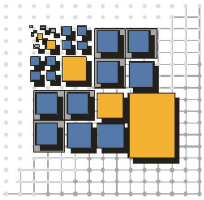

Otto-Friedrich-Universität Bamberg

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:

Vasilyev Petr

Themensteller:
Prof. Dr. Guido Wirtz

Abgabedatum:
17.05.2017

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Abkürzungsverzeichnis

API Application Programming Interface

IaaS Infrastructure-as-a-Service

PaaS Platform-as-a-Service

SaaS Software-as-a-Service

DSS Decision Support System

DOM Document Object Model

DTD Document Type Definition

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

REST Representational State Transfer

JSON JavaScript Object Notation

MVVM Model View ViewModel

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 Anliegen, ein intelligentes System zu schaffen, das mithilfe des spezifischen Wissens die Menschen bei den Problemlösungen unterstützt [RR10, S.18]. Darin besteht der Unterschied zwischen einem konventionellen Informationssystem und einem wissensbasierten System. Informationssysteme sind schließlich datenbasiert und konzentrieren sich auf die Datenverarbeitung [RR10, S.19]. Eine Spezialisierung der wissensbasierten Systeme stellen Expertensysteme dar, in denen das Wissen letztendlich von Experten stammt. Ein wissensbasiertes System bzw. ein Expertensystem ist durch die Trennung zwischen 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 initiale Datenbank wird in der Regel mithilfe manueller Datenerfassung aufgebaut [Kar92, S.70]. Ein Interview zwischen einem Wissensingenieurs und einem Wissensträger wird beispielsweise oft 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, kosten- und zeitintensiv sind. Aus diesem Grund wird seit langem versucht, den Prozess der Datenerfassung zu automatisieren. Eine Auswahl an bestehenden Ansätzen, die bezüglich der vorliegenden Arbeit relevant sind, wird im Abschnitt zu den verwandten Arbeiten gegeben. Trotz der vielen Herangehensweisen gibt es kein einheitliches Konzept, das einen allgemeinen Rahmen für die Automatisierung der Datenerfassung bildet.

1.2 Verwandte Arbeiten

Ein Konzept der Automatisierung der Wissenserfassung wird in [SK09] vorgestellt. Die Entwicklung der Wissensbasis umfasst dabei drei Phasen. In der ersten Phase wird die initiale Wissensdatenbank aufgebaut, die unvollständig und teilweise widersprüchlich sein kann. Die zweite Phase umfasst inkrementelle Erweiterung und Verbesserung der Wissensbasis. Dabei werden Methoden des maschinellen Lernens eingesetzt. Im Laufe der dritten Phase wird die Wissensbasis in Bezug auf Effizienz optimiert [Ghe92, S.1444]. Gheorghe Tecuci betont, dass bei jeder Phase eine Kooperation zwischen dem fachlichen Experten und dem System notwendig ist. Nach der maschinellen Datenerschließung werden die Ergebnisse manuell kontrolliert und anschließend in der Wissensbasis gespeichert. In dieser Weise lassen sich die Vorteile maschineller und manueller Datenerfassung optimal kombinieren [GD94, S.137], [Ghe92, S.1445].
Ein weiterer Ansatz wird in [SK09] thematisiert. Dabei handelt es sich um ein datenbasiertes Decision Support System (DSS), das die Unternehmensführung bei den Entscheidungen in Bezug auf die Produktionsoptimierung unterstützen soll. Allerdings werden die Störungen in der Produktion von Anlagenbedienern (Experten) mittels Erfahrungswissen intern behoben, ohne dieses Wissen weiterzugeben. Als Folge hat die Unternehmensführung kein umfangreiches Bild der Produktion und kann keine optimalen Entscheidungen treffen. Aus diesem Grund erweitern Gebus und Leiviskä das bestehende DSS mit einer Schnittstelle, um das Expertenwissen in die Datenbank zu integrieren [SK09, S.94]. All-

gemeiner betrachtet geht es um die Transformation eines datenbasiertes System in ein wissensbasiertes System. Auch hier wird versucht, die Daten mithilfe der engen Kooperation zwischen dem Menschen und dem System zu erfassen.

1.3 Zielsetzung

Das Ziel der vorliegenden Arbeit ist die Erarbeitung eines allgemeinen Konzeptes zur Automatisierung der Datenerfassung. Da die komplett automatisierte Datenerfassung sehr schwierig umzusetzen ist, liegt der Schwerpunkt dieser Arbeit auf der Kombination zwischen den manuellen und maschinellen Vorgehensweisen.
Beim technischen Kontext wird angedeutet, dass die Umsetzung im Rahmen eines bestimmten Anwendungsbereichs erfolgt. Es wird also kein Allgemeinwissen wie in [Nik16], sondern ein anwendungsbezogenes Wissen betrachtet. In diesem Zusammenhang wird das Konzept auf Basis eines Expertensystems entwickelt, da die Wissensbasis eines Expertensystems spezifischer ist als bei einem wissensbasierten System.
Die praktische Umsetzung soll auf Basis von *PaaSfinder*[1] erfolgen. Bei *PaaSfinder* handelt es sich um eine Web-Anwendung, die über eine Wissensdatenbank im Bereich Platform-as-a-Service (PaaS) verfügt. PaaS gehört neben Infrastructure-as-a-Service (IaaS) und Software-as-a-Service (SaaS) zum Themengebiet von Cloud Computing [PT11] und soll die Anwendungsentwicklung erleichtern, indem die Laufzeit- ober Entwicklungsumgebungen von einem PaaS-Anbieter dem Kunden vorkonfiguriert angeboten werden [Geo08, S.14]. Da *PaaSfinder* ein Open-Source Projekt ist, kann jeder der Datenbank von *PaaSfinder* beitragen. Allerdings ist die Mitwirkung mit einem hohen Aufwand verbunden und setzt Informatikkenntnisse voraus. Außerdem werden die Daten von *PaaSfinder* hauptsächlich auf den Webseiten von PaaS-Anbietern manuell erfasst. Dies stellt ein hohes Entwicklungspotential, die Erfassung von solchen Daten zu automatisieren.

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 das Kontext der Datenerfassung dargestellt und Automatisierungsmöglichkeiten angesprochen. 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 Erkenntnisse, in Kapitel 2 und 3 gewonnen wurden. In Kapitel 5 wird die Implementierung evaluiert und die Ergebnisse der Arbeit zusammengefasst. Anschließend wird in Kapitel 7 ein Ausblick für zukünftige Forschung im betrachteten Bereich gegeben.

[1] https://paasfinder.org

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 vom bestehenden Wissen abzuleiten [Ghe92]. Im Laufe der Zeit hat sich das Konzept eines Expertensystems auf andere Anwendungsbereiche verbreitet. 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 drauf 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 soll [CG14, S.12]. Im Rahmen dieser Arbeit sind folgende Eingenschaften 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.

Es ist auch wichtig anzumerken, dass die Begriffe “Künstliche Intelligenz“, “Wissensbasiertes System“ und “Expertensystem“ in einer engen Beziehung zueinander stehen. Haun gibt eine systematische Abgrenzung dieser Begriffe, 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 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 der Abbildung 1 darstellen:

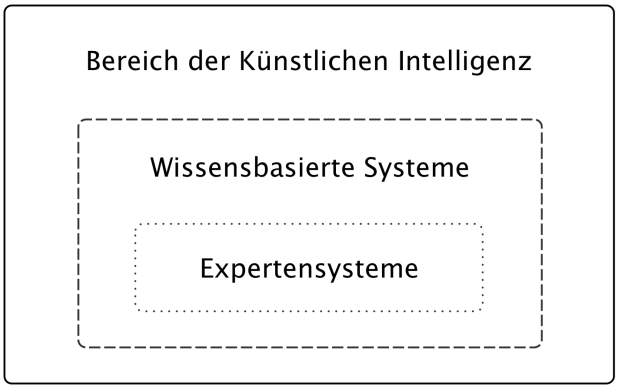

Abbildung 1: Begriffsabgrenzung, [Mat00, S.30]

Nach dieser Agbrenzung 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 zu einfach und nicht besonders aussagekräftig. Beierle und Kern-Isberner weisen drauf hin, dass nach diesem Kriterium viele existierenden Wissensbasierten Systeme als Expertensysteme bezeichnet werden könnten [CG14, S.11]. Aufgrund dessen 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 den 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 eine hohes Potenzial besitzt. Allerdings kann ein Expertensystem nicht als Ersatz eines menschlichen Experten betrachtet werden. Vielmehr geht es um eine Erfassung, Darstellung und Pflege des Expertenwissens in einem Expertensystems, 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]:

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

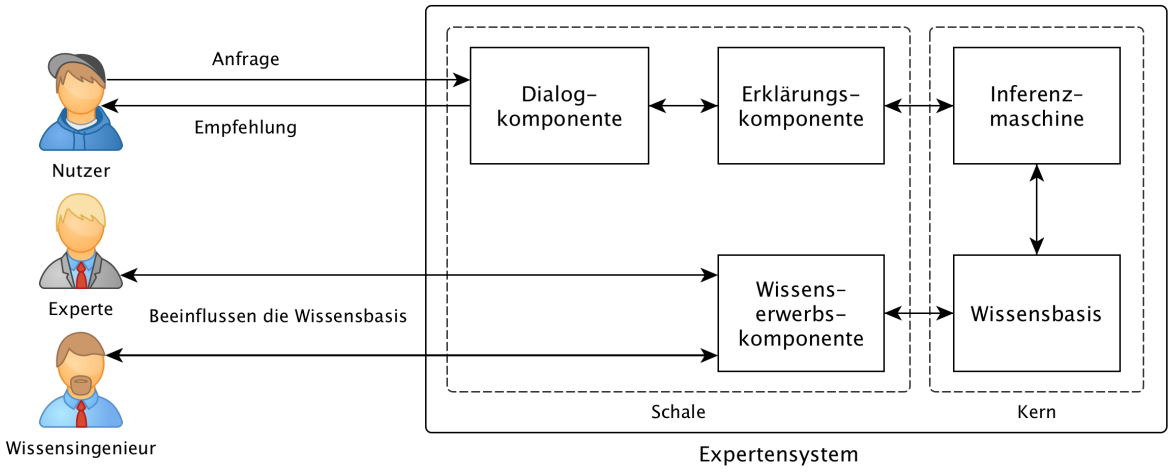

Abbildung 2: Expertensystem nach Haun, [Mat00, S.126]

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 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 der verfügbaren Wissensquellen wie Text, 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 den Experten ermöglicht, ihr Erfahrungswissen selbst zu formalisieren und gegebenenfalls zu warten [JFI93, S.743]. Die Überprüfung des Dateninputs ist ebenfalls die Aufgabe der Wissenserwerbskomponente. Dies kann mittels Durchführung automatisierten Tests bei jeder Änderungsanfrage erfolgen, um die Konsistenz der Wissensbasis zu gewährleisten [JFI93, S.743].
Um ein geeignetes Konzept der automatisierten Datenerfassung zu entwickeln, ist ein grundlegendes Verständnis von der Struktur und Funktionsweise der Wissensbasis sowie der Wissenserwerbskomponente erforderlich. Im weiteren Verlauf der Arbeit werden 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, 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]:

1. Systematische Erfassung vom Expertenwissen
2. Verfeinerung der Wissensbasis
3. Reorganisation der Wissensbasis

In der ersten Phase werden das Vokabular und die geeignete Wissensrepräsentation festgelegt. Gebus und Leiviskä betonen, dass die Wissensrepräsentation den entscheidenden Einfluss auf die Generierung und spätere Handhabung der Wissensbasis hat [SK09, S.95]. Die initialen Daten werden meistens im Rahmen eines Interviews zwischen dem Wissensingenieur und dem Experten erfasst [Ghe92, S.1444]. Das Ergebnis der ersten Phase ist eine initiale Wissensbasis, die unvollständig und teilweise widersprüchlich ist. In der zweiten Phase wird die initiale Wissensbasis mithilfe der geeigneten 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 der Abbildung 3 dargestellt.

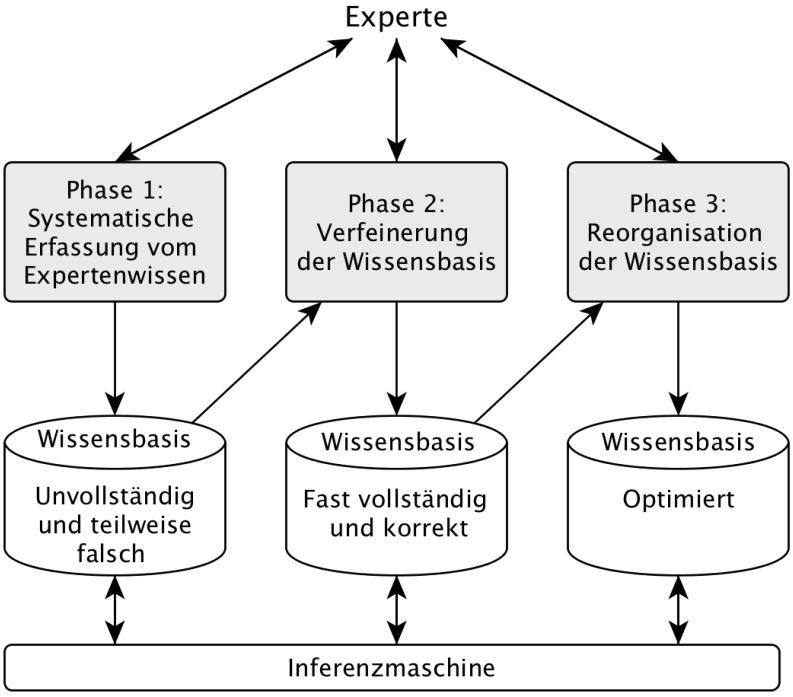

Abbildung 3: Phasen der Expertensystementwicklung, [GD94, S.138]

In der Abbildung 3 sieht man, dass der Autor 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 sollen.
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 spezifischste Wissen, das sich nur auf den gerade betrachteten Problemfall bezieht. Das sind z.B. Fakten, die von Beobachtungen oder Untersuchungsergebnissen stammen.
- *Regelhaftes Wissen*, das den eigentlichen Kern der Wissensbasis darstellt. Dieses Wissen kann noch genauer differenziert werden:
  - *Bereichsbezogenes Wissen*, das sich auf den gesamten Problembereich beziehen. Das kann sowohl theoretisches Fachwissen als auch Erfahrungswissen sein. Anders gesagt handelt es sich um generisches Wissen.
  - *Allgemeinwissen*, das z.B. um 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 der Abbildung 4 dargestellt wird.

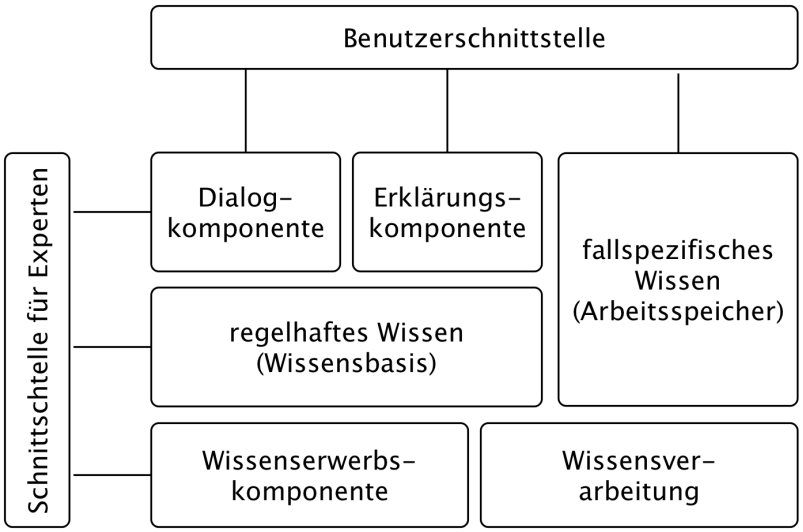

Abbildung 4: Expertensystem nach Beierle und Kern-Isberner, [CG14, S.18]

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 Formularisierung von Wissen, um eine maschinelle Verarbeitung erst 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-Regeln eingesetzt.
- *Objektorientierte Darstellung*, die das Konzept der Objekttypen übernimmt und mit deklarativen Operatorbeschreibungen verbindet.
- *Constraints Darstellung*, die Modellbeschreibungen aus dem Operations 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 Aufgaben. 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 deklarativen Form erfasst.
- *Wissensverfügbarkeit*: Die Codewartung von einem imperativen Programm ist fehleranfällig, 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 solcher Umsetzung besteht in der Modularität des Wissens. Das Wissen in unabhängige Module aufgeteilt wird. Da die einzelnen Module unabhängig voneinander sind, können sie getrennt getestet und modifiziert werden, ohne den Rest der Wissensbasis zu beeinträchtigen. Dies ermöglicht 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 Funktionen thematisiert.

2.4 Wissenserwerbskomponente

Bei der Wissenserwerbskomponente handelt es sich um einen Bestandteil eines 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. 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 ins Expertensystem ein.
- *Automatisierter Wissenserwerb:* Die Wissensbasis wird entweder mithilfe der automatisierten Datenerschließung aus verfügbarer Dokumente 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 der 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 der Benutzerfreundlichkeit ist 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 das nur in gut verstandenen und strukturierten Anwendungsbereichen möglich sei.
Zu dem automatisierten Wissenserwerb gibt es am wenigsten Erkenntnisse, die allgemein anwendbar sind. Meistens handelt es sich um Lösungen, die nur innerhalb eines spezifischen Anwendungsbereichs funktionieren. Nichtsdestotrotz lässt sich sagen, dass das

Wissen sich 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 den Fall großer Datenmenge und Anwendung des maschinellen Lernens ist ein Diagnosesystem, das für mögliche Diagnose Fälle mit Symptomen enthält. Für die Textanalyse können beispielsweise Bedienungsanleitungen analysiert werden, wobei diese Vorgehensweise gewisse Einschränkungen ausweist. Die Schwierigkeit dabei 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:

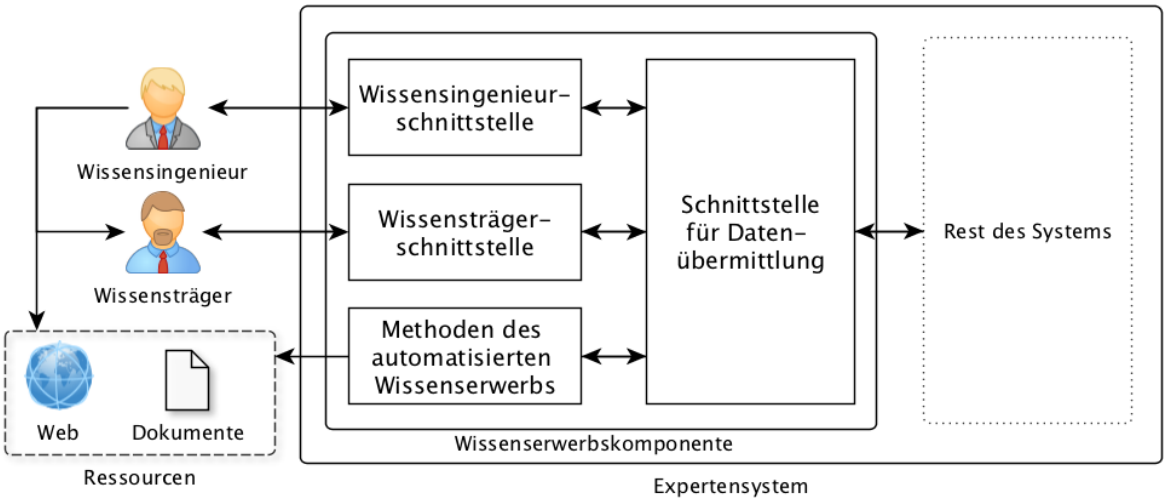

Abbildung 5: Wissenserwerbskomponente

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 gewünschte Bestandteile der Wissenserwerbskomponente gelegt wird. Beispielsweise kann sich ein Expertensystem mit umfangreichen Falldaten auf das maschinelle Lernen konzentrieren. Ein System, dass auf der Datenerfassung mithilfe der Wissensträgerschnittstelle basiert ist, wird in [SK09, S.97] vorgestellt.
Im weiteren Verlauf der Arbeit werden die Wissensträgerschnittstelle, die Komponente mit Methoden des automatisierten Wissenserwerbs und die Schnittstelle für Validierung und Speicherung der Daten thematisiert, da sie ein Potenzial für die Automatisierung der Datenerfassung aufweisen.

3 Methoden der Wissens- und Datenerfassung

Mit der Wissensakquisitionskomponente 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 Stärke 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 es im Weiteren auf automatisierte und halb-automatisiere Bestandteile der Wissensakquisitionskomponente beschränkt.

3.1 Schnittstelle zur Dateneingabe

Die Schnittstelle zur Dateneingabe (auch als Wissensträgerschnittstelle im Kapitel 2.4 bekannt) ist in der Regel ein Bestandteil eines Tools zum Wissenserwerb. Diese Schnittstelle soll dem Wissensträger ermöglichen, sein Wissen ins System auf eine einfache Weise 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 der [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 den Entscheidungen von nicht-strukturieren Problemen 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 allerdings von Experten gesteuert, die mithilfe des Erfahrungswissens Störungen in der Produktion beseitigen. Dieses Erfahrungswissen über Störungen wird im System nicht erfasst. Als Folge hat die Unternehmensführung 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 von einem Prototyp. Der Entwicklungsprozess lässt sich allgemein wie folgt beschreiben:

1. Die Definition der Zielgruppe, die für die Schnittstelle relevant ist.
2. Ausgehen 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 der Abbildung 6 vereinfacht nachbilden.

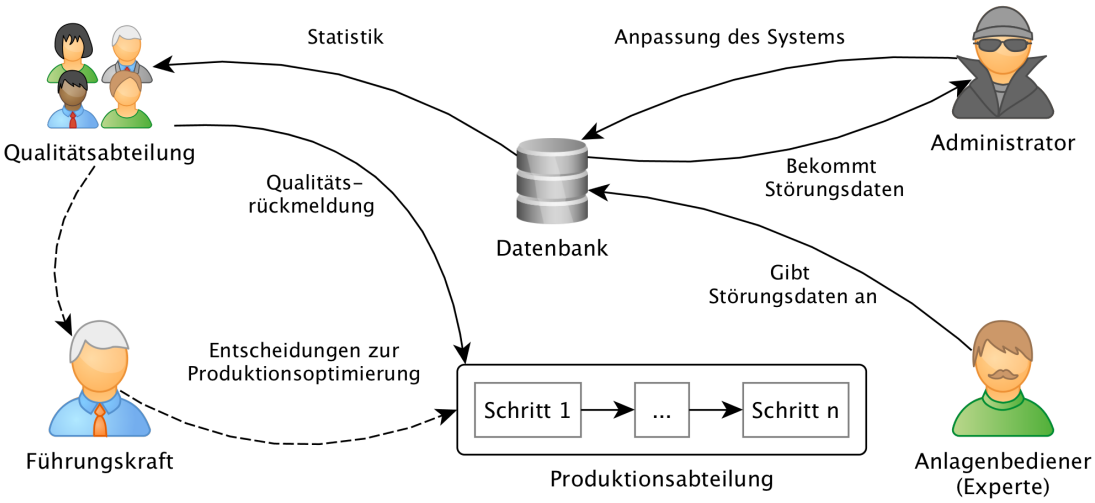

Abbildung 6: Die Struktur und die Informationsflüsse im DSS, [SK09, S.98]

Bei der Informationsrepräsentation werden die Anlagenbilder benutzt. Die Autoren betonen, dass die Expertenschnittstelle 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 Butzerinteraktion 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]. In Bezug auf Usefulness ist eine interessante Ansicht in [Ste14] gegeben. Steve Krug bezeichnet einen Gegenstand 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 die Nutzungsrate von 38% ergibt. Laut Gebus und Leiviskä lag es daran, dass der Protoryp aus Usability-Perspektive nicht optimal war. Beispielsweise waren einige Elemente der Schnittstelle eher verwirrend. Nach den notwendigen Anpassungen konnte die Nutzungsrate fast bei 100% erreicht 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 der 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 Informationen zur Berichterstellung für die Führungskräfte und die Produktion. 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 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 den 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. Darauffolgend 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 identifizieren folgende Herausforderungen [EPGR14, S.302]:

- *Automatisierungsgrad*: Die Erfassung der Webdaten soll oft von einem menschlichen Experten überwacht werden, um die Genauigkeit der Daten zu gewährleisten.
- *Skalierbarkeit*: Bei den umfangreichen Webressourcen soll innerhalb kürzer Zeit schnell 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 der Webressourcen ändert sich oft. Die Datenerfassungsmethoden für das Web sollen eine gewisse Flexibilität besitzen, um weiterhin korrekt zu funktionieren.
- *Trainingsdaten*: Bei der Einsetzung des maschinellen Lernen ist eine ausreichende Trainingsmenge an Webseiten erforderlich, die manuell vorbereitet wird. Dies ist eine schwierige 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)[2] definiert. Die Idee vom DOM besteht darin, dass die HTML-Webseite ein 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 effiziente Datensuche in einer HTML-Seite [EPGR14, S.303].
Da HTML ein Dialekt von Extensible Markup Language (XML) ist, kann XML Path Language (XPath)[3] für die Navigation in 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 Reihe) ausgewählt. Im Beispiel (B) werden mehrere Elemente (alle Zellen der zweiten Reihe) angesprochen [EPGR14, S.303].

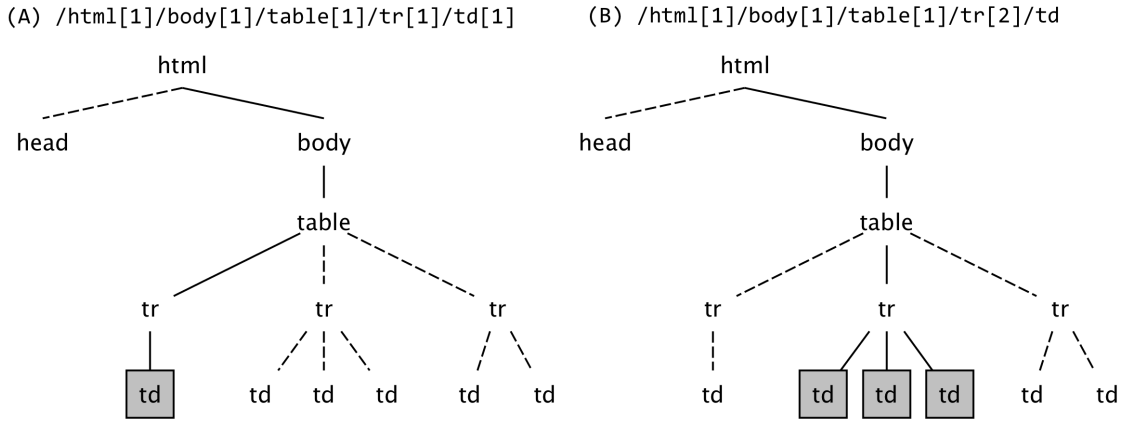

Abbildung 7: XPath im Dokumentenbaum, [EPGR14, S.304]

[2]https://www.w3.org/DOM
[3]https://www.w3.org/TR/xpath

Der Hauptnachteil von XPath besteht darin, dass XPath-Ausdrücke strikt an der 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. In Bezug auf dieses Problem wurde im letzten Release von XPath[4] relative XPath-Ausdrücke eingeführt [EPGR14, S.304].
Ein weiterer Ansatz, die Webdaten zu erfassen, stellt ein Web Wrapper dar. Das Web Wrapper umfasst in der Regel einen oder mehreren Algorithmen, die zur Datenerfassung aus den Webdokumenten eingesetzt werden. Anschließend werden die erfassten Daten in eine strukturierte Form transformiert und für 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 kann ein Web Wrapper mittels folgender Ansätze generiert und ausgeführt werden [EPGR14, S.306]:

- *Reguläre Ausdrücke*: Daten werden gemäß Regeln (expressions) gewonnen. Im Rahmen dieser Arbeit wird es im Weiteren aus reguläre Ausdrücke beschränkt.
- *Logikbasierter Ansatz*: Zur Datenerfassung wird eine Wrapper Programmiersprache eingesetzt (wrapper programming language).
- *Baumbasierter Ansatz*: Dabei 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 Patterns in der unstrukturierten bzw. semi-strukturierten Dokumenten unter Verwendung der Regeln, die z.B. in Form von Wortgrenzen oder HTML-Tags definiert werden. Der Vorteil der regulären Ausdrücken besteht in der Möglichkeit, beliebiges Element auf einer Webseite anzusprechen. Außerdem bieten einige Implementierungen ein grafisches Benutzerinterface, sodass der Benutzer die Elemente auf einfache Weise auswählen kann. Die Regeln werden dann automatisch generiert. Eine mögliche Umsetzung des Wrappers wird in [AF99] mit W4F vorgestellt. Das Tool verfügt über eine Hilfsmethode, die den Benutzer bei der Auswahl der Elementen unterstützt. Auf Basis der ausgewählten Elementen 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 die Veränderungen der DOM-Struktur. Die Anpassung erfolgt entweder manuell oder in gewisser Weise automatisiert. Ferrara et al betonen, dass der Automatisierungsgrad der Wartung

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

besonders kritisch ist [EPGR14, S.308]. Bei einer kleinen Dokumentenanzahl ist die manuelle Anpassung noch akzeptabel. Allerdings ist die manuelle Wartung bei einer großen Menge der Dokumente nicht mehr denkbar.
Ein Beispiel der automatisierter 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 der Abbildung 8 dargestellt.

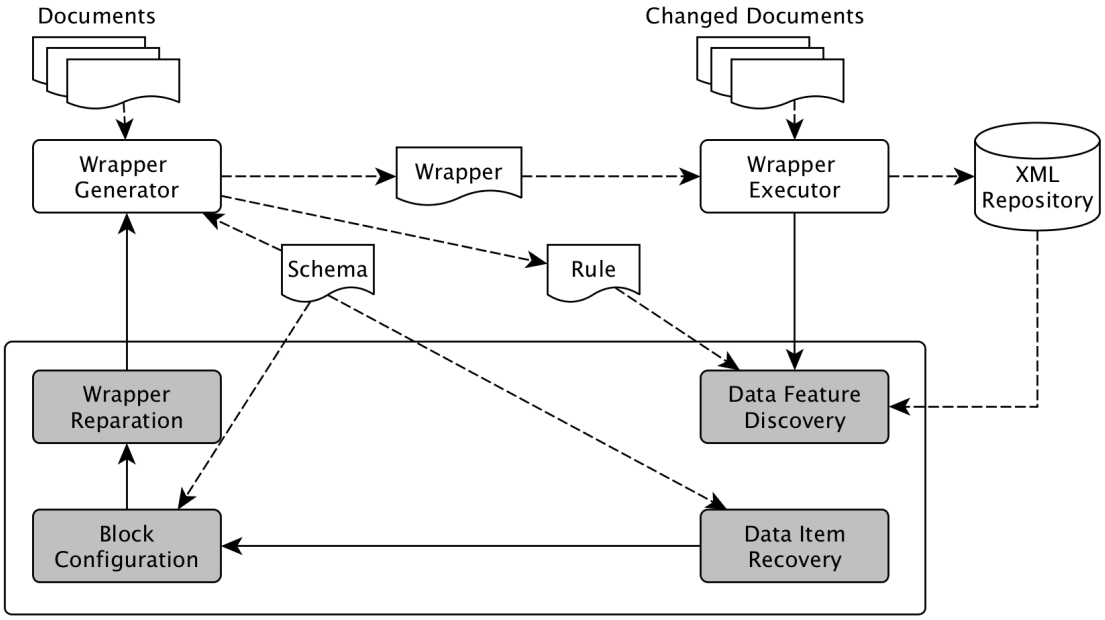

Abbildung 8: Die Architektur vom SG-WRAM, [XDC03, S.2]

Im ersten Schritt werden die HTML-Seite, XML-Schemata und das Mapping zwischen diesen beim Wrapper-Generator eingegeben. XML-Schema wird in Document Type Definition (DTD) beschrieben. Darauffolgend 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. Beim Fehlschlag werden Warnungen und Benachrichtigungen angezeigt, dass die Regeln nicht mehr funktionieren. In diesem Fall sollen die Regeln manuell von einem Experten angepasst werden [EPGR14, S.308].
Als ein 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 Patterns zu finden. Allgemein kann das Verfahren die Daten aus beliebiger Quelle erfassen, die mindestens zwei Seiten mit ähnlicher Struktur 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 handelt es sich um einen Ausblick in weitere Wissenserwerbsmethoden, und zwar die Anwendung des maschinellen Lernens. In der Literatur gibt es eine Reihe der Ansäzte, die sich damit beschäftigen (z.B. [JJJ01] und [Geo96]). Trotz der Unterschieden in der Umsetzung gibt es eine entscheidende Gemeinsamkeit. Beide Methoden basieren auf der Zusammenarbeit zwischen dem Experten und dem Lernalgorithmus. 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 des Expertenwissens im Medizinbereich. Das Konzept der Wissensbasiserweiterung orientiert sich an [Ghe92] und besteht darin, dass anfangs eine initiale Wissensbasis gebaut und inkrementell verbessert wird, indem der Experte die Fragen vom System beantwortet. Bei der Erstellung der Fragen wird ein Lernverfahren eingesetzt. Die Idee des Lernverfahrens besteht in der Erschließung des naheliegenden Expertenwissens aus den Trainingsdaten der Schlussfolgerungen eines Experten.
Der Kern des Verfahrens besteht aus dem induktiven Lernen und dem generischen Algorithmus von [JJJ99], der auf der Fuzzylogik basiert ist. Bei der Induktion handelt es sich um das Erschließen der allgemeinen Schlussfolgerungen aus den Einzelfällen. Die Fuzzylogik beschäftigt sich mit den graduellen Aussagen, die nicht eindeutig falsch oder wahr bezeichnet werden können. Diese Aussagen werden mithilfe der vagen Prädikaten beschrieben. Ein typisches Beispiel ist das Prädikat „groß“. Eine 1,80 m große Person wird als „groß“ bezeichnet, während eine 1,75 m Person ist 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 des Rausches aus den Trainingsdaten.
3. Ermittlung der Genen, 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 Teil anderer Regeln sind.
7. Ende.

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

- *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-Werten $X^n \times Y$ 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 treffen die Annahmen, 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 $\mathcal{P}(\mathcal{L}_i)$, oder zusammenfassend $E_i \in \mathcal{P}(\mathcal{L}_i)$.
Das Ergebnis der Durchführung des Lernverfahrens ist eine Menge der Regeln, die bei der Erstellung der Fragen verwendet werden. 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, handelt es sich in diesem Kapitel um die Umsetzung der Wissensträgerschnittstelle am Beispiel von *PaaSfinder*. Am Anfang wird ein kurzer Bezug auf das *PaaSfinder* genommen. 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 vom *PaaSfinder*[5], einem Expertensystem und Open-Source-Projekt im Bereich von Platform-as-a-Service (PaaS). Das Ziel des Systems besteht darin, zahlreiche PaaS-Anbieter vergleichbar zu machen. *PaaSfinder* verfügt bereits über eine Wissensdatenbank, die PaaS-Profile beinhaltet und aufgrund der häufigen Änderungen im gegebenen Anwendungsbereich regelmäßig aktualisiert werden muss. Die Struktur von *PaaSfinder* lässt sich in der Abbildung 9 folgendermaßen beschreiben.

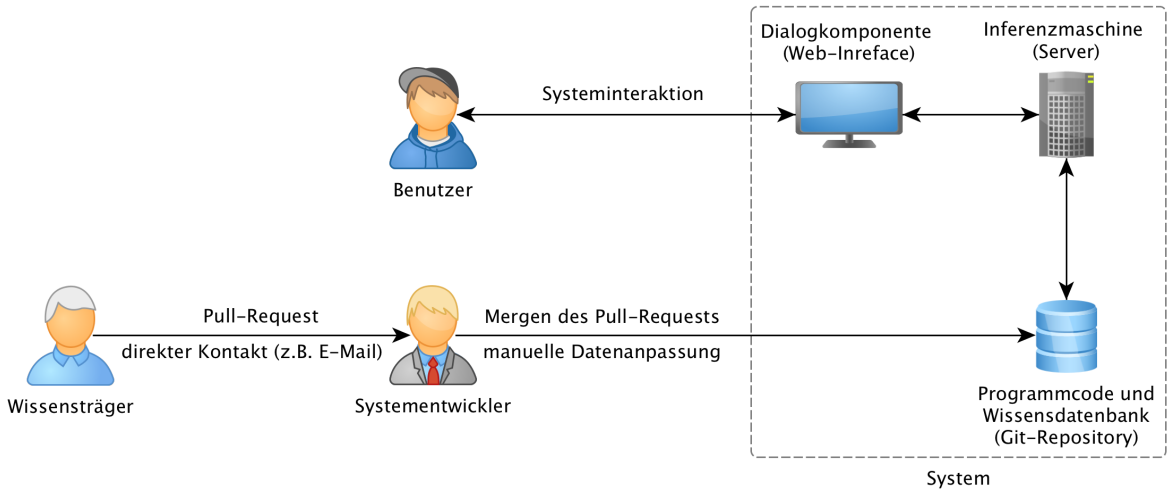

Abbildung 9: Die Struktur vom *PaaSfinder*

PaaSfinder setzt sich aus dem Programmcode mit der Wissensdatenbank und der Webanwendung zusammen. Die Webanwendung wird wiederum auf einem Server betrieben. Ein PaaS-Anbieter wird in einem Profil[6] beschrieben, das wie folgt aufgebaut ist:

1. **General Properties**: allgemeine Eingenschaften des PaaS-Anbieters (z.B. Name, URL, Status, Type, etc.)
2. **Extensible**: generelle Erweiterungsmöglichkeit nach Kundenbedarf (z.B. spezielle Laufzeitumgebung)

[5] https://paasfinder.org
[6] 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[7])
8. **Middleware** (z.B. Tomcat[8])
9. **Frameworks** (z.B. Play[9])
10. **Services** (z.B. Datenspeicher)
11. **Infrastructures** (z.B. Informationen zum Standort)

Ein Profil wird als ein JSON[10]-Eintrag gespeichert. JSON stellt ein Datenaustauschformat dar und basiert auf zwei Datenstrukturen:

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

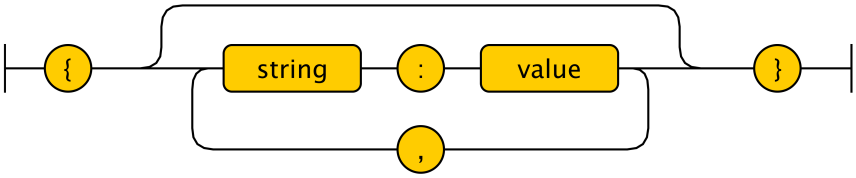

Abbildung 10: JSON-Objekt Spezifikation, 10

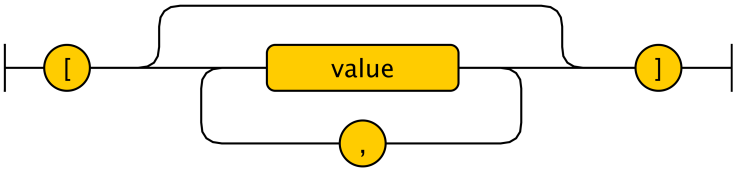

Abbildung 11: JSON-Array Spezifikation, 10

Die Wissensdatenbank und der Programmcode von *PaaSfinder* befinden sich in einem Git-Repository. Bei Git[11] handelt es sich um ein verteiltes System zur Projektverwaltung.

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

Da *PaaSfinder* ein Open-Source-Projekt ist, kann jeder der Wissensdatenbank beitragen. Momentan kann das auf zwei Weisen erfolgen (siehe Abbildung 9):

- *Pull-Request*[12]
- *Direkter Kontakt mit dem Systementwickler*

Beim Pull-Request wird das gesamte Projekt-Repository lokal gespeichert. Danach werden Änderungen vorgenommen und anschließend einen 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 ein 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 Kontakt mit dem Systementwickler entspricht dem direkten Wissenserwerb gemäß der Klassifikation in Kapitel 2.4. Der Vorteil dabei ist, dass die Daten nicht manuell gesucht werden sollen, sondern gleich vom Wissensträger stammen. Anderseits 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 PaaS-Eintrag 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 Schnittstele zur Dateneingabe entwickelt (Wissensträgerschnittstelle). Mithilfe der Schnittstelle soll die Aktualisierung eines bestehenden Profils ermöglicht werden. Das Anlegen oder das Löschen eines Profils liegt nicht im Anwendungsbereich der Schnittstelle. Der Ablauf wird in Abbildung 12 veranschaulicht.

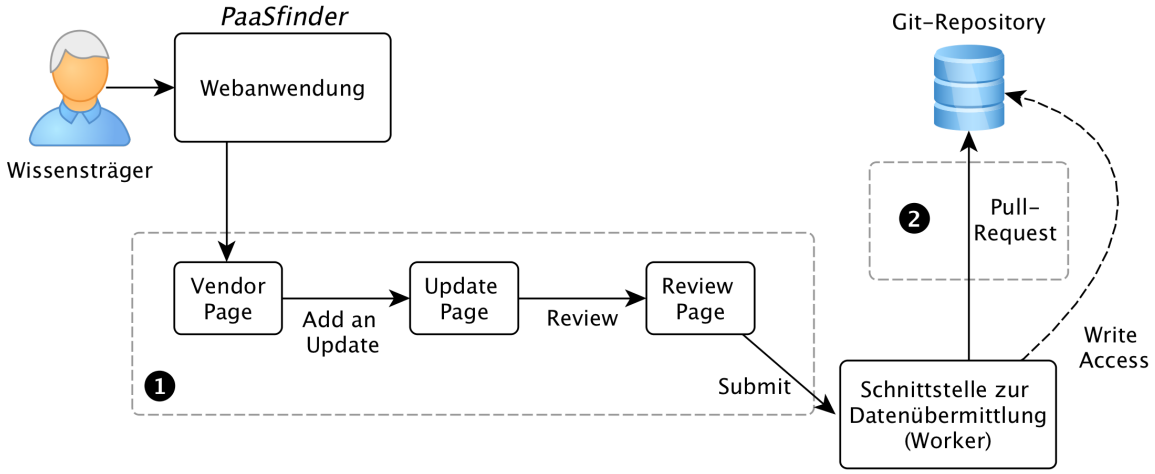

Abbildung 12: Profilaktualisierung mittels Wissensträgerschnittstelle

[12]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. *Profilauswahl* (Vendor Page)
2. *Aktualisieren* des Profils (Update Page)
3. *Überprüfung* der Daten (Review Page)
4. *Absenden* der Daten (Sumbit)

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

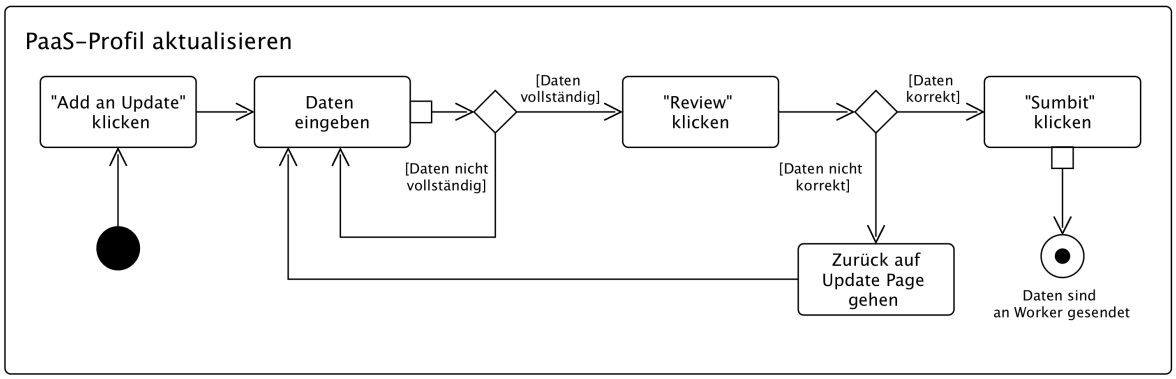

Abbildung 13: Aktivitätsdiagramm der Aktualisierung des PaaS-Profils

Ausgehend von einer Vendor Page wird der Ablauf mit „Add an Update“ gestartet. Als nächstes werden die Daten eingegeben. Wenn das Profil 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, kann das aktualisierte Profil mit dem Klick auf „Submit“ an den Worker gesendet werden. Anderenfalls kann der Wissensträger zurück zur Dateneingabe gehen und die Daten überarbeiten. In 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 wird. Im weiteren Verlauf wird die Implementierung der Page und Review Page beschrieben.

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

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

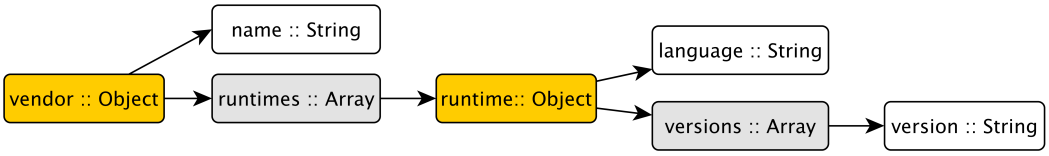

Abbildung 14: Komplexe Datentypen im PaaS-Profil

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[14] eingesetzt. Knockout.js ist eine JavaScript Bibliothek zur Erstellung dynamischer Webseiten. Das Framework basiert auf Model View ViewModel (MVVM) Entwurfsmuster[15]:

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

Als erstes werden die für das Modell erforderliche Daten über *PaaSfinder*-API[16] geholt und das Modell initialisiert. Da das Modell komplexe Datentypen umfasst, sind explizite JavaScript Klassen nötig. Beispielsweise gibt es für „runtime“ und „version“ jeweils eigene Klassen mit unterschiedlichem Verhalten. Die Klasse „runtime“ beinhaltet z.B. die Methode zum Hinzufügen neuer 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 zusammenfasst, so dass der Wissensträger die Eingaben überprüfen kann. Die Profildaten von der Update zur Review Page werden mithilfe von Web Storage[17] ausgetauscht. W3C definiert zwei Arten der Speicherung, nämlich Session und Local Storage. Session Storage gilt ausschließlich innerhalb Browserfenster. Die Inhalte von Local Storage können dagegen innerhalb einer Domain zugegriffen werden und gelten zeitlich unbeschränkt. Selbst beim Fensterschließen bleiben die Daten erhalten und können bei Bedarf gelesen werden. Daher wird im Weiteren Local Storage betrachtet.

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

Auf der Update Page wird das Profil beim Klicken auf „Review“ in die Local Storage gespeichert. Das Speichern bzw. Lesen basiert wie bei JSON auf Schlüssel/Wert-Paar-Prinzip. Der Schlüssel ist dabei der Profilname, 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, das Profil auf der Update Page zurückzusetzen, indem die Profildaten aus der Local Storage gelöscht und erneut von der *PaaSfinder*-API angefordert werden.
Im folgenden Beispiel wird das Profil „Heroku“ aktualisiert, indem eine neue Version (1.9) zu Java in „runtimes“ hinzugefügt wird.

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

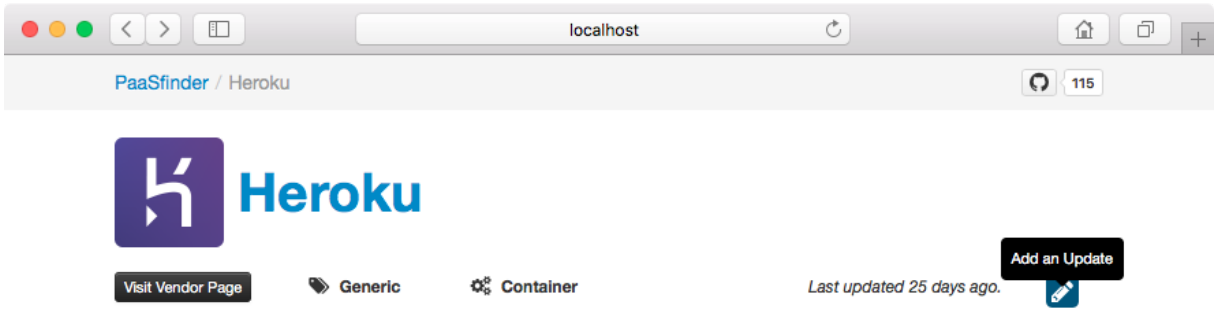

Abbildung 15: Vendor Page

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

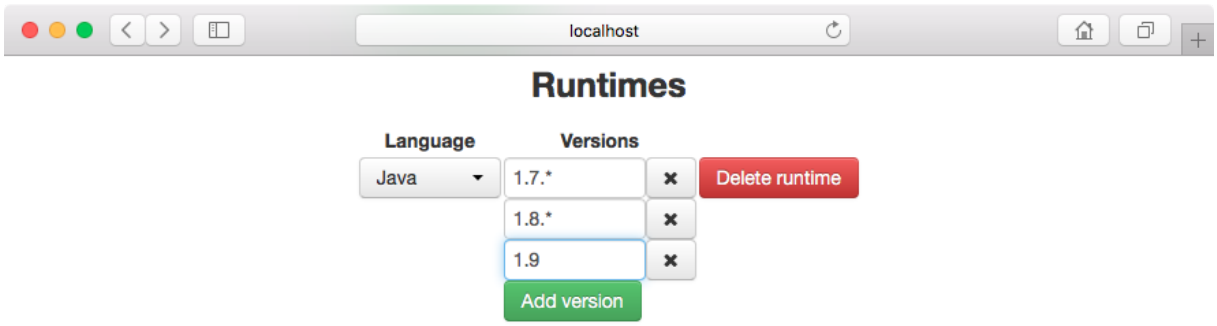

Abbildung 16: Update Page

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.

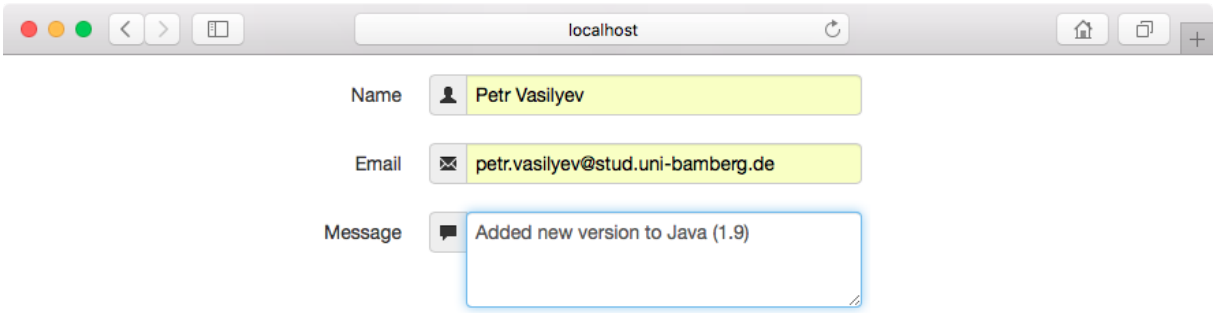

Abbildung 18: Kontaktdatenform

5. Anschließend werden die Daten mit dem Klick auf den „Sumbit"-Button an die für die Datenübermittlung zuständige Schnittstelle abgeschickt.

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 (User Interface) verhindert werden. Als Beispiel wird in der Kontaktform (Abbildung 18) bei der E-Mail Adresse mittels HTML-Attributes type=„email" sichergestellt, dass der Input formal der E-Mail Struktur einspricht. Allerdings kann der vergleichbare Ansatz nicht immer angewandt werden. Ein Beispiel ist eine Version, die ein Sonderzeichen (z.B. *) oder einen Buchstaben enthalten kann.

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 vom Worker wird in Abbildung 19 wie folgt dargestellt:

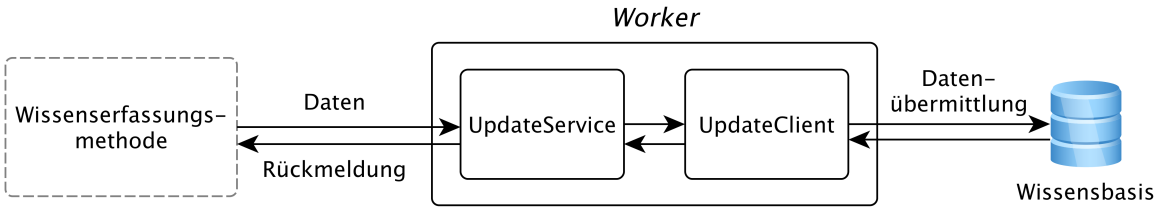

Abbildung 19: Anwendungsbereich vom Worker

Grundsätzlich besteht der Worker aus zwei Komponenten, nämlich „*UpdateService*" und „*UpdateClient*". Die Aufgabe vom 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 der 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.
Im 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[18]. In anderen Worten wird das ursprüngliche Git-Repository von *PaaSfinder* kopiert, sodass der Bot-Account einen schreibenden Zugriff auf die Wissensbasis hat.
Technisch gesehen erfolgt der Nachrichtenaustausch zwischen Wissenserfassungsmethode, dem Worker und der Wissensbasis auf Basis von Hypertext Transfer Protocol (HTTP)[19] und Representational State Transfer (REST) Prinzip, das ursprünglich aus der Dissertation von Fielding [R.T00] stammt. Das zentrale Konzept von REST basiert auf Ressourcen, die im globalen Raum mithilfe von Uniform Resource Identifier (URI)[20] eindeutig identifiziert werden [SMSO15, S.11,35]. Ein Vendor wird also als Ressource in JSON Format zunächst zum Worker und darauffolgend 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[21] und aktepriert die Daten in JSON Format. Für die Implementierung der Route wurde das Framework Spark[22] verwendet. Auf der Seite vom UpdateClient werden die Nachrichten als Anfragen an die Github API[23] mithilfe von OkHttp[24] 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

[18] https://github.com/update-bot/paas-profiles
[19] https://www.w3.org/Protocols
[20] https://tools.ietf.org/html/rfc3986
[21] https://tools.ietf.org/html/rfc7231#section-4.3.3
[22] http://sparkjava.com
[23] https://developer.github.com/v3
[24] https://square.github.io/okhttp

Als erstes werden die Daten geparsed. Danach wird ein neuer Branch erzeugt und an den UpdateClient zum Absenden weitergegeben. Sobald der Branch auf der Github-Seite erfolgreich erstellt wurde, wird die Vendor-Datei im erstellen 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 Schnitt die erfolgreiche Ausführung des vorherigen Schrittes voraussetzt. Beispielsweise setzt die Aktualisierung der Datei die Erstellung vom Branch 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 vom UpdateService. Das erfolgreiche Branch bzw. Pull-Request-Erstellen wird durch den 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 der Update von Heroku auf Github betrachtet 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).

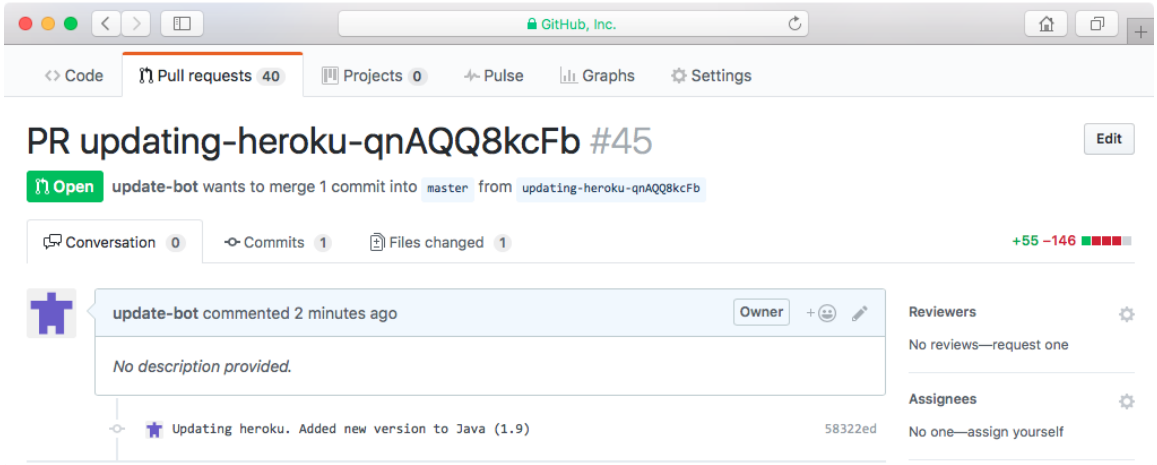

Abbildung 20: Pull-Requests Ansicht auf Github

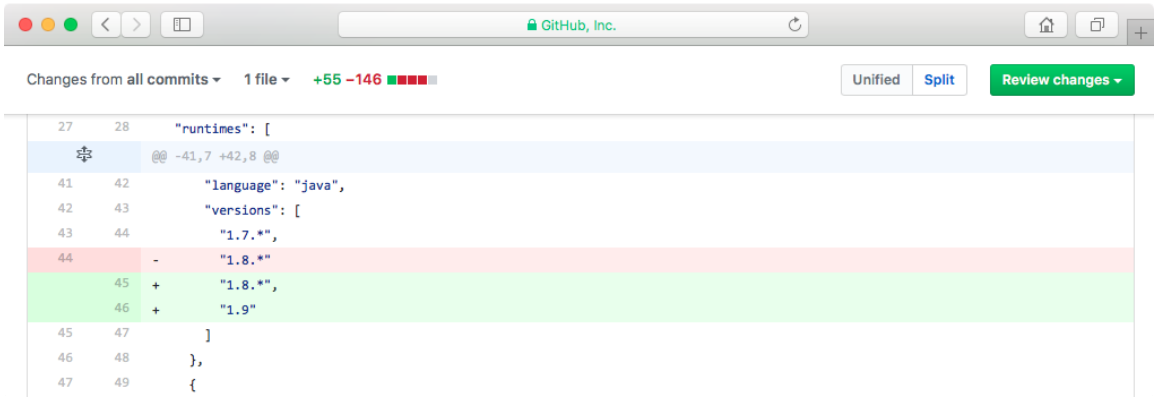

Abbildung 21: Heroku Pull Request

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 bei der Route in 1 ausgeführt, sodass der Client eine aussagekräftige Fehlermeldung bekommt.

5 Automatisierung der Webdatenerfassung

Die Besonderheit von *PaaSfinder* besteht darin, dass der überwiegende Teil der gesuchten Daten sich im Web befindet. Aus diesem Grund befasst sich dieses Kapitel mit den Informationsquellen des Web, die bei der Datenerfassung für *PaaSfinder* thematische Relevanz ausweisen können.

5.1 Abgrenzung der Informationsquellen

Allgemein beschränkt sich die Datenerfassung wie im Kapitel 4 auf die Aktualisierung der bestehenden Einträgen der Wissensbasis von *PaaSfinder*. Dabei lassen sich folgende Informationsquellen identifizieren:

- Webseiten von PaaS-Anbietern
- Web-Feeds (z.B. News-Feeds)
- Soziale Netzwerke

Die Analyse der Webseiten wäre die naheliegende Vorgehensweise, neue Daten über einen PaaS-Anbieter zu erschließen. Die Wissensbasis von *PaaSfinder* beinhaltet bereits die Homepage-URLs, die als Ausgangspunkt (Seed) bei einem Web-Crawler benutzt werden können. Auch die Fachliteratur bietet zahlreiche Informationen bezüglich der Umsetzung eines Web-Crawlers (z.B. [WDT10, S.35-50]). Allerdings macht das Webseiten-Crawling aus folgenden Gründen wenig Sinn:

- *Webseitenstruktur*. Unter den PaaS-Anbietern gibt es keine einheitliche Sicht, wie die Informationen auf der Webseiten strukturiert werden können.
- *Datenverarbeitung (Parsen)*. Bei der kleinsten Änderung der Seitenstruktur kann die gesamte Anfrage (meist HTML-Tag-Selektoren) zusammengebrochen werden.
- *Neuheit der Information*. Im Kontext von bestehenden Daten ist es höchstwahrscheinlich, dass die Daten bereits erfasst wurden.

In Bezug auf Parsen der Webseiten wurde im Abschnitt 3.2 das Konzept von Wrapper angesprochen. Allerdings ist an der Stelle die Verwendung der REST-API einfacher zu nutzen und zu warten. Das Argument mit der Neuheit der Information kann umstritten werden, wenn es um die Erfassung eines noch nicht in der Wissensbasis vorhandenen PaaS-Anbieters geht. Aber selbst in diesem Fall stellt sich die Frage der Glaubwürdigkeit eines PaaS-Profils, das auf maschinelle Weise erstellt wurde.
Eine weitere Kategorie der Informationsquelle stellen Web-Feeds und soziale Netzwerke dar. Aus Sicht der Datenaktualität sind Web-Feeds und soziale Netzwerke hervorragend geeignet. Die Daten werden in kleinen zeitlich geordneten Informationseinheiten dargestellt. Außerdem bieten einige PaaS-Anbieter einen Service, der sich auf ein bestimmtes Thema spezialisiert. Aufgrund dessen wird im Weiteren der Bereich von Web-Feeds und sozialen Netzwerken genauer betrachtet.

5.2 Web-Feeds und soziale Netzwerke

In Bezug auf Web-Feeds spricht man von „Content Syndication“, was im deutschsprachigen Raum als „Syndikation“ bezeichnet wird. Unter Syndikation wird Bereitstellung von Daten für Übertragung, Aggregierung und Online-Publikation[25] verstanden. Ein verbreitetes Format für Web-Feeds ist RSS. Ursprünglich fand RSS besondere Verbreitung bei Webbloggern [SMSO15, S.103]. Aufgrund des einfachen Konzeptes ist RSS in kürzer Zeit zum meistgenutzten Format zur Änderungsbenachrichtigung geworden.
Je nach Quelle wird die Abkürzung RSS unterschiedlich interpretiert. In der ersten Version (RSS 1.0) stand die Abkürzung für „RDF Site Summary“. In der Version 2.0 wird jedoch „Really Simple Syndikation“[26] erwähnt. Ein RSS 2.0 Dokument wird in XML-Format definiert und als Wurzel den Rss-Tag enthält, der den Namensraum der Elemente und die Version von RSS bestimmt. Im Rss-Tag wird der Channel-Tag definiert, der folgende Elemente enthalten muss (siehe 26):

- Title: Name des Channels (engl. Kanal)
- Link: URL zur Webseite des entsprechenden Channels
- Description: kurze Zusammenfassung des Channels

Die Inhalte des Channels werden durch Items dargestellt. Laut der Spezifikation in 26 entspricht ein Item einer Story (ein kurzer Artikel). Bei einem Item gibt es keine Pflichtfelder. Allerdings muss entweder Title oder Description vorhanden sein. Ein Beispiel für einen RSS 2.0 Feed[27] wird in Listing 5 dargestellt.

```
<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: Beispiel eines Eintrages in RSS 2.0

[25] http://web.resource.org/rss/1.0/
[26] https://validator.w3.org/feed/docs/rss2.html
[27] https://blog.heroku.com/news/feed

Im Beispiel von Heroku werden außerdem separate Feed-Channels für die unterschiedlichen Themenbereiche angeboten, wie es in Abbildung 22 zu sehen ist.

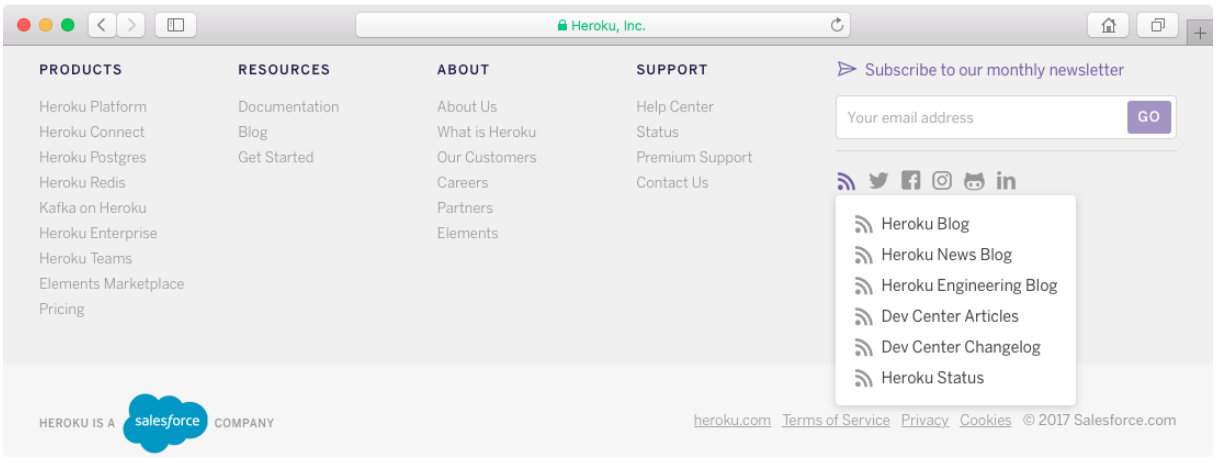

Abbildung 22: Feed-Channels von Heroku

Der Themenbereich umfasst sowohl allgemeine (z.B. Heroku Blog) als auch spezifische Informationen (z.B. Dev Center Articles). Im Anwendungsfall von *PaaSfinder* ist der Dev Center Changelog Channel besonders relevant, da der die Benachrichtigungen im technischen Bereich enthält (siehe Abbildung 23).

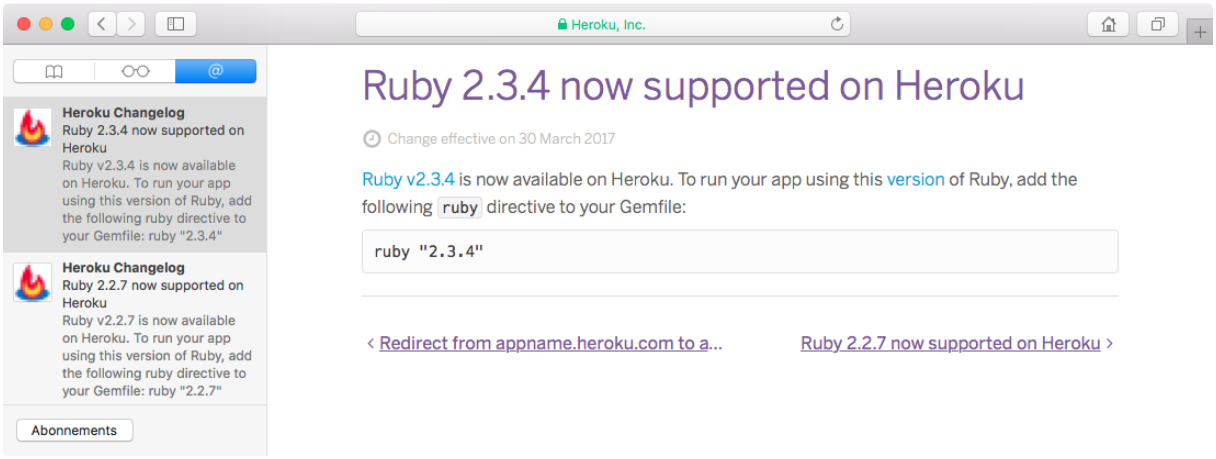

Abbildung 23: Heroku Feed

Neben RSS können die Daten aus sozialen Netzwerken erschlossen werden. Zu diesem Zweck stellen die Anbieter der sozialen Netzwerken (z.B. Facebook[28], Twitter[29]) eine REST-API zur Verfügung, die das Lesen/Schreiben der Daten ermöglicht. Im Beispiel von Heroku hat sich herausgestellt, dass sich Twitter am besten für Benachrichtigungen geeignet ist. Bei diesem konkreten Fall handelt es sich sogar um die gleiche Information wie in RSS-Feeds (vgl. Heroku in Abbildung 23 und 24).

[28] https://developers.facebook.com/docs/graph-api/
[29] https://dev.twitter.com/overview/api

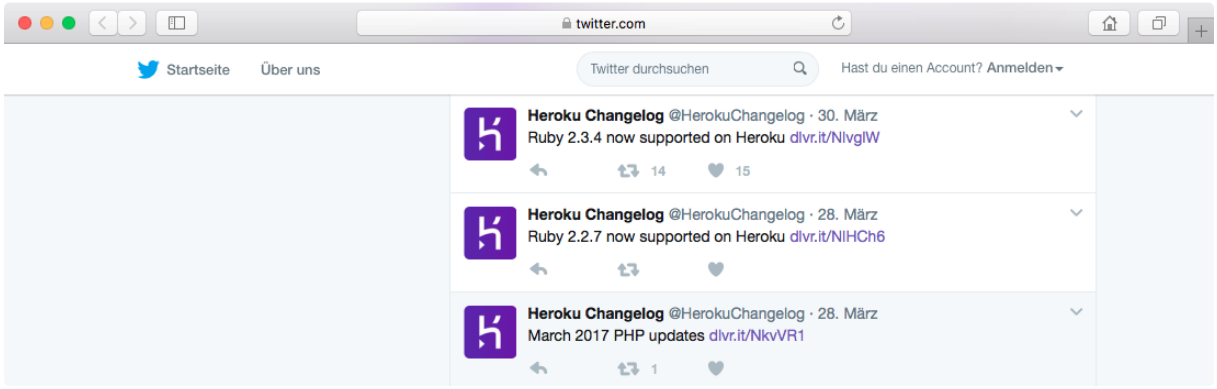

Abbildung 24: Heroku Twitter

Es bleibt also dem Entwickler offen, ob man sich für einen News-Dienst wie Twitter oder Web-Feeds entscheidet. In beiden Fällen kommt man auf die gleiche Information, die unterschiedlich repräsentiert wird. Während Twitter eine umfangreiche REST-API mit Daten in JSON Format anbietet, setzt RSS auf das XML-Format. Die RSS-Daten können mit dem HTTP-GET-Request aufgerufen werden.
Unabhängig davon, ob die Datenerfassung über Twitter API oder RSS erfolgt, können die Daten wiederum an die Worker-API gesendet werden. Im einfachsten Fall werden sie an das Github-Repository weitergegeben. In der fortgeschrittenen Variante können die Daten nach bestimmten Kriterien gefiltert bzw. verarbeitet werden.

6 Fazit

Literatur

[AF99] A. Sahuguet und F. Azavant: *Building light-weight wrappers for legacy Web data-sources using W4F.* In: *VLDB*, 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.

[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.

[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.

[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.

[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.

[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.

[PT11] P. Mell und T. Grance: *The NIST Definition of Cloud Computing.* NIST Special Publication 800 - 145, 2011.

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

[R.T00] R.T. Fielding: *Architectural Styles and the Design of Network-based Software Architectures.* Doktorarbeit, University of California, Irvine, 2000.

[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: *VLDB*, 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 31.03.2017 Petr Vasilyev