei ist, dass die Daten nicht manuell gesucht werden müssen, sondern gleich vom Wissensträger stammen. Andererseits wird die Datenerfassung durch Git-Kenntnisse des Wissensträgers beschränkt. Um den direkten Wissenserwerb zu erleichtern, wird im weiteren Verlauf die Wissensträgerschnittstelle entwickelt, die es dem Wissensträger ermöglicht, einen bestehenden Vendor zu aktualisieren. Nach dem erfolgreichen Absenden der Änderungen soll automatisch ein Pull Request erstellt werden.

4.2 Wissensträgerschnittstelle

Als erster Schnitt in der Automatisierung der Datenerfassung bei *PaaSfinder* wird eine Schnittstelle zur Dateneingabe entwickelt (Wissensträgerschnittstelle). Mithilfe der Schnittstelle soll die Aktualisierung eines bestehenden Vendors ermöglicht werden. Das Anlegen oder Löschen eines Vendors liegt nicht im Anwendungsbereich der Schnittstelle. Der Ablauf wird in Abbildung 12 veranschaulicht.

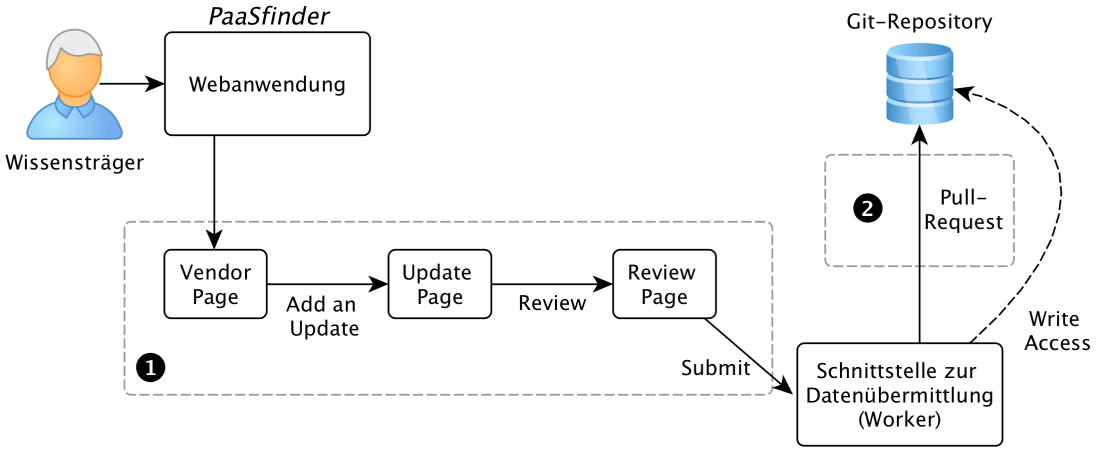

Abbildung 12: Profilaktualisierung mittels Wissensträgerschnittstelle

[14]https://git-scm.com/docs/git-request-pull

Der Ablauf in Abbildung 12 lässt sich in zwei Schritte aufteilen. Dieser Abschnitt beschäftigt sich mit dem ersten Schritt, der die Daten von einem Wissensträger entgegennimmt, zusammenfasst und abschickt. Die Verarbeitung dieser Daten erfolgt im zweiten Schritt, der im nächsten Abschnitt besprochen wird. Allgemein besteht der erste Schritt aus folgenden Aktionen:

1. *Vendorauswahl* (Vendor Page)
2. *Aktualisieren* des Profils (Update Page)
3. *Überprüfung* der Daten (Review Page)
4. *Absenden* der Daten (Submit)

Der gesamte Prozess lässt sich in Abbildung 13 als Aktivitätsdiagramm darstellen. Die verwendete Notation entspricht dem UML-Standard 2.5 [15]. Es werden Knoten (Rechtecke) für Aktivitäten und Kanten (Pfeile) für Verbindungen verwendet. Bei den Verbindungen werden Kontrollfluss- und Datenflussverbindungen unterschieden. Im Fall einer Datenflussverbindung gibt es einen Ein- und einen Ausgabe-Pin (kleiner Quadrat). Im vorliegenden Fall werden zusätzlich Fallunterscheidungen verwendet, die durch eine Raute gekennzeichnet sind.

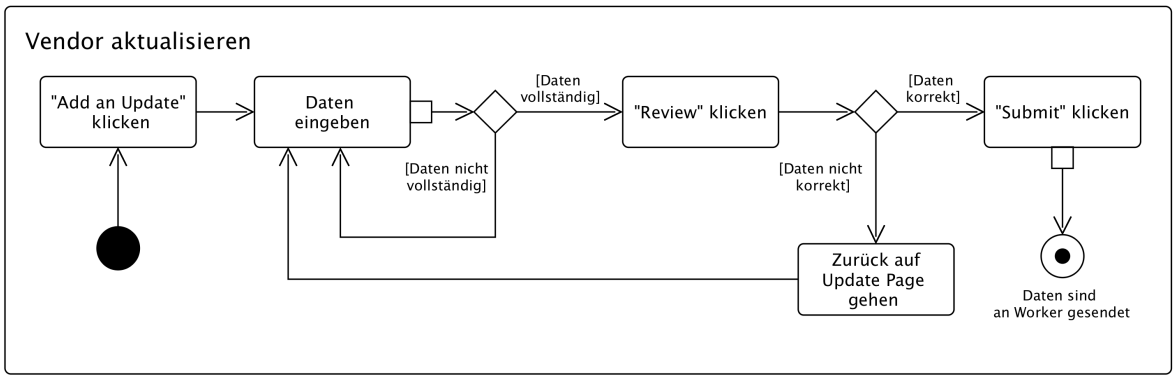

Abbildung 13: Aktivitätsdiagramm der Aktualisierung eines Vendors

Ausgehend von einer Vendor Page wird der Ablauf mit „Add an Update“ gestartet. Als nächstes werden die Daten eingegeben. Wenn ein Vendor aus Sicht des Wissensträgers vollständig ist, wird auf „Review“ geklickt. Sonst wird der Schritt der Dateneingabe wiederholt. Falls der „Review“-Button geklickt wurde, gelangt man zur Review Page. Hier können die Daten überprüft werden. Wenn die Daten korrekt sind, können die Daten mit dem Klick auf „Submit“ an den Worker gesendet werden. Andernfalls kann der Wissensträger zurück zur Dateneingabe gehen und die Daten überarbeiten. Zu jedem Zeitpunkt besteht die Möglichkeit, den Vorgang abzubrechen, indem das Browserfenster geschlossen wird. Die Daten werden nur dann gesendet, wenn der „Submit“-Button auf der Review Page geklickt wurde. Im weiteren Verlauf wird die Implementierung der Update Page und Review Page beschrieben.

[15] http://www.omg.org/spec/UML/2.5/PDF

Die Update Page umfasst zwei Aspekte. Erstens sollen die Vendordaten für den Wissensträger geeignet dargestellt werden. Zweitens soll die Eingabe neuer und die Änderung bestehender Daten ermöglicht werden. Außerdem sollen die Änderungen am Profil dynamisch zwischengespeichert werden. Das ist eine herausfordernde Aufgabe, da ein Vendor, abgesehen von allgemeinen Eigenschaften (z.B. „name“), aus komplexen und teils verschachtelten Datentypen besteht. In Abbildung 14 wird dies am Beispiel von „runtimes“ (Laufzeitumgehungen) gezeigt. Gelb steht dabei für ein Objekt, grau für eine Liste und weiß für einen String (Zeichenkette). Hier umfasst ein Vendor eine Liste von „runtimes“. „Runtime“ besteht wiederum aus „language“ und „versions“.

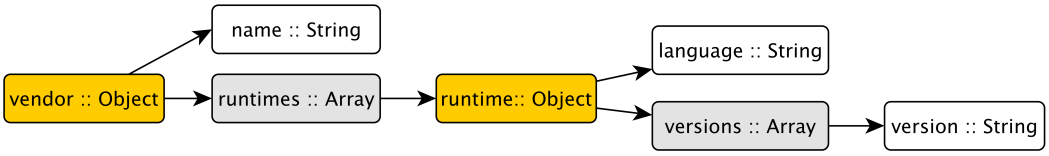

Abbildung 14: Ausschnitt aus Vendor Datentyp

Um diese Aufgabe zu lösen, werden die Daten in zwei Richtungen gebunden (two-way data binding). Zu diesem Zweck wird ein Open-Source Framework Knockout.js[16] eingesetzt. Knockout.js ist eine JavaScript Bibliothek zur Erstellung dynamischer Inhalte. Das Framework basiert auf dem Model View ViewModel (MVVM) Entwurfsmuster[17]:

- *Model*: Daten, die unabhängig auf einem Server liegen (hier: verfügbare Vendors).
- *ViewModel*: Programmcode der Daten und Operationen auf der Benutzerschnittstelle (hier: JavaScript Klasse für Vendor (Modell)).
- *View*: Benutzerschnittstelle, die den Zustand des ViewModel abbildet und bei Benutzeraktionen aktualisiert (hier: Update Page).

Als erstes werden die für das Modell erforderlichen Daten über die *PaaSfinder*-API[18] geholt und das Modell initialisiert. Da das Modell komplexe Datentypen umfasst, sind explizite JavaScript Klassen nötig. Beispielsweise gibt es für „runtime“ und „version“ eigene Klassen mit jeweils unterschiedlichem Verhalten. Die Klasse „runtime“ beinhaltet z.B. die Methode zum Hinzufügen einer neuen Version. Nachdem das Modell initialisiert wurde, wird das Data-Binding zwischen dem Modell und der View aktiviert.
Während die Update Page die Benutzerdaten entgegennimmt, werden die Daten auf der Review Page zusammengefasst, sodass der Wissensträger die Eingaben überprüfen kann. Die Profildaten von der Update zur Review Page werden mithilfe von Web Storage[19] ausgetauscht. W3C definiert zwei Arten der Speicherung, nämlich Session und Local Storage. Session Storage gilt ausschließlich innerhalb des Browserfensters. Auf die Inhalte des Local Storage kann dagegen innerhalb einer Domain zugegriffen werden und diese gelten zeitlich unbeschränkt. Aus diesem Grund wird im Weiteren das Local Storage betrachtet.
Auf der Update Page wird ein Vendor beim Klicken auf „Review“ in das Local Storage

[16] http://knockoutjs.com
[17] http://knockoutjs.com/documentation/observables.html
[18] https://paasfinder.org/api/vendors/
[19] https://www.w3.org/TR/webstorage/

gespeichert. Das Speichern bzw. Lesen basiert wie bei JSON auf dem Schlüssel/Wert-Paar-Prinzip. Der Schlüssel ist dabei der Name des Vendors, der bei der API benutzt wird. Da der Schlüssel als Parameter an die Review Page geschickt wird, können die Daten problemlos gelesen werden. Es besteht ebenso die Möglichkeit, den Vendor auf der Update Page zurückzusetzen, indem die Vendordaten aus dem Local Storage gelöscht und erneut von der *PaaSfinder*-API angefordert werden.
Im folgenden Beispiel wird der Vendor „Heroku“ aktualisiert, indem eine neue Version (1.9) zu Java in „runtimes“ hinzugefügt wird.

1. Auf der Vendor Page auf „Add an Update“ klicken.

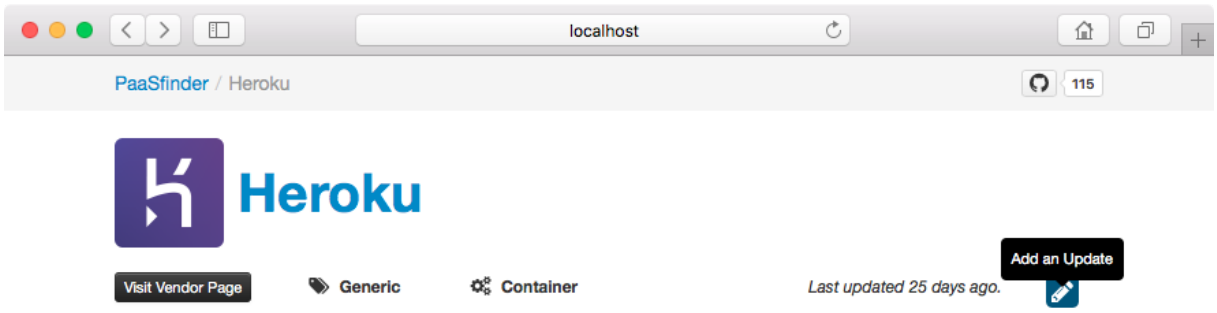

Abbildung 15: Vendor Page

2. Eine neue Java Version (1.9) hinzufügen (siehe Abbildung 16).

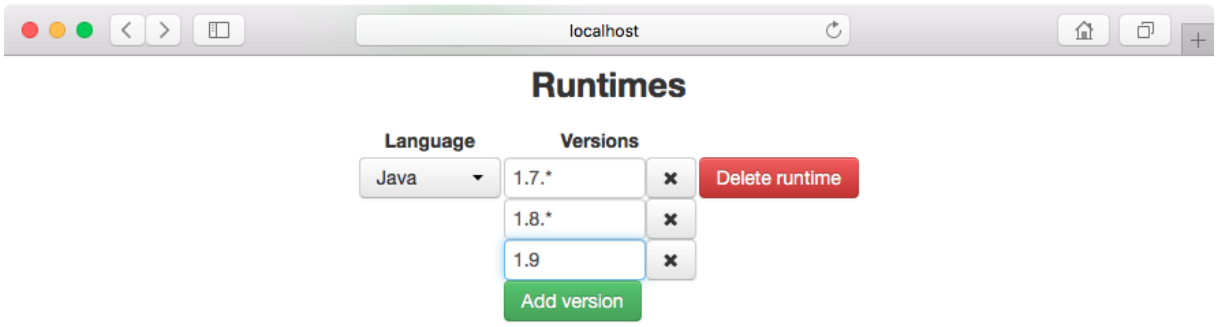

Abbildung 16: Update Page

3. Die Daten auf der Review Page überprüfen (siehe Abbildung 15).

Abbildung 17: Review Page

4. Optional kann der Wissensträger seine Kontaktdaten (Name und E-Mail) sowie die Nachricht zum Update hinterlassen.

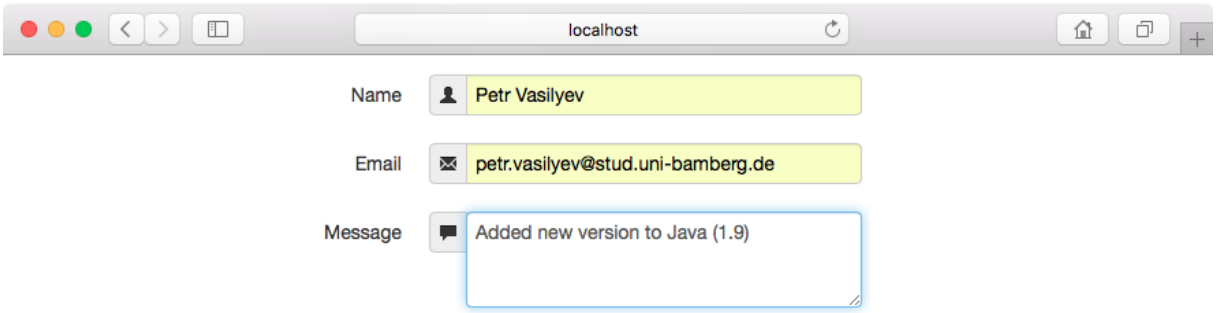

Abbildung 18: Kontaktdatenform

5. Anschließend werden die Daten mit dem Klick auf den „Submit“-Button an die für die Datenübermittlung zuständige Schnittstelle geschickt.

Ein wichtiger Aspekt bei der Entwicklung der Wissensträgerschnittstelle ist die Validierung der Eingabedaten. In einigen Fällen können problematische Eingaben bereits auf der Ebene der Benutzerschnittstelle (Update Page) verhindert werden. Als Beispiel wird bei einem Kontaktformular (Abbildung 18) bei der E-Mail Adresse mittels des HTLM-Attributs type=„email“ sichergestellt, dass dass der Input der formalen Input-Struktur entspricht. Allerdings kann dieser Ansatz nicht immer angewendet werden. Ein Beispiel ist eine Version, die ein Sonderzeichen (z.B. *) oder einen Buchstaben enthält.

4.3 Worker für Datenübermittlung

Der vorliegende Abschnitt befasst sich mit dem Teil der Wissenserwerbskomponente, der als Bindeglied zwischen den Wissenserfassungsmethoden und der Wissensbasis auftritt und in Abbildung 5 als Schnittstelle für die Datenübermittlung bezeichnet wird. Im Rahmen dieser Arbeit wird der Begriff „*Worker*“ verwendet, der für die Ausprägung dieser Schnittstelle steht. Im Folgenden wird das Konzept hinter dem Worker abstrakt skizziert und im weiteren Verlauf mit der konkreten Implementierung verdeutlicht. Der Anwendungsbereich des Workers wird in Abbildung 19 wie folgt dargestellt:

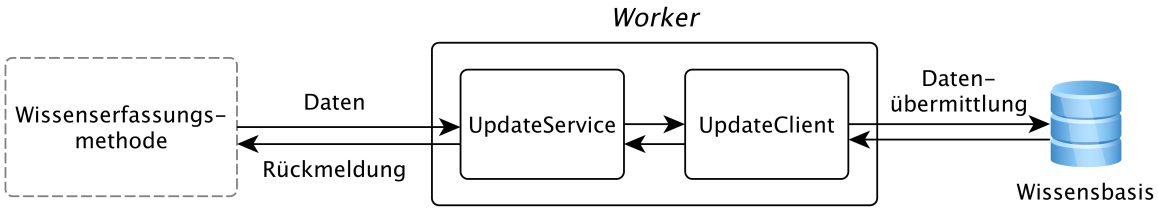

Abbildung 19: Anwendungsbereich vom Worker

Grundsätzlich besteht der Worker aus zwei Komponenten, nämlich „*UpdateService*“ und „*UpdateClient*“. Die Aufgabe des UpdateService besteht in der Bereitstellung einer Schnittstelle, die die Daten von außen aufnimmt und die Datenübermittlung an den UpdateClient

delegiert. Auf der anderen Seite stellt der UpdateClient die Methoden zur Verfügung, die für die Datenübermittlung zuständig sind.
Generell lässt sich der Ablauf gemäß Abbildung 19 folgendermaßen beschreiben. Als erstes werden die Daten von der Wissenserfassungsmethode an den UpdateService gesendet. Darauffolgend werden die Daten vom UpdateService verarbeitet und für den UpdateClient vorbereitet. Im nächsten Schritt wird die Aufgabe der Datenübermittlung an den UpdateClient delegiert. Der UpdateClient erstellt die Anfrage an die Wissensbasis und teilt die Antwort dem UpdateService mit. Anschließend verschickt der UpdateService die Rückmeldung an die Wissenserfassungsmethode.
Des Weiteren wird die Wissensträgerschnittstelle aus dem Abschnitt 4.2 als Wissenserfassungsmethode betrachtet. Die Wissensbasis stellt das Git-Repository von *PaaSfinder* dar, das von einem Bot-Account auf Github „geforkt“ wird[20]. In anderen Worten wird das ursprüngliche Git-Repository von *PaaSfinder* kopiert, sodass der Bot-Account einen schreibenden Zugriff auf die Wissensbasis erhält.
Technisch gesehen erfolgt der Nachrichtenaustausch zwischen Wissenserfassungsmethode, dem Worker und der Wissensbasis auf Basis des Hypertext Transfer Protocols (HTTP) und des Representational State Transfer (REST) Prinzips, das ursprünglich aus der Dissertation von Fielding [Roy00] stammt. Das zentrale Konzept von REST basiert auf Ressourcen, die im globalen Raum mithilfe eines Uniform Resource Identifier (URI)[21] eindeutig identifiziert werden [SMSO15, S.11,35]. Ein Vendor wird also als Ressource im JSON Format zunächst zum Worker und anschließend zum Git-Repository geschickt.
Um die Daten von außen empfangen zu können, implementiert der UpdateClient eine REST API, die in Form einer Route („/vendor“) definiert wird. Die Route entspricht dem HTTP-Standardverb POST[22] und akzeptiert die Daten im JSON Format. Für die Implementierung der Route wurde das Framework Spark[23] verwendet. Auf der Seite vom UpdateClient werden die Nachrichten als Anfragen an die Github API[24] mithilfe von OkHttp[25] gesendet. In Java Pseudocode lässt sich die Route folgendermaßen beschreiben (siehe Listing 1):

```
post("/vendor", "application/json", (request, response) -> {
  JsonObject data = jsonParser.parse(request.body());

  Branch branch = new Branch(..);
  client.postBranch(branch);

  File file = new File(..);
  client.putFile(file);

  PullRequest pullRequest = new PullRequest(..);
  client.postPullRequest(pullRequest);
});
```

Listing 1: „/vendor“ Route

Als erstes werden die Daten in ein Java Objekt transformiert. Danach wird ein neu-

[20] https://github.com/update-bot/paas-profiles
[21] https://tools.ietf.org/html/rfc3986
[22] https://tools.ietf.org/html/rfc7231#section-4.3.3
[23] http://sparkjava.com
[24] https://developer.github.com/v3
[25] https://square.github.io/okhttp

er Branch erzeugt und an den UpdateClient zum Absenden weitergegeben. Sobald der Branch auf der Github-Seite erfolgreich erstellt wurde, wird die Vendor-Datei im erstellten Branch aktualisiert. Anschließend wird ein Pull-Request erzeugt und vom UpdateClient an das Git-Repository geschickt. Bedauerlicherweise lässt sich der Ablauf nicht parallelisieren, da der nächste Schritt die erfolgreiche Ausführung des vorherigen Schrittes voraussetzt. Beispielsweise setzt die Aktualisierung der Datei die Erstellung des Branches voraus, da die Datei im erstellten Branch aktualisiert wird.
Wenn man den oben beschriebenen Ablauf auf das Beispiel mit „Heroku“ aus dem Abschnitt 4.2 überträgt, ergibt sich Folgendes: sobald der Benutzer auf „Submit“ klickt, werden die Daten als JSON in der POST-Anfrage an den Worker gesendet, nämlich an die Schnittstelle des UpdateService. Das erfolgreiche Branch- bzw. Pull-Request-Erstellen wird durch den HTTP-Code 201 („Created“) im Response der Github-API mitgeteilt (siehe Listing 2 und 4). Beim erfolgreichen Updaten der Datei wird der Statuscode 200 („OK“) zurückgeliefert (siehe Listing 3).

```
protocol=http/1.1,
code=201,
message=Created,
url=https://api.github.com/repos/update-bot/paas-profiles/
   git/refs
```

Listing 2: Response beim erfolgreichen Branch-Erstellen

```
protocol=http/1.1,
code=200,
message=OK,
url=https://api.github.com/repos/update-bot/paas-profiles/
   contents/profiles/heroku.json
```

Listing 3: Response beim erfolgreichen File-Update

```
protocol=http/1.1,
code=201,
message=Created,
url=https://api.github.com/repos/update-bot/paas-profiles/
   pulls
```

Listing 4: Response beim erfolgreichen Pull-Request-Erstellen

Nach der erfolgreichen Pull-Request-Erstellung kann die aktualisierte Version von Heroku auf Github angeschaut werden (siehe Abbildung 20). In der Detailansicht werden ebenso die Änderungen explizit gezeigt. Dabei werden die gelöschten Zeilen als rot markiert und die hinzugefügten als grün (siehe Abbildung 21).

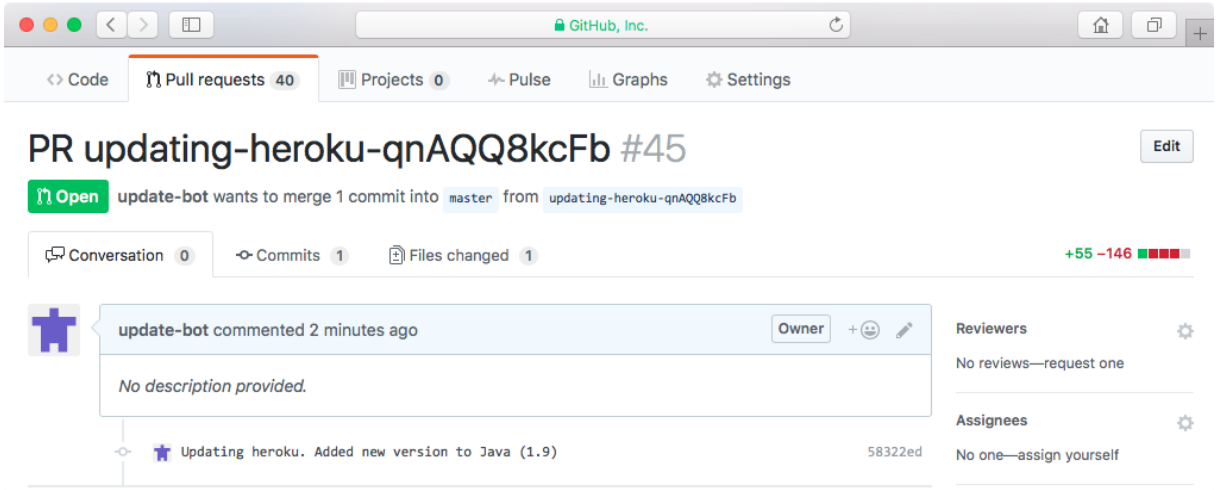

Abbildung 20: Pull-Requests Ansicht auf Github

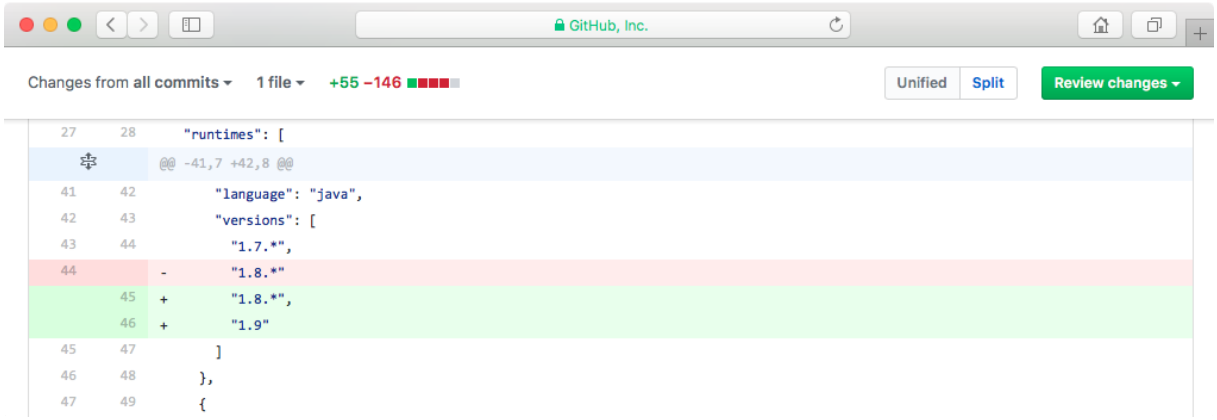

Abbildung 21: Heroku Pull Request

Schließlich wird im Erfolgsfall der Statuscode 200 („OK“) der Wissenserfassungsmethode mitgeteilt. In Bezug auf die Fehlerbehandlung wird jeder Schritt, der sich logisch abgrenzen lässt (Branch erstellen, Datei aktualisieren und Pull-Request erstellen), in einem eigenen try-catch-Block innerhalb der Route in Listing 1 ausgeführt, sodass der Client eine aussagekräftige Fehlermeldung erhält.

5 Ausblick

Nachdem die Wissensträgerschnittstelle zur Datenkorrektur und der Worker zum Erstellen automatischer Pull Requests betrachtet wurden, handelt es sich im vorliegenden Abschnitt um einen Ausblick auf weitere Datenquellen, die ein Automatisierungspotential im Rahmen der Datenerfassung für *PaaSfinder* aufweisen. Nach einer kurzen Erläuterung der möglichen Datenquellen findet ein Vergleich der Quellen statt. Anschließend wird ein Bezug auf zukünftige Arbeiten genommen.

5.1 Abgrenzung der Datenquellen

Die Auswahl der Datenquellen erfolgt unter der Annahme, dass die Datenerfassung sich auf die Aktualisierung der bestehenden Vendors beschränkt. Der Grund dafür besteht darin, dass die maschinelle Erfassung eines neuen Vendors die fachliche Expertise eines Experten erfordert.
Insgesamt können folgende Datenquellen relevant sein:

- Webseite des Vendors
- Web-Feeds
- Soziale Netzwerke
- Newsletter
- Blogs

Die erste Quelle, die bei der Datenerfassung eines Vendors in Frage kommt, ist der Webauftritt des betrachteten Vendors. Zur Datenerfassung kann ein Web-Crawler benutzt werden, der ausgehend von einem Startlink (Seed) und einem Schlüsselwort die Webseite durchsucht [WDT10, S.32]. Als Seed wird die Homepage des Vendors verwendet. Die weiteren Seiten ergeben sich durch Verfolgen der Links auf der besuchen Seite. Allerdings ist dieser Ansatz hier weniger sinnvoll, da die Mehrheit der Daten bereits erfasst wurde.
Bei Web-Feeds handelt es sich um „Content Syndication“, was als Bereitstellung von Daten für Übertragung, Aggregierung und Online-Publikation[26] bezeichnet werden kann. Ein verbreitetes Beispiel für Web-Feeds ist RSS. Die Abkürzung steht je nach Quelle für „RDF Site Summary“[27], „Really Simple Syndikation“[28] und „Rich Site Summary“. Es wird zwischen Push und Pull RSS unterschieden. Bei Push RSS werden die Benachrichtigung vom Absender angestoßen. Bei Pull RSS soll man die Benachrichtigen manuell abrufen. In der Praxis werden Pull RSS am meisten eingesetzt. Ein RSS-Dokument wird in XML beschrieben, was ein effizientes Parsen der Inhalte ermöglicht. Außerdem enthalten RSS-Einträge das Erstellungsdatum, das für die Zuordnung der Aktualisierung hilfreich ist. Ein RSS Beispiel von Heroku[29] wird in Listing 5 dargestellt.

[26] http://web.resource.org/rss/1.0/
[27] http://web.resource.org/rss/1.0/spec
[28] https://validator.w3.org/feed/docs/rss2.html
[29] https://www.heroku.com/

```
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" version
   ="2.0">
 <channel>
  <title>Heroku</title>
  <link>http://blog.heroku.com</link>
  <description>The Heroku Blog</description>
  <item>
   <title>The Heroku-16 Stack is Now Available</title>
   <link>https://blog.heroku.com/heroku-16-is-generally-
      available</link>
   <pubDate>Thu, 20 Apr 2017 15:06:00 GMT</pubDate>
   <guid>https://blog.heroku.com/heroku-16-is-generally-
      available</guid>
   <description>
    <p>Your Heroku applications run on top of ...</p>
   </description>
   <author>Jon Byrum</author>
  </item>
 </channel>
</rss>
```

Listing 5: Ein RSS Beispiel von Heroku

Neben RSS können soziale Netzwerke für das Beziehen von Updates verwendet werden. Twitter[30] ist ein Beispiel dafür. Der Vorteil dieses Kurznachrichtendienstes besteht darin, dass er für den Vendor keinen Entwicklungsaufwand mitbringt. Zum Abrufen der Daten bietet Twitter eine eigene REST API[31] an.
Newsletter und Blogs können ebenso als potentielle Datenquellen benutzt werden. Beim Abonnieren von Newslettern werden die Updates an die E-Mail Adresse regelmäßig vom Vendor verschickt. Außerdem können Vendor Blogs zur Datenerfassung durchsucht werden. In einem Blog werden meist Informationen veröffentlicht, die für Entwickler interessant sind. Ein Blogeintrag ist meist ein kurzer Artikel, der ein Update enthält. Aus diesem Grund werden Blogs ebenso für die Datenerfassung genutzt.

5.2 Vergleich der Informationsquellen

Im Zuge des Vergleichs wurden 30 Vendors nach Unterstützung durch RSS-Feeds, Twitter, Newsletter und Blogs analysiert. Zusätzlich wurde das Datum des letzten Updates im Format TT.MM.JJ erfasst, das nach dem neuesten Eintrag aus den betrachteten Datenquellen zum Zeitpunkt vom 10. Mai 2017 bestimmt wird. Die Ergebnisse werden in Tabelle 1 zusammengefasst.

[30] https://twitter.com/
[31] https://dev.twitter.com/rest/public

Vendor	RSS	Twitter	Newsletter	Blogs	Last Update
Anynines	nein	ja	ja	nein	18.08.15
App42 PaaS	nein	ja	ja	ja	10.05.17
AppFog	nein	ja	ja	ja	10.05.17
AppHarbor	nein	ja	nein	ja	22.05.16
BitNami	nein	ja	ja	ja	10.05.17
Brightbox	nein	ja	nein	ja	25.04.17
Clever Cloud	nein	ja	nein	ja	26.04.17
Cloudnode	nein	ja	nein	ja	23.04.17
CloudUnit	nein	ja	nein	nein	10.05.17
Cloudways	nein	ja	ja	ja	10.05.17
EngineYard	nein	ja	nein	ja	29.04.17
Flynn	nein	ja	ja	ja	17.10.16
fortrabbit	nein	ja	nein	ja	04.05.17
Getup Cloud	nein	ja	nein	nein	09.05.17
Google App Engine	nein	ja	ja	ja	10.05.17
Heroku	ja	ja	ja	ja	10.05.17
Jelastic	nein	ja	ja	ja	10.05.17
Mendix	nein	ja	nein	ja	28.04.17
mOSAIC	ja	nein	nein	nein	05.04.13
OpenShift Container Platform	nein	ja	nein	ja	22.04.17
Oracle Cloud PaaS	ja	ja	nein	ja	01.12.16
OrangeScape	nein	ja	nein	ja	05.04.17
Pagoda Box	nein	ja	nein	ja	06.05.16
Platformer.com	ja	ja	nein	ja	01.04.17
SAP HANA Cloud Platform	nein	ja	nein	ja	02.05.17
Software AG Live	nein	ja	ja	ja	10.05.17
Tsuru	nein	ja	ja	ja	12.04.17
Voxoz	nein	ja	nein	nein	08.01.17
WSO2 App Cloud	nein	ja	ja	ja	10.05.17

Tabelle 1: Vergleich von RSS, Twitter, Newsletter und Blogs

In Tabelle 1 sieht man deutlich, dass die Nutzung des Kurznachrichtendiensts Twitter die höchste Verbreitung unter den betrachteten Quellen besitzt. Nur in einem Fall wird es nicht unterstützt, und zwar bei mOSAIC[32].
RSS wird im Allgemeinen deutlich weniger verwendet. Während es bei bekannten Anbietern wie Heroku umfangreich unterstützt wird (siehe Abbildung 22), bieten kleinere Vendors diesen Service nicht an. Ein möglicher Grund könnte die dazu im Gegensatz einfachere Handhabung von Twitter sein. Während RSS deutlich aufwendiger ist, erreichen beide Dienste dasselbe Ziel.
Auch Blogs erfreuen sich großer Popularität/sind relativ weit verbreitet. Diese zielen jedoch auf ein Publikum mit technischem Hintergrund ab. Aber wie bereits in Abschnitt 5.1 erwähnt wurde, werden dabei die Updates in Form eines Artikels publiziert. Bezüglich der Newsletter gibt es kein klares Bild. Etwa ein Drittel der betrachteten Vendors sehen diese Option vor. Dabei muss beachtet werden, dass Newsletter fallweise zu einem Blog angeboten werden.

[32] http://www.mosaic-cloud.eu/

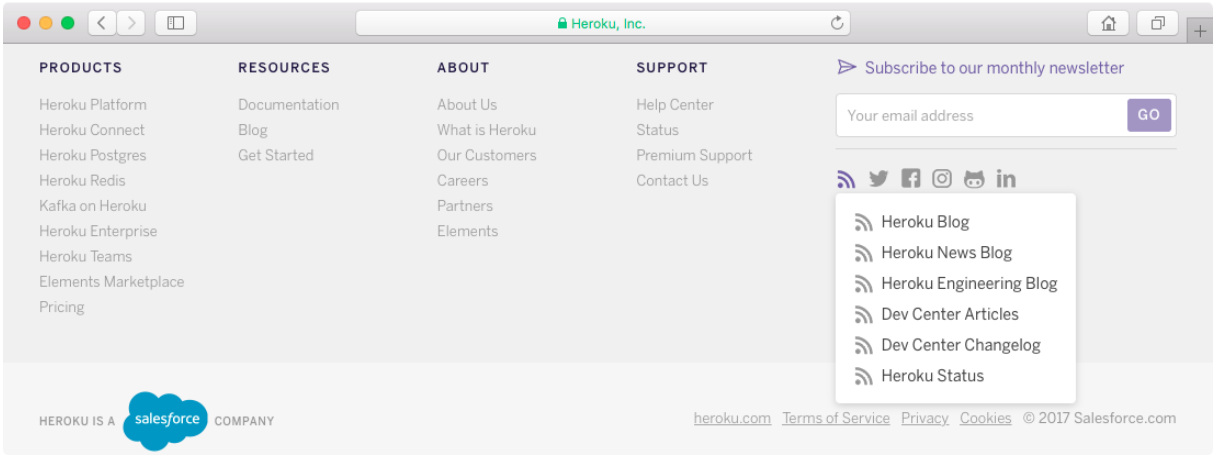

Abbildung 22: RSS Angebot bei Heroku

5.3 Future Work

Im folgenden Abschnitt werden zukünftige Arbeitsrichtungen in Bezug auf die Automatisierung der Datenerfassung von *PaaSfinder* erläutert, die auf den Erkenntnissen der Arbeit aus 4.2, 4.3 und 5.2 basieren.
Als Weiterentwicklung der Arbeit aus 4.2 sollen die Hilfestellungen in das Interface eingebaut werden. Da es nicht immer für alle ersichtlich ist, was ein bestimmtes Feld bedeutet, sollte neben einem Feld ein Element (z.B. Fragezeichen Tooltip) hinzugefügt werden, das eine kurze Erklärung zum Feld enthält.
Zunächst soll die REST API von Twitter genauer untersucht werden. Da Twitter enorm verbreitet ist, lassen sich Nachrichten vieler Vendors auf einmal erfassen. Die Aufgabe besteht darin, die Daten nach Vendor zusammenzustellen. Als nächstes können die Daten nach einem Pattern ausgewertet werden. Wenn beispielsweise das Wort „Ruby“ vorkommt, ist die Wahrscheinlichkeit hoch, dass es sich um ein Update der unterstützen Version von Ruby handelt. Hier besteht auch Potential für das Anwenden des maschinellen Lernens, indem die Trainingsmengen von Twitter ausgewertet werden.
Im Weiteren soll das Potential von Blogs ausgenutzt werden. Die Artikel, die in Blogs publiziert werden, können zusätzliche Information und externe Links auf wichtige Ressourcen enthalten. Dabei bietet es sich an, einen Web-Crawler zu benutzen. Wenn ein Newsletter allgemein oder im Kontext eines Blogs angeboten wird, soll diese Möglichkeit in Anspruch genommen werden. Ansonsten kann ein Service entwickelt werden, der den Update Reminder an verantwortliche Personen der Firmen via E-Mail nach einem bestimmten Zeitraum nach dem letzten Update verschickt.
Schließlich sollen die automatischen Tests von *PaaSfinder* erweitert werden, um die Konsistenz der Daten sicherzustellen. Bei der Implementierung der Wissensträgerschnittstelle stellte sich als häufiges Problem heraus, dass einige Felder auf null gesetzt werden, was zu Problemen bei der Darstellung der Daten führt.

6 Fazit

Im folgenden Abschnitt werden die Ergebnisse der Arbeit zusammengefasst. Des Weiteren werden die Probleme angesprochen, die in zukünftiger Arbeit behandelt werden sollen.
Im theoretischen Teil der Arbeit wurde ein Modell des automatisierten Wissenserwerbs erarbeitet (siehe Abbildung 5). Das Modell sieht vor, dass die Datenerfassung je nach Anwendungsbereich unterschiedlich automatisierbar ist. Aus diesem Grund wurden drei Kategorien des Wissenserwerbs unterschieden: indirekter, direkter und automatisierter Wissenserwerb. Darüber hinaus wurde die Komponente zur Datenübermittlung definiert, die als Bindeglied zwischen den Wissenserwerbsmethoden und der Wissensbasis auftritt.
Die Anwendung des theoretischen Modells wurde am Beispiel von *PaaSfinder* behandelt. Gemäß der Zielsetzung wurde im ersten Schritt die Benutzerschnittstelle zur Aktualisierung eines Vendors entwickelt. Als nächstes wurde ein Service implementiert, der für die automatische Erstellung von Pull Requests zuständig ist. Des Weiteren wurden verschiedene Datenquellen untersucht, die bei der Automatisierung der Datenerfassung in zukünftiger Arbeit eingesetzt werden können.
In Bezug auf zukünftige Arbeit gibt es sich folgende Herausforderungen. Erstens sollen Verfahren entwickelt werden, die die Daten aus unterschiedlichen Quellen sinnvoll zusammenfassen und persistent speichern. Zweitens soll es ermöglicht werden, relevante Informationen aus den erfassten Daten maschinell zu erzielen. Dabei besteht Potential für die Anwendung des maschinellen Lernens. Drittens soll die Erweiterung der automatischen Tests in Zukunft stärker berücksichtigt werden, um die Datenbank konsistent und fehlerfrei zu halten.

Literatur

[AF99] A. Sahuguet und F. Azavant: *Building light-weight wrappers for legacy Web data-sources using W4F.* In: *Proceedings of the 25th International Very Large Data Bases Conference*, Seiten 738 – 741, 1999.

[AHM03] A. Rafea, H. Hassen und M. Hazman: *Automatic knowledge acquisition tool for irrigation and fertilization expert systems.* Expert systems with Applications, Seiten 49 – 57, 2003.

[Ben05] Ben Hammersley: *Developing Feeds with RSS and Atom: Developers Guide to Syndicating News & Blogs.* O'Reilly Media, 2005.

[BZL16] B. Song, Z. Jiang und L. Liu: *Automated experiential engineering knowledge acquisition through Q&A contextualization and transformation.* Advanced Engineering Informatics, Seiten 467 – 480, 2016.

[CG14] C. Beierle und G. Kern-Isberner: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen.* Vieweg, 2014.

[EPGR14] E. Ferrara, P. D. Meo, G. Fiumara und R. Baumgartner: *Web data extraction, applications and techniques: A survey.* Knowledge-Based Systems, Seiten 301 – 323, 2014.

[ER11] E. Ferrara und R. Baumgartner: *Intelligent Self-repairable Web Wrappers.* In: *Congress of the Italian Association for Artificial Intelligence*, Seiten 274 – 285. Springer, 2011.

[GD94] G. D. Tecuci und D. Duff: *A framework for knowledge base refinement through multistrategy learning and knowledge acquisition.* Knowledge Acquisition, Seiten 137 – 162, 1994.

[Geo96] Geoffrey I. Webb: *Integrating machine learning with knowledge acquisition through direct interaction with domain experts.* Knowledge-Based Systems, Seiten 253 – 266, 1996.

[Geo08] George Lawton: *Developing Software Online with Platform-as-a-Service Technology.* Computer, 2008.

[Ghe92] Gheorghe D. Tecuci: *Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base.* Transactions on Systems, Man, and Cybernetics, Seiten 1444 – 1460, 1992.

[Gre92] Gregory B. Winter: *An automated knowledge acquisition system for model-based diagnostics.* In: *Autotest Conf. Record.*, Seiten 167 – 174, 1992.

[GTW90] G. Gottlob, T. Frühwirth und W. Horn: *Expertensysteme.* Springer-Verlag, 1990.

[HDNR97] H. Fujihara, D. B. Simmons, N. C. Ellis und R. E. Shannons: *Knowledge Conceptualization Tool.* Transactions on Knowledge and Data Engineering, 1997.

[IKTH91] I. Okamura, K. Baba, T. Takahashi und H. Shiomi: *Development of automatic knowledge-acquisition expert system.* In: *the international conference on industrial electronics, control and instrumentation*, Seiten 37–41, 1991.

[ISO98] ISO: *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability.* Technischer Bericht, International Organization for Standardization, 1998.

[JFI93] J. Meyer-Fujara, F. Puppe und I. Wachsmuth: *Expertensysteme und Wissensmodellierung*, Kapitel 7, Seiten 714 – 766. Addison-Wesley, 1993.

[JJJ99] J.L. Castro, J.J. Castro-Schez und J.M. Zurita: *Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems.* Fuzzy Sets and Systems, Seiten 331 – 342, 1999.

[JJJ01] J.L. Castro, J.J. Castro-Schez und J.M. Zurita: *Use of a fuzzy machine learning technique in the knowledge acquisition process.* Fuzzy Sets and Systems, Seiten 307 – 320, 2001.

[Kar92] Karl Kurbel: *Entwicklung und Einsatz von Expertensystemen: Eine anwendungsorientierte Einführung in wissensbasierte Systeme.* Springer-Verlag, 1992.

[KM90] K. S. Leung und M. H. Wong: *An Expert-System Shell Using Structured Knowledge: An Object-Oriented Approach.* Computer, Seiten 38 – 47, 1990.

[LHCP02] L. M. Su, H. Zhang, C. Z. Hou und Pan X. Q.: *Research on an improved genetic algorithm based knowledge acquisition.* In: *1st international conference on machine learning and cybernetic*, Seiten 455 – 458, 2002.

[Mat00] Matthias Haun: *Wissensbasierte Systeme.* Expert Verlag, 2000.

[Nik16] Niket Tandon: *Commonsense Knowledge Acquisition and Applications.* Doktorarbeit, Saarland University, Saarbrücken, Germany, 2016.

[OE13] O. K. Ferstl und E. J. Sinz: *Grundlagen der Wirtschaftsinformatik.* Oldenbourg Verlag, 2013.

[Pet10] Peter Zöller-Greer: *Künstliche Intelligenz: Grundlagen und Anwendungen.* Composia Verlag, 2010.

[PFW+12] P. Mertens, F. Bodendorf, W. König, A. Picot, M. Schumann und T. Hess: *Grundzüge der Wirtschaftsinformatik.* Springer-Verlag, 2012.

[Roy00] Roy T. Fielding: *Architectural Styles and the Design of Network-based Software Architectures.* Doktorarbeit, University of California, Irvine, 2000.

[RR10] R. Akerkar und R. Sajja: *Knowledge-Based Systems.* Jones and Bartlett Publishers, 2010.

[RSG01] R. Baumgartner, S. Flesca und G. Gottlob: *Visual Web Information Extraction with Lixto.* In: *Proceedings of the 27th International Very Large Data Bases Conference*, Seiten 119 – 128, 2001.

[SK09] S. Gebus und K. Leiviskä: *Knowledge Acquisition for Decision Support Systems on an Electronic Assembly Line.* Expert Systems with Applications, Seiten 93 – 101, 2009.

[SMSO15] S. Tilkov, M. Eigenbrodt, S. Schreier und O. Wolf: *REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web.* dpunkt, 2015.

[Ste14] Steve Krug: *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability.* New Riders, 2014.

[VGP01] V. Crescenzi, G. Mecca und P. Merialdo: *RoadRunner: Towards Automatic Data Extraction from Large Web Sites.* In: *Proceedings of the 27th International Very Large Data Bases Conference*, Seiten 109 – 118, 2001.

[WDT10] W. B. Croft, D. Metzler und T. Strohman: *Search Engines. Information Retrieval in Practice.* Pearson, 2010.

[XDC03] X. Meng, D. Hu und C. Li: *Schema-Guided Wrapper Maintenance for Web-Data Extraction.* In: *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, Seiten 1 – 8, 2003.

Ich erkläre hiermit gemäß §17 Abs. 2 APO, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Bamberg, den 16.05.2017 Petr Vasilyev