Erfahrungsmanagement mit fallbasierten Assistenzsystemen

Prozesse, Konzepte und Anwendungsbeispiele in einem ganzheitlichen Rahmenwerk

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Frau Dipl.-Inf. Mirjam Minor
geboren am 04.07.1972 in Heilbronn

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Uwe Küchler

Gutachter:

1. Prof. Dr. Hans-Dieter Burkhard
2. Prof. Dr. Ralph Bergmann
3. Prof. Dr. Gerd Stumme

eingereicht am:	13. Februar 2006
Tag der mündlichen Prüfung:	18. Mai 2006

Abstract

Experience Management (EM) is a special form of Knowledge Management that deals with task-based knowledge. This thesis provides a framework for assistant systems that support human beings in EM tasks. It deals not only with technical issues (how to collect, structure, store, retrieve, and reuse experiential knowledge), but als with organizational issues (how to evaluate and maintain it) and psychosocial questions (how to integrate an EM system, how to avoid barriers, how to evaluate the success of the whole system).

Case-based sample applications from both, industrial and experimental scenarios, show to what extend the particular EM processes can be supported or which sub-processes can even be automated. By means of experiments with these implemented samples, we evaluate the topics that are discussed at the beginning of each application chapter.

Keywords:
case-based reasoning, knowledge management, agents, socio-technical systems

Zusammenfassung

Erfahrungsmanagement (EM) ist eine Spezialform des Wissensmanagements, die sich mit aufgabenbezogenem Wissen beschäftigt. Diese Arbeit entwickelt ein Rahmenwerk für Assistenzsysteme, die Menschen bei EM-Aufgaben unterstützen. Es untersucht nicht nur technische Fragen (Erfahrungswissen sammeln, strukturieren, speichern und wiederverwenden) sondern auch organisatorische (Erfahrungswissen evaluieren und pflegen) und psychosoziale Aspekte (ein EM-System integrieren, Barrieren vermeiden, den Systemeinsatz bewerten).

Fallbasierte Anwendungsbeispiele für industrielle und experimentelle Szenarien zeigen, welche Prozesse wo unterstützt oder gar teilautomatisiert werden können. Sie dienen der experimentellen Evaluierug der Fragen, die ich zu Beginn jedes Anwendungskapitels formuliert habe.

Schlagwörter:
Fallbasiertes Schließen, Wissensmanagement, Agentenorientierte Programmierung, Soziotechnische Systeme

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die vorliegende Dissertation entwickelt ein Rahmenwerk für Assistenzsysteme zum Erfahrungsmanagement, erläutert an praktischen Anwendungen des Fallbasierten Schließens. Abkürzend wird in dieser Arbeit von Erfahrungsmanagement-Systemen (EM-Systemen) gesprochen. Das Rahmenwerk orientiert sich an Probsts Modell von den Bausteinen des Wissensmanagements. Die sechs Kernprozesse des Wissensmanagements bei Probst werden im Licht von Bergmanns ganzheitlichem Experience-Management-Modell betrachtet und auf Erfahrungsmanagement-Systeme übertragen. Die in dieser Arbeit vorgestellten neuen Konzepte und Systeme dienen zur technischen Begleitung von Erfahrungsmanagement. Sie werden anhand praktischer Anwendungen und einer Interaktivitätsstudie experimentell evaluiert. Die praktischen Anwendungen sind in eigenen Projekten entstanden, zum Teil im Rahmen von mir betreuter Diplomarbeiten.

Die wichtigsten wissenschaftlichen Fragestellungen dieser Arbeit sind folgende:

- Probst 2003 fordert eine ganzheitliche Herangehensweise an das Wissensmanagement. Wie kann man neben der rein technischen Sichtweise eine organisatorische Perspektive tatsächlich in EM-Systeme einbringen?
- Wie kann man Erfahrungswissen gewinnen (Wissenserwerb, Bezug zur Wissensentwicklung)?
- Wie kann man Erfahrungswissen weiterentwickeln (Wissensentwicklung)?
- Wie kann man Erfahrungswissen austauschen (Wissensverteilung)?

- Wie kann man ein EM-System organisieren? Wie kann man dafür sorgen, dass Erfahrungswissen genutzt wird (Wissensnutzung)?
- Wie kann man Erfahrungswissen aktuell halten (Wissensbewahrung)?
- Welche Teilprozesse lassen sich automatisieren? Welche Teilprozesse sollten beim Menschen bleiben?

1.1 Begriffliche Präliminarien

Bevor die Konzepte und Verfahren dieser Arbeit vorgestellt werden, nehme ich in eigenen Worten zu Begriffen Stellung, die sehr kontrovers diskutiert werden.

Definition 1.1.1 („Intelligentes“ Computersystem)

> Ein *„intelligentes“ Computersystem* ist ein Computersystem, das Methoden der Künstlichen Intelligenz einsetzt.

Die Anführungszeichen in dieser Definition bringen meine Ansicht zum Ausdruck, dass die Intelligenz eines Systems eine Metapher für intelligent erscheinendes Verhalten ist. Echte Intelligenz ist meines Erachtens an ein Individuum mit einer eigenen Lebenserfahrung und einem Körper gebunden. Es gibt derzeit (noch) keine künstlichen Systeme, die die Bezeichnung Individuum verdienen.

Alle Anwendungsbeispiele dieser Arbeit beziehen sich auf eine Domäne:

Definition 1.1.2 (Domäne)

> Eine *Domäne* ist ein Aufgabenbereich oder Weltausschnitt, in dem ein Computersystem eingesetzt werden kann.

Ein Beispiel für eine sehr breite Domäne ist die Biologie, eine engere Domäne ist zum Beipiel die technische Diagnose von Druckerproblemen.

Definition 1.1.3 (Assistenzsystem)

> Ein *Assistenzsystem* ist ein „intelligentes“ Computersystem, das Menschen oder andere Computersysteme bei einer zielgerichteten Tätigkeit unterstützt.

Assistenzsysteme für Wissens- und Erfahrungsmanagement müssen „intelligente“ Systeme sein, weil sie einen eigenen Anteil an kognitiven Prozessen übernehmen sollen. Zum Beispiel kann ein Assistenzsystem durch Inferenz Zusammenhänge zwischen Informationen entdecken und in Form von Querverweisen ablegen (siehe Kapitel 5). Oder es kann zu einer aktuellen Aufgabe passende Beschreibungen suchen, indem es inhaltliche Vergleiche anstellt (siehe Kapitel 4). Natürlich gibt es auch ganz einfache, wirkungsvolle Hilfsmittel beim Wissens- und Erfahrungsmanagement, wie zum Beispiel eine FAQ-Datenbank, eine Verzeichnisstruktur oder gar ein Telefon, mit dem man einen Experten befragen kann. Diese Hilfsmittel zählen aber nicht zu den Assistenzsystemen.

Definition 1.1.4 (Akteur)

> Ein *Akteur* ist ein Mensch oder ein technisches Artefakt, der oder das in Eigeninitiative Handlungen ausführen kann.

Agentenorientierte Programme (siehe unten) und autonome Roboter sind Beispiele für technische Artefakte, die zu den Akteuren gehören. Manche Assistenzsysteme für das Wissens- und Erfahrungsmanagement können jetzt schon in sehr eingeschränktem Rahmen autonom agieren, zum Beispiel neues Wissen ansammeln (siehe Kapitel 5) oder Wissen austauschen (siehe Kapitel 7). In der Literatur gibt es „Mixed-initiative“- [Cox99, TAB+03] oder hybride Systeme [RSS02], bei denen die Initiative wechselseitig von natürlichen und künstlichen Akteuren ausgeht. In dieser Arbeit verwende ich den Begriff eines Akteurs immer dann, wenn ich nicht festlegen möchte, ob es sich beim Handlungsträger um einen Menschen oder einen Agenten handelt.

Definition 1.1.5 (Agent, Software-Agent, agentenorientiertes Programm)

> Ein *Agent* (*Software-Agent*, *agentenorientiertes Programm*) ist ein künstlicher Akteur in Form eines Computerprogramms.

Diese Definition ist bewusst pragmatisch gewählt und spiegelt nicht die umfangreiche Diskussion des Agentenbegriffs in der Literatur.

1.2 Leitfaden dieser Arbeit

Diese Arbeit ist in drei Teile gegliedert:

Teil I	beschäftigt sich mit den grundlegenden Prozessen und Konzepten von Wissensmanagement, Erfahrungsmanagement und Fallbasiertem Schließen.
Teil II	beschreibt meine eigenen Anwendungsbeispiele zu den jeweiligen Kernprozessen.
Teil III	ordnet diese Arbeit in den wissenschaftlichen Kontext ein.

Hervorheben möchte ich zwei Kapitel des ersten Teils, aus denen sich die Gliederung für Teil II dieser Arbeit ableitet:

- In Kapitel 2.3 referiere ich die **Kernprozesse des Wissensmanagements** nach dem Modell von Probst et al. [PRR99].
- In Kapitel 2.4 diskutiere ich für jeden der Probst-Prozesse das **Potential zur Automatisierung** und den heutigen Forschungsstand von **Assistenzsystemen für Wissens- und Erfahrungsmanagement**.

Zugleich enthalten die Grundlagen in Teil I einige neue Konzepte, die ich entwickelt oder mitentwickelt habe, zum Beispiel die ganzheitliche Herangehensweise an Erfahrungsmanagement, die psychologische Erkenntnisse mit einschließt (siehe Kapitel 3.3), oder das Fallbasierte Schließen für Texte (siehe Kapitel 4.3). Die vorliegende Arbeit und insbesondere das Raster in Kapitel 2.4 kann als handlungsorientiertes Rahmenwerk für ganzheitliches Erfahrungsmanagement eingesetzt werden.

Teil I

Hintergrund

Kapitel 2

Wissensmanagement

Das Thema Wissensmanagement (Knowledge Management, KM) hat seit Mitte der neunziger Jahre einen enormen Aufschwung erlebt [Leh00]. In verschiedenen Wissenschaftsdisziplinen sind Theorien entstanden, inspiriert und belegt durch zahlreiche praktische Anwendungen. Alle großen Unternehmen haben sich inzwischen mit Wissensmanagement auseinander gesetzt. In letzter Zeit flaut dieser Trend wieder ab [Pro03].

In der Künstlichen Intelligenz (KI) wird jenseits von Modetrends seit Jahren daran geforscht, wie man Wissen nutzbringend in Systeme bringen kann. Wissensmanagement ist sowohl als modernes Anwendungsgebiet für KI-Systeme von Bedeutung als auch für die Organisation der wissensbasierten Systeme selbst.

In Kapitel 2.1 setze ich mich mit Definitionen von Wissen und Wissensmanagement auseinander. In Kapitel 2.2 stelle ich das klassische Modell der Wissensschaffung nach Nonaka und Takeuchi vor, gefolgt von dem Prozessmodell von Probst et al. in Kapitel 2.3. In Kapitel 2.4 diskutiere ich Assistenzsysteme für Probsts Kernprozesse und entwickle daraus die Gliederung für Teil II dieser Arbeit.

2.1 Wissensbegriff und Wissensmanagement

Seit Jahrtausenden ringen Wissenschaftler verschiedenster Disziplinen um eine Definition des Begriffs „Wissen“. In der Philosophie ist der Wissensbegriff seit Plato und Aristoteles eng verknüpft mit der Vorstellung über die „Wahrheit“. Platos Ideenlehre [Pla63][Vor02, §21] geht davon aus, dass hinter unseren Vorstellungen und Begriffen absolute, göttliche Ideen stehen. Unsere Wahrnehmungen in der diesseitigen Welt lösen Erinnerungen an das aus, was wir im Jenseits gesehen haben. So sind unsere Bilder von Pferden, die wir im

Kopf haben, lediglich Abglanz der absoluten Idee „Pferd“. Im Höhlengleichnis [Pla88, Siebentes Buch, 514, S. 301ff.][Pla05] ringen die Menschen um die Erkenntnis der Wahrheit, die sich hinter den von ihnen wahrgenommenen Schatten verbirgt.

Es würde den Rahmen dieser Arbeit sprengen, die interessante Entwicklung verschiedenster Wissensbegriffe über die Jahrhunderte zu verfolgen. Deshalb konzentriere ich mich in der nachfolgenden Diskussion auf einen kleinen Ausschnitt heutiger Wissensbegriffe aus der Psychologie, den Wirtschaftswissenschaften und der Informatik und bilde dabei einen eigenen, praxisorientierten Wissensbegriff für diese Arbeit. Für ausführlichere Betrachtungen des Wissensbegriffs verweise ich auf die Literatur [WDA99, Rol00, GS03].

Aus **psychologischer Perspektive** ist ein lernorientierter Wissensbegriff am besten geeignet. Als ein Beispiel unter zahlreichen Definitionen stelle ich die von Friedhart Klix vor:

Definition 2.1.1 (Wissen aus psychologischer Sicht nach [Kli92])

> Wissensstrukturen im menschlichen Gedächtnis sind Resultate von Lernprozessen.

Klix beschreibt vier Bedingungskomplexe, aus denen heraus Wissen entsteht:

- aus Wahrnehmungen der Sinnesorgane,
- aus individuellen Erfahrungen, sogenannten „Situations-Aktions-Lernvorgängen“,
- aus sprachlicher Belehrung und
- aus Nachdenken, also auf reflexiver Ebene.

Diese Beschreibung kommt der Informatik-Perspektive sehr entgegen, da sie die (menschliche) Informationsverarbeitung einbezieht, ohne direkt auszuschließen, dass auch in Computern Wissen dokumentiert werden kann.

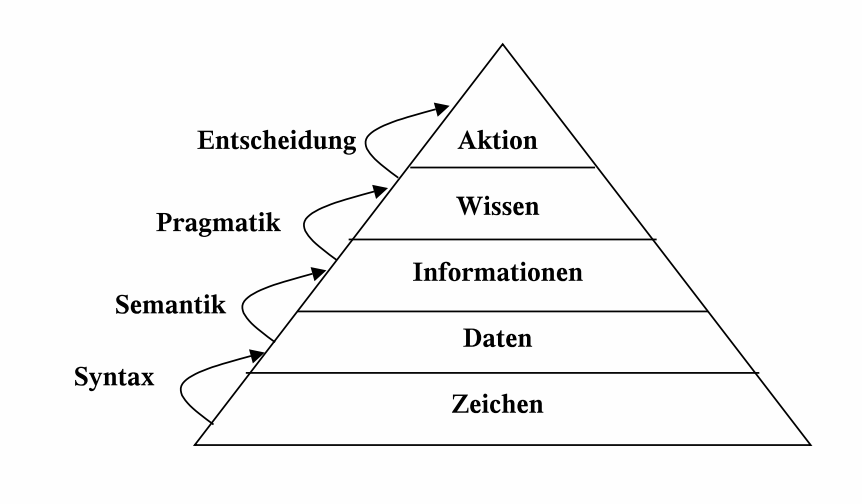


Abbildung 2.1: Die Wissenspyramide nach [AN95, Übersetzung aus [WDA99]].

In den **Wirtschaftswissenschaften** wird meist ein prozessorientierter Wissensbegriff verwendet.

Definition 2.1.2 (Wissen aus wirtschaftswissenschaftlicher Sicht nach [NT97, S. 70])

> Wissen ist ein dynamischer menschlicher Prozess der Erklärung persönlicher Vorstellungen über die „Wahrheit“.

In der **Künstlichen Intelligenz** wird in vielen Publikationen gar keine Definition des Wissensbegriffs gegeben. Statt dessen wird gleich von „Wissensrepräsentation“ gesprochen und ein intuitives Verständnis dieses Begriffs einfach vorausgesetzt, um Formalismen zur Wissensrepräsentation wie Regeln, Logik, Frames, Falldaten oder Ontologien zu beschreiben. Im Hintergrund steht – oft unausgesprochen – ein technikorientierter Wissensbegriff, der durch die Entwicklung von Expertensystemen motiviert ist. Ein gutes Beispiel dafür ist die Wissenspyramide nach Aamodt und Nygård (siehe Abbildung 2.1), die die klassische Dreiteilung in Daten, Informationen und Wissen aufgreift. Zum Beispiel ist die Zahl „100“ mit der Syntax, dass Zahlen aus Ziffern, höchstens einem Komma und einem Vorzeichen bestehen, ein abstraktes Datum. Durch den Zusatz einer Geschwindigkeitseinheit wie „100 km/h“ wird daraus eine Information, das heißt eine Interpretation des Datums mit Hilfe der Semantik ist möglich. Wissen wird daraus erst durch eine

Pragmatik, eine Anwendungsmöglichkeit, zum Beispiel wird ein Autofahrer zu der Entscheidung gebracht zu bremsen, wenn er auf dem Tacho die Geschwindigkeit 100 km/h erkennt und gleichzeitig ein Ortsschild wahrnimmt. Die Repräsentation von Daten, Informationen oder Wissen in Computersystemen ist im allgemeinen gleich, sie besteht nämlich aus Zeichen. Bergmann leitet aus der Wissenspyramide folgende Definition von Wissen ab:

Definition 2.1.3 (Wissen aus technischer Sicht nach [Ber02, S. 26])

> Wissen ist eine Menge verknüpfter Informationen mit Pragmatik. Wissen setzt Informationen in einen aufgabenbezogenen oder zielorientierten Kontext.

Die Pragmatik gibt an, wie das Wissen angewendet werden kann. Ich ergänze Bergmanns Definition um eine subjektive Komponente, die den Anspruch auf die allgemeingültige Wahrheit von Wissen aufgibt. Meine Definition gilt für die Betrachtung von Assistenzsystemen.

Definition 2.1.4 (Wissen aus Assistenzsystem-Sicht)

> Wissen ist eine Menge verknüpfter Informationen mit Pragmatik im Konsens einer Gruppe von Individuen. Wissen setzt Informationen in einen aufgabenbezogenen oder zielorientierten Kontext.

Diese Definition schlägt eine Brücke zwischen der technischen und der lernorientierten Sicht: Einerseits kann Wissen damit für Assistenzsysteme in Wissenscontainern dokumentiert werden (vergleiche dazu S. 39), andererseits ist aber klar, dass das Wissen nicht losgelöst von individuellen Sichtweisen ist. Ich entziehe mich einer erkenntnistheoretischen Diskussion über die Wahrheit von Wissen und berufe mich darauf, dass mein Wissensbegriff ausschließlich in Bezug auf Assistenzsysteme verwendet wird, die jeweils nur einen bestimmten Weltausschnitt betrachten. In diesem Weltausschnitt ist das Wissen aus der Sicht der beteiligten Individuen wahr. Dabei ist nicht ausgeschlossen, dass in Zukunft auch Agentenprogramme oder Roboter die Rolle von Wissensträgern einnehmen. Dies geht aber nach obiger Definition erst dann, wenn ihnen eine Individualität zum Beispiel durch eigenes „Hinzulernen“ zugesprochen werden kann. Dann könnte meine Definition aus der Sicht der Individuen durch eine Definition aus der Sicht der Akteure ersetzt werden. Im Moment spreche ich lieber davon, dass das Wissen in den Assistenzsystemen dokumentiert ist.

Bergmann [Ber02] stellt verschiedene Unterteilungen des Wissensbegriffs vor, von denen in dieser Arbeit die folgenden eine Rolle spielen: die Unterteilung in implizites und explizites Wissen nach Polanyi (siehe S. 12), die

Unterteilung in allgemeines und spezifisches Wissen in Kapitel 3.1, und die Dokumentation des Wissens in Wissenscontainern in Kapitel 4 auf S. 39. Ich ignoriere in dieser Arbeit die in der klassischen KI typische Unterteilung in deklaratives und prozedurales Wissen, da alle Wissenscontainer einen deklarativen und gleichzeitig einen prozeduralen, aufgabenbezogenen Charakter haben.

Nachdem nun der in dieser Arbeit verwendete Wissensbegriff geklärt ist, können wir uns dem Begriff „Wissensmanagement“ zuwenden.

Definition 2.1.5 (Wissensmanagement, WM nach [Epp05, eigene Übersetzung])

Wissensmanagement (WM) ist ein

- systematischer Ansatz (mit Wurzeln in der Informationstechnologie, der Personalwirtschaft, der Firmenstrategie und der Leitung von Unternehmen),
- bei dem implizites und explizites Wissen eine strategische Schlüsselrolle spielt
- und der zum Ziel hat, den Umgang mit Wissen auf individueller Ebene, im Team, in der Organisation und zwischen Organisationen zu verbessern,
- um die Innovation, die Qualität, die Effektivität der Kosten und die Vorlaufzeit von Produkten zu verbessern.

Aus dieser Definition wird ersichtlich, dass WM Wurzeln in verschiedenen Disziplinen hat. Deshalb ist eine interdisziplinäre Herangehensweise an WM sehr natürlich, zumindest reicht eine rein technische Herangehensweise an WM-unterstützende Systeme nicht aus, wie Abbildung 2.2 durch die drei Säulen veranschaulicht. Wolf et al. gehen sogar so weit zu sagen: „Alle Wissensmanagementaktivitäten müssen von den Bausteinen Organisation, Menschen und Technologie, die in eine adäquate Unternehmenskultur eingebettet sind, getragen werden.“ [WDA99, S. 752]. Ist dies nicht der Fall, stürzt das „Wissensmanagementgebäude“ ein. Auch Probst spricht davon, dass WM „klar in einen wirtschaftlichen und sozialen Kontext eingebettet sein“ muss [Pro03]. In Kapitel 3.3 wird diese Forderung in Bezug auf EM-Systeme näher beleuchtet. Bei den wirtschaftlichen Aspekten liegt der Schwerpunkt dabei nicht auf finanzwirtschaftlichen Berechnungen zum return-on-invest sondern auf organisatorischen Belangen.

Die Begriffe „implizites“ und „explizites“ Wissen werden genau wie die Ebenen, auf denen mit Wissen umgegangen wird, im folgenden Kapitel ausführlich diskutiert. Der letzte Punkt der Definition erfordert eigentlich die

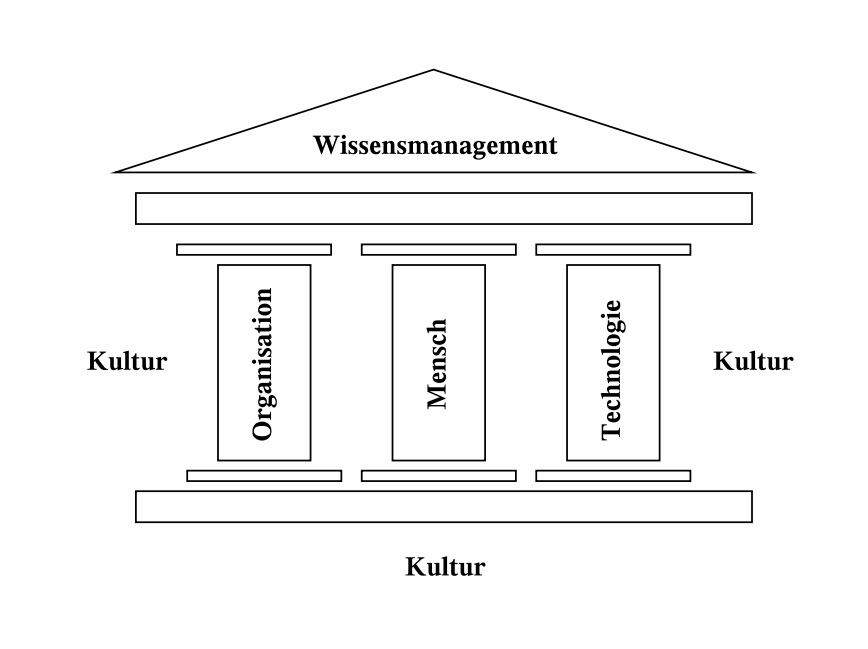


Abbildung 2.2: Die drei Säulen des Wissensmanagements nach [WDA99].

Entwicklung von Messmethoden für den Erfolg oder das Scheitern von Wissensmanagementaktivitäten. Noch gibt es aber nur wenige praktikable Ansätze dazu [Lee00, Wur05]. Dieser Punkt wird deshalb in der Arbeit nur durch Analyse von Benutzungsdaten in Kapitel 8 umgesetzt.

2.2 Das Spiralmodell der Wissensschaffung nach Nonaka und Takeuchi

Die beiden japanischen Wirtschaftswissenschaftler Ikujiro Nonaka und Hirotaka Takeuchi zählen zu den Begründern des WM. In den neunziger Jahren entstand ihr Spiralmodell der Wissensschaffung in Organisationen [NT95, NT97]. Es ist das Ergebnis einer Analyse von japanischen Unternehmen, die erfolgreich Innovationen erzeugt haben. Das Spiralmodell ist das meistzitierte Modell für WM, obwohl die beiden Autoren die westlichen WM-Anstrengungen und den Begriff „Wissensmanagement“ sehr kritisch sehen [NT00]. Statt vorhandenes Wissen zu managen fordern sie, die Schaffung von neuem Wissen ins Zentrum zu rücken. In dieser Arbeit bleibe ich trotz der fernöstlichen Kritik bei meinem WM-Begriff, da WM sich in der westlichen Welt eingebürgert hat und die Bemühungen zur Wissensschaffung mit

einschließt. Im Folgenden wird das Spiralmodell von Nonaka und Takeuchi zunächst vorgestellt und dann für eine naturwissenschaftliche Herangehensweise an WM präzisiert.

Zwei „Dimensionen“ der Wissensschaffung

Das Spiralmodell benutzt zwei „Dimensionen“, die nicht im mathematischen Sinn zu verstehen sind: die ontologische und die epistemologische Dimension. Die ontologische Dimension ist nicht zu verwechseln mit dem Ontologie-Begriff auf S. 73. Sie unterscheidet vier Ebenen:

1. Wissenserzeugung beim Individuum,
2. Wissenserzeugung in der Gruppe,
3. Wissenserzeugung im gesamten Unternehmen und
4. Wissenserzeugung bei der Interaktion zwischen Unternehmen.

Damit auf allen vier Ebenen neues Wissen entstehen kann, sollten folgende drei Voraussetzungen gegeben sein: Einzelpersonen sind kreativ, Unternehmen verstärken dies und die Unternehmen selbst sind in einem Wissensnetz verankert.

Die epistemologische Dimension unterscheidet zwei Formen von Wissen: explizites und implizites Wissen. Nonaka und Takeuchi beziehen sich in ihrer Definition auf den ungarischen Philosophen Michael Polanyi [Pol85], formulieren aber prägnanter als das Original:

Definition 2.2.1 (Implizites und explizites Wissen nach [NT97, S. 72])

> *Implizites Wissen* ist persönlich, kontextspezifisch und daher nur schwer kommunizierbar.
>
> *Explizites Wissen* lässt sich in formaler, systematischer Sprache weitergeben.

Polanyi sieht das explizite Wissen eines Menschen nur als die Spitze des Eisbergs an, wenn er in [Pol85, S. 14] feststellt, „...dass wir mehr wissen, als wir zu sagen wissen.“ Polanyi geht davon aus, dass die Menschen durch aktive Schaffung und Organisation von Erfahrungen Wissen erwerben. Dabei spielt das implizite Wissen eine wichtige Rolle für das menschliche Erkennen. Wir können beispielsweise den Gesichtsausdruck eines Nachbarn, dem wir begegnen, mit Hilfe von implizitem Wissen deuten, um seine Gefühle zu bestimmen.

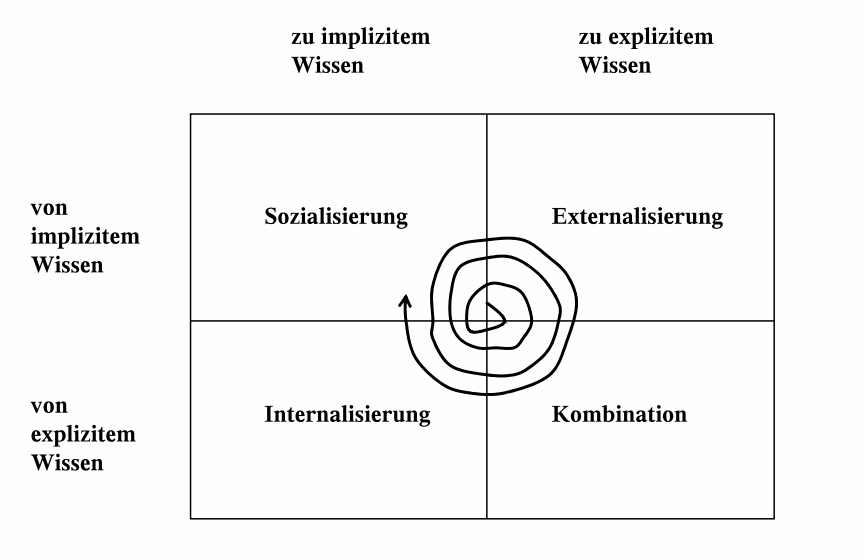


Abbildung 2.3: Die Wissensspirale mit den vier Formen der Wissensumwandlung nach [NT97].

Wissensschaffung durch Wissensumwandlung

Nonaka und Takeuchi bezeichnen das Wechselspiel von implizitem und explizitem Wissen als „Wissensschaffung durch Wissensumwandlung“. Im Folgenden wird ihr semi-formales Spiralmodell dazu vorgestellt. Die beiden Wissensformen implizites und explizites Wissen können in einem sozialen Prozess zwischen Menschen ineinander überführt werden. Dabei entsteht neues Wissen in impliziter oder expliziter Form.

Definition 2.2.2 (Das Seki-Modell nach [NT97])

Es gibt vier Formen der Wissensumwandlung (siehe Abbildung 2.3):

1. Die **Sozialisation** erzeugt durch Austausch von implizitem Wissen neues implizites Wissen.
2. Die **Externalisierung** macht implizites Wissen explizit.
3. Die **Kombination** von explizitem Wissen produziert neues explizites Wissen.
4. Die **Internalisierung** ist das Verinnerlichen von explizitem Wissen.

Die Spirale im Modell deutet an, dass es sich bei der Wissenserzeugung um einen iterativen Prozess handelt. Er beginnt mit der Sozialisation, der Weitervermittlung von implizitem Wissen. Dabei entstehen gemeinsame mentale Modelle oder gemeinsame technische Fertigkeiten. Auf die Sozialisation folgt die Externalisierung, üblicherweise im Dialog zwischen Menschen, bei dem durch Artikulation von implizitem Wissen explizite Konzepte entstehen. Nach der Externalisierung wird bei der Kombination explizites Wissen mit anderem explizitem Wissen verbunden, wobei weiteres Wissen neu entstehen kann. Nonaka und Takeuchi sprechen selbst davon, dass die Wissenskombination gut durch Computersysteme unterstützt werden kann. Der vierte Prozess, die Internalisierung von explizitem Wissen, ist nahe verwandt mit dem „learning by doing“. Dokumente helfen bei der Internalisierung von Erfahrungen und erleichtern die Übermittlung von explizitem Wissen an andere. Nach der Internalisierung folgt wieder die Sozialisation, dann die Externalisierung und so weiter.

Verbindet man beide Dimensionen miteinander, entsteht ein dreidimensionales Modell. Die Spirale zieht immer weitere Kreise und geht vom Individuum aus in immer höhere ontologische Ebenen. Dabei durchläuft sie wiederholt alle vier Prozesse der Wissensumwandlung.

Diskussion und Präzisierung

Die Begriffe aus Nonaka und Takeuchis Modell sind zu unpräzise, um als Grundlage für ein naturwissenschaftliches Modell zu dienen. Vielleicht ist dies einer der Gründe für den in [SKMH04] diagnostizierten Bruch zwischen der Theorie und Praxis des WM, dass nämlich implizites Wissen in der Praxis so gut wie nicht betrachtet wird, obwohl das Seki-Modell (siehe S. 13) häufig als theoretische Grundlage zitiert wird. In dieser Arbeit spielen beide Formen, explizites und implizites Wissen, eine Rolle, so dass ich auf die Ungenauigkeiten und Differenzen innerhalb des Modells von Nonaka und Takuchi im Folgenden eingehe.

Nonaka und Takeuchis Erläuterungen zu implizitem und explizitem Wissen sind widersprüchlich. Sie beschreiben beispielsweise Erfahrung als implizites Wissen, das im Gegensatz zu explizitem Verstandeswissen steht. Andererseits sagen die Autoren, dass Erfahrungen internalisiert werden können, das heißt es muss auch eine explizite Ausdrucksweise für Erfahrung geben. Letzteres entspricht auch meiner Anschauung (vergleiche Kapitel 3), steht aber im Widerspruch zu Nonaka und Takeuchis Charakterisierung von implizitem Wissen. Es gibt Versuche, die Begriffe anders zu definieren, zum Beispiel statt von implizitem von stillschweigendem Wissen zu sprechen (wie in Claudia Müllers Vortrag zu [GMU+04]) oder anstelle von explizitem und

implizitem Wissen codierbares und nicht codierbares Wissen zu unterscheiden. Ich löse das Durcheinander der Begriffe für diese Arbeit so auf, dass ich nur die wörtlichen Definitionen für implizites und explizites Wissen von Polanyi (siehe S. 12) übernehme und Nonaka und Takeuchis widersprüchliche Erläuterungen ignoriere.

Ich präzisiere den Begriff „implizites Wissen“, so dass folgende Unterteilung entsteht, die zweckdienlich für das Dokumentieren von Wissen in Wissenscontainern (siehe S. 39) für Assistenzsysteme ist:

- **implizites** Wissen
  - **explizierbares implizites** Wissen, das heißt implizites Wissen, das in eine explizite Form gebracht und in Wissenscontainern dokumentiert werden kann
  - **nicht explizierbares implizites** Wissen, das ausschließlich durch Sozialisation oder gar nicht weitergegeben werden kann
- **explizites** Wissen

Explizites und explizierbares Wissen kann so umgeformt werden, dass es in Wissenscontainern abgespeichert werden kann. Nicht explizierbares implizites Wissen kann im Moment nur außerhalb von Assistenzsystemen berücksichtigt werden, da Assistenzsysteme ja noch keine eigene Körperlichkeit haben und deshalb kein implizites Wissen austauschen können, ohne es vorher explizit zu machen (siehe auch die Diskussion zum allgemeinen Wissensbegriff auf S. 9). Um den oben genannten Bruch zwischen Theorie und Praxis in der WM-Forschung aber aufzulösen, ist es unerlässlich, nicht explizierbares implizites Wissen in das Rahmenwerk der Prozesse im Umfeld der Assistenzsysteme zu integrieren (vergleiche dazu die jeweiligen Anmerkungen in den Anwendungskapiteln in Teil II dieser Arbeit).

2.3 Die Bausteine des Wissensmanagements nach Probst et al.

Gilbert Probst et al. [PRR99] haben ein Wissensmanagement-Konzept für Führungskräfte entwickelt, das im deutschsprachigen Raum inzwischen weit verbreitet ist. Es ist ein wirtschaftswissenschaftliches Modell, das nach den Prinzipien des Action Research [Qui92, Qui93] entwickelt wurde, das heißt es verbindet theoretische und praktische Forschung. So ist ein handlungsorientiertes Analyseraster für Wissensmanagement entstanden, das sich auch

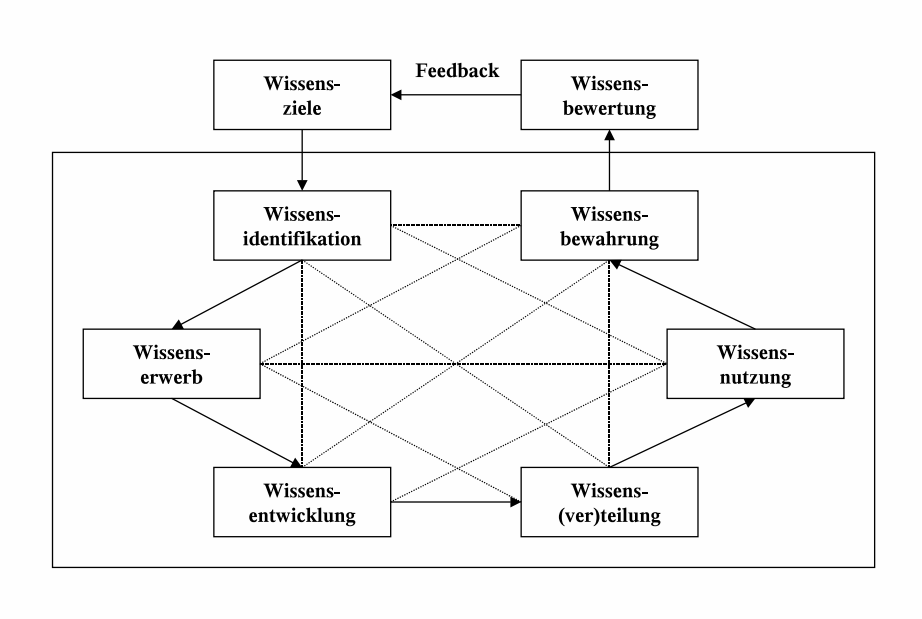


Abbildung 2.4: Die Kernprozesse des Wissensmanagements.

vorzüglich dafür eignet, ein Rahmenwerk für EM-Systeme zu strukturieren. Probsts Modell von 1999 und dessen Aktualisierung in [Pro03] ist Motivation und Leitfaden für Teil II dieser Arbeit.

Probst et al. haben sechs Kernprozesse des Wissensmanagements in Organisationen identifiziert (siehe Abb. 2.4). Sie stehen alle mehr oder weniger miteinander in Verbindung und sollten nicht isoliert betrachtet werden.

- Die *Wissensidentifikation* soll Transparenz über vorhandenes Wissen schaffen. Dazu ist es nötig herauszufinden, wo sich innerhalb und außerhalb der eigenen Organisation nützliches Wissen befindet. Der häufig zitierte Spruch „Wenn Siemens wüsste, was Siemens so alles weiß“ ist ein prominentes Beispiel für die Notwendigkeit des Wissensidentifikations-Prozesses.

- Der *Wissenserwerb* beschäftigt sich mit der Nutzbarmachung externen Wissens, sei es durch Rekrutierung von Wissensträgern, den Erwerb von Wissen anderer Firmen, zum Beispiel Kundenfirmen, oder den Kauf von Wissensprodukten.

- Die *Wissensentwicklung* hat die Entstehung neuen Wissens zum Ziel. Freiräume zum Beispiel durch Familiensinn, Fehlerfreundlichkeit oder

das Honorieren langfristiger Erfolge in einem Unternehmen fördern die Entwicklung neuen Wissens. So entsteht Wissen auch bei Aktivitäten, die traditionell nur als Leistungserstellung betrachtet werden.

- Die *Wissens(ver)teilung* betrifft den Prozess der Verbreitung bereits vorhandenen Wissens innerhalb der Organisation. Oft kann Wissen nur in persönlichem Austausch zwischen Individuen übertragen werden.
- Die *Wissensnutzung* ist Ziel und Zweck des Wissensmanagements. Die Anwendung fremden Wissens in typischen Arbeitssituationen wird allerdings durch eine Reihe von Barrieren beschränkt.
- Die *Wissensbewahrung* erzeugt ein organisatorisches Gedächtnis. Sie umfasst drei Teilprozesse, nämlich bewahrungswürdiges Wissen selegieren, in angemessener Form speichern und die Aktualisierung des organisatorischen Gedächtnisses sicherstellen.

Diese sechs Kernprozesse sind in einem Kreislauf angeordnet, an dem sich Wissensmanager orientieren können. Zusätzlich definiert Probst zwei strategische Bausteine:

- Die Bestimmung von *Wissenszielen* sollte den Anfang aller Wissensmanagement-Aktivitäten bilden.
- Eine *Wissensbewertung* misst den Erfolg der Lernprozesse und gibt die Möglichkeit zu Kurskorrekturen.

2.4 Assistenzsysteme für Wissensmanagement-Prozesse

In diesem Teilkapitel wird für jeden der sechs Kernprozesse des Probst-Modells diskutiert, ob und wie er auf sinnvolle Weise technisch unterstützt oder gar automatisiert werden kann. Der Fokus liegt auf EM-Systemen (vergleiche die Definition auf S. 27), das heißt die Assistenzsysteme sind nicht wie in Teil II der Arbeit auf fallbasierte EM-Systeme beschränkt. Die Übertragung von Probsts Modell passt auch auf klassische ontologiebasierte Systeme, „intelligente“ Groupware-Systeme oder weitere Alternativen mit Methoden der Künstlichen Intelligenz. Bordmittel wie die Suchfunktion in einem Editor oder eine SQL-Anfrage an eine Datenbank können zwar auch das Wissensmanagement unterstützen, zählen aber nicht zu den „intelligenten“ Assistenzsystemen (siehe Abgrenzung auf S. 3). Bei dem größten Teil des Wissens, das

bei EM-Systemen eine Rolle spielt, handelt es sich naturgemäß um Erfahrungswissen (für eine exakte Definition siehe Kapitel 3).

Es gibt Teilprozesse von Probsts Kernprozessen, für die eine Automatisierung sehr gut möglich ist. Andere sollten besser von Menschen ausgeführt oder zumindest kontrolliert werden. Die beiden strategischen Bausteine *Wissensziele* und *Wissensbewertung* werden nicht betrachtet, da sie in der Hand von Menschen bleiben sollen. Die *Wissensidentifikation* bleibt in dieser Arbeit als einziger der sechs Kernprozesse komplett in der Obhut des Menschen. Die anderen fünf Kernprozesse werden teilweise oder ganz automatisiert (siehe unten bzw. die Anwendungsbeispiele dazu in Teil II dieser Arbeit). Nonaka und Takeuchis vier Prozesse der Wissensschaffung (siehe Definition des Seki-Modells auf S. 13) werden in den Prozess der Wissensentwicklung integriert.

Wissensidentifikation

Die *Wissensidentifikation* wird in dieser Arbeit hauptsächlich durch die Entdeckung von Wissensquellen für die Assistenzsysteme umgesetzt.

Definition 2.4.1 (Wissensquelle für ein Assistenzsystem)

> Eine *Wissensquelle für ein Assistenzsystem* ist eine Quelle von Informationen oder Wissen, aus der Wissen für ein Assistenzsystem gewonnen werden kann.

Eine gute Wissensquelle für Assistenzsysteme ist der Erfahrungsschatz eines Menschen, sobald dieser Mensch sein Wissen explizit macht. Aber auch elektronische Datenbestände werden in dieser Arbeit als Wissensquelle genutzt. Darüber hinaus sind schriftliche Quellen und sogar beobachtete Objekte oder Systeme als Wissensquelle denkbar, sofern es einen Mechanismus gibt, wie das Assistenzsystem die Wissensquelle erschliessen kann. Systeme zur Schrifterkennung, Kameras und andere Sensoren bieten hier neue Möglichkeiten.

Folgende Kriterien spielen in dieser Arbeit neben der grundsätzlichen Erschließbarkeit eine Rolle für die Auswahl einer Wissensquelle:

- die Qualität,
- der passende Fokus und
- die Aktualität der Quelle

als die drei obersten Kriterien, außerdem nachgeordnet

- die einfache Erschließbarkeit der Quelle und
- die Orientierung am Netzwerkgedanken.

Der Netzwerkgedanke gibt vor, dass so weit als möglich Verbindungen vom Assistenzsystem zu Quellen des Erfahrungswissens hergestellt werden, statt die Wissensinhalte in den Erfahrungsschatz des Assistenzsystems zu kopieren. Bei Menschen als Wissensquellen ist der hohe Aufwand für Wissensakquisition abzuwägen gegen die Gefahr, dass der Erfahrungsschatz in dem Moment, wo er gebraucht wird, nicht zugänglich ist. Arbeitsüberlastung, Fluktuation oder vorübergehende Abwesenheit der betreffenden Person sind häufige Ursachen dafür.
Automatisierbarkeit: In der Literatur werden Assistenzsysteme beschrieben, die den Wissensidentifikationsprozess unterstützen und Teile davon automatisieren. Eines der ersten Beispiele für den Teilprozess Human Resource Management (HRM) ist das System ProPer [SMS00], das eine Ontologie von Mitarbeitern und Abteilungen einer Organisation auswertet. Der Aufwand für eine automatische Identifikation von Wissensquellen ist allerdings sehr hoch und diese ist nur dort zu verwirklichen, wo ein vollständiges und sauber strukturiertes Verzeichnis möglicher Wissensquellen vorliegt. Dies traf in keiner der in dieser Arbeit untersuchten Anwendungsdomänen zu und hätte sich auch nicht mit vernünftigem Aufwand erstellen lassen. Deshalb konzentriere ich mich in Teil II dieser Arbeit auf die fünf weiteren Kernprozesse.

Wissenserwerb

Der *Wissenserwerb* ist in dieser Arbeit der Prozess, Wissensquellen zu erschließen und Wissensinhalte in Assistenzsysteme zu integrieren. Für den Wissenserwerb gelten dieselben Kriterien wie für die oben diskutierte Auswahl von Wissensquellen. Die klare Zuordnung von Aktivitäten zu den Prozessen Wissenserwerb und Wissensentwicklung ist nicht leicht. Zum Beispiel könnte man die zeitliche Ausdehnung der Aktivität als Zuordnungskriterium verwenden: Alle automatischen und manuellen Verfahren, bei denen Wissen initial aus einer Wissensquelle extrahiert oder externalisiert wird, zählen nach dieser Aufteilung zum Wissenserwerb und alle Verfahren, bei denen Wissen nach einer gewissen Zeit wieder aufgegriffen und verändert wird, gehören zur Wissensentwicklung. Die Grenze dabei ist fließend. Ich habe mich gegen dieses Modell entschieden und argumentiere eher inhaltlich als zeitlich: Meine Zuordnung richtet sich danach, wann Wissen lediglich umgeformt oder extrahiert wird und wann neues Wissen gelernt oder entwickelt wird. Auch hier sind die Übergänge fließend, so dass ich zum Beispiel die Zuordnung

der Querverweis-Erzeugung in Kapitel 5.2 nach praktischen Gesichtspunkten entschieden habe. In Kapitel 5 werden verschiedene Umsetzungen des Wissenserwerbsprozesses beleuchtet.
Automatisierbarkeit: Der Wissenserwerb lässt sich unter bestimmten Voraussetzungen zu großen Teilen automatisieren. Dazu gehört zum Beispiel, dass das gesuchte Wissen bereits in elektronischer Form zugänglich ist und dass dem Assistenzsystem klar gemacht werden kann, welche Art von Wissen benötigt wird.

Wissensentwicklung

Die *Wissensentwicklung* kann in Bezug auf Assistenzsysteme auf zwei Arten erfolgen: Man kann Wissen *für* Assistenzsysteme entwickeln, also die Systeme mit neuem Wissen füllen. Auf der anderen Seite entwickelt sich neues Wissen *mit Hilfe* der Assistenzsysteme. Hier ergibt sich ein enger Bezug zu Nonaka und Takeuchi (siehe oben in Kapitel 2.2), die statt von Wissensmanagement von Wissensschaffung sprechen. Wir integrieren die vier Prozesse aus dem Seki-Modell (siehe S. 13) in den Prozess der Wissensentwicklung:

Probst et al. teilen mit Nonaka und Takeuchi die Beobachtung, dass bei der Externalisierung von Wissen neues Wissen in den Köpfen entsteht. Im Idealfall verbleibt das neue Wissen nicht in den Köpfen, sondern wird umgehend in das Assistenzsystem eingespeist. In dieser Arbeit wird die `Externalisierung` von Erfahrungswissen *für* Assistenzsysteme in Kapitel 6 beleuchtet. Sie besteht aus dem Beschreiben und Modellieren von Erfahrungen, so dass diese in den Erfahrungsschatz der Assistenzsysteme integriert werden können (siehe Tabelle 2.1). Die Kreativität der Autoren bei der Externalisierung *mit Hilfe* von Assistenzsystemen wird dabei gerne in Kauf genommen.

Auch die drei weiteren Prozesse der Wissensumwandlung und Wissenserzeugung nach Nonaka und Takeuchi werden in Tabelle 2.1 in Bezug zu Assistenzsystemen für das Erfahrungsmanagement gesetzt. Die `Kombination` von Wissen *für* die Assistenzsysteme wird in Kapitel 5 behandelt. Die Kombination *mit Hilfe* der Systeme passiert beim automatischen Wissenserwerb und in den Köpfen der Benutzer. Kombiniertes Wissen kann sich im Wissensschatz der Systeme niederschlagen. Die `Internalisierung` von Wissen bei Benutzung eines Assistenzsystems geschieht quasi nebenbei durch das Lesen und Anwenden fremder Erfahrungen. Internalisierung erzeugt *mit Hilfe* der Systeme neues Wissen in den Köpfen. Die `Sozialisation` von Erfahrungswissen findet ohne die Assistenzsysteme statt. Sie kann höchstens durch das Arbeiten mit den Systemen angeregt werden. Auch das Hinterlegen von Kontaktdaten der Autoren und die Werbung für die Systeme, so dass sie im Gespräch bleiben, mag zur Sozialisation von Erfahrungswissen

Tabelle 2.1: Die Wissensentwicklung bei Assistenzsystemen

Umwandlung von Wissen nach Nonaka und Takeuchi	Entwicklung von Erfahrungswissen	für AS	mit AS
Externalisierung	Erfahrungen beschreiben und modellieren	x	x
Kombination	verschiedene Erfahrungen heranziehen (lassen)	x	x
Internalisierung	fremde Erfahrungen lesen und anwenden		x
Sozialisation	außerhalb des Systems	ohne	

beitragen (siehe Kapitel 8). Im Spiralmodell von Nonaka und Takeuchi ist die Wissensentwicklung ein iterativer Prozess. Dies manifestiert sich in dieser Arbeit im Wiederaufgreifen und Weiterentwickeln von Erfahrungswissen für Assistenzsysteme (siehe Kapitel 6).

Automatisierbarkeit: Bei der Wissensentwicklung kann nur wenig automatisiert werden. Die Externalisierung von Erfahrungswissen muss natürlich durch diejenigen Menschen geschehen, in deren Köpfen sich das Wissen befindet. Die Assistenzsysteme können diesen Prozess anregen und fördern, aber nichts davon ersetzen. Auch die Internalisierung und die Sozialisation kann nur von Menschen durchgeführt werden. Es bleibt also nur die Kombination von Erfahrungswissen für die Automatisierung übrig. Es ist eine strittige Frage, ob die automatische Kombination von Wissen zur Erzeugung von Querverweisen (siehe Kapitel 5) tatsächlich neues Wissen erzeugt. Der größte Teil der Wissensentwicklung wird auf jeden Fall von Menschen geleistet, sei es nun *für* oder *mit Hilfe* von Assistenzsystemen.

Wissensverteilung

Die *Wissensverteilung* bei Assistenzsystemen kann nach dem *pull*- oder nach dem *push*-Prinzip geschehen. *Pull* heißt hier, dass Wissen bei Bedarf angefordert wird, zum Beispiel durch eine Anfrage oder einen Auftrag an ein Assistenzsystem. Voraussetzung dafür ist natürlich der freie Zugang zu den Wissensbeständen des Systems. Viele Systeme sind in Netzwerke wie zum Beispiel ein Intranet eingebunden. Andere müssen regelmäßig mit Updates versorgt werden. So wird an anderer Stelle eingespeistes Wissen verteilt. Kapitel 7 beschäftigt sich mit der Wissensverteilung. *Push* ist die proaktive Verteilung von Wissen. Unsere Beispielagenten in Kapitel 7 wenden das

Push-Prinzip nicht an, es wäre aber ein interessante Forschungsaufgabe, dies einmal mit ihnen auszuprobieren.
Automatisierbarkeit: Handelt es sich bei einem Assistenzsystem um ein agentenorientiertes Programm, kann das Abrufen von Wissen zu einem Teil automatisiert werden, weil das Programm selbst Wissen anfordern kann. Kooperierende Agenten tauschen Wissen und Fähigkeiten aus, um ihre Aufgaben zu bewältigen. Dann entsteht auf künstliche Weise genau das, was Probst et al. Wissensnetzwerke nennen. Sie funktionieren hier nach dem *pull*-Prinzip. Außerhalb dieser Arbeit gibt es Assistenzsysteme, die selbstständig E-Mails versenden, zum Beispiel das Medizinstudien-Managementsystem in [ML05]. Solche Systeme zeigen, dass sich auch die Wissensverteilung nach dem *push*-Prinzip automatisieren lässt.

Wissensnutzung

Die *Wissensnutzung* sicherstellen heißt in dieser Arbeit, dass für die Benutzung der Assistenzsysteme gesorgt wird. Die Barrieren für die Wissensnutzung sind hier nicht kaufmännischer Natur wie zum Beispiel fehlende Patente oder Lizenzen für Wissensbestände. Sie entsprechen eher organisatorischen und psychologischen Barrieren. Um diese auszuräumen, wird eine Strategie entwickelt, wie Assistenzsysteme bekannt gemacht werden können oder wie Mitarbeiter zur Wissensnutzung und -bereitstellung motiviert werden (siehe auch weiter oben zum Thema Wissensentwicklung und Kapitel 8).
Automatisierbarkeit: Teile der Werbe- und Motivationsstrategie zur Wissensnutzung sind automatisierbar.

Wissensbewahrung

Die *Wissensbewahrung* kann wie die Wissensentwicklung (siehe oben) in Bezug auf Assistenzsysteme auf zwei Arten erfolgen: *für* und *mit Hilfe* der Systeme. Die Systeme tragen durch das Speichern von dokumentiertem Erfahrungswissen zum Bewahren dieses Wissens bei. In dieser Arbeit wird die Wissensbewahrung *für* Assistenzsysteme durch die Aktualisierung des organisatorischen Gedächtnisses näher beleuchtet. Wie dies für Assistenzsysteme verwirklicht werden kann, zeigt Kapitel 9.
Automatisierbarkeit: Die Organisation der Wissensbewahrung lässt sich automatisieren. Die Durchführung hingegen braucht die Mitarbeit von Menschen.

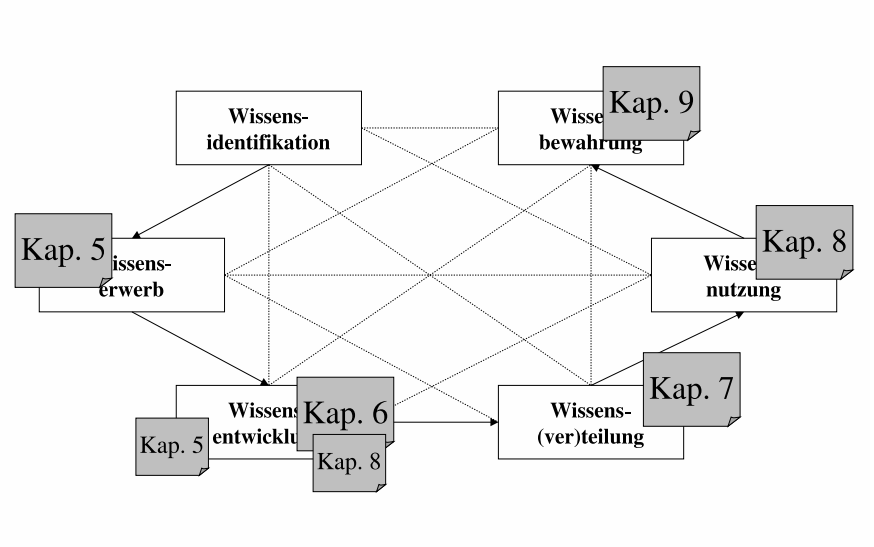


Abbildung 2.5: Zuordnung der Kernprozesse von Probst et al. zu den Kapiteln dieser Arbeit.

Das Automatisierungspotential der sechs Kernprozesse ist bei dem heutigen Stand der Forschung sehr unterschiedlich: Wissenserwerb und Wissensverteilung sind unter gewissen Voraussetzungen (siehe oben) schon zu großen Teilen automatisierbar, während bei den vier anderen Kernprozessen in Zukunft sicher noch mehr möglich ist. Zum Beispiel sind Techniken des maschinellen Lernens für die Wissensentwicklung vielversprechend. Auch für die Identifikation von Wissensquellen sehe ich Potential, beispielsweise durch die Klassifikation von Webseiten oder durch Data-Mining-Techniken.

Abbildung 2.5 zeigt die Zuordnung der Prozesse im Probst-Modell zu den folgenden Kapiteln der Arbeit:

- Kapitel 5 beschäftigt sich damit, Erfahrungswissen zu gewinnen (Wissenserwerb).
- Kapitel 6 untersucht, wie Erfahrungswissen weiterentwickelt werden kann (Wissensentwicklung).
- Kapitel 7 beschreibt, wie Akteure Erfahrungswissen austauschen (Wissens(ver)teilung).

- Kapitel 8 behandelt Vorgehensweisen, um ein EM-System zu organisieren (Wissensnutzung).
- Kapitel 9 enthält eine Strategie, um Erfahrungswissen aktuell zu halten (Wissensbewahrung)

Kapitel 3

Erfahrungsmanagement

Erfahrungsmanagement ist ein Teilgebiet des Wissensmanagements, das sich mit Erfahrungswissen befasst (siehe S. 26). Ich beschreibe im folgenden Kapitel die Grundlagen und Aufgaben des Erfahrungsmanagents. Dabei stelle ich ein ganzheitliches Modell von Ralph Bergmann vor, das neben dem technischen Kern von Erfahrungsmanagement-Systemen auch Wert auf organisatorische und kulturelle Aspekte legt. In Kapitel 3.3 erweitere ich die ganzheitliche Herangehensweise um psychologische Aspekte, die im Erfahrungsmanagement noch stärker als im allgemeinen Wissensmanagement zum Erfolg oder Misserfolg von Assistenzsystemen beitragen.

3.1 Grundbegriffe und Aufgaben des Erfahrungsmanagements

Zur Einführung des relativ neuen Forschungsgebiets „Erfahrungsmanagement“orientiere ich mich an den Arbeiten von Ralph Bergmann [Ber02], [ADH+01] und [Nic05]. Die folgende Definition ist eine Abwandlung der Definition in [Ber02, S. 28].

Definition 3.1.1 (Erfahrungswissen)

> *Erfahrungswissen* ist spezielles Wissen, das ein Akteur oder eine Gruppe von Akteuren in einem bestimmten Problemlösungskontext erworben hat.

Erfahrungswissen ist aus den Erfahrungen eines oder mehrerer Akteure entstanden. Es gilt in Situationen, die zu einem bestimmten Aufgabenbereich gehören, zum Beispiel zum Konfigurieren von Mobiltelefonen oder zum Anleiten von Arbeitsgruppen. Im Gegensatz zu Bergmanns Definition muss

bei mir Erfahrungswissen nicht von vornherein wertvoll sein. Wenn Erfahrungswissen in Computersystemen gespeichert werden soll, kann es sinnvoll sein, offensichtlich wertloses oder gar störendes Wissen auszusortieren. Will man sich diesen Aufwand ersparen, muss bei der Wiederverwendung des Wissens sichergestellt sein, dass das System die Benutzer nicht mit offensichtlich unnützem Wissen überschwemmt (vergleiche dazu das Anwendungsbeispiel **SimLex** in Kapitel 5). Den tatsächlichen Wert von Erfahrungswissen in einer bestimmten Situation kann man oft erst nach der Wiederverwendung des Wissens beurteilen.

Ich grenze meine Definition von Erfahrungswissen gegen die Gebrauchsweise des Begriffs bei [NT97] ab (siehe Kapitel 2.2). Nonaka und Takeuchi sprechen davon, dass Erfahrungswissen „meist implizit, körperlich und subjektiv“ist und stellen es Verstandeswissen gegenüber. Bei mir ist Erfahrungswissen meist explizierbar, das heißt man kann es auch in elektronischer Form speichern und weitergeben.

Erfahrungswissen bildet in meinem Verständnis einen Gegensatz zu allgemeinem, regelhaftem Wissen, das in allen Situationen gültig ist. Ein Beispiel für allgemeines Wissen ist das Wissen über die Schwerkraft. Allgemeines und Erfahrungswissen sind jedoch nicht klar voneinander getrennt. So kann die Schwerkraft in einer bestimmten Situation auch einmal aufgehoben sein. Diese Aufhebung eines Naturgesetzes ist dann Teil einer Erfahrung in einem speziellen Problemlösekontext. Für diese Arbeit lasse ich solche philosophischen Überlegungen außer Acht. Wie beim allgemeinen Wissensbegriff verwende ich eine praxisorientierte Definition von Erfahrungswissen. Wichtig ist nicht zu vergessen, dass Erfahrungswissen kontextbezogen ist und deshalb keinen Anspruch auf Allgemeingültigkeit erheben kann.

Definition 3.1.2 (Erfahrungsmanagement (Experience Management, EM nach [Ber02, S. 14]))

> *Erfahrungsmanagement (Experience Management, EM)* ist eine Spezialform von Wissensmanagement, die sich mit Erfahrungswissen befasst.

Definition 3.1.3 (EM-Aufgaben nach [Ber02, S. 14])

> EM beschäftigt sich mit dem Sammeln, Strukturieren, Speichern, Wiederverwenden, Evaluieren und Pflegen von Erfahrungswissen.

Neben Bergmanns Definitionen von EM und EM-Aufgaben gibt es eine lernorientierte Definition im Umfeld der **Experience Factory**, die mit EM „jede organisatorische und technische Unterstützung des Lernens aus Erfahrung“[ADH+01, S. 2, eigene Übersetzung] beschreibt. Die daraus abgeleiteten EM-Aufgaben sind sehr ähnlich zu denen in Bergmanns Definition (siehe oben). Wir haben Bergmanns Definitionen für EM und EM-Aufgaben übernommen, weil sie den Bezug zum WM herstellen und die vorliegende Arbeit sich ebenfalls an einem WM-Modell orientiert (vergleiche Kapitel 2.3 und 2.4). Bergmanns Definitionen werden um die folgenden eigenen Definitionen und Überlegungen ergänzt.

Definition 3.1.4 (EM-System, EM-Assistenzsystem)

> *EM-Systeme (EM-Assistenzsysteme)* sind Assistenzsysteme, die Menschen bei der Bearbeitung von EM-Aufgaben unterstützen.

Dieser Begriff bezeichnet also Computersysteme, die über Informationssysteme hinausgehen: Nach dem FRISCO-Ansatz [BHA+00] unterstützen Informationssysteme den Austausch von Informationen. Mein Begriff hingegen schließt nicht aus, dass ein Assistenzsystem als Akteur in einer hybriden Gemeinschaft auftritt und selbstständig Teilaufgaben des EM organisiert oder bearbeitet. Beispielsweise kann ein Assistenzagent eine Maintenance-Strategie organisieren (siehe Kapitel 9) oder selbst Wissen mit anderem Wissen kombinieren (siehe Kapitel 5 oder 7). Im Gegensatz zu meinem Begriff unterscheidet Markus Nick [Nic05] zwischen technischen „experience based information systems“und sozio-technischen EM-Systemen. Die nicht-technischen Aspekte werden in dieser Arbeit direkt in das Rahmenwerk der WM-Prozesse integriert (siehe Kapitel 2.4), das heißt ich trenne nicht zwischen technischem und sozio-technischem System.

Setzt man ein EM-System ein, kann das System Teile der ursprünglichen EM-Aufgaben übernehmen. Dafür kommen in Definition 3.1.6 zwei neue Aufgaben hinzu, nämlich die Integration des Systems und die Modellierung und Pflege des systeminternen Hintergrundwissens.

Definition 3.1.5 (Hintergrundwissen eines EM-Systems)

> Das *Hintergrundwissen eines EM-Systems* ist das systeminterne Wissen, das das System für die Bewältigung seiner Aufgaben benötigt.

Das Hintergrundwissen eines EM-Systems wird meist zum Wiederfinden und Wiederverwenden von Erfahrungswissen benötigt (vergleiche dazu den Wissenskern des EM-Modells auf S. 30 und ganz konkret im Fallbasierten Schließen für Texte das Index- und das Ähnlichkeitslexikon auf den S. 45 und 46). Das Spektrum der EM-Aufgaben verschiebt sich durch den Einsatz eines EM-Systems wie folgt:

Definition 3.1.6 (EM-Aufgaben unter Einsatz eines EM-Systems)

1. Sammeln, Niederschreiben und Pflegen von Erfahrungswissen,
2. Modellieren und Pflegen des systeminternen Hintergrundwissens,
3. Wiederverwenden von Erfahrungswissen,
4. Integrieren des EM-Systems (einschließlich psychosozialer und kultureller Aspekte),
5. Evaluieren der Wissensinhalte und des Systemeinsatzes.

Das Sammeln von Erfahrungswissen wird vom System unterstützt (siehe Kapitel 5). Die Strukturierung des Erfahrungswissens wird vom System stark erleichtert, so dass es sich eher um ein Niederschreiben durch einen Autor handelt (siehe auch Kapitel 5). Das Speichern wird vom EM-System realisiert. Das Pflegen rückt in die Nähe des Niederschreibens, da beide Aufgaben auf ähnliche Weise mit Hilfe eines EM-Systems organisiert werden können (siehe Kapitel 6 und 9).

Das Modellieren und Pflegen des systeminternen Hintergrundwissens ist eine neue Aufgabe, die für die Assistenzsysteme erledigt werden muss (siehe Kapitel 5). Sie kann teilweise automatisiert werden. Beim Erfahrungswissen und beim systeminternen Hintergrundwissen gehen das Arbeiten *für* und *mit* dem Assistenzsystem fließend ineinander über (vergleiche auch die Diskussion auf S. 20).

Das Wiederverwenden von Erfahrungswissen wird zwar auch technisch unterstützt, kann aber zumindest in den Anwendungsbeispielen dieser Arbeit (siehe Teil II der Arbeit) nicht vollautomatisiert werden.

Die Integration des EM-Systems kommt neu hinzu (siehe Kapitel 8). Dies beinhaltet neben technischen und organisatorischen auch psychosoziale und kulturelle Aspekte (siehe Kapitel 3.3).

Es genügt nun nicht mehr, die Wissensinhalte des Systems zu evaluieren. Der Erfolg des Systemeinsatzes zum Beispiel in Form von Zeitersparnis oder in Form von Inspiration zur Entwicklung neuen Wissens sollte ebenfalls bewertet werden, sofern dies möglich ist (vergleiche Kapitel 8).

Diese fünf EM-Aufgaben lassen sich folgenden Kernprozessen des WM zuordnen (vergleiche Kapitel 2.3 und 2.4):

- das Sammeln und Niederschreiben von Erfahrungswissen und das Modellieren von Hintergrundwissen dem Prozess des *Wissenserwerbs* und der *Wissensentwicklung* (siehe Kapitel 5 und 6), je nachdem, ob eher neu geschrieben und modelliert wird oder ob eher Wissen wieder aufgegriffen und weiterentwickelt wird,
- das Pflegen von Erfahrungswissen und Hintergrundwissen dem Prozess der *Wissensbewahrung* (siehe Kapitel 9),
- das Wiederverwenden von Erfahrungswissen dem Prozess der *Wissens-(ver)teilung* (siehe Kapitel 7),
- die Integration des EM-Systems dem Prozess der *Wissensnutzung* (siehe Kapitel 8) und
- das Evaluieren ebenfalls dem Prozess der *Wissensnutzung*.

Zur technischen Unterstützung dieser Aufgaben können verschiedenste Methoden eingesetzt werden. In Teil II meiner Arbeit sind dies fallbasierte Methoden, in Kapitel 10 beschreibe ich alternative Ansätze zum Fallbasierten Schließen.

3.2 Das EM-Modell nach Bergmann

Im diesem Kapitel werden die EM-Systeme in ein Prozessmodell von Ralph Bergmann [Ber02] eingebettet, das über rein technische Gesichtspunkte hinausgeht. Es besteht aus drei Teilen: einem Wissenskern, einem inneren und einem äußeren Kreislauf.

Der Wissenskern enthält das Erfahrungswissen und das Wissen bezüglich der Wiederverwendung des Erfahrungswissens sowie das Vokabular, um diese beiden Wissensarten zu beschreiben. Der Wissenskern lehnt sich an Richters Wissenscontainer an (vergleiche deren Definition auf S. 39 und in [Ric98]).

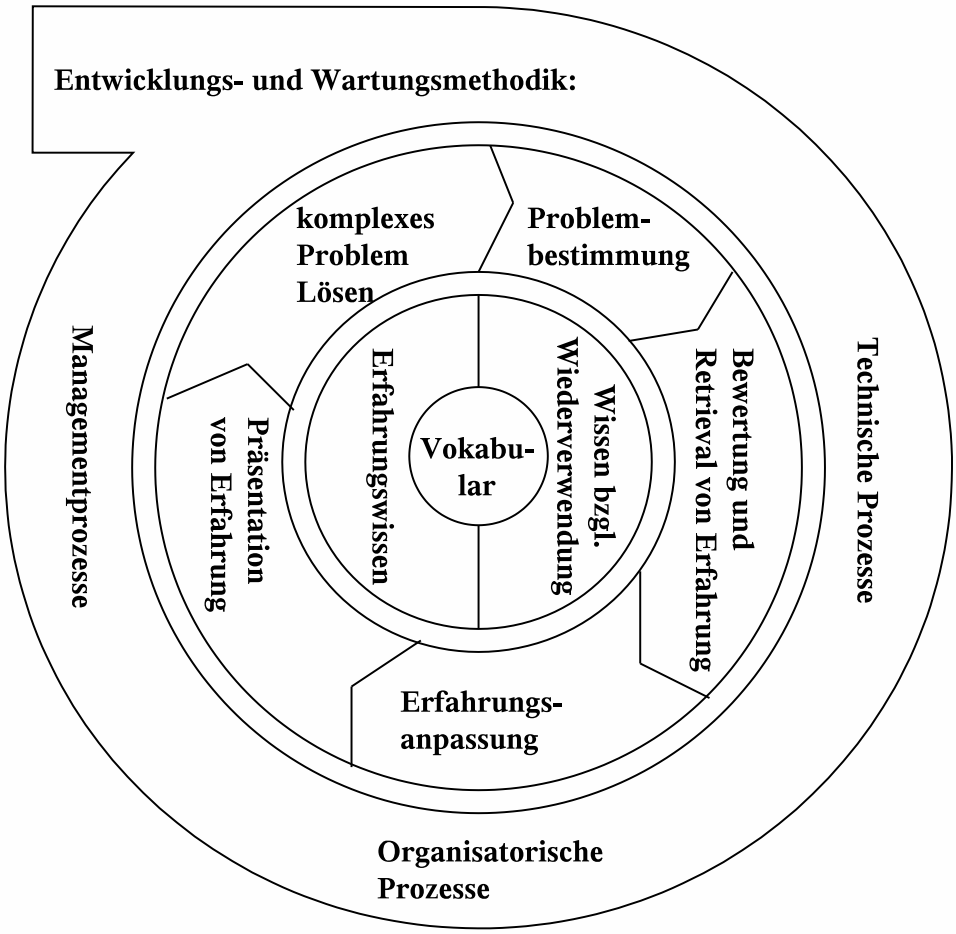


Abbildung 3.1: Prozessmodell für EM nach [Ber02, eigene Übersetzung].

Der innere Kreislauf bildet einen Problemlöse-Zyklus, der das Vorgehen beim Problemlösen mit Hilfe von wiederverwendetem Erfahrungswissen beschreibt. Bis auf das Problemlösen selbst können die Prozesse des inneren Kreislaufs durch Computersysteme automatisiert werden. Manche kleineren Probleme, wie zum Beispiel das Ausdrucken einer Datei, können auch jetzt schon von Agentenprogrammen gelöst werden.

Der äußere Kreislauf beschreibt die Entwicklungs- und Wartungsmethodik für den Wissenskern und den inneren Kreislauf. Diese Methodik verlässt die technische Ebene und wendet sich auch organisatorischen und Management-Prozessen zu. Damit ist sie grob vergleichbar mit dem Wissensmetaprozess bei Steffen Staab [Sta02], der die Entwicklung und Pflege einer Ontologie auch unter organisatorischen Gesichtspunkten beschreibt. Bergmanns Methodik geht jedoch mit ihrem Bezug zu Geschäftsprozessen (siehe unten) über Staabs Wissensmetaprozess hinaus.

Problemlöse-Zyklus des EM-Modells

Der Problemlöse-Zyklus im inneren Kreislauf des Modells unterstützt das Lösen komplexer Probleme durch Erfahrungswissen.

Bei der *Problembestimmung* wird der Verantwortliche dazu veranlasst, sein Problem zu äußern. Eventuell sollte auch der Kontext des Problems beschrieben werden. Dieser Prozess kann zum Beispiel durch Fragebögen, geführte Dialoge oder Auswahllisten unterstützt werden. Ergebnis der Problembestimmung ist eine Problembeschreibung, die unvollständig, ungenau oder sogar teilweise falsch oder inkonsistent sein kann.

Die Bewertung und das Retrieval von Erfahrung haben zum Ziel, relevante Erfahrungen aus den im System abgespeicherten Erfahrungsbeschreibungen herauszusuchen („retrieve“= etwas wiederfinden). Dieser Prozess stellt den Systementwicklern zwei Hauptaufgaben:

1. Es muss festgelegt werden, wann eine Erfahrung relevant für eine bestimmte Situation ist. Eventuell geht die Bewertung, wie nützlich eine im System beschriebene Erfahrung allgemein oder in Bezug auf spezielle Charakteristika des Problemkontexts ist, mit in die Bestimmung der Relevanz ein.

2. Eine effiziente Suchmethode muss implementiert werden.

Das Wissen, das zur Berechnung der Relevanz von Erfahrungen nötig ist, steckt im Wissenskern des Systems zu „Wissen bezüglich der Wiederverwendung“. Ergebnis des Retrievals ist eine Menge relevanter Erfahrungsbeschreibungen, die in den Anpassungsprozess weitergeleitet werden.

Die *Erfahrungsanpassung* überträgt gefundene Erfahrungsbeschreibungen auf die aktuelle Problemsituation. Dieser Prozess ist sehr wissensintensiv und bezieht bei automatischer Ausführung zum Beispiel Adaptionsregeln aus dem Wissenskern des Systems („Wissen bezüglich der Wiederverwendung“).

Die *Präsentation von Erfahrung* muss so gestaltet sein, dass der Verantwortliche sie verstehen und zum Lösen des Problems heranziehen kann. Der Problemlöse-Zyklus kann auch Backtracking beinhalten, etwa wenn der Anpassungsprozess gescheitert ist und alternative Erfahrungsbeschreibungen gesucht werden sollen.

Entwicklungs- und Wartungsmethodik des EM-Modells

Eine Entwicklungs- und Wartungsmethodik im äußeren Kreislauf des Modells soll den Entwicklern eines EM-Systems eine systematische Vorgehensweise erleichtern. Sie besteht im Wesentlichen aus Prozessmodellen und Checklisten.

Die technischen Prozesse der Methodik betreffen das Erzeugen oder Verändern von EM-Softwarekomponenten oder von darin repräsentiertem Wissen.

Die organisatorischen Prozesse der Methodik beinhalten Veränderungen an Geschäftsprozessen, in die das EM-System eingebettet wird, und neu zu beschreibende Geschäftsprozesse wie Benutzerschulung oder Maintenance-Aktivitäten.

Die Management-Prozesse laufen eine Stufe höher ab: Sie betreffen die Metaprozesse der Software-Entwicklung, z.B. die generelle Projektplanung, das Monitoring oder die Qualitätssicherung.

Bergmann nutzt die recht umfangreiche INRECA-Methodik [BBG+99] als Beispielframework für den industriellen Einsatz von EM-Systemen. Ich verzichte darauf, sie hier zu beschreiben, da sie für meine kommerziellen und nicht-kommerziellen Anwendungen von EM-Systemen (siehe Teil II der Arbeit) zu umfangreich und zeitaufwändig gewesen wäre.

3.3 Ganzheitliches EM

Ich übertrage aus dem Modell von Bergmann, das im vorigen Kapitel beschrieben wurde, für diese Arbeit vor allem die ganzheitliche Herangehensweise. „Ganzheitlich“im Zusammenhang mit Assistenzsystemen für das EM meint, dass es nicht ausreicht, sich auf die technische Seite von Assistenzsystemen für das Erfahrungsmanagement zu beschränken. Für die Entwicklung solcher Systeme ist es unverzichtbar, organisatorische, psychologische

und kulturelle Aspekte im Blick zu behalten. Dies geht über eine benutzerfreundliche Gestaltung der Bedienoberflächen wie in [DWar] hinaus. Für allgemeines WM wurde der Ansatz der Interdisziplinarität oder Ganzheitlichkeit schon durch das Drei-Säulen-Modell in Abbildung 2.2 veranschaulicht. Für EM-Systeme ist die ganzheitliche Herangehensweise noch wichtiger als für allgemeines Wissen. Im Folgenden wird dies anhand einer psychologischen Studie zu möglichen Ursachen für das Scheitern von WM-Systemen [MS05] konkret erläutert. Die wesentlichen Aspekte der Studie werden referiert und der Bezug zu EM-Systemen hergestellt.

Meyer und Scholl haben vier Gruppen von möglichen Ursachen für das Fehlschlagen von WM-Systemen identifiziert:

- organisatorische Barrieren,
- einseitiger Fokus der WM-Aktivitäten entweder auf Codifizierung oder auf Personalisierung von Wissen,
- fehlende oder ungeeignete Definition des Wissensbegriffs und
- dass man Wissen überhaupt nicht managen kann.

Die organisatorischen Barrieren können Zeitmangel, schlechte Motivation der Mitarbeiter, eine zu kleine finanzielle oder personelle Ausstattung von WM-Projekten oder einschränkende Paradigmen und Traditionen sein. Die meisten dieser Barrieren treten bei allgemeinem WM und EM gleichermaßen auf. Die einschränkende Tradition, schlechte Erfahrungen nicht publik zu machen, trifft besonders EM-Systeme, die auch negatives Erfahrungswissen zulassen. Ein Beispiel aus der Medizin sind die fehlenden Publikationen negativer Studienergebnisse, die dazu führen, dass Studien von mehreren Forschungsgruppen wiederholt werden, weil die schlechte Erfahrung nicht weitergegeben wird. Gibt es solch eine Konvention, Wissen zu verschweigen, kann das beste EM-System nichts ausrichten.

Der einseitige Fokus auf Codifizierung von Wissen, das heißt Wissen in elektronischer Form explizit zu machen, oder das Gegenteil davon, nämlich Personen zur Internalisierung von Wissen anzuregen, kann sowohl bei allgemeinen WM-Systemen als auch bei EM-Systemen gesetzt werden. Für ein erfolgreiches System ist eine Integration von Codifizierung und Personalisierung nötig.

Irreführende Wissensbegriffe betrachten Wissen als ein „Ding“oder gehen davon aus, dass Menschen und Computerprogramme Wissen in gleicher Weise verarbeiten. Sie können ein sozio-technisches Systemdesign verhindern oder die Wichtigkeit eines gemeinsamen Kontexts, in dem das Wissen angewendet

werden kann, herunterspielen. Besonders die falsche Annahme, dass Computer wie Menschen mit Wissen umgehen könnten, gerät in Konflikt mit dem Charakter von Erfahrungswissen, wie es in EM-Systemen vorkommt: Teile des Erfahrungswissens sind diffus, werden in natürlicher Sprache beschrieben und lassen sich nicht immer klar strukturieren. Das Erfahrungswissen ist hochgradig abhängig vom Problemlösekontext, das heißt es wäre ein grober Fehler, nur eine Art des Zugriffs, zum Beispiel über einen Verzeichnisbaum zu ermöglichen. Anders als bei Regelwissen können in verschiedenen Situationen dieselben Erfahrungen einmal gültig und einmal falsch sein.

Die Behauptung, dass Wissen gar nicht zu managen sei, lehne ich ab. Dass es oft schwer ist, Erfahrungswissen zu teilen, ist unbestreitbar. Das mag unter anderem auch damit zusammenhängen, dass der Erfahrungsschatz eines Menschen etwas Persönlicheres ist als eine allgemeine Regel.

Meyer und Scholl haben diese vier Ursachengruppen näher untersucht und geben infolgedessen drei Handlungsempfehlungen für den Entwurf von WM-Systemen:

1. Eine genaue Kenntnis, welche Unterstützung die Mitarbeiter überhaupt brauchen, sollte schon beim Entwurf des Systems vorliegen. Das heißt, eine frühe Partizipation zukünftiger Benutzer ist wünschenswert.

2. Der Mensch sollte im Mittelpunkt stehen. Das heißt, es ist hilfreich zu erfragen, welche Haltung die Mitarbeiter zum WM haben, ob sie vielleicht Ängste haben kontrolliert zu werden. Zum Erreichen positiver Einstellungen sollten die Ziele des WM-Systems und deren Verankerung in der Firmenleitung allen Mitarbeitern bekannt gemacht werden.

3. Organisatorische Barrieren sollten ausgeräumt werden. Dazu gehört ein ausreichendes Maß an Zeit und Schulungen, das Ausräumen gesetzlicher Schranken und das Motivieren der Mitarbeiter vor allem durch Bekanntmachen der Vorteile, die der Einsatz eines Systems bringt.

Die zweite Empfehlung ist bei EM-Systemen besonders wichtig, weil das Mitteilen von persönlichem Erfahrungswissen den Wissensgeber angreifbar macht. Vertrauen in das System und in die Wissensempfänger und -anwender ist deshalb bei EM-Systemen besonders wichtig. In Kapitel 8 schildere ich eigene Erfahrungen mit einer organisatorischen Barriere, nämlich der Angst der Benutzer vor Kontrolle. Im selben Kapitel wird eine Werbestrategie für EM-Systeme vorgestellt, die die organisatorische Barriere einer mangelnden Motivation der Benutzer ausräumen soll und im Anwendungsbeispiel auch erfolgreich ausgeräumt hat.

Kapitel 4

Grundlagen des Fallbasierten Schließens

In diesem Kapitel stelle ich die Grundlagen des Fallbasierten Schließens vor. In Kapitel 4.1 präsentiere ich Fallbasierte Systeme und in Kapitel 4.2 die Technik der Case Retrieval Netze von Mario Lenz und Hans-Dieter Burkhard [LB96]. Kapitel 4.3 enthält die von mir mitentwickelte Technik des Fallbasierten Schließens für Texte [LHK98b].

Fallbasiertes Schließen ist ein Teilgebiet der Künstlichen Intelligenz und bildet die menschliche Fähigkeit des Erinnerns nach. Die Grundlagen wurden Anfang der achtziger Jahre von Roger Schank [Sch82] und Janet Kolodner [Kol84] gelegt. Fallbasiertes Schließen basiert auf der kognitionspsychologischen Erkenntnis, dass Menschen auf ihren Erfahrungsschatz zurückgreifen, wenn sie ein Problem lösen wollen [Ros89]. Die Erinnerung an vergangene Situationen trägt - wie beim Arzt oder Juristen - oft zur Lösung eines aktuellen Problems bei.

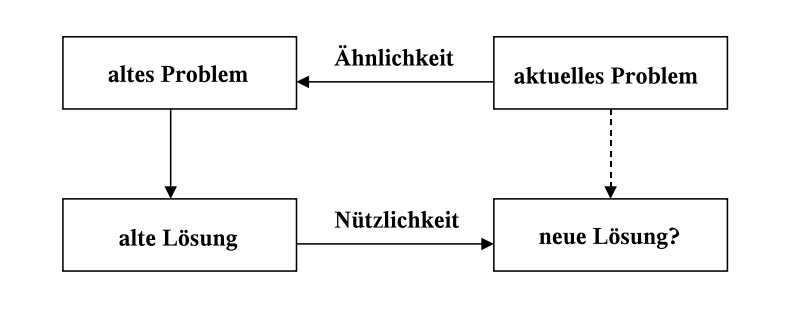


Abbildung 4.1: Grundschema des Fallbasierten Schließens.

Dieses Grundprinzip des Erinnerns wurde auf Computersysteme übertragen. Daraus ist das in Abbildung 4.1 beschriebene Grundschema des Fallbasierten Schließens entstanden. Zu einem aktuellen Problem wird ein ähnlicher Problemfall aus der Vergangenheit gesucht, der bereits gelöst ist. Die Ähnlichkeit zwischen den beiden Problemen lässt darauf hoffen, dass die Lösung des alten Problems nützlich für das neue Problem ist. Aus dem a priori feststellbaren Kriterium der Ähnlichkeit von Problemen kann man auf die angenommene Nützlichkeit einer Lösung schließen. Die tatsächliche Nützlichkeit lässt sich erst a posteriori überprüfen. Eine detaillierte Diskussion der Begriffe Ähnlichkeit und Nützlichkeit wird in [BR01] geführt.

Die folgende Definition des fallbasierten Schließens orientiert sich an [Weß95, S. 26]:

Definition 4.0.1 (Fallbasiertes Schließen, FBS, Case-Based Reasoning, CBR)

> *Fallbasiertes Schließen (FBS, Case-Based Reasoning, CBR)* ist ein Ansatz zur Wiederverwendung von spezifischem Problemlösungswissen im Kontext eines aktuell zu lösenden Problems.

Im Gegensatz zu allgemeinen Problemlöse-Ansätzen aus den frühen Tagen der KI (siehe zum Beispiel [NSS59]) wendet das Fallbasierte Schließen sich immer einer konkreten Domäne zu, zum Beispiel dem elektronischen Verkauf von Jeanshosen oder dem alltäglichen Umgang mit Computerproblemen professioneller Software-Entwickler. Das Fallbasierte Schließen setzt spezifisches Wissen über die jeweilige Domäne ein.

4.1 Fallbasierte Systeme

Ein *fallbasiertes System* verwaltet Problemlösungswissen in der Form von Fällen. Das System beantwortet Anfragen, indem es möglichst ähnliche Fälle in seiner Fallbasis sucht und diese dem Benutzer im Kontext der Anfrage anbietet.

Definition 4.1.1 (Fall, Fallbeispiel)

> Ein *Fall* oder *Fallbeispiel* ist die Beschreibung einer Problemsituation zusammen mit den Erfahrungen, die während der Bearbeitung des Problems gewonnen werden konnten.

Diese Erfahrungen müssen nicht prinzipiell eine Lösung enthalten. Es ist auch möglich, negative Erfahrungen wiederzuverwenden, um Fehler nicht zu wiederholen.

Definition 4.1.2 (Fallbasis)

> Eine *Fallbasis* ist eine geeignet organisierte Sammlung von Fallbeispielen.

Definition 4.1.3 (Anfrage, Query)

> Eine *Anfrage (Query)* ist die Beschreibung einer Problemsituation, die aktuell zu lösen ist.

Definition 4.1.4 (Fallbasiertes System, FBS-System, CBR-System)

> Ein *fallbasiertes System (FBS-System, CBR-System)* ist ein Computersystem, das den Ansatz des Fallbasierten Schließens implementiert.

Das Prozessmodell von Aamodt und Plaza [AP94] charakterisiert den fallbasierten Ansatz durch den Zyklus in Abbildung 4.1. Dieses klassische Modell für fallbasierte Systeme besteht aus vier Subprozessen.

Definition 4.1.5 (R4-Modell des Fallbasierten Schließens nach [AP94, eigene Übersetzung])

> Das *R4-Modell des Fallbasierten Schließens* besteht aus den Prozessen *Retrieve*, *Reuse*, *Revise* und *Retain*.
>
> | | |
> |---|---|
> | **Retrieve:** | Bereitstellen eines geeigneten Fallbeispiels. |
> | **Reuse:** | Wiederverwenden von gespeichertem Problemlösewissen. |
> | **Revise:** | Testen und Modifizieren der gefundenen Lösung. |
> | **Retain:** | Speichern der neu gewonnenen Erfahrungen. |

Der *Retrieval-Prozess* besteht aus einer Suche in der Fallbasis nach dem Fallbeispiel, das am ähnlichsten zur Anfrage ist. Im *Reuse-Prozess* wird die im Antwortfall gefundene Lösung an den neuen Kontext angepasst. Die adaptierte Lösung wird im *Revise-Prozess* auf die Anfrage angewendet und gegebenenfalls modifiziert. Die dabei gewonnene Erfahrung kann im *Retain-Prozess* in die Falldaten oder in das allgemeine Wissen, das im System gespeichert ist, einfließen. Zur Bestimmung der Ähnlichkeit eines Falls zu einer Anfrage und zur Adaption eines Falles ist zusätzlich zur Fallbasis allgemeines Wissen nötig.

Viele fallbasierte Systeme, vor allem die kommerziell genutzten, konzentrieren sich auf den Retrieve-Prozess und überlassen die anderen drei Prozesse weitestgehend den Benutzern. In dieser Arbeit ist dies prinzipiell genauso.

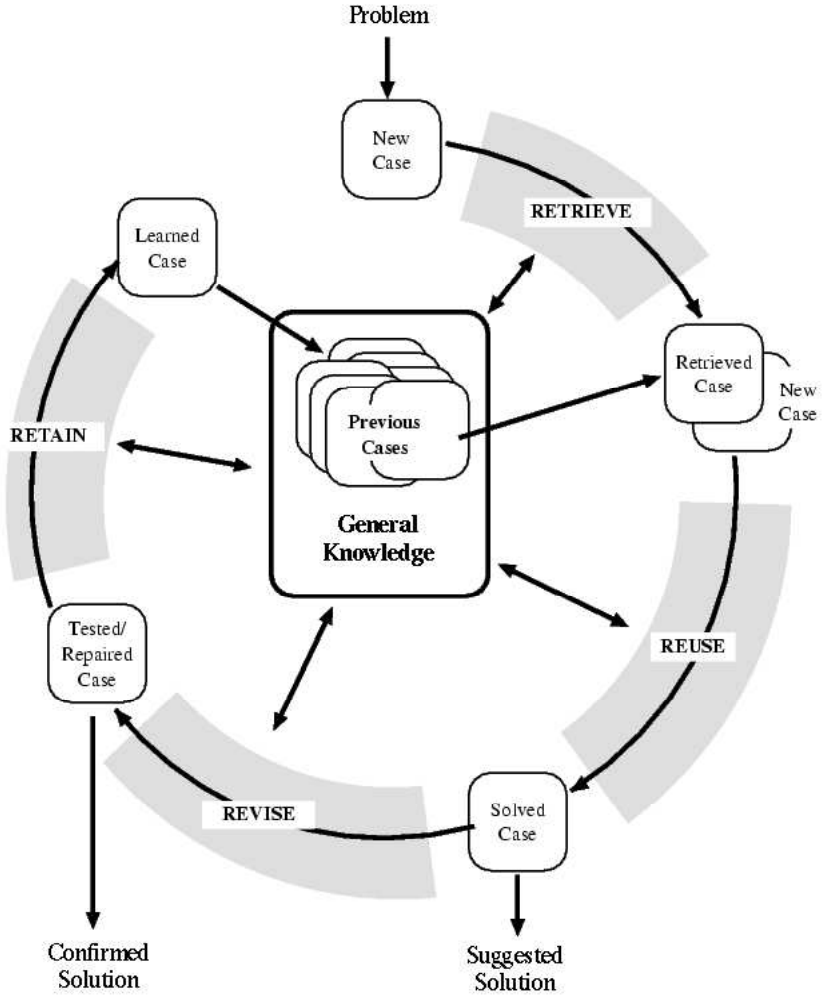


Abbildung 4.2: Prozessmodell des Fallbasierten Schließens [AP94].

Zusätzlich haben wir den Reuse-Prozess und Teilprozesse des Retain implementiert: Agenten automatisieren den Reuse-Prozess durch selbstständiges Kombinieren („Generative Reuse“) und Ausprobieren von Lösungen (siehe Kapitel 7). Der Retain-Prozess hat sich bei uns differenziert in Sammeln, Niederschreiben und Pflegen von Erfahrungswissen (vergleiche mit den EM-Aufgaben unter Einsatz eines EM-Systems in Kapitel 3.1, S. 28). Teilaufgaben davon werden auch in unseren Anwendungsbeispielen automatisiert (siehe Kapitel 5, 6 und 9).

Michael M. Richter [Ric98, Ric00] hat Wissenscontainer eines fallbasierten Systems beschrieben, mit deren Hilfe alle vier Prozesse implementiert werden können:

Definition 4.1.6 (Wissenscontainer eines fallbasierten Systems nach [Ric00])

> *Wissenscontainer* sind strukturelle Elemente, die Wissen enthalten, das für viele Aufgaben relevant ist. In fallbasierten Systemen lassen sich vier Container identifizieren:
>
> 1. Die Repräsentationssprache und das Vokabular
> 2. Das Ähnlichkeitsmaß mit allen Bestandteilen, zum Beispiel Wertetabellen
> 3. Die Fallbasis
> 4. Die Lösungstransformation

Die Fallbasis ist bereits definiert. Es müssen im Folgenden also noch weitere drei Wissenscontainer eingeführt werden. Auf S. 39 diskutiere ich eine Repräsentationssprache und ein Vokabular, auf S. 41 Ähnlichkeit und auf S. 42 die Lösungstransformation.

Repräsentation von Fällen

Ein fallbasiertes System braucht eine gute interne Notation für die Falldaten, um die Fälle im Retrieval mit Anfragen vergleichen zu können und sie gegebenenfalls beim Reuse und Revise zu modifizieren, wenn dies möglich ist. Bei manchen, gut strukturierten systemexternen Daten kann die innere und die äußere Darstellung übereinstimmen. Oft bilden die für die Suche relevanten Attribute eine Teilmenge des Falls. Bei Texten (vergleiche Kapitel 4.3), Bild- und Audiodaten hingegen ist es sinnvoll, die Daten in originaler Form für die Benutzer darzustellen; für das System müssen sie dann vorverarbeitet und auf interne Strukturen abgebildet werden.

In dieser Arbeit wird der Begriff „Fall“ für zwei verschiedene Repräsentationsformen verwendet: Er gilt sowohl für Beschreibungen, die unabhängig von dem konkreten fallbasierten System existieren, als auch für deren systeminterne Form innerhalb einer Fallbasis. Auch der Begriff „Anfrage“ bezeichnet eine Problembeschreibung, wie ein Benutzer des Systems sie formuliert, und deren systeminterne Form. Dieser Sprachgebrauch hat sich eingebürgert, so dass in dieser Arbeit nur aus dem Kontext ersichtlich wird, welche Repräsentationsform jeweils gemeint ist.

In der Literatur (z. B. in [Weß95]) wird die Anfrage oft als spezielle Art von Fall interpretiert, dem nur noch die Lösung fehlt. Oft ist es aber schwer, Problemteil und Lösungsteil eines Falls klar zu unterscheiden. Im E-Commerce ist zum Beispiel die Beschreibung des gesuchten Produkts schon ein Teil der Problemlösung. Ein Fall kann auch selbst die Funktion einer Anfrage übernehmen (vergleiche Kapitel 5). Wir verwenden deshalb die Begriffe „Fall“ und „Anfrage“ je nach Funktion, nicht aufgrund des Inhalts oder der Struktur.

Für die systeminterne Repräsentation von Fällen und Anfragen verwende ich das Konzept der Informationseinheiten:

Definition 4.1.7 (Informationseinheit, IE)

> Eine *Informationseinheit* (ein *IE*[1]) ist ein atomares Element eines Falls oder einer Anfrage. E ist die Menge aller (möglichen) IEs in einer Domäne. Jedes IE wird durch einen IE-Namen eindeutig bezeichnet. Damit gilt im Innern eines fallbasierten Systems:
>
> > Ein Fall ist eine Menge von IEs: $c \subseteq E$.
> > Die Menge aller Fälle (in der Fallbasis) wird mit C bezeichnet, wobei gilt: $C \subseteq \mathcal{P}(E)$.
> > Eine Anfrage ist eine Menge von IEs: $q \subseteq E$.
> > Die Menge aller Anfragen wird mit Q bezeichnet, wobei gilt: $Q \subseteq \mathcal{P}(E)$.

Ein Beispiel für ein IE in einer Anwendung zur technischen Diagnose bei der Produktion von Autokarosserien (siehe [MCG+04]) ist die Abweichung des Messwerts für die obere Länge der rechten Vordertür vom gewünschten Wert. Die konkrete Repräsentationssprache und das Vokabular, das in den Anwendungsbeispielen dieser Arbeit verwendet wird, ist in Kapitel 4.3 beschrieben.

[1] Die Abkürzung wird im deutschen Sprachgebrauch in der Neutrum-Form verwendet.

Ähnlichkeit

Mit Hilfe der oben definierten Repräsentation von Fällen und Anfragen durch Mengen von IEs kann das System numerische Werte für das Kriterium „Ähnlichkeit“ ermitteln:

Definition 4.1.8 (Ähnlichkeitsfunktion, globale Ähnlichkeitsfunktion)

> Eine *(globale) Ähnlichkeitsfunktion* $SIM : Q \times C \rightarrow \mathbb{R}$ berechnet den Grad der Ähnlichkeit eines Fallbeispiels c zu einer Anfrage q. Wir interpretieren
>
> $SIM(q, c_i) > SIM(q, c_j)$ als „c_i ist ähnlicher zu q als c_j“.

Definition 4.1.9 (Lokale Ähnlichkeitsfunktion)

> Eine *lokale Ähnlichkeitsfunktion* sim bezeichnet den Grad der Ähnlichkeit eines IEs zu einem anderen IE:
>
> $sim : E \times E \rightarrow \mathbb{R}$.

In dieser Arbeit wird die globale Ähnlichkeitsfunktion auf der Basis der an der Anfrage und dem jeweiligen Fall beteiligten IEs gebildet. Ein einfaches Beispiel für eine solche kompositorische Ähnlichkeitsfunktion ist die ungewichtete Summe der lokalen Ähnlichkeitswerte:

$$SIM(q, c) = \sum_{e_i \epsilon q} \sum_{e_j \epsilon c} sim(e_i, e_j).$$

Im allgemeinen sind auch negative Werte für sim zulässig, die dann den Ähnlichkeitswert eines Falls verkleinern können. Zum Beispiel könnte das IE für „Windows“ sich negativ auf Fälle auswirken, die das IE für „Linux“ enthalten. In dieser Arbeit nehmen die Funktionen sim und damit SIM aber nur positive Werte an. Die 0 bezeichnet die Unähnlichkeit, die 1 die totale Übereinstimmung. Dies ist zum Beispiel die Ähnlichkeit eines IEs zu sich selbst. Der Wertebereich von sim ist das Intervall $[0, 1]$.

Der Wertebereich von SIM wird sehr einfach zwischen 0 und 1 gehalten, indem das Ergebnis der Summe normiert wird (vergleiche S. 51). In Kapitel 4.3 gebe ich ein Beispiel an, wie mit Hilfe eines Lexikons eine konkrete Instanz der lokalen Ähnlichkeitsfunktion erzeugt werden kann.

Diffizilere Beispiele für die globale Ähnlichkeitsfunktion stelle ich in den einzelnen Anwendungskapiteln in Teil II der Arbeit vor (vergleiche auch [BR01]).

Lösungstransformation

Wenn passende Fälle zur Anfrage gefunden wurden, kann man selten eine Lösung unverändert auf das in der Anfrage beschriebene Problem anwenden, sondern muss eine oder mehrere der vorgeschlagenen Lösungen anpassen. Viele CBR-Systeme überlassen diese Lösungstransformation den Benutzern. Wir tun dies in den Anwendungsbeispielen dieser Arbeit auch, weil es sich hier um Falldaten mit Texten handelt und eine automatische Anpassung von Texten sehr schwierig ist (vgl. Kapitel 4.3). Deshalb fällt in dieser Arbeit der Wissenscontainer für die Lösungstransformation weg.

Für Falldaten, die nur aus Attribut-Werte-Paaren bestehen, kann der Wissenscontainer für die Lösungstransformation Adaptionsregeln enthalten. Zum Beispiel kann eine Adaptionsregel für die Konfiguration von PCs besagen, dass für ein PC-Angebot ein 17" Röhrenbildschirm durch einen 15" Flachbildschirm ersetzt werden darf.

4.2 Case Retrieval Nets

Eine an der Humboldt-Universität entwickelte Technik zur Strukturierung des Fallspeichers für den Retrieval-Prozess sind die *Case Retrieval Nets* [LB96]:

Definition 4.2.1 (Case Retrieval Net, CRN)

> Ein *Case Retrieval Net (CRN)* N ist definiert als kanten- und knotengewichtetes Netz $N = [E, C, \sigma, \rho, \Pi]$ mit
>
> - E als endlicher Knotenmenge von Informationseinheiten (IE-Knoten),
> - C als endlicher Knotenmenge von Falldeskriptoren,
> - $\sigma : E \times E \rightarrow \mathbb{R}$ als Gewichtsfunktion für die Kanten zwischen den IEs (die Ähnlichkeitskanten),
> - $\rho : E \times C \rightarrow \mathbb{R}$ als Gewichtsfunktion für die Kanten von den IEs zu den Falldeskriptoren (die Relevanzkanten) und
> - Π als endlicher Menge von Propagierungsfunktionen π_n für jeden Knoten $n \in E \cup C$ mit $\pi_n : \mathbb{R}^E \rightarrow \mathbb{R}$.

Fälle werden als Teilgraphen eines Netzes dargestellt, das gewichtete Knoten $E \cup C$ und mit σ oder ρ gewichtete Kanten enthält. C sind die Knoten zur Identifikation von Fällen, E die Knoten zur Repräsentation der IE-Menge E aus Definition 4.1.7.

ρ gibt die Relevanz je eines IEs für einen Fall an. In dieser Arbeit ist der Wertebereich von ρ auf $\{0, 1\}$ eingeschränkt. Dies entspricht der charakteristischen Funktion für die Mengendarstellung der Fälle (siehe Definition 4.1.7). $\sigma(e_i, e_j)$ übernimmt den Wert der lokalen Ähnlichkeitsfunktion *sim* (siehe S. 41) für das Paar von IEs, das im Netz durch die Knoten e_i und e_j repräsentiert ist.

Das Retrieval wird über einen *spreading activation*-Mechanismus ausgeführt, der die Propagierungsfunktionen Π implementiert: Die Anfrage aktiviert IE-Knoten, die ihre Aktivierung dann über die Ähnlichkeits- und Relevanzkanten verbreiten. So entsteht eine Halbordnung auf den Falldeskriptoren, die durch die Ähnlichkeit jedes Falls zur Anfrage induziert wird. Die Aktivierungsfunktion α_2 auf S. 49 beschreibt ein Beispiel für diesen Mechanismus.

CRNs eignen sich besonders gut dazu, große Mengen von Falldaten zu verwalten. Die Fälle müssen im Retrieval-Prozess nicht sequentiell durchsucht werden, sondern können in der Netzstruktur effizient geordnet werden (vgl. [Len99]). Durch Hinzufügen neuer Knoten und Kanten ist das CRN leicht erweiterbar.

4.3 Fallbasiertes Schließen für Texte

Fallbasiertes Schließen für Texte (Textual CBR, TCBR) arbeitet mit Falldaten, die Texte enthalten. Ich betrachte in dieser Arbeit leicht strukturierte Falldaten. Unstrukturierte Texte sind ein Spezialfall dieser Fallstruktur:

Definition 4.3.1 (Fallstruktur im TCBR)

> Ein Fall im Fallbasierten Schließen für Texte besteht aus der Sicht des Benutzers aus
>
> - einem Fall-Bezeichner,
> - einem oder mehreren Textabschnitten und
> - einem oder mehreren Abschnitten mit Attribut-Werte-Paaren.
>
> Aus der Sicht des Systems besteht ein Fall aus einer Menge von IEs.

Diese Struktur ist eine Konkretisierung der allgemeinen Definition eines Falls auf S. 36. Ein Beispielfall findet sich in einem der Anwendungskapitel auf S. 62. Oft werden die Fälle für TCBR-Systeme aus bereits bestehenden elektronischen Quellen automatisch generiert (siehe Kapitel 5). Zum Aufbau

einer Fallbasis werden die Fälle auf Mengen von IEs abgebildet, das heißt von einer Repräsentationsart in eine andere überführt (vergleiche die Definition eines Falls innerhalb eines fallbasierten Systems auf S. 40).

Im Folgenden stelle ich ein wortbasiertes Verfahren zur Indizierung und zum Vergleich von Texten vor, das Mario Lenz mit André Hübner und mir noch während meines Studiums Ende der neunziger Jahre an der Humboldt-Universität entwickelt hat [LHK98b]. Im Anschluss daran diskutiere ich kurz einige Alternativen aus der Literatur.

Repräsentation von Texten durch Text-IEs

Die Repräsentationssprache für Texte wird durch Text-IEs gebildet:

Definition 4.3.2 (Text-Informationseinheit, Text-IE)

> Eine *Text-Informationseinheit* (ein *Text-IE*) ist ein IE, das eine Menge von Wörtern und Wortfolgen repräsentiert, die für ein Konzept stehen, zum Beispiel grammatikalische Formen, synonyme Formulierungen, Übersetzungen, Abkürzungen oder verschiedene Schreibweisen eines Wortes.

Die Schreibweisen schließen auch häufige Tippfehler ein. Ein Beispiel für ein Text-IE zum Begriff „Anwender“ ist `__ANWENDER__ = {Anwender, Anwenders, Anwendern, Anw, Anwedner, Benutzer, Benutzers, Benutzern, User}`. `__ANWENDER__` ist der IE-Name.

Definition 4.3.3 (Indexvokabular)

> Das Indexvokabular eines fallbasierten Systems für Texte ist die Menge der Wörter und Wortfolgen, die in Text-IEs vorkommen.

Die Text-IEs sind in verschiedene Wortschatz-Kategorien eingeteilt:

Definition 4.3.4 (Wortschatz-Kategorie)

> Eine *Wortschatz-Kategorie* ist eine Teilmenge der IE-Menge.

Eine Wortschatz-Kategorie drückt die semantische Zugehörigkeit der IEs zu einer bestimmten Begriffswelt aus. Beispiele sind der allgemeinsprachliche Grundwortschatz oder Fachbegriffe verschiedener Domänen, zum Beipiel der allgemeinen IT-Welt oder einer bestimmten Betriebssystem-Linie.

Definition 4.3.5 (Indexlexikon)

> Ein Indexlexikon enthält die Text-IEs und die Zuordnung genau einer Wortschatz-Kategorie zu jedem Text-IE. Die Zugehörigkeit des Indexvokabulars zu den Text-IEs ist eindeutig.

Denkbar sind auch multiple Zugehörigkeiten von Indexvokabular zu Text-IEs oder von Text-IEs zu Wortschatz-Kategorien. Mit eindeutigen Zuordnungen lassen sich eine Reihe von Entscheidungen vermeiden, sobald IEs unterschiedlich behandelt werden. Zum Beispiel können Text-IEs einer spezielleren Kategorie mit einem höheren Gewicht in Retrieval-Prozesse eingehen als allgemeinere Text-IEs (vergleiche Kapitel 4.3). Ein Abschnitt aus einem Beispiel-Indexlexikon ist auf S. 73 zu sehen.

Ermitteln der IE-Mengen für die Fälle

Zum Aufbau der Fallbasis werden alle für den Retrieval-Prozess relevanten Text-Komponenten der Fälle mit Hilfe des Indexlexikons auf Text-IEs abgebildet: Der Indizierungs-Algorithmus liest jeden Text wortweise ein und sucht jeweils nach der längsten Folge von Wörtern, die in einem Eintrag des Indexlexikons vorkommt. Das zugehörige Text-IE wird in die Menge der IEs aufgenommen, die den untersuchten Text repräsentieren. Die Häufigkeit des Vorkommens wird auf S. 4.3 diskutiert.

Die Gesamtmenge der IEs besteht aus der Menge aller Text-IEs vereinigt mit der Menge der bekannten Attribut-Werte-Paare:

Definition 4.3.6 (Attribut-IE)

> Ein Attribut-IE ist ein Attribut-Werte-Paar.

Die Attribut-IEs bilden eine spezielle Wortschatz-Kategorie. Die für den Retrieval-Prozess relevanten Attribute werden in Attribut-IEs erfasst. Nicht belegte Attribute werden in der Mengendarstellung eines Falls ignoriert, mehrere Werte desselben Attributs ergeben mehrere Attribut-IEs. Es gibt auch Attribute, die nicht in das Retrieval eingehen sollen, sondern nur zur Information der Nutzer gedacht sind. Aus ihnen werden keine Attribut-IEs gebildet. Aus der Sicht der Fallbasis besteht der Fall also dann aus einem Fallbezeichner und einer Menge von Text- und Attribut-IEs (siehe Beispiel auf S. 62).

Lokale Ähnlichkeit zwischen Text-IEs

Die Werte der lokalen Ähnlichkeitsfunktion für je zwei IEs sind wie die IEs selbst in einem Lexikon organisiert. Viele gleiche oder ähnliche IEs in einem Fall und einer Anfrage sind Indikatoren für die Nützlichkeit dieses Falls

zur Lösung der Aufgabe, die hinter der Anfrage steckt. Die lokalen Ähnlichkeitswerte sind deshalb stark aufgabenbezogen und werden auch immer im Hinblick auf den Aufgabenbereich des fallbasierten Systems modelliert. Sprachliche Ähnlichkeiten wie in einem Thesaurus und domänenspezifische Ähnlichkeiten zum Beispiel zwischen Produktfamilien können gut verwendet werden, um Ähnlichkeitswerte für IEs abzuleiten.

Der Großteil der Ähnlichkeitsbeziehungen ist automatisch aus elektronischen Thesauri wie WordNet [Wor05] und aus anderen Quellen generiert (siehe Kapitel 5.4). Diese Basismenge wird dann gegebenenfalls manuell nachbearbeitet, zum Beispiel gefiltert, und mit semantischem Wissen über die Domäne ergänzt. Um die Modellierung zu vereinfachen, wird von den absoluten Zahlwerten der Ähnlichkeitsfunktion abstrahiert auf Ähnlichkeitstypen:

Definition 4.3.7 (Ähnlichkeitstyp)

> Ein *Ähnlichkeitstyp* S ist eine binäre Relation zwischen IEs, das heißt $S \subseteq E \times E$. Eine Menge $\mathfrak{S}$ von Ähnlichkeitstypen nennen wir Ähnlichkeitslexikon.

Jeder Ähnlichkeitstyp hat einen quantitativen oder qualitativen Bezeichner. Quantitative Bezeichner sind zum Beispiel „WEAK_SIM“ für „leichte Ähnlichkeit“ oder „STRONG_SIM“ für „starke Ähnlichkeit“. Ein Beispiel für einen qualitativen Ähnlichkeitstyp ist „HAS_PART“. Eine Verbindung der qualitativen Bezeichner zu ontologiebasierten Systemen liegt nahe und wird in Kapitel 5.4 beschrieben. Die Elemente des Ähnlichkeitslexikons haben die Form $e'[S_i]; e''$, also zum Beispiel (für ein längeres Beispiel siehe S. 77):

$$\_\_NETSCAPE\_\_[ABSTRACTION]; \_\_NETSCAPE4\_\_$$

Aus den Ähnlichkeitstypen eines Paares von IEs e' und e'' können Werte für die lokale Ähnlichkeitsfunktion $sim(e', e'')$ abgeleitet werden. Dazu wird jedem Ähnlichkeitstyp ein Wert zugewiesen:

Definition 4.3.8 (Gewichtungsfunktion für Ähnlichkeitstypen)

> Eine *Gewichtungsfunktion für Ähnlichkeitstypen* ist eine Funktion $g : \mathfrak{S} \to \mathbb{R}$.

Wir betrachten im Folgenden eine beliebige, aber fest gewählte Gewichtungsfunktion g. Mit Hilfe von g kann man den Grad der Ähnlichkeit für ein Paar von IEs berechnen, der aus der Zugehörigkeit des Paars zu S resultiert:

Definition 4.3.9 (Partielle Ähnlichkeitsfunktion)

> Die zur Gewichtungsfunktion g und zum Ähnlichkeitstyp S gehörende *partielle Ähnlichkeitsfunktion* $sim_{g,S} : E \times E \rightarrow \mathbb{R} \cup \perp$ ist folgendermaßen definiert:

$$sim_{g,S}(e', e'') = \begin{cases} g(S) & , \quad falls\ e'Se'' \\ \perp & , \quad sonst. \end{cases}$$

Für Paare von IEs, die zu verschiedenen Ähnlichkeitstypen gehören, muss ein eindeutiger numerischer Wert für *sim* bestimmt werden. Dazu definiere ich eine aggregierende Funktion, die die Werte von n verschiedenen partiellen Ähnlichkeitsfunktionen zusammenfasst:

Definition 4.3.10 (Aggregierende Funktion zur Bestimmung von *sim*)

> Eine Funktion $aggr : (\mathbb{R} \cup \perp)^n \rightarrow \mathbb{R}$ aggregiert n Werte zu einem einzigen Wert.
>
> Wird die aggregierende Funktion auf ein Ähnlichkeitslexikon angewendet, ist n die Anzahl der Ähnlichkeitstypen im Lexikon und wir können *sim* bestimmen als:
>
> $sim(e', e'') = aggr(sim_{S_1}(e', e''), ..., sim_{S_n}(e', e''))$.

Für *aggr* kommen zum Beispiel das Maximum, die Summe oder das Produkt aller partiellen Werte ungleich $\perp$ in Frage. Im Spezialfall, dass alle Argumente $\perp$ sind, kann man 0 als Defaultwert annehmen. In den Anwendungsbeispielen dieser Arbeit ist eine Ordnungsrelation über $\mathfrak{S}$ definiert, mit deren Hilfe aus allen partiellen Werten derjenige mit der höchsten Priorität für S_i ausgewählt wird (siehe S. 104). g und damit alle sim_{S_i} liegen in dieser Arbeit im Intervall $[0, 1]$, für die Identität I gilt $g(I) = 1$. Ein besonderer Ähnlichkeitstyp „NO_SIM“, der die explizite Nicht-Ähnlichkeit zweier IEs durch $g(NO\_SIM) = 0$ repräsentiert, hat die höchste Priorität, so dass sim_{NO_SIM} alle anderen sim_{S_i} mit einer 0 „überschreibt“.

Retrieval von Falldaten mit Texten

An eine Fallbasis, die Fälle mit Texten enthält, können Anfragen in Textform gestellt werden. In manchen Anwendungsbeispielen in Teil II sind die Anfragetexte durch eine Auswahl von Attributwerten ergänzt. Der Retrieval-Prozess läuft mit Hilfe des CRNs nach folgendem Schema ab:

- Die Anfrage wird auf eine Menge von IEs abgebildet.
- Die IE-Knoten im CRN werden initial aktiviert.
- Die initiale Aktivierung wird entlang der Ähnlichkeitskanten propagiert.
- Die Aktivierung wird entlang der Relevanzkanten zu den Fallknoten weiterpropagiert.
- Die Fallknoten mit den höchsten (gegebenenfalls normierten) Aktivierungswerten werden ermittelt. Dabei werden die zugehörigen Fälle nach absteigender Aktivierung in die Ergebnisliste einsortiert.

Die Anfrage wird nach derselben Methode auf IEs abgebildet, mit der beim Aufbau des Systems die Falldaten indiziert werden (siehe S. 45). Die Propagierungsfunktionen Π des Netzes sind in dieser Arbeit nicht für jeden Knoten individuell verschieden, sondern es ist eine Funktion für IE-Knoten und eine zweite Funktion für Falldeskriptoren definiert. Sie sind in einer dreistufigen Aktivierungsfunktion für alle Knoten des Netzes zusammengefasst, die aus einem initialen Aktivierungsschritt α_0 und zwei Propagierungsschritten α_1 und α_2 besteht (siehe unten und meine Diplomarbeit [Kun98]).

$w_{cat} : E \rightarrow [0, 1]$ gibt die Höhe der potentiellen initialen Aktivierung für α_0 an. Der Index *cat* steht für die Wortschatz-Kategorie des zu aktivierenden IEs, weil das initiale Gewicht dafür anhand dieser Kategorie definiert ist. So kann zum Beispiel allen Produktnamen aus dem Gegenstandsbereich des EM-Systems ein höheres Gewicht zugewiesen werden als der Klasse allgemeinsprachlicher Wörter. Mehrfaches Vorkommen eines IEs im Text wird ignoriert. Dies weicht bewußt vom klassischen Information Retrieval [Rij79] ab, wo die Häufigkeit für die Gewichtung eines Terms eine große Rolle spielt. Bei den oft ins Unreine geschriebenen Texten aus unseren Anwendungsbeispielen kommen sehr viele Wortwiederholungen vor. Diese Texte sollen nicht durch mehrfaches Vorkommen einzelner IEs gegenüber prägnant formulierten Texten bevorzugt werden.

Definition 4.3.11 (Dreistufige Aktivierungsfunktion)

Sei q eine Anfrage. Die drei Stufen $\alpha_0, \alpha_1, \alpha_2$ der Aktivierungsfunktion mit $\alpha_i : E \cup C \rightarrow \mathbb{R}$ beschreiben den „spreading activation"-Mechanismus im CRN wie folgt:

$$\alpha_0(n) = \begin{cases} w_{cat}(n) & , \quad n \,\epsilon\, q \\ 0 & , \quad sonst. \end{cases}$$

(initiale Aktivierung)

$$\alpha_1(n) = \begin{cases} \max_{e \epsilon E} (\sigma(e,n) \cdot \alpha_0(e)) & , \quad n \,\epsilon\, E \\ 0 & , \quad n \,\epsilon\, C. \end{cases}$$

(Propagierung entlang der Ähnlichkeitskanten)

$$\alpha_2(n) = \begin{cases} \alpha_1(n) & , \quad n \,\epsilon\, E \\ \sum_{e \epsilon E} \rho(e,n) \cdot \alpha_1(e) & , \quad n \,\epsilon\, C \end{cases}$$

(Propagierung entlang der Relevanzkanten).

Eigenschaft

Setzt man w_{cat} auf 1 und σ auf sim, nimmt man für ρ die charakteristische Funktion (Wertebereich $\{0,1\}$) und ersetzt man das Maximum in α_1 durch eine Summe über alle IEs, dann nimmt α_2 für einen Falldeskriptor $cn \,\epsilon\, C$ bei einer Anfrage q gerade den Wert der einfachen kompositorischen Ähnlichkeitsfunktion von S. 41 für den durch cn repräsentierten Fall c an:

$$\alpha_2(cn) = SIM(q,c) = \sum_{e_i \epsilon q} \sum_{e_j \epsilon c} sim(e_i, e_j).$$

Beweis:

$$\begin{aligned}
\alpha_2(cn) &= \sum_{e_j \epsilon E}(\rho(e_j, cn) \cdot \alpha_1(e_j)) \\
&= \sum_{e_j \epsilon c} \alpha_1(e_j) \\
&= \sum_{e_j \epsilon c} \sum_{e_i \epsilon E}(\sigma(e_i, e_j) \cdot \alpha_0(e_i)) \\
&= \sum_{e_j \epsilon c} \sum_{e_i \epsilon q} \sigma(e_i, e_j) \\
&= SIM(q, c).
\end{aligned}$$

□

Fallknoten mit hoher Aktivierung zeigen an, dass ihr Fall ein hohes Maß an Ähnlichkeit zur Anfrage besitzt. Weniger ähnliche Fälle haben nach dem Retrieval-Prozess eine niedrigere oder gar keine Aktivierung ihrer Fallknoten. In den Anwendungsbeispielen dieser Arbeit wurde die Maximumfunktion für α_1 gewählt: Bei einer Summe kann ein IE, das eingehende Kanten von vielen Nachbar-IEs hat, eine sehr hohe Aktivierung bekommen, wenn viele dieser Nachbarn in der Anfrage vorkommen. In den Anwendungsbeispielen haben die Benutzer die Anfragen eher knapp formuliert. Empirisch macht es deshalb keinen großen Unterschied, ob man die Summe oder das Maximum nimmt. Bei Anfragen in weniger technischem Stil - etwa in einer geisteswissenschaftlichen Domäne - könnten jedoch nennenswerte Schmutzeffekte durch paraphrasierende Formulierungen entstehen. Dann wäre es auch geboten, eine Lösung für Falltexte zu suchen, die sehr viele ähnliche IEs enthalten und deshalb durch ein einziges IE in einer Anfrage unverhältnismäßig viel Aktivierung bekommen können.

Um zu erreichen, dass ein Fall zu sich selbst immer die Ähnlichkeit 1 hat, kann man die obige Ähnlichkeitsformel normieren. Es ist möglich, dazu (4.1) über die Länge (Anzahl der IEs) der Anfrage q, (4.2) über die Länge des Antwortfalles c oder (4.3) über beide zu normieren.

Drei Normierungsvarianten

$$SIM_{norm}(q,c) = min\{\frac{SIM(q,c)}{\sum_{e_i \epsilon q} w_{cat}(e_i)}, 1\} \tag{4.1}$$

oder

$$SIM_{norm}(q,c) = min\{\frac{SIM(q,c)}{\sum_{e_i \epsilon c} w_{cat}(e_i)}, 1\} \tag{4.2}$$

oder

$$SIM_{norm}(q,c) = min\{\frac{SIM(q,c)}{\sum_{e_i \epsilon q} w_{cat}(e_i)} \cdot \frac{SIM(q,c)}{\sum_{e_i \epsilon c} w_{cat}(e_i)}, 1\} \tag{4.3}$$

Eigenschaft

$SIM_{norm}(c,c) = 1.$

Plausibilitätskontrolle:

Es gilt $SIM(c,c) \geq \sum_{e_i \epsilon q} w_{cat}(e_i)$, da jedes IE des Falls ja ein IE der Anfrage ist und schon initial mit w_{cat} aktiviert wird. Aus demselben Grund gilt auch $SIM(c,c) \geq \sum_{e_i \epsilon c} w_{cat}(e_i)$.

Damit ergeben die Brüche in allen drei Varianten Werte, die größer oder gleich 1 sind.

Ersetzt man in SIM die Summe $\sum_{e_i \epsilon q}$ durch das Maximum $\max_{e_i \epsilon q}$, gilt die Normierungseigenschaft, wie man leicht sieht, nach derselben Argumentation. Die Normierung über Anfrage und Fall (4.3) eignet sich gut, wenn die Anfrage aus einer präzisen Beschreibung des Problems besteht und die Fälle der Fallbasis eine relativ homogene Größe haben. Hat man eher kurze, stichwortartige Anfragen, werden von dieser Formel kleine Fälle bevorzugt. Diese enthalten weniger inaktive IEs, die das Ergebnis verschlechtern. Ausserdem werden manche Fälle im Laufe der Zeit immer wieder aufgerollt und wachsen dadurch in die Länge (siehe Kapitel 6). Sie haben dann immer schlechtere Chancen, bei der gleichen Anfrage gefunden zu werden. Hier ist jeweils eine Normierung über die Länge der Anfrage sinnvoll.

Beispiel:

Sei $SIM(q,c) = \sum_{e_j \epsilon c} \max_{e_i \epsilon q} sim_(e_i, e_j)$. Sei eine Anfrage mit den IEs *{spooler, paper}* und zwei Fällen der Fallbasis gegeben, wobei Fall 1 = *{spooler, queue}* und Fall 2 ausführlich beschrieben sei durch die IEs *{spooler, paper,* ie_1*,* ie_2*,* ie_3*,* ie_4*,* ie_5*,* ie_6*,* ie_7*}*. Seien weiterhin die IEs der Anfrage nicht ähnlich zu ie_1 bis ie_7.
Formel (4.1) bevorzugt Fall 2, während bei beiden anderen Formeln die Menge nicht aktivierter IEs in Fall 2 sich darauf auswirkt, dass Fall 1 gewinnt, obwohl das IE *queue* auf ein anderes Thema hindeutet.

Ist die Anfrage sehr ausführlich und vage, kann man über die Länge des Anwortfalls normieren, da so nur IEs zählen, die in Fällen gefunden wurden. Überflüssige IEs aus der Anfrage verschlechtern das Ergebnis nicht. Ein eher psychologisches Problem bei Normierung (4.1) ist die häufige Vergabe der Gesamtaktivierung 1.0, die eigentlich die Identität von Fall und Anfrage suggeriert. Insbesondere wenn die Anfrage viele Wörter enthält, die gar nicht auf IEs abgebildet werden können, ärgert der Benutzer sich über die Behauptung des Systems, Fälle mit hundertprozentiger Ähnlichkeit gefunden zu haben. Die ausreichende Abdeckung der Fälle durch IEs ist bei einer sorgfältigen Modellierung und Pflege der Lexika gewährleistet. Um eine Normierung auszuwählen, muss man Untersuchungen dazu anstellen, wie prägnant die Nutzer ihre Anfragen stellen und wie groß die Fälle sein werden. In den Anwendungsbeispielen dieser Arbeit wurde Variante (4.1) ausgewählt.

Die Länge der gesamten Ergebnisliste kann entweder fest definiert oder durch eine untere Schranke für die Werte der Ähnlichkeitsfunktion bestimmt werden. Die Wahl hängt davon ab, wie tolerant die Anwender gegenüber Fällen mit niedriger Ähnlichkeit wohl sind. Die erste Variante wird in den Kapiteln 5, 6, 8 und 9 benutzt, die zweite in Kapitel 7.

Setzt man TCBR-Systeme als EM-System ein, dann wird das Erfahrungswissen in Form von Fällen abgespeichert. Der erste Wissenscontainer (siehe S. 39), die Fallbasis, enthält also die für die Benutzer dokumentierten Erfahrungen. Das Hintergrundwissen des Systems, das zum Abspeichern und Wiederfinden des Erfahrungswissens benötigt wird, steckt im zweiten und dritten Wissenscontainer, der Repräsentationssprache samt Vokabular und dem Ähnlichkeitsmaß. Das Hintergrundwissen besteht im TCBR hauptsächlich aus dem Index- und dem Ähnlichkeitslexikon. Die Ähnlichkeitsfunktion selbst und die Attribut-IEs gehören natürlich auch zum Hintergrundwissen in den entsprechenden Containern. Sie müssen aber im Vergleich zu den Lexika

selten verändert oder erweitert werden. Deshalb spielen sie in den Anwendungskapiteln in Teil II dieser Arbeit nur eine kleine Rolle. Das fallbasierte Retrieval unterstützt die Benutzer vor allem bei den WM-Prozessen der Wissens(ver)teilung (siehe Kapitel 7) und der Wissensnutzung (siehe Kapitel 8), aber auch bei der Wissensentwicklung (siehe Kapitel 6) und dem Wissenserwerb (siehe Kapitel 5).

Alternative Methoden des Fallbasierten Schließens für Texte

Seit Jahrzehnten beschäftigt sich die Forschungsrichtung *Information Retrieval* mit der automatischen Verwaltung von Texten. Einige fallbasierte Systeme (zum Beispiel [DC04]) bedienen sich der Standard-Cosinus-Ähnlichkeit für das Vector-Space-Modell [Rij79]. Vereinfacht gesprochen zählt und gewichtet diese Methode des *Information Retrieval* das gemeinsame Vorkommen von Wörtern („string matching“). Grammatikformen, Abkürzungen und verschiedene Schreibweisen von Wörtern bleiben unberücksichtigt. Ungelöst bleibt auch das Problem der unterschiedlichen Formulierungen („paraphrase problem“) für dieselben Sachverhalte. Wiratunga et al. [WKM04] lösen einen Spezialfall dieses Problems, nämlich die Verwendung von Unter- und Oberbegriffen. Dazu treffen sie eine Auswahl der signifikantesten Wörter („feature selection“) und generalisieren diese mit Hilfe von Assoziationsregeln über den Vektoren zu abstrakteren Oberbegriffen. Diese Technik ist ähnlich zur formalen Begriffsanalyse [GW99, SW00]. Unbestreitbarer Vorteil dieser Verfahren ist, dass sie vollautomatisch angewendet werden können. Eine „ablation study“ in [Len98] zeigt, dass das Weglassen von Modellierungsschritten bei unserer Methode (zum Beispiel der Grammatikformen oder gar der lokalen Ähnlichkeitsbeziehungen) signifikante Verluste bei der Genauigkeit der Retrieval-Ergebnisse bringt. Die schlechtesten Werte erzielt das einfache „string matching“. Ein detaillierter Vergleich von TCBR mit den verwandten Gebieten *Information Retrieval* und *Information Extraction* ist in der Dissertation von Mario Lenz [Len99] publiziert.

Noch einfacher kann man Texte durch n-Gramme repräsentieren. N-Gramme sind Zeichenketten der Länge n. Das Zählen übereinstimmender n-Gramme zweier Texte liefert eine sehr einfache, schwache Ähnlichkeitsfunktion. Das Help-Desk-System **CBR Express** der Firma Inference Corp. wendet dieses Verfahren an (**CBR Express** ist mittlerweile nicht mehr auf dem Markt).

In die andere Richtung, nämlich eine stärkere Ähnlichkeitsfunktion durch Semantik oberhalb der Ebene einzelner Wörter, gehen Methoden des „Natural Language Processing“ (NLP). Besonders die automatische Extraktion

von Attributwerten („feature extraction“) mit Hilfe von Satzmustern wird in fallbasierten Systemen derzeit eingesetzt, zum Beispiel in [BA01, GA05, MLB05]. Bei unserer Methode ist der Modellierungsaufwand, um Fälle einer Domäne indizieren zu können, wesentlich geringer. Ganz starke NLP-Verfahren, etwa zur Behandlung der Negation, sind aufgrund mangelnder Robustheit noch nicht für fallbasierte Systeme geeignet.

Teil II

Anwendungen

Kapitel 5

Wissenserwerb: Erfahrungswissen gewinnen

Vor der Einführung eines EM-Systems ist in einer Organisation oft schon Wissen in elektronischer Form dokumentiert. Textdokumente und E-Mail-Archive sind meist unübersichtlich organisiert, bergen aber wertvolle Inhalte. Glossare, Thesauri und Leitliniendokumente enthalten Hintergrundwissen. Deshalb lohnt es sich, für den Wissenserwerb Wissen aus vorhandenen elektronischen Quellen zu extrahieren. Ein EM-System kann dann auf diese Wissensquellen zugreifen und das darin enthaltene Erfahrungswissen für die Benutzer in einem problembezogenen Kontext anbieten. Neben der Auswertung elektronischer Quellen sollte natürlich auch das Wissen in den Köpfen für das EM-System erfasst werden („case authoring“ für fallbasierte EM-Systeme). Die Erschließung dieser natürlichen Wissensquellen gehört eher zur Wissensentwicklung als zum Wissenserwerb (siehe Diskussion auf S. 19). und wird in Kapitel 6 behandelt.

Im Folgenden geht es um die automatische Wissensakquisition aus vorhandenen elektronischen Quellen. Die Fragestellungen in diesem Kapitel sind:

- Wie kann man eine Fallbasis automatisch mit Falldaten füllen?
- Wie geht man dabei mit heterogenen Wissensquellen um, die sich hinsichtlich ihrer Struktur, ihrer Qualität und ihrer Aktualität unterscheiden?
- Wie kann man Querverweise erzeugen lassen?
- Wie kann man Hintergrundwissen aus vorhandenen Quellen gewinnen?

Zunächst diskutiere ich in Kapitel 5.1 allgemeine Eigenschaften der elektronischen Wissensquellen. Kapitel 5.2 stellt eine Akquisitionsstrategie für

Falldaten mit automatischer Vernetzung vor. In Kapitel 5.3 folgt das praktische Anwendungsbeispiel **SimLex** für diese Strategie, das gleichzeitig den TCBR-Ansatz aus Kapitel 4.3 veranschaulicht. Danach beschreibe ich in Kapitel 5.4 die semi-automatische Akquisition von Hintergrundwissen für die Assistenzsysteme mit Hilfe unserer Tools **OntoCBR** und **MultiLingual**.

5.1 Eigenschaften der elektronischen Wissensquellen

Die Struktur, Qualität und Aktualität der vorhandenen Wissensquellen kann sehr heterogen sein. Die Struktur der Quellen wird in Kapitel 5.2 diskutiert. Hier betrachte ich die beiden Eigenschaften Qualität und Aktualität (siehe Abbildung 5.1).

Manche Quellen enthalten fundiertes, sorgfältig redigiertes Wissen. Dazu gehören beispielsweise Glossare, wissenschaftliche Veröffentlichungen oder Handbücher. Andere Quellen wie Newsgroups oder E-Mails bestehen meist aus ungesicherten Erkenntnissen, deren Qualität stark von den jeweiligen Autoren abhängt.

Im Gegenzug sind die ungesicherten Inhalte oft aktueller als die sorgfältig überprüften. Ideal wären hochaktuelle, gut abgesicherte Quellen, die in Abbildung 5.1 im rechten unteren Quader angesiedelt sind. Diese kommen aber nur sehr selten vor, da es ja Zeit kostet, die Inhalte zu prüfen und aufeinander abzustimmen oder gar einen Konsens verschiedener Expertenmeinungen zu bilden. Es ist zum Beispiel unüblich, bei jeder kleinen Änderung an einer Software ein neues Benutzerhandbuch herauszugeben, weil die Entwickler oder Dokumentatoren sich nicht ständig mit der Durchsicht des ganzen Werks beschäftigen können. Auch die Benutzer von Software sind daran gewöhnt, sich an Versionen zu orientieren.

Die meisten Wissensquellen sind entweder neu und ungesichert oder schon älter, aber abgesichert, befinden sich in Abbildung 5.1 also rechts oben oder links unten. Bei der automatischen Integration verschiedener Quellen in ein EM-System muss man mit dieser Heterogenität umgehen. Die beiden folgenden Unterkapitel beschreiben zwei Möglichkeiten, dies zu tun.

Transparenz der heterogenen Wissensquellen

Es gibt verschiedene Ansätze zur Behandlung von Wissensquellen unterschiedlicher Qualität. Ich betrachte erst die Falldaten, dann das Hintergrundwissen. Bei konventionellen CBR-Systemen kann man die Qualität von Falldaten bei der ohnehin manuellen Akquisition sicherstellen. „Schlechte“ Fälle

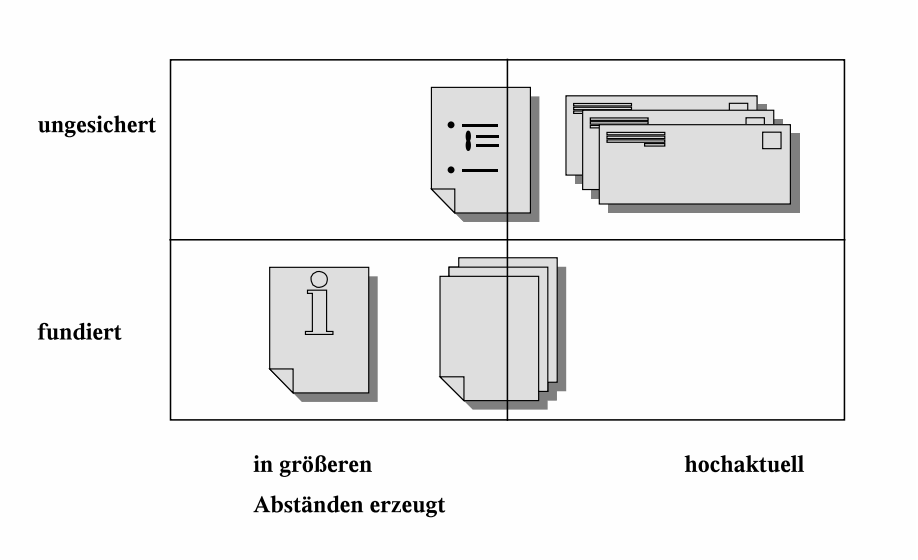


Abbildung 5.1: Heterogenität der Wissensquellen.

werden gar nicht erst modelliert werden. Bei TCBR-Systemen, die die Falldaten automatisch erheben, ist eine manuelle Bewertung ein hoher Zusatzaufwand. Eine automatische Bewertung nach Art der Quelle ist fehleranfällig, da es auch schlechte Handbücher und exzellente Newsgroup-Einträge gibt. Ich habe mich dagegen entschieden, die Qualität einer Quelle im System zu bewerten. Es ist eine interessante Forschungsfrage für die Zukunft, wie man eine solche Bewertung gewinnt und sie dann zum Beispiel in Form von Wahrscheinlichkeiten oder modallogischen Prädikaten sinnvoll mit einer Ähnlichkeitsberechnung kombiniert. Statt einer Bewertung halte ich für die Falldaten am Prinzip der „Transparenz der Wissensquellen“ fest, das heißt es soll für die Wissensanwender immer erkennbar sein, aus welcher Quelle eine Erfahrungsbeschreibung stammt. Damit kann jeder Benutzer selbst die Qualität einschätzen. Auch die Aktualität von automatisch integrierten Falldaten wird in Form einer Datumsangabe transparent gemacht. Zusätzlich können Fälle zeitlich gruppiert werden (vergleiche S. 60). Schon im Vorfeld einer Anfrage kann ein Benutzer auswählen, welche Quellen überhaupt durchsucht werden sollen. Ein anderes Thema ist die Qualitätssicherung und Pflege der Falldaten im Laufe der Zeit. Mit dieser Problematk setzt sich Kapitel 9 auseinander.

Die Qualität des Hintergrundwissens wird durch ausschließliche Verwendung qualitativ hochwertiger Quellen sichergestellt. Bei der automatischen Ergänzung des Hintergrundwissens (siehe Kapitel 5.4, 6.3, 6.4 und 7.3) spielt natürlich auch die Qualität des Verfahrens eine wichtige Rolle. Viele automa-

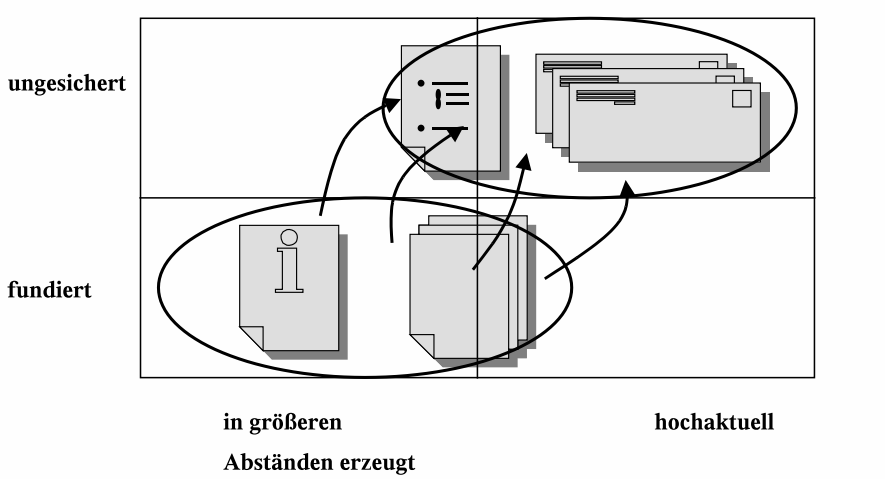


Abbildung 5.2: Vernetzung heterogener Wissensquellen.

tische Verfahren sind deshalb in einen halb-automatischen Entwicklungsprozess eingebettet und werden durch versierte Benutzer oder Administratoren kontrolliert. Da das Hintergrundwissen den Benutzern im Normalfall nicht gezeigt wird und einzelne Modellierungsfehler nur geringe Auswirkungen auf die Retrieval-Ergebnisse haben (siehe Kapitel 7.3), ist die Qualitätssicherung der Falldaten weitaus wichtiger.

Vernetzte Falldaten

Um den Benutzern den Umgang mit den heterogenen Quellen weiter zu erleichtern, ist eine automatische Vernetzung von Wissensquellen für die Falldaten implementiert. Mit Hilfe von Verweisen können sie so zwischen Falldaten aus verschiedenen Wissensquellen navigieren (siehe Abbildung 5.2).

Im folgenden Kapitel stelle ich eine automatische Vernetzungsstrategie dafür vor. Kapitel 5.3 enthält ein fallbasiertes Anwendungsbeispiel.

5.2 Erzeugen von Falldaten aus heterogenen Wissensquellen

Ich habe eine Strategie für die Extraktion und Vernetzung von Fällen aus elektronischen Textdaten entwickelt, bei der die Wissensquellen Unterschiede hinsichtlich Struktur, Qualität und Aktualität aufweisen. Ich verweise auch

auf meinen Konferenzbeitrag [MB05] und die von mir betreute Diplomarbeit [Bel04].

Unsere Strategie berücksichtigt folgende grundlegenden Fragen:

- Wie viele und welche Fallbasen werden für die Organisation der Falldaten benötigt?
- Welche Struktur sollen die Fälle haben?
- Wie werden die Daten aus den Wissensquellen in ein Fallformat transformiert?
- Wie kann man Fälle durch Querverweise semantisch miteinander verbinden?

Jedes der folgenden Unterkapitel beantwortet eine der genannten Fragen. Eine praktische Anwendung unserer Akquisitionsstrategie wird in Kapitel 5.3 vorgestellt.

Arrangement der Fälle in Fallbasen

Bei der Aufteilung der Fälle in verschiedene Fallbasen folgen wir den natürlichen Gegebenheiten: Jede Art von Wissensquelle (E-Mails, Handbücher, wissenschaftliche Artikel) bekommt eine eigene Fallbasis. Außerdem wird bei jeder neuen Version des inhaltlichen Gegenstands des EM-Systems eine neue Runde von Fallbasen für die oben genannten Wissensquellen eröffnet: Alle Texte, die nach einem bestimmten Stichtag geschrieben wurden, kommen in die neue Fallbasis des ihrer Quelle entsprechenden Typs. Jede Fallbasis wird mit Hilfe der in Kapitel 4.3 beschriebenen Technik auf ein eigenes CRN abgebildet, wobei für alle CRNs dasselbe Indexlexikon und dasselbe Ähnlichkeitslexikon verwendet wird. Bei einer Auswahl mehrerer Fallbasen werden die Retrieval-Ergebnisse einzeln dargestellt.

Alternativ dazu hätte man alle Fälle in einer einzigen Fallbasis erfassen können. Die verschiedenen Fallbasen wären dann verschiedene Teil-Fallbasen geworden. Dann wäre aber die Organisation des Fallspeichers komplizierter geworden, um selektive Anfragen zu ermöglichen. Bei der ersten Aufteilung bleibt das in Kapitel 5.1 erläuterte Kriterium der Transparenz der Quellen auf einfache Weise gewahrt. Die Anwender können bei der Benutzung des EM-Systems die Herkunft der Fallbasen sehen und selektiv oder über alle Fallbasen suchen lassen.

Fallstruktur

Die Struktur der originalen Textdaten ist heterogen. Artikel und Handbücher bestehen hauptsächlich aus fortlaufendem Text, E-Mails haben einen „header“ und einen Textteil, eventuell auch noch Anhänge.

Innerhalb des EM-Systems haben trotzdem alle Falldaten dieselbe Struktur, damit sie einheitlich durchsucht werden können. Die Fallstruktur ist eine Instanz der allgemeinen Definition auf S. 43. Die Falldaten unseres Anwendungsprojekts werden in zwei Repräsentationsformen abgespeichert: Einmal in XML-Dateien (Beispielfall siehe linke Seite von Tabelle 5.1) und einmal als Mengen von IEs (siehe rechte Seite der Tabelle). Die Elemente *TITLE* und *DESCRIPTION* enthalten den Titel und die Beschreibung des Falls. Drei weitere Elemente sind mit Attribut-Werte-Paaren gefüllt: *RETRIEVAL_ATTRIBUTES* (Retrieval-Attribute), *PRESENTATION_ATTRIBUTES* (Darstellungs-Attribute) und *INFORMATION_ATTRIBUTES* (Informations-Attribute). Die Retrieval-Attribute sind die einzigen, die beim Retrieval berücksichtigt werden. Die Darstellungs-Attribute speichern die Herkunft des Falls und die Querverweise zu anderen Fällen. Diese Informationen werden zum Aufbau der Präsentation des Falls für die Benutzer benötigt. Die Informations-Attribute enthalten zusätzliche Informationen zur Entstehungsgeschichte des Falls („change history“), die Revisionsnummer und die systeminterne Sortierzeit. Die IE-Darstellung wird mit Hilfe des Index-Lexikons aus der XML-Darstellung automatisch ermittelt (vergleiche S. 45 in Kapitel 4.3).

Transformation der Daten in die Fallstruktur

Für jede Art der Wissensquelle wird ein eigenes Verfahren zur Transformation der Daten in die oben beschriebene Fallstruktur (vgl. S. 61 und Tabelle 5.1) entwickelt. Ich stelle ein Verfahren für E-Mail-Daten und ein zweites Verfahren für fortlaufende Texte in LaTeX vor. Weitere Verfahren für weitere Arten von Wissensquellen lassen sich leicht integrieren. Alle Verfahren sollten dieselben drei Teilaufgaben umsetzen:

1. Segmentierung der Quelldaten,
2. Transformation in XML,
3. Extraktion von Attributwerten („feature extraction“).

Die erste Teilaufgabe ist nötig, weil kleinere Einheiten sich besser für das Retrieval und die Erzeugung von Querverweisen eignen als lange Texte. Es wurden dafür sehr einfache Verfahren gewählt: Fortlaufende Texte werden in ihre Textabsätze zerlegt, so dass jeder Absatz ein Fall wird. Die Überschrift

Tabelle 5.1: Beispiel eines E-Mail-Falls in XML und als Menge von IEs

$< CASE >$ $< CASE\_NUMBER >$ 324 $< /CASE\_NUMBER >$ $< TITLE >$ Re: [robocup-sim] about connection on Robocup Soccer Simulator $< /TITLE >$ $< RETRIEVAL\_ATTRIBUTES >$ author = Tom Howard version = 7.07 media = EMAIL $< /RETRIEVAL\_ATTRIBUTES >$ $< PRESENTATION\_ATTRIBUTES >$ msgId = <200304282310.h3SNANu101116@ pcthoward-2529.tsi-woywoy.tibco.com> email = tomhoward@users.sourceforge.net previous = 320 next = 330 simCaseTo = 21;22;23;124;172 $< /PRESENTATION\_ATTRIBUTES >$ $< INFO\_ATTRIBUTES >$ createdOn = 29 Apr 2003 createdBy = Tom Howard changedOn = 31 Dec 2003 changedBy = Tom Howard revision = 11 sortTime = 1072890340637 $< /INFO\_ATTRIBUTES >$ $< DESCRIPTION >$ What port do you send to? What message did you send? If you do a recvfrom() (or actually the java equivilent), what message do you get and where from? Cheers, Tom Howard $< /DESCRIPTION >$ $< /CASE >$	$CASE\_IDENTIFIER = 324$ $IE\_LIST\_TITLE =$ (CONNECTION, ROBOCUP, SOCCER, SIMULATOR) $IE\_LIST\_RATTR =$ ((TOM, author), (HOWARD, author), (7.07, version), (EMAIL, media)) $IE\_LIST\_DESCRIPTION =$ (PORT, SEND, MESSAGE, SEND, RECVFROM, JAVA, EQUIVALENT, MESSAGE, GET)

im fortlaufenden Text ergibt den Titel des Falls oder auch mehrerer Fälle, wenn der Text bis zur nächsten Überschrift mehrere Absätze hat (siehe Beispiel in Abbildung 5.3). Jede E-Mail wird ein Fall. Die Betreff-Zeile liefert den Titel des Falls. Anhänge (zum Beispiel Dateien, Bilder, alte E-Mails) werden gelöscht. Alte E-Mails dürfen gelöscht werden, weil ich davon ausgehe, dass sie als eigene Fälle im System erfasst und via Hyperlinks erreichbar sind (vergleiche dazu den Abschnitt zur automatischen Vernetzung auf S. 63).

Die zweite Teilaufgabe erzeugt als Zwischenformat XML. Dies erleichtert das Debugging des Systems und den Austausch von Falldaten (vgl. Kapitel 7). Die Transformation der fortlaufenden Texte aus LaTeX in XML geht aus technischen Gründen leider noch nicht vollautomatisch. So wird zum Beispiel die Nummerierung von Aufzählungen nicht korrekt übertragen. Bessere latex2html-Werkzeuge werden ein Korrekturlesen der Fälle durch einen Administrator hoffentlich bald überflüssig machen. Da das Einpflegen fortlaufender Texte zum Glück nicht allzu oft geschieht, hält sich der manuelle Aufwand auch jetzt schon in Grenzen. Kommerzielle Software könnte eingesetzt werden, um weitere Dateiformate in XML zu transformieren. E-Mails werden durch Perl-Skripte vollautomatisch in Fälle umgeformt. Dies ist wichtig, weil über die E-Mail-Listen ständig neue E-Mails verschickt werden, die das neueste Erfahrungswissen enthalten. Durch die automatische Umformung können auch nutzlose Fälle entstehen, etwa wenn jemand die E-Mail-Liste zweckentfremdet, um sich mit jemand zum Mittagessen zu verabreden. Für das EM-System ist das nicht weiter schlimm, weil solche Fälle kaum relevante Begriffe enthalten. So werden sie auch nicht durch IEs indiziert und können nicht in den Retrieval-Ergebnissen stören. Problematischer sind einige E-Mails aus dem arabischen Sprachraum, die sich nicht an den RFC-Standard 822 [Cro82] halten. Wir haben Regeln zur Behandlung der häufigsten Abweichungen definiert und in den Transformationsprozess integriert.

Einfache Informationsextraktion ermittelt den Absender einer E-Mail und trägt ihn als Wert für das Attribut Autor ein.

Automatische Vernetzung der Falldaten

Die Falldaten werden auf zwei verschiedene Arten miteinander vernetzt: innerhalb einer Fallbasis und zwischen verschiedenen Fallbasen. Die Fälle werden innerhalb ihrer eigenen Fallbasis in einen Kontext aus Vorgänger- und Nachfolgerfällen gestellt. Querverweise verbinden sie zusätzlich mit ähnlichen Fällen aus anderen Fallbasen. Für Originaldokumente in Formaten, die Links unterstützen, könnte man natürlich auch die Originale mit den Links anreichern. Das erfordert allerdings einen höheren Aufwand, weil für jedes

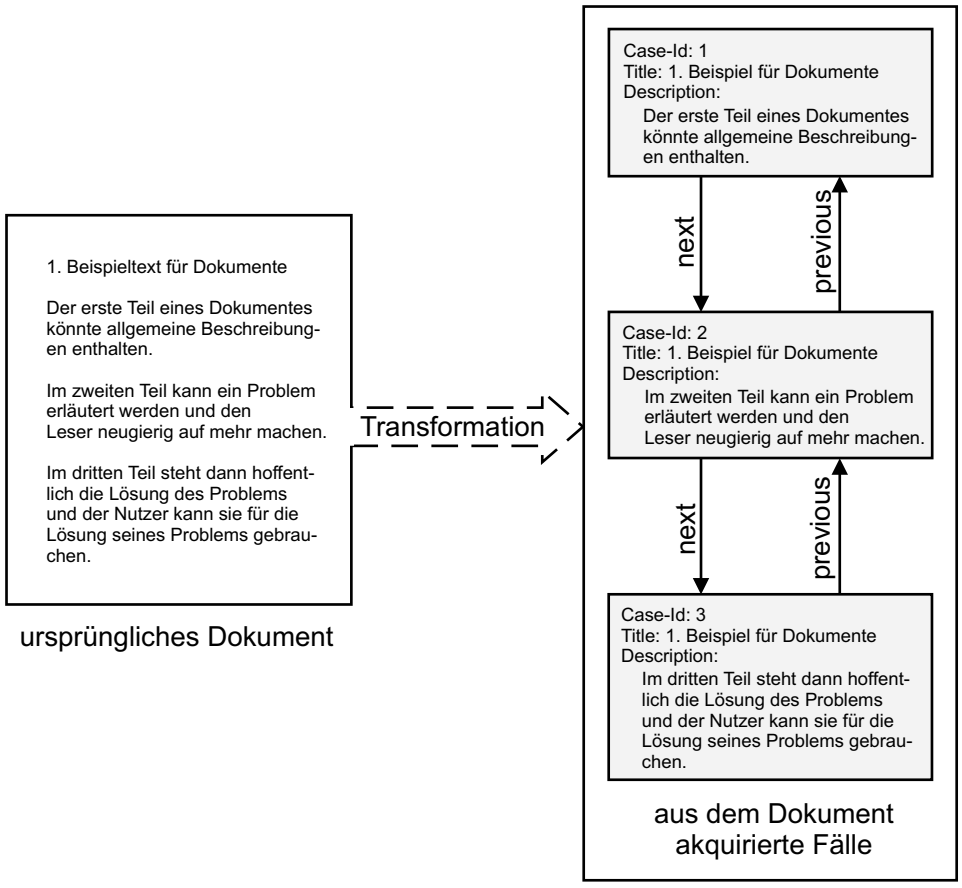


Abbildung 5.3: Erzeugen von Falldaten aus fortlaufenden Texten.

Format zusätzlich zur Transformation in die Fallstruktur ein Verfahren zur Anpassung des Originaldokuments geschrieben werden muss.

Abbildung 5.3 zeigt, wie Fälle, die aus fortlaufenden Texten stammen, mit ihrem jeweiligen Vorgänger und Nachfolger verbunden werden. Diese Verbindungen sind wichtig, weil bei der einfachen Segmentierungsmethode nicht gewährleistet werden kann, dass jeder Fall ein Problem und dessen Lösung enthält. Die Benutzer müssen unter Umständen in den zusammenhängenden Falldaten blättern, um zur Lösung zu gelangen.

Fälle, die aus E-Mails entstanden sind, werden in eine Baumstruktur aus Vorgängern und Nachfolgern eingebettet. Ein Beispiel ist in Abbildung 5.4 dargestellt. Durch die Baumstruktur sind zusätzliche Lösungen und Bemerkungen zu einem Problem leicht erreichbar.

Die Erzeugung von Querverweisen ist komplexer als die Vernetzung der Fälle mit Vorgängern und Nachfolgern. Die Querverweise verbinden die Fäl-

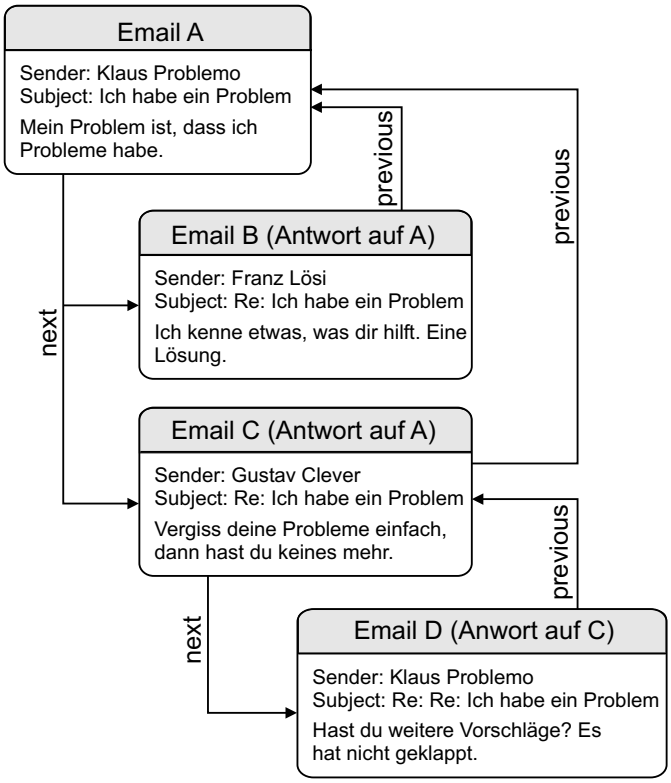


Abbildung 5.4: Erzeugen von Falldaten aus E-Mail-Listen.

le aus fortlaufenden Texten mit den E-Mail-Fällen. Wir benutzen die Ähnlichkeit als semantisches Kriterium für die Erstellung eines Querverweises. Dafür wird jeder Fall, der nicht aus einer E-Mail erzeugt wurde, als Anfrage an die E-Mail-Fallbasis gestellt. Dann wird er mit den E-Mail-Fällen verbunden, die die höchsten Ähnlichkeitswerte im Retrieval-Ergebnis erreicht haben. Der Retrieval-Prozess läuft mit Hilfe des CRNs ab, wie es in Kapitel 4.3 beschrieben ist. Sind mehrere CRNs beteiligt (siehe oben), werden die Ergebnisse einfach gemischt und anhand der Ähnlichkeitswerte sortiert. Die Gegenrichtung, nämlich E-Mail-Fälle mit Fällen aus den anderen Fallbasen zu verknüpfen, wurde von den befragten Benutzern unserer Beispielanwendung nicht gewünscht. Solche Querverweise ließen sich aber leicht nach derselben Methode erzeugen. Das Verfahren ist je nach Größe der Fallbasen mehr oder weniger zeitaufwändig, da für jeden Fall außer den E-Mails ein eigener Retrieval-Prozess durchlaufen werden muss. Deshalb werden die semantischen Links außerhalb des Normalbetriebs des EM-Systems erzeugt und im Fallformat mit abgespeichert.

Alle erzeugten Verweise gehen nicht in spätere Retrieval-Prozesse mit ein, sondern bieten lediglich den Benutzern eine Navigationsmöglichkeit. In Zukunft könnte man darüber nachdenken, ob auch Konglomerate von Fällen oder Meta-Fälle selbst im Retrieval zur Verfügung stehen sollten. Natürlich müsste man die Größe irgendwie beschränken, damit keine „Killer“-Fälle dabei entstehen: Ein ganzes Manual als Fall würde sicherlich zu nahezu jeder Anfrage gefunden.

5.3 Das Web-Lexikon SimLex für die RoboCup-Community

Unser Anwendungsbeispiel der Akquisitionsstrategie für Falldaten (siehe voriges Kapitel) heißt **SimLex** und ist in der RoboCup-Domäne angesiedelt. Die RoboCup-Community ist eine Gemeinschaft von Wissenschaftlern aus aller Welt, die sich mit Fußball spielenden Robotern beschäftigt. Während des Spiels sind die Roboter völlig autonom und spielen in Mannschaften zusammen. Einmal im Jahr treten die Mannschaften aller Forschergruppen bei der RoboCup-Weltmeisterschaft gegeneinander an. Der RoboCup hat sich dem „open source“-Gedanken verpflichtet, das heißt nach jeder Weltmeisterschaft wird das Wissen der Forschergruppen veröffentlicht und ausgetauscht. Dies geschieht in Form von Quellcode, Manuals, Artikeln und über E-Mail-Listen. Da viele Studierende beteiligt sind, ist die Fluktuation der Community-Mitglieder hoch. Es müssen sich immer wieder neue Leute einarbeiten. In

den einzelnen Forschergruppen wird ständig neues Wissen entwickelt. Ein EM-System passt also sehr gut in dieses Szenario.

Wir haben **SimLex** als Web-Lexikon für die Simulationsliga des RoboCup entwickelt. In der Simulationsliga spielen 22 autonome Computerprogramme in einem simulierten Fußballspiel auf einem sogenannten „Soccer-Server“. Wir integrieren und vernetzen in **SimLex** das Erfahrungswissen, das auf der Webseite der RoboCup Simulationsliga[1] veröffentlicht wird, und alle E-Mails aus der E-Mail-Liste der Community[2]. Neueinsteiger können **SimLex** zur Einarbeitung in die Spielregeln, den RoboCup-Wettbewerb und das „Soccer-Server“-Programm nutzen. Erfahrene Entwickler finden in **SimLex** Unterstützung für aktuelle Probleme. In beiden Szenarien hilft unser fallbasiertes Retrieval dabei, das explizite Wissen der Community zu durchsuchen.

Die Fallbasen von **SimLex** werden mehrere hundert Mal pro Jahr mit neuen E-Mails versorgt. Die Manuals des „Soccer-Servers“ erscheinen ca. alle zwei oder drei Jahre neu. Weitere Dokumente kommen in unregelmäßigen Abständen hinzu. Den Quellcode selbst haben wir nicht in **SimLex** integriert. Da die Kommentare im Quellcode mit Hilfe einer Dokumentationssoftware geschrieben wurden, sind sie im Manual enthalten. Die Herausgabe eines neuen Manuals zeigt an, dass eine neue Version des „Soccer-Servers“ in Umlauf gebracht wurde. Deshalb ist dies der Auslöser daür, dass eine neue Runde von Fallbasen eröffnet wird.

Die Anfragen an **SimLex** können über ein einfaches Texteingabefeld auf einer Webseite eingegeben und durch Spezifikation von Versionen („version“) und Quellarten („media“) auf bestimmte Wissensquellen eingeschränkt werden (siehe Abbildung 5.5). Die dreißig Fälle mit den höchsten Ähnlichkeitswerten nach der Maximum-Formel (wie im Beispiel auf S. 52) werden als Retrieval-Ergebnis präsentiert. Für ungefähr 16.000 IEs, 22.300 Einträge im Ähnlichkeitslexikon und 1100 Fälle hat das Retrieval auf einem 750 MHz Linux PC zwischen 0,5 und 0,7 Sekunden gedauert. Aus der Ergebnisliste können die Benutzer einen Fall per Mausklick auswählen, der dann im Detail angezeigt wird (siehe Abbildung 5.6 und 5.7). Die automatisch erzeugten Verbindungen zu anderen Fällen werden mit angezeigt und können durch einen weiteren Mausklick verfolgt werden.

Fälle, die aus kontinuierlichem Text erzeugt wurden, werden mit ihrem Vorgängerfall, ihrem Nachfolgerfall und bis zu fünfzehn E-Mails verbunden, die einen Ähnlichkeitswert höher als 0,6 erreicht haben. Diese beiden Zahlen sind willkürlich gewählt und können nach weiteren praktischen Experimenten in der Konfigurationsdatei des Systems angepasst werden.

[1] http://www.robocup.org

[2] robocup-sim@robocup.biglist.com

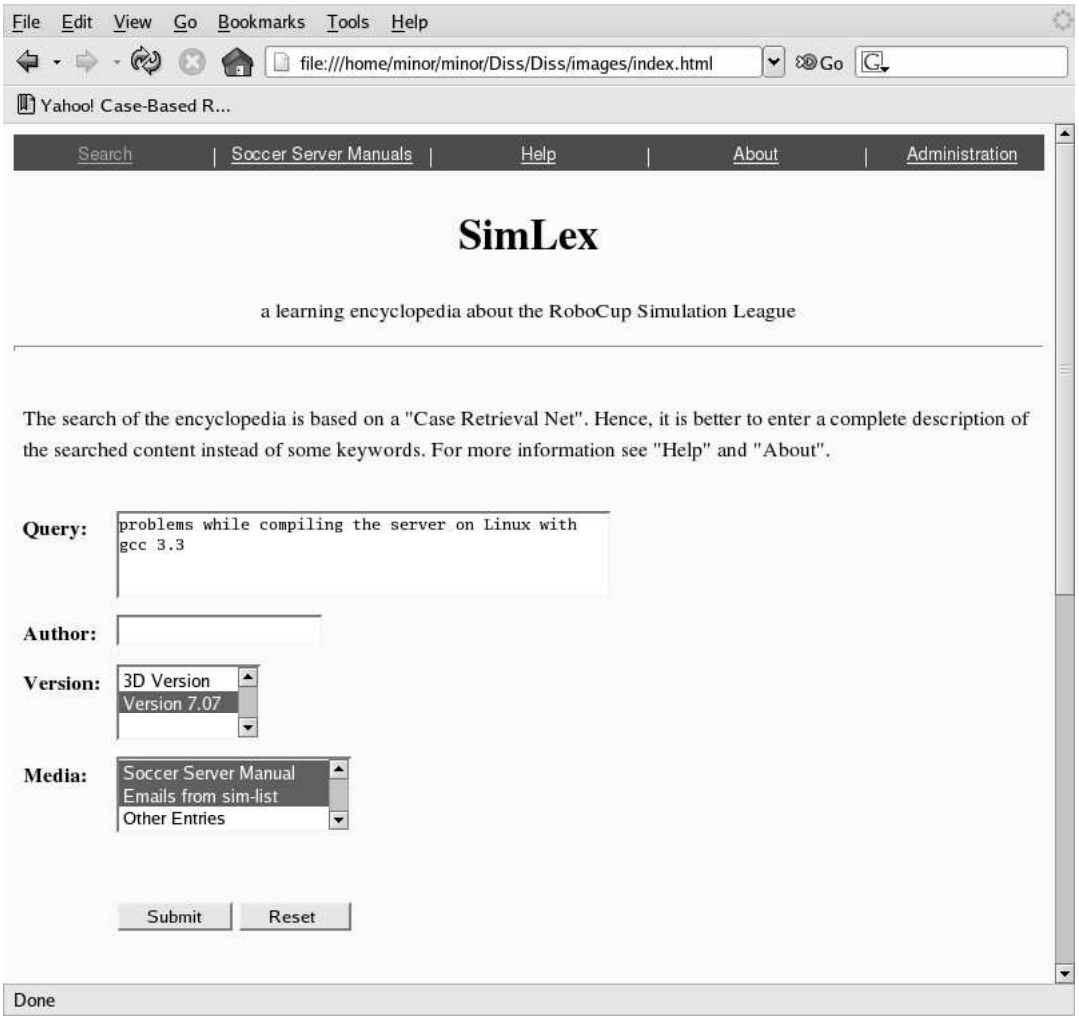


Abbildung 5.5: Anfrageseite von **SimLex**.

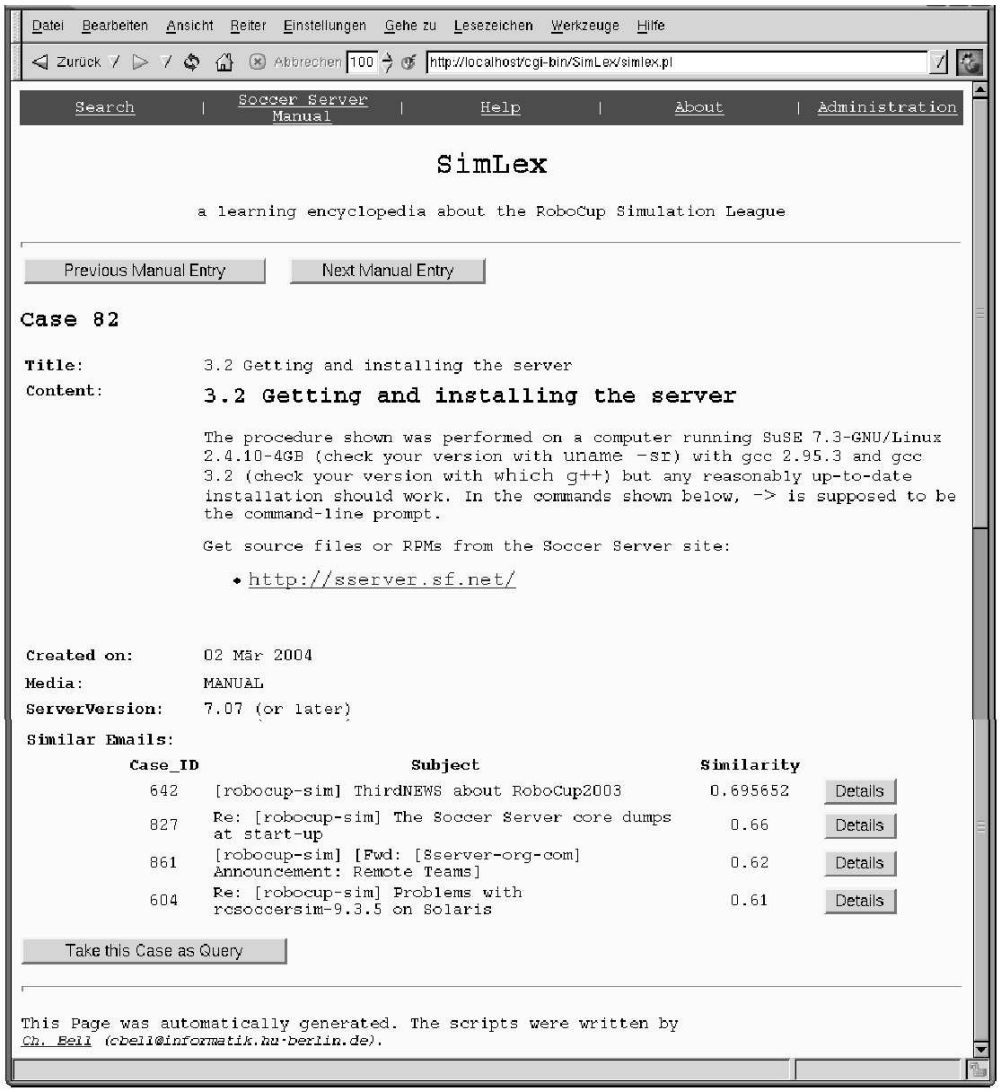


Abbildung 5.6: Fall in **SimLex**, der aus kontinuierlichem Text erzeugt wurde.

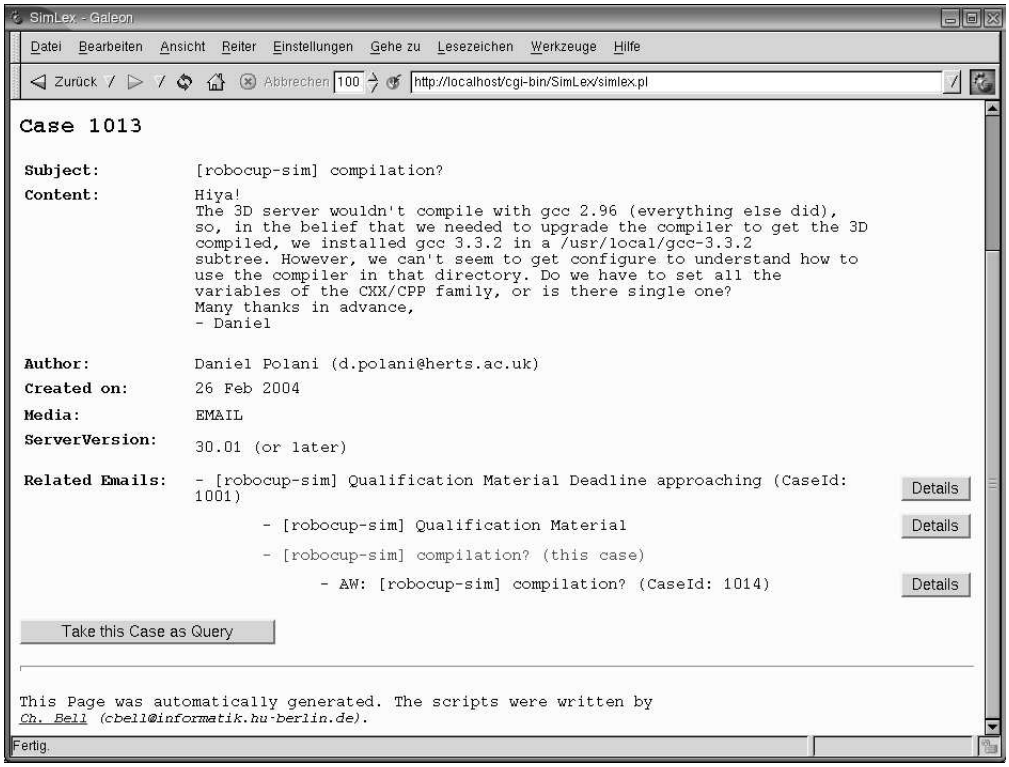


Abbildung 5.7: Fall in **SimLex**, der aus einer E-Mail erzeugt wurde.

E-Mail-Fälle sind, wie im vorigen Kapitel beschrieben, in eine Baumstruktur eingebettet. Die Navigation im Baum funktioniert genau so, wie es aus Newsgroups und Diskussionsforen bekannt ist. Aus Implementierungsgründen speichern wir intern nur den direkten Vorgänger und Nachfolger einer E-Mail ab und erzeugen den Baum erst, wenn der Fall zur Laufzeit des Systems angeklickt wurde. Dies dauert zwischen 0,4 und 1,2 Sekunden, könnte also noch verbessert werden.

Wir haben das fallbasierte Retrieval ergänzt um eine einfache Stichwortsuche und um eine flache Katalogstruktur, mit der man direkt durch die Fallbasen navigieren kann. Dies ist eine Vorsichtsmaßnahme für den Fall, dass die internen Lexika mit Hintergrundwissen in Zukunft stark vernachlässigt werden sollten. Dann können die Benutzer immer noch auf diese konventionellen Suchmechanismen zurückgreifen. Ein Vergleich von Stichwortsuche und TCBR ist auf S. 53 beschrieben.

SimLex ist als Prototyp in C++ implementiert und befindet sich zur Zeit in einer ersten Erprobungsphase mit ausgewählten RoboCup-Mitgliedern, bevor es der ganzen Community zur Verfügung gestellt werden soll. Es hat eine Client-Server-Architektur, das heißt es gibt einen CRN-Server, der durch CGI-Skripte mit Hilfe eines Apache-Webservers Anfragen zugestellt bekommt. Die Anfragen landen in einer Warteschlange und werden sequentiell abgearbeitet, so dass der Mehrbenutzerbetrieb sichergestellt ist.

Teile der Arbeiten an **SimLex** wurden im Rahmen des DFG Schwerpunktprogramms Nr. 1125 gefördert, wofür ich mich herzlich bedanke.

5.4 Erzeugen von Hintergrundwissen mit OntoCBR und MultiLingual

Das Hintergrundwissen von TCBR-Systemen besteht aus einem IE-Lexikon und einem Ähnlichkeitslexikon (vergleiche S. 52). Das Füllen dieser Lexika kann ganz gut automatisch unterstützt werden. Je nach Verfügbarkeit geeigneter Wissensquellen lassen sich die hier beschriebenen Verfahren auf strukturiertes CBR oder andere Techniken für EM-Systeme, zum Beispiel auf die automatische Population von Ontologien (siehe S. 73), übertragen. Dies liegt aber nicht im Fokus dieser Arbeit.

Sowohl für das IE-Lexikon als auch für das Ähnlichkeitslexikon eines TCBR-Systems ist eine gemischte Strategie zum Wissenserwerb und zur Wissensentwicklung sinnvoll, die sich aus folgenden Teilen zusammensetzt:

- aus der Extraktion von Lexikoneinträgen aus vorhandenen Quellen, soweit dies möglich ist,
- aus der manuellen Ergänzung der Lexika, vor allem durch Fachbegriffe und deren Beziehungen untereinander, und
- durch semi-automatische Verfahren zur Erweiterung und Veränderung der Lexika.

Die Extraktion von Lexikoneinträgen beschreibe ich unten für das Indexlexikon und dann für das Ähnlichkeitslexikon. Zu Punkt zwei, der manuellen Ergänzung, ist zu bemerken, dass sie von Personen durchgeführt werden sollte, die sich sowohl in der Domäne des Systems gut auskennen als auch eine ungefähre Vorstellung davon besitzen, was mit den Lexikoneinträgen im Retrieval geschehen soll. Unsere Erfahrungen zum Beispiel im Projekt **Simatic Knowledge Manager** [LHK98a] haben gezeigt, dass dies für versierte Dokumentatoren kein Problem darstellt, aber natürlich Zeit kostet. Der dritte Teil, die automatischen Akquisitionsverfahren, werden nicht hier, sondern in den Kapiteln 6.3 und 6.4 zur Wissensentwicklung beschrieben (vergleiche zu dieser Zuordnung S. 19). Auch das Lernen durch Wissensaustausch erweitert die eigenen Wissensbestände und wird in Kapitel 7 dieser Arbeit erläutert.

Erzeugen von Indexlexikon-Einträgen

Das automatische Erzeugen von Indexlexikon-Einträgen geschieht im Wesentlichen durch Extraktion von Wörtern aus Lexika, Glossaren und Ontologien. Wir haben in verschiedenen Projekten einfache **Perl**-Skripte dafür implementiert.

Beispielsweise haben wir das elektronische Computerlexikon **FOLDOC** [How05] für englische Begriffe verwendet. Jeder Begriff, der in **FOLDOC** erläutert ist, wird als Text-IE mit der Wortschatz-Kategorie „IT-Begriff“ in das Indexlexikon geschrieben. Für die Erfassung verschiedener Schreibweisen wird jeder Begriff blind grammatikalisch gebeugt, indem ein -s, ein -ing oder ein -ed angehängt wird. Dieses Verfahren wendet den „stemming“-Algorithmus von Porter [Por80] in umgelehrter Richtung an. Abbildung 5.8 zeigt ein Beispiel. Durch das Verfahren entsteht zwar ein zu großes Indexvokabular, weil ein Teil der Formen so nie in natürlicher Sprache vorkommt, es deckt aber die meisten tatsächlichen Grammatikformen im Englischen ab.

```
__PRINT__[IT-TERM];PRINT;PRINTS;PRINTING;PRINTED
__PRINTER__[IT-TERM];PRINTER;PRINTERS;PRINTERING;PRINTERED
```

Abbildung 5.8: Beispielabschnitt des erzeugten IE-Lexikons.

Die überzähligen Formen stehen unbenutzt in den Hash-Tabellen des Textparsers (siehe S. 45), stören aber die Performance des Systems nicht, da die Falldaten ja vor Start des Systems indiziert werden. Leider kann man dieses Verfahren im Deutschen nicht anwenden, weil dort die Grammatikformen wesentlich komplexer sind. In Kapitel 6.4 ist ein etwas diffizileres Verfahren mit Hilfe von „Part-of-Speech-Taggern“ beschrieben.

Für die Modellierung eines Grundwortschatzes zu UNIX wurden alle Einträge der „UNIX-man-pages“ zu Text-IEs umgeformt. Diese haben leider keine automatisch erzeugten Grammatikformen, sondern müssen nach und nach manuell damit versorgt werden. Dennoch ist es schon eine große Hilfe, die Grundformen als Anhaltspunkt bei der Modellierung zu haben. In älteren Industrieprojekten außerhalb dieser Arbeit [KH98, LHK98a] haben wir firmeneigene Glossare, Leitlinien und Spezifikationen verwendet, um semiautomatisch die Indexlexika für die jeweilige Domäne aufzubauen.

Mit **OntoCBR** (siehe unten) kann man auch die Konzepte aus vorhandenen Ontologien in Text-IEs transformieren.

Definition 5.4.1 (Ontologie nach [Gru93])

> Eine *Ontologie* ist die formale, explizite Spezifikation einer gemeinsamen Konzeptionalisierung.

Ontologien werden verwendet, um Konzepte und Beziehungen in einer Domäne zu modellieren. Die Ontologie sollte in einer formalen Sprache beschrieben sein, die maschinell verarbeitet werden kann. Die gängigsten Sprachen sind zur Zeit **RDF** (Resource Description Framework, siehe [McB04]) und als Erweiterung dazu **OWL** (Web Ontology Language, siehe [AH04]). Ich benutze zur Erläuterung meiner Beispiele im Folgenden **F-Logik** [AL04] als Ontologiesprache. **RDF** kann durch **F-Logik** wesentlich übersichtlicher repräsentiert werden. „Gemeinsame Konzeptionalisierung“ in Grubers Definition heißt, dass eine Ontologie einen Konsens verschiedener individueller Sichtweisen abbildet. So ist zum Beispiel der Begriff „Flasche“ bei verschiedenen Firmen unterschiedlich belegt: Für die einen bezeichnet er einen Teil des Leerguts, für die anderen eine Getränkeflasche mit Inhalt. Diese beiden Sichtweisen führen zu einer empirischen Inkonsistenz, zum Beispiel innerhalb der *is_part*-Relation mit $bierflasche[is\_part- >> leergut]$ aus der

einen Sichtweise und $bier[is\_part- >> bierflasche]$ aus der anderen. Kombiniert man die beiden *is_part*-Paare in einer Ontologie, so wird unter der Annahme der Transitivität von *is_part* Bier auf einmal als Teil des Leerguts betrachtet. Zum Aufbau einer gemeinsamen Ontologie gehört also die Konsensbildung zwischen beteiligten Personen oder Organisationen dazu.

Nicht alle Ontologien halten sich streng an Grubers Definition. **W**ordNet [Fel98] zum Beispiel macht Abstriche bei der formalen, maschinenverständlichen Sprache zu Gunsten der Lesbarkeit für Menschen (vergleiche das Beispiel in Abbildung 5.9 auf S. 76).

Nach Fensel [Fen01] gibt es verschiedene *Typen von Ontologien*:

- *Domänenontologien* beschreiben die wichtigsten Konzepte und Beziehungen in einer Domäne. Sie sind der am häufigsten anzutreffende Typ von Ontologien.

- *Generelle Ontologien* definieren Begrifflichkeiten, die über Domänengrenzen hinweg gelten, zum Beispiel Raum oder Zeit. Ein noch ungelöstes Problem stellen unterschiedliche Konzepte auf der obersten Ebene dar.

- *Methodenontologien* enthalten Konzepte, auf denen Problemlösungsmethoden arbeiten, zum Beispiel „korrekter Zustand“ oder „verletzte Bedingung“.

- *Aufgabenontologien* beschreiben Begrifflichkeiten verschiedener Typen von Aufgaben, zum Beispiel „Ursache“, „Symptom“ oder „erzeugt“.

- *Ontologien für Metadaten* liefern das Vokabular zur Beschreibung von Inhalten. Prominentestes Beispiel ist der **D**ublin Core [Dub05].

Die von mir betreute Diplomarbeit [Mül05] hat die Integration von Domänenontologien und dem TCBR-Ansatz aus Kapitel 4.3 näher untersucht. Das im Rahmen dieser Diplomarbeit entwickelte Tool **OntoCBR** hat unter anderem eine Importfunktionalität für spezielle Domänenontologien aus **OWL**. Die Implementation in **Perl** demonstriert sie am Beispiel einer einfachen Ontologie zu Computerwissen. In dieser Beispielontologie (siehe S. 109) sind die Konzepte als OWL-Klassen modelliert, die verschiedenen Grammatikformen sind als Instanzen der Konzepte angegeben. Sind in der Ontologie keine Grammatikformen spezifiziert, müssen diese wie beim UNIX-Wortschatz von Hand nachmodelliert werden. Beim Import einer Ontologie in ein IE-Lexikon wird aus jeder OWL-Klasse ein IE-Name. Die zugehörigen Instanzen bilden das Indexvokabular dieses IEs. Die Transformation geschieht also vollautomatisch.

Leider ist keines der hier beschriebenen automatischen Erzeugungsverfahren vollständig: In beinahe jedem neuen Fall sind ein oder zwei wichtige Wörter enthalten, die bislang noch nicht im Indexlexikon stehen. Diese Beobachtung deckt sich mit der Erkenntnis aus dem Projekt **Deutscher Wortschatz** [Pro05] an der Universität Leipzig, dass die natürliche Sprache unglaublich kreativ verwendet wird und immer wieder neue Wörter geschaffen werden. So muss also mit Hilfe der tatsächlichen Falldaten das Lexikon überprüft und ergänzt werden, damit das Retrieval genaue Ergebnisse liefern kann. Der Zeitaufwand pro Fall, um wichtige zusätzliche Begriffe ins Lexikon einzutragen, variiert je nach Länge des Falls und je nach Erfahrung des Administrators. Zum Beispiel dauert beim **E**xperienceBookII (für eine Beschreibung des Systems siehe Kapitel 8) das Einpflegen eines neuen Falls inklusive der Erzeugung neuer Einträge im Ähnlichkeitslexikon (siehe unten) zwischen einer halben und zehn Minuten.

Erzeugen von Ähnlichkeitslexikon-Einträgen

Im Folgenden beschreibe ich zwei automatische Verfahren, um das Ähnlichkeitslexikon mit Hilfe vorhandener Wissensquellen zu füllen. Das erste Verfahren nutzt semantische Relationen zwischen Konzepten, wie sie in einer Domänenontologie modelliert sind. Das zweite Verfahren nimmt die Übersetzung von Wörtern aus einer Sprache in eine andere als Grundlage für einen neuen Ähnlichkeitstyp.

```
Sense 1
printer, pressman -- (someone whose occupation is printing)
       => skilled worker, trained worker -- (a worker who has
          acquired special skills)
Sense 2
printer -- ((computer science) an output device that prints
the results of data processing)
       => peripheral, computer peripheral, peripheral device --
          ((computer science) electronic equipment connected
          by cable to the CPU of a computer; "disk drives and
          printers are important peripherals")
Sense 3
printer, printing machine -- (a machine that prints)
       => machine -- (any mechanical or electrical device that
          transmits or modifies energy to perform or assist in
          the performance of human tasks)
```

Abbildung 5.9: Ausgabe von **WordNet** zu `printer`.

Wir haben für die meisten Anwendungsbeispiele dieser Arbeit viele Einträge für das Ähnlichkeitslexikon halbautomatisch aus der frei verfügbaren Ontologie **WordNet** [Fel98, Wor05] extrahiert:

- Jedes IE aus dem Indexlexikon wird automatisch in **WordNet** nachgeschlagen.
- Aus der Antwort werden alle Beziehungen, die für die Domäne wichtig sind, von Hand markiert.
- Jeder Relation in **WordNet** wird von Hand ein entsprechender Ähnlichkeitstyp im Ähnlichkeitslexikon zugeordnet.
- Die Paare von IEs werden von einem **Perl**-Skript mit dem entsprechenden Ähnlichkeitstyp ins Ähnlichkeitslexikon eingetragen.
- Fehlende Partner-IEs werden automatisch identifiziert und manuell im Indexlexikon nachgetragen.

Abbildung 5.9 zeigt eine Beispielausgabe von **W**ordNet zum IE `printer`. WordNet kennt drei verschiedene Bedeutungen von `printer`: `Sense 1` beschreibt eine Person, die eine Druckmaschine bedient, `Sense 2` beschreibt einen Drucker in der Computerwelt, und `Sense 3` eine Druckmaschine. Ist die gewünschte Domäne die IT-Welt, so werden nur die Beziehungen aus `Sense`

```
__PRINTER__[STRONG_SIM];__PERIPHERAL__
__PRINTER__[STRONG_SIM];__COMPUTER PERIPHERAL__
__PRINTER__[STRONG_SIM];__PERIPHERAL DEVICE__

und in der Gegenrichtung:

__PERIPHERAL__[STRONG_SIM];__PRINTER__
__COMPUTER PERIPHERAL__[STRONG_SIM];__PRINTER__
__PERIPHERAL DEVICE__[STRONG_SIM];__PRINTER__
```

Abbildung 5.10: Beispieleinträge im Ähnlichkeitslexikon, die aus **W**ordNet abgeleitet wurden.

2 untersucht. Aus den drei angegebenen Synonymen `peripheral`, `computer peripheral` und `peripheral device` werden sechs neue Einträge im Ähnlichkeitslexikon erzeugt, die in Abbildung 5.10 zu sehen sind.

Die sechs neuen Einträge im Ähnlichkeitslexikon werden dem Typ „STRONG_SIM“ für „starke Ähnlichkeit“ zugeordnet, da wir Synonymität in beiden Richtungen als starke Ähnlichkeit interpretiert haben[3]. Entsprechende Zuordnungen haben wir für weitere Beziehungsarten in **WordNet** getroffen, zum Beispiel für Antonyme (Gegenteile) und Hypernyme bzw. Hyponyme (Oberbegriffe bzw. Unterbegriffe).

Nach demselben Grundschema arbeitet **O**ntoCBR [Mül05], um Einträge für das Ähnlichkeitslexikon aus den Relationen einer Domänenontologie zu gewinnen. Im Gegensatz zu dem oben beschriebenen Verfahren mit **WordNet** arbeitet **OntoCBR** vollautomatisch (vgl. auch S. 108). Es sei hier eine kurze Bemerkung zum Export des Ähnlichkeitslexikons gestattet, obwohl dies nicht zum Wissenserwerb gehört: Die Ähnlichkeitstypen mit qualitativen Bezeichnern lassen sich einfach exportieren. Die Ähnlichkeitstypen mit quantitativen Bezeichnern sind untypisch für Ontologien, weil dort normalerweise nur qualitative Bezeichner verwendet werden. Aber selbst ein numerischer Bezeichner für einen Ähnlichkeitstyp, etwa 0.3, lässt sich in der Ontologie syntaktisch korrekt darstellen, zum Beispiel durch eine Relation $SIM_{0.3}$.

MultiLingual [MDP01] erzeugt für ein vorhandenes Indexlexikon in einer Sprache ein neues Indexlexikon in einer anderen Sprache bzw. ergänzt das Indexlexikon in der Zielsprache um neue Einträge. Dazu generiert es Einträge zu einem neuen Ähnlichkeitstyp für „Übersetzungs-Ähnlichkeit“. Die

[3] In der neuen Version von WordNet wird für diese Wörter zusätzlich zur Synonymität die Hypernomie verwendet, so dass „ABSTRACTION“ statt „STRONG_SIM“ erzeugt würde.

```
__ABSEITS__[TRANSLATION_SIM];__OFFSIDE__
__ABSEITS__[TRANSLATION_SIM];__REMOTE__
...

__OFFSIDE__[TRANSLATION_SIM];__ABSEITS__
__OFFSIDE__[TRANSLATION_SIM];__ABSEITSSTELLUNG__
...
__REMOTE__[TRANSLATION_SIM];__ENTFERNT__
...
```

Abbildung 5.11: Beispieleinträge im Ähnlichkeitslexikon, die mit **M**ultiLingual neu erzeugt wurden.

Analyse von übersetzten Texten hat ergeben, dass sich Begriffe schlecht eins zu eins in eine andere Sprache übersetzen lassen und deshalb mehrsprachige IEs die Retrievalergebnisse verfälschen. Besser sind Begriffsspektren, die die verschiedenen Bedeutungen eines Worts in der anderen Sprache abdecken.

Am Beispiel des Internet-Lexikons **Leo** [Leo05] in der Version von 2001 und einer Beispielmenge von Fußballbeschreibungstexten wurde dieser Ansatz für die Übersetzung von Englisch und Deutsch vollautomatisiert. Jedes deutsche IE wird dabei als Anfrage an **Leo** gestellt.

Leos Übersetzung ins Englische liefert die IEs, zu denen eine „Übersetzungs-Ähnlichkeit“ gelten soll. Macht man dasselbe für die Rückrichtung, ergeben sich weitere „Übersetzungs-Ähnlichkeiten“. In der Rückrichtung werden allerdings keine neuen IEs mehr erzeugt, sondern nur Einträge zu bereits bekannten deutschen IEs zugelassen.

Abbildung 5.11 zeigt ein Beispiel für das (auszugsweise dargestellte) englische Begriffsspektrum zu „ABSEITS“, nämlich „OFFSIDE“ und „REMOTE“. „OFFSIDE“ wird wieder in „ABSEITS“ und zusätzlich in das ebenfalls bekannte „ABSEITSSTELLUNG“ zurückübersetzt, während „REMOTE“ nur mit „ENTFERNT“ übersetzt wird und deshalb in dieser Richtung nicht mit „ABSEITS“ verbunden wird. Das englische Indexlexikon wird um die beiden IEs „OFFSIDE“ und „REMOTE“ erweitert. Die Grammatikformen können entweder von Hand oder durch ein automatisches Verfahren (siehe S. 72) ergänzt werden. In der neuen Version von **Leo** wird „REMOTE“ auch mit „ABSEITS“ übersetzt, da inzwischen alle Einträge bidirektional sind. Wir halten aber dennoch an unseren Erkenntnissen zu unterschiedlichen Begriffsspektren fest und müssten demnach für eine Wiederholung des Experiments ein anderes Lexikon einsetzen.

Multilingual wurde in einer kleinen Beispieldomäne mit EU-Richtlinien

formativ evaluiert [MDP00]. Das Experiment ist aus 28 englischen Texten und deren professioneller Übersetzung ins Deutsche und Spanische aufgebaut. Die Qualität der Ergebnisse wird anhand von zwei Kriterien gemessen:

1. Ein Fall gilt als gefunden, wenn er unter den ersten zehn Fällen im Retrieval-Ergebnis steht.

2. Ein Fall gilt als Retrieval-Gewinner, wenn er den höchsten Ähnlichkeitswert der Fälle hat, die zu einer Anfrage gefunden wurden.

Beide Kriterien sind wie die *Precision*- und *Recall*-Werte im Information Retrieval abhängig vom betrachteten Textkorpus[Rij79]. Die Bestimmung von *Precision* und *Recall* eignet sich nicht zur Evaluierung von fallbasierten Systemen, da die Retrieval-Ergebnisse ja eine Ordnung der Fallbasis liefern. Im klassischen Information Retrieval gibt es hingegen eine klare Trennung der Texte in eine Ergebnismenge und eine nicht zum Ergebnis gehörige Menge. Diese Trennung kann in einem fallbasierten System durch die Bestimmung einer Schranke für den Ähnlichkeitswert ebenfalls erreicht werden [Len99, S. 138 ff.]. Dann gehören beispielsweise alle Fälle, deren Ähnlichkeitswert zur Anfrage im Intervall $[0.8, 1.0]$ liegt, zur Ergebnismenge dazu, und die Fälle im Intervall $[0, 0.8[$ gehören nicht dazu. Diese Trennung entspricht aber nicht der Intention des Fallbasierten Schließens: Hier ist man ja gerade daran interessiert, die besten Fälle auf einen Blick zu sehen. Steht der Fall, der am besten zur Anfrage passt, an hundertster Stelle, ist dies nicht befriedigend, selbst wenn sein Ähnlichkeitswert über einer bestimmten Schranke liegt. Umgekehrt stört ein unpassender Fall unter den ersten zehn Fällen nur wenig, wenn die anderen neun Fälle bei der Lösung eines Problems wirklich helfen. Deshalb habe ich mich für das Kriterium entschieden, dass ein empirisch passender Fall unter den zehn besten Fällen sein soll.

Nun stellt sich natürlich die Frage, was ein „empirisch passender Fall“ ist. Dies kann sehr subjektiv sein und eigentlich nur durch unabhängige Benutzer festgelegt werden. Bei der Evaluierung von **Multilingual** ist dieses Problem etwas einfacher zu lösen, da ja die Übersetzung der Fälle in zwei andere Sprachen ein objektives Kriterium für die inhaltliche Ähnlichkeit von jeweils drei Texten darstellt.

Zwei verschiedene Verfahren zur Modellierung des Hintergrundwissens wurden auf die Testfallbasis mit den 84 Fällen angewendet:

Tabelle 5.2: Die Sprach-Kombinationen in den vier Testläufen

Testlauf	Sprachen der Fälle	Sprache(n) der Anfragen
A	Deutsch, Englisch, Spanisch	Deutsch, Englisch, Spanisch
B	Deutsch, Englisch	Spanisch
C	Deutsch, Spanisch	Englisch
D	Englisch, Spanisch	Deutsch

Experimentelle Methode 1

Das Indexlexikon besteht aus mehrsprachigen IEs. Alle Übersetzungen und Formen eines Worts werden in einem Text-IE zusammengefasst, zum Beispiel:

```
__POWER__ = { POWER; POWERS; BEFUGNIS; BEFUGNISSE; PODER;
              PODERES}
```

Die Ähnlichkeitstypen des Ähnlichkeitslexikons sind wie im einsprachigen Fallbasierten Schließen für Texte spezifiziert (vgl. das Beispiel auf S. 105).

Experimentelle Methode 2

Das Indexlexikon besteht aus einsprachigen IEs, zum Beispiel:

```
__POWER__ = { POWER; POWERS}
__BEFUGNIS__ = { BEFUGNIS; BEFUGNISSE}
__PODER__ = { PODER; PODERES}
```

Die Ähnlichkeitstypen des Ähnlichkeitslexikons sind um einen speziellen Typ — die „Übersetzungs-Ähnlichkeit“ — mit dem Gewicht $0,5$ ergänzt. Dieser hat für das obige Beispiel folgende Einträge:

```
__BEFUGNIS__[TRANSLATION_SIM];__POWER__
__BEFUGNIS__[TRANSLATION_SIM];__PODER__
__POWER__[TRANSLATION_SIM];__PODER__
__PODER__[TRANSLATION_SIM];__POWER__
```

Vier Testläufe mit je 21 Anfragen in verschiedenen Kombinationen von Sprachen (siehe Tabelle 5.2) wurden durchgeführt, wobei Fälle aus der Fallbasis als Anfragen genommen wurden.

Tabelle 5.3: Die Retrieval-Ergebnisse für Methode 1

Testlauf	beide Übersetzungsfälle wurden gefunden	beide Übersetzungsfälle waren Gewinner
A	17 von 21: 85,95%	13 von 21: 61,90%
B	21 von 21: 100%	17 von 21: 85,95%
C	19 von 21: 90,48%	17 von 21: 85,95%
D	20 von 21: 95,24%	16 von 21: 76,19%

Tabelle 5.4: Die Retrieval-Ergebnisse für Methode 2

Testlauf	beide Übersetzungsfälle wurden gefunden	beide Übersetzungsfälle waren Gewinner
A	21 von 21: 100%	16 von 21: 76,19%
B	21 von 21: 100%	17 von 21: 85,95%
C	21 von 21: 100%	21 von 21: 100%
D	21 von 21: 100%	15 von 21: 71,43%

Die Ergebnisse beider Methoden in den vier Testläufen waren sehr gut, was an der relativ kleinen Testfallbasis liegen mag. Tabelle 5.3 und 5.4 zeigen, dass für beide Methoden durchweg mehr als 85% der übersetzten Fälle gefunden wurden. Die zweite Methode fand 100% der übersetzten Fälle und lieferte damit die besseren Ergebnisse im Testszenario.

Wir danken dem **Leo**-Team von der Technischen Universität München recht herzlich für die freundliche Unterstützung bei **MultiLingual** durch die „Eintrittserlaubnis“ zu **Leo** für unseren Bot.

Kapitel 6

Wissensentwicklung: Erfahrungswissen weiterentwickeln

Erfahrungswissen ist nicht statisch, sondern unterliegt Veränderungen. Aufgabe der Wissensentwicklung ist es, die Weiterentwicklung von Wissen zu ermöglichen, das in Assistenzsystemen dokumentiert ist. Die Fragestellungen in diesem Kapitel sind:

- Kann man den Lebenszyklus von Erfahrungswissen nachbilden?
- Welche technischen Konsequenzen hat ein Life-Cycle-Modell für die Falldaten und die Ähnlichkeitsberechnung in einem fallbasierten System?
- Wie muss die Fallbasis organisiert werden, damit mehrere Autoren die Falldaten verändern können, ohne die Suche zu stören?
- Kann man kollaboratives Lernen durch EM-Systeme unterstützen?
- Kann Erfahrungswissen durch automatische Analyse und Erweiterung von Hintergrundwissen „vertieft“ werden?

Meine Anwendungsbeispiele, die im Folgenden vorgestellt werden, beschränken sich auf zwei Felder der Wissensentwicklung: einmal die Unterstützung menschlicher Lern- und Wissensentwicklungsprozesse sowie zweitens automatische oder besser gesagt halbautomatische Akquisition von Hintergrundwissen. Kapitel 6.1 beschreibt ein Life-Cycle-Modell für Falldaten, die im Laufe der Zeit von der vagen Idee bis zum ausgereiften Fall entwickelt werden. In Kapitel 6.2 stelle ich die Beispielanwendung **TestManager** dazu vor.

Kapitel 6.3 erläutert automatische Verfahren zur Erweiterung und Veränderung eines Ähnlichkeitslexikons, die in **OntoCBR** implementiert sind. Und Kapitel 6.4 beschäftigt sich mit Akquisitionsverfahren für Hintergrundwissen aus Textkorpora in **OntoDigger**.

6.1 Ein Life-Cycle-Modell für Falldaten

Beim Einsatz fallbasierter EM-Systeme über einen längeren Zeitraum, das heißt über mehrere Monate oder Jahre, hat sich gezeigt, dass die Falldaten nichts Statisches sind [Arc04, Nic05]. Das in ihnen gespeicherte Erfahrungswissen entwickelt sich weiter, sei es durch Anwendung des Falls in neuen Situationen oder sei es einfach durch Veränderung der Umwelt, zum Beispiel während des Reifeprozesses eines Produktes. Es ist die Aufgabe der Maintenance (siehe Kapitel 9), dieses neue Erfahrungswissen zu dokumentieren und ausgereifte Fälle an neue Gegebenheiten anzupassen.

Hier in diesem Kapitel geht es um Szenarien, in denen ganz bewusst unreife oder unvollständige Fälle schon in die Fallbasis aufgenommen werden sollen, um sie später wieder aufzugreifen und zu vervollständigen. Das EM-System begleitet die Benutzer während der Entwicklung der Fälle über einen längeren Zeitraum. Deshalb muss das System Mechanismen besitzen, um mit solchen „mitwachsenden“ Fällen umzugehen. Mein Ansatz dazu betrachtet zwei Aufgaben:

- das Retrieval unvollständiger Fälle und
- die Unterstützung beim Weiterschreiben unvollständiger Fälle.

Das Retrieval sollte so gestaltet werden, dass unvollständige Fälle in verschiedenen Reifegraden neben ausgereiften Fällen in der Fallbasis stehen und alle Arten von Fällen im Retrieval gefunden werden können. Zum anderen ist natürlich die Unterstützung der Schreibprozesse selbst ganz wichtig. Dabei gehen Suchen und Weiterschreiben von Falldaten Hand in Hand, um den bisherigen Entwicklungsstand der Fallbasis zu überprüfen und bereits fertiggestellte Fälle als Vorlagen zu nutzen. Das System muss also eine Verschränkung der Prozesse zum Retrieval und zum Verändern der Fallbasis bieten, besonders wenn mehrere Benutzer gleichzeitig damit arbeiten. Die Grundlage für eine Lösung der beiden Aufgaben bilden Fallrevisionen (Der Begriff „Version“ ist schon in Kapitel 5.2 belegt, wo verschiedene Versionen des inhaltlichen Gegenstands des EM-Systems unterschieden werden):

Definition 6.1.1 (Fallrevision, Revision eines Falls)

> Eine *Fallrevision* (*Revision eines Falls*) ist die Momentaufnahme eines Falls zu einem bestimmten Zeitpunkt. Sie wird durch eine Revisionsnummer eindeutig identifiziert. Mehrere Revisionen desselben Falls werden durch aufsteigende Revisionsnummern in eine zeitliche Reihenfolge gebracht.

Es gibt Revisionen, in denen der Fall noch fragmentarisch oder vage beschrieben ist. Eventuell besitzt er noch nicht einmal eine Lösung, sondern nur die Idee eines potentiellen Problems. Eine frühe Fallrevision kann also leere Abschnitte enthalten, wenn eine Fallstruktur, wie in Kapitel 4.3, beschrieben verwendet wird (siehe auch Anwendungsbeispiel in Kapitel 6.2). Der Reifegrad einer Revision wird explizit in einem Attribut erfasst. Leere Abschnitte lassen sich zwar auch automatisch identifizieren, Skizzen einer Lösung oder eines Problems lassen sich hingegen schwer von vollständigen Beschreibungen unterscheiden. Deshalb wird der Reifegrad eines Falles nicht automatisch bestimmt. Mit Hilfe des Attributwerts sehen die Benutzer auf einen Blick, wie weit ein Fall gediehen ist, und können auch selektiv nach Fällen mit einem bestimmten Reifegrad suchen. Das System kann mehrere Revisionen desselben Falls parallel im System verwalten und für verschiedene Sichten zur Verfügung stellen (siehe unten).

Definition 6.1.2 (Lebenszyklus eines Falls, Life-Cycle)

> Der *Lebenszyklus eines Falls* (*Life-Cycle*) spiegelt seine Entwicklung von der Entstehung bis zum Entfernen aus dem System wieder.

Der Lebenszyklus-Gedanke hat drei Auswirkungen auf den Entwurf des Beispielsystems:

- Ein Fall durchläuft verschiedene Revisionen.
- Jeder Fall hat eine eindeutige, persistente Identifikationsnummer.
- Die Ähnlichkeitsfunktion behandelt Fälle mit leeren Abschnitten dergestalt, dass sie nicht gegenüber vollständig ausgefüllten Fällen benachteiligt werden.

Die Vergabe einer eindeutigen Identifikationsnummer folgt der Beobachtung, dass viele Autoren sich die Nummern der Fälle merken, die sie selbst bearbeitet oder häufig gelesen haben. Bei der Erstellung einer neuen Revision wird nur die Revisionsnummer des Falles erhöht. In einer nach Fallnummern

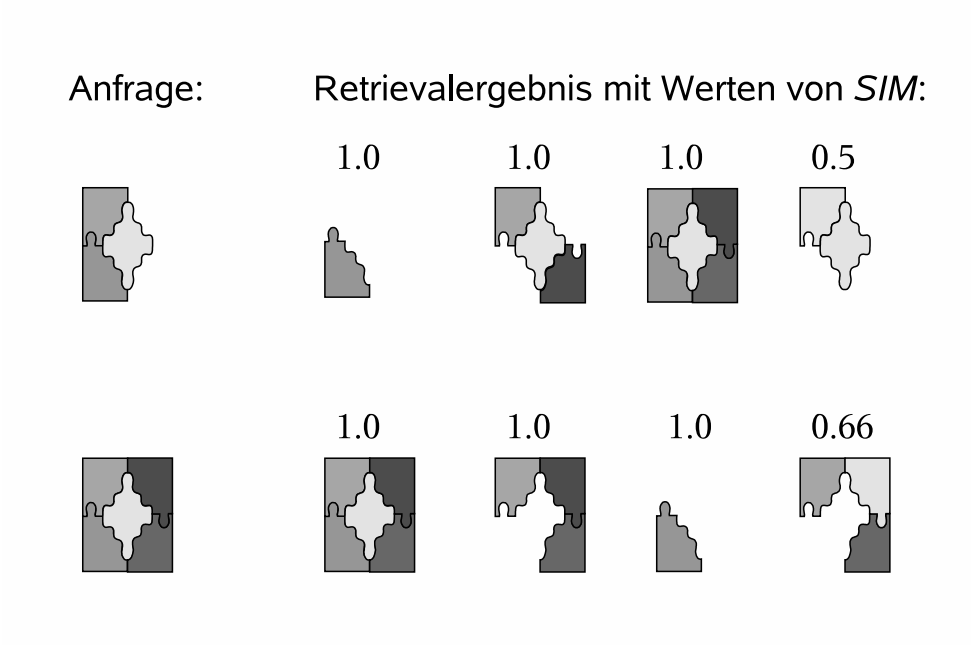


Abbildung 6.1: Vergleich mit vollständigen und unvollständigen Fällen.

sortierten Liste hat der Fall nach wie vor die gleiche Position. Werden Fälle gelöscht, entstehen Lücken in der Fallnummerierung.

Die Ähnlichkeitsfunktion muss sicherstellen, dass Fälle mit leeren Abschnitten genauso gefunden werden können wie vollständige Fälle, die zu einer Anfrage passen. Abbildung 6.1 veranschaulicht dies durch ein symbolisches Puzzle: Jedes Puzzleteil entspricht einem ausgefüllten Abschnitt der Anfrage oder des Falls. Fehlende Puzzleteile entsprechen leeren Abschnitten. In der ersten Zeile werden zu einer unvollständigen Anfrage sowohl Fälle mit weniger Abschnitten identischen Inhalts gefunden, als auch Fälle mit mehr ausgefüllten Abschnitten als die Anfrage. Der letzte Fall dieser Reihe hat in der Mitte ein identisches Puzzlestück, in der linken oberen Ecke jedoch ein stark abweichendes. Deshalb kann dieser Fall nicht den maximalen Ähnlichkeitswert von 1,0 zur Anfrage erreichen. Die zweite Zeile beschreibt die Situation, dass eine vollständig ausgefüllte Anfrage gestellt wird, aber auch teilweise ausgefüllte Fälle gefunden werden. Der vierte Fall wird auch hier schlechter bewertet, da das rechte obere Teil abweicht. Dies bedeutet, dass ein Benutzer die Fälle, die zum höchsten Reifegrad der Anfrage gut passen, auch schon in früheren Stadien gefunden hätte, und dass unvollständige Fälle somit nicht benachteiligt werden.

Zur Umsetzung dieses nichtmonotonen Verhaltens einer Ähnlichkeitsfunk-

tion mussten wir die übliche Vorstellung einer kompositorischen Ähnlichkeit verwerfen. In der Literatur wird anstelle von „Ähnlichkeit“ auch der Begriff der „Nützlichkeit“ verwendet [BR01]. Ein Fall ist nützlich oder akzeptabel, wenn er zur Lösung einer aktuellen Aufgabe beiträgt. Für die Aufgabe, einen Fall weiterzuentwickeln oder einen neuen Fall zu schreiben, sind diejenigen Fälle als Vorlagen nützlich, die Textabschnitte mit ähnlichem Inhalt haben. Um diese zu ermitteln, kann der Fall, der gerade editiert wird, als Anfrage an die Fallbasis gestellt werden. Jeder Textabschnitt der Anfrage wird einzeln mit seinem Partnerabschnitt im Fall verglichen. Dazu muss die Mengendarstellung eines Falls (vgl. S. 40) durch eine Liste von Mengen ersetzt werden. k ist die Anzahl der Abschnitte in der Fallstruktur (vgl. S. 62) und $S^i \subseteq E$ ist die Menge der IEs im i-ten Abschnitt. Ein Fall c kann dann dargestellt werden als:

$$c = [S^1, S^2, ..., S^k].$$

Dieselbe Darstellung ist für eine Anfrage mit k Abschnitten möglich. So kann die Ähnlichkeitsfunktion SIM leere und gefüllte Abschnitte unterscheiden. SIM wird in drei Stufen unterteilt:

1. Die lokalen Ähnlichkeitswerte zwischen IEs (siehe S. 41),
2. die partielle Ähnlichkeit SIM_{part} zwischen zwei Partnerabschnitten S^i in Fall und Anfrage, die die lokalen Werte eines Abschnitts summiert (nicht zu verwechseln mit $SIM_{g,S}$ auf S. 47), und
3. die Gesamtähnlichkeit SIM eines Falls zur Anfrage.

Das heißt, wenn ein Abschnitt in der Anfrage oder im Fall leer ist, nimmt SIM_{part} automatisch den Wert 0 an. Der entscheidende Trick liegt bei der Zusammenfassung der Werte von SIM_{part}. Ein Normierungsfaktor $\frac{1}{\alpha}$ berücksichtigt nur Abschnitte, die auf beiden Seiten gefüllt sind: α ist die Anzahl der IEs aus denjenigen Abschnitten der Anfrage, die im gerade betrachteten Fall ebenfalls gefüllt sind. Kommt dasselbe IE in mehreren Abschnitten vor, kann es mehrfach in die Ähnlichkeitsberechnung eingehen. Die IEs aus den Abschnitten der Anfrage, die einen leeren Partnerabschnitt im Fall haben, werden ignoriert. Die neue Formel für SIM ist eine normierte Summe der partiellen Ähnlichkeiten:

$$SIM(q, c) = \frac{1}{\alpha} \sum_{i=1}^{k} SIM_{part}(S_q^i, S_c^i)$$

Wird die Anfrage in bisher leeren Abschnitten erweitert, kann dies auch zur Verringerung des Ähnlichkeitswerts eines Falls führen (siehe unten). Dieser Effekt ist gewünscht, falls Anfrage und Vergleichsfall sich „auseinanderentwickeln“. Die leeren Abschnitte werden also optimistisch bewertet: Sie gehen nicht negativ in die Gesamtbewertung mit ein, weil angenommen wird, dass sie sich beim Ausfüllen ähnlich zum bereits gefüllten Partnerabschnitt entwickeln. Tun sie dies nicht, wird der Ähnlichkeitswert infolgedessen schlechter.

Eigenschaft

Sei q eine Anfrage mit leeren und ausgefüllten Abschnitten. Sei q' aus q durch Ausfüllen leerer Abschnitte entstanden. Sei c ein Fall, der sowohl zu den in q ausgefüllten Abschnitten als auch zu den neu ausgefüllten Abschnitten in q' jeweils mindestens einen nichtleeren Partnerabschnitt hat. Buchstabe A bezeichne die Summe der partiellen Ähnlichkeitswerte für das originale q und c: $A = \sum_{i=1}^{k} SIM_{part}(S_q^i, S_c^i)$ und α_A den zu q und c gehörigen α-Wert. Analog dazu werden für die erweiterte Anfrage q' die Buchstaben $B = \sum_{i=1}^{k} SIM_{part}(S_{q'}^i, S_c^i)$ und α_B verwendet. Sei weiterhin der neue Anteil der Summe $N = B - A$ und $\alpha_N = \alpha_B - \alpha_A$.

Es gilt:

$$SIM(q', c) < SIM(q, c) \text{ gdw. } \frac{N}{\alpha_N} < SIM(q, c).$$

Beweis:

Zuerst zeige ich die eine Richtung:

$$\frac{N}{\alpha_N} < SIM(q, c) \Rightarrow SIM(q', c) < SIM(q, c).$$

Für $\frac{N}{\alpha_N} < SIM(q, c)$ kann man schreiben: $\frac{N}{\alpha_N} < \frac{A}{\alpha_A}$ oder

$$\frac{N \cdot \alpha_A}{A} < \alpha_N. \quad \text{(I)}$$

Annahme: Es gelte (I) und es sei $SIM(q', c) \geq SIM(q, c)$.

Für $SIM(q', c) \geq SIM(q, c)$ kann man schreiben:

$$\frac{A+N}{\alpha_A+\alpha_N} \geq \frac{A}{\alpha_A}.$$

$$\frac{(A+N)\cdot\alpha_A}{\alpha_A+\alpha_N} \geq A.$$

$$\frac{(A+N)\cdot\alpha_A}{A} \geq \alpha_A + \alpha_N.$$

$$\frac{(A+N)\cdot\alpha_A - A\cdot\alpha_A}{A} \geq \alpha_N.$$

$$\frac{A\cdot\alpha_A + N\cdot\alpha_A - A\cdot\alpha_A}{A} \geq \alpha_N.$$

Das ist ein Widerspruch zu (I)!

Die Gegenrichtung $SIM(q', c) < SIM(q, c) \Rightarrow \frac{N}{\alpha_N} < SIM(q, c)$ gilt analog.

□

Wie man leicht sieht, kann man die Eigenschaft durch dieselbe Argumentation auch für eine Erhöhung des Ähnlichkeitswerts (>) und für einen gleichbleibenden Ähnlichkeitswert zeigen, so dass die Vorstellung eines Puzzles (siehe Abbildung 6.1) damit mathematisch hinterlegt ist.

6.2 Unterstützung bei der Erzeugung von Testfällen durch TestManager

Das Anwendungsbeispiel **TestManager** implementiert das oben beschriebene Modell von Fällen mit Lebenszyklus. **TestManager** ist ein fallbasiertes Assistenzsystem für das funktionale Testen von Software, das in Kooperation mit der Firma PSI AG, Berlin, entstanden ist (siehe dazu auch unseren Workshopbeitrag [MH00]).

In allen Entwicklungsphasen von Software entstehen Testfälle, die am Schluss eine Testspezifikation bilden. Abbildung 6.2 zeigt das V-Modell der Softwareentwicklung nach [PHW94]. Schon bei der Anforderungsanalyse entstehen erste Ideen für Testfälle. Bis zur tatsächlichen Implementierung entstehen in jeder Phase weitere Testfälle, die vor der Auslieferung der Software ausgeführt werden müssen. Die ISO 9001 schreibt die Spezifikation von Testfällen als Teil der Entwicklungsleistung vor. Das Assistenzsystem **TestManager** begleitet die Entwickler in allen Projektphasen. Die systematische Testphase am Ende der Implementierung wird viel einfacher, wenn die Entwickler in allen Entwicklungsphasen Testfälle festgehalten haben. Auch Ideen

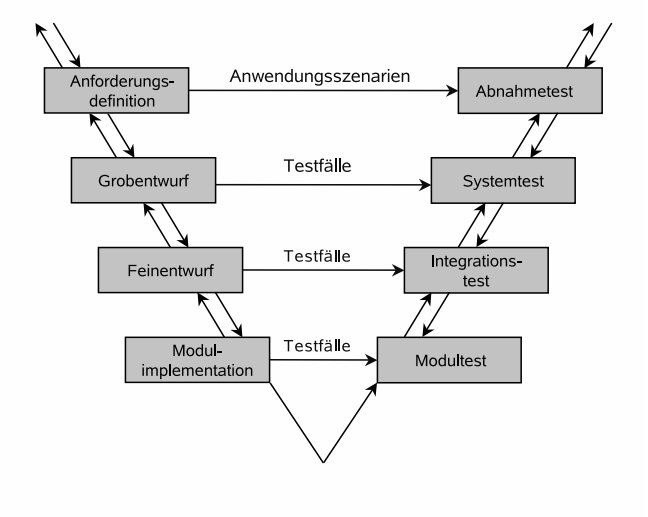


Abbildung 6.2: Das V-Modell der Softwareentwicklung.

für Testfälle sind beim Erstellen einer Testfallspezifikation wertvoll. Sowohl Testfälle aus alten Projekten als auch aktuelle Testfälle helfen als Vorlagen bei der Formulierung neuer Fälle. Mit Hilfe der neuen Fälle lassen sich außerdem Duplikate vermeiden.

Wird anstelle des V-Modells ein anderes Softwareentwicklungsmodell, etwa das Wasserfall- oder das Spiralmodell gewählt, kann der **TestManger** ohne Veränderung übernommen werden.

Ein Fall enthält die Spezifikation eines Testfalls, die aus drei Textabschnitten besteht:

- aus der Beschreibung der Testaufgabe,
- aus Instruktionen, wie der Testfall ausgeführt wird,
- aus dem erwarteten Testergebnis.

Diese drei Abschnitte finden sich natürlich auch in der Fallstruktur wieder. Dazu kommen Attribute für das Software-Modul, zu dem der Testfall gehört, für die Kategorie von Testfällen, zum Beispiel die „Simulation“, und für den Reifegrad des Falls. Ein typischer Lebenszyklus eines Tesfalls ist in Abbildung 6.3 dargestellt.

Die Abstände zwischen den Entwicklungsschritten eines Testfalls können mehrere Monate betragen. Zwischenzeitlich stehen die unvollständigen Testfälle für das Retrieval zur Verfügung. Normalerweise wird als erstes nur die

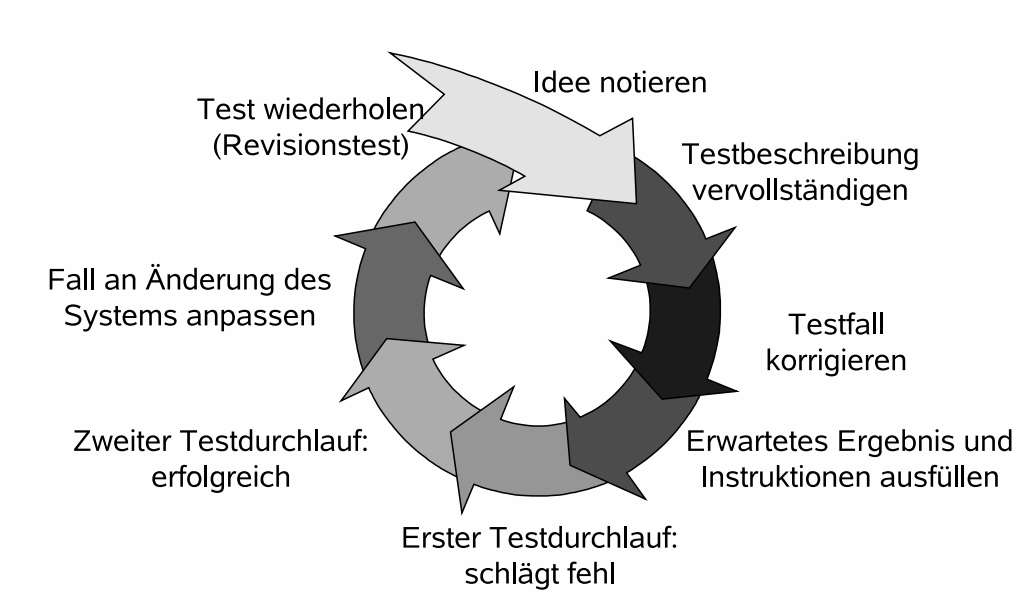


Abbildung 6.3: Ein typischer Lebenszyklus eines Testfalls.

Beschreibung des Testfalls ausgefüllt, später kommen dann das erwartete Ergebnis und konkrete Anweisungen zur Durchführung des Testfalls dazu. Oft entstehen aus einer Testidee später mehrere Testfälle, zum Beispiel wenn das Systemverhalten bei Überschreiten verschiedener Grenzwerte überprüft werden soll. Eine abgeschlossene Testspezifikation beinhaltet viele Überlappungen von einzelnen Testfällen. Jeder Softwareentwickler ist durch die ISO-Norm zur Spezifikation von Testfällen verpflichtet. Deshalb darf jedes Team-Mitglied alle Testfälle lesen und weiterschreiben. Ein Fall kann so im Laufe seines Lebens von mehreren Autoren bearbeitet werden.

Implementierung

Der Prototyp ist wie **SimLex** in C++ implementiert, hat aber statt einfacher CGI-Skripte ein Java-Applet als grafische Benutzerschnittstelle. Das Java-Applet benutzt CGI-Skripte, um die Anfragen vom Apache-Webserver zum CRN-Server durchstellen zu lassen. Wir haben mit der Geschwindigkeit des Java-Applets schlechte Erfahrungen gemacht und greifen deshalb in den folgenden Anwendungskapiteln wieder auf die einfache CGI-Technik zurück.

Der Aufbau des CRNs in Vorbereitung des Retrievals dauert einige Minuten. Ein Autor, der gerade einen Fall bearbeitet und vielleicht mehrfach zwischenspeichert, würde bei einem Neuaufbau des CRNs alle anderen Be-

nutzer stören und sie mit ihren Anfragen viel zu lange warten lassen. Deshalb verzögern wir den Neuaufbau des CRNs, bis ihn jemand per Mausklick explizit aufruft oder bis ein zeitgesteuerter Prozess dies zum Beispiel in der Nacht erledigt. Schreibkonflikte an einem einzigen Fall werden mit Hilfe der Revisionsnummern entdeckt und zur Lösung an den Benutzer kommuniziert, der die Änderungen eines anderen überschreiben wollte, ohne die neueste Revision zu kennen. Wegen der Zeitverzögerung des Netzaufbaus benutzt der **TestManager** manchmal zwei Revisionen eines Falls: eine hochaktuelle, die beim Editieren benötigt wird, und eine zweite, die ins CRN eingebunden ist. Diese beiden Sichten des Systems auf die Fallbasis werden wiedervereinigt, sobald das CRN neu aufgebaut wird.

Die Fallbasis des **TestManagers** hat ca. 1400 Testfällen aus einem Projekt zu Gasleitsystemen. Ein Indexlexikon und ein Ähnlichkeitslexikon aus einem unserer früheren Projekte außerhalb dieser Arbeit konnte wiederverwendet werden, wurde aber um fachspezifische Begriffe und neue Ähnlichkeitswerte aus den Fällen und aus einer Softwarespezifikation zu Gasleitsystemen ergänzt. Wir bedanken uns sehr herzlich bei der PSI AG für die zur Verfügung gestellten Daten und Dokumente im Rahmen des Verbundprojekts „Werkzeuge und Verfahren zur Modellierung technologischer Prozesse“.

6.3 Veränderung des Ähnlichkeitsmodells mit OntoCBR

Der Import und Export von IE- und Ähnlichkeitslexika in Ontologien (siehe Kapitel 5.4) hat mich bei der Entwicklung von **OntoCBR** dazu angeregt, über Eigenschaften von Ähnlichkeitstypen zur Veränderung des Ähnlichkeitslexikons nachzudenken. In der Ontologiesprache OWL ist es möglich, den Beziehungen zwischen Konzepten eine oder mehrere der folgenden vier Eigenschaften zuzuweisen:

- `owl:TransitiveProperty` für die Transitivität,
- `owl:SymmetricProperty` für die Symmetrie,
- `owl:FunctionalProperty` für die Eigenschaft, dass es sich bei der Relation um eine Funktion handelt, das heißt $[e, e'], [e, e''] \in R \rightarrow e' = e''$ oder, anders ausgedrückt, R ist beschreibbar durch $f(e) = e'$,
- `owl:InverseFunctionalProperty` für die Eigenschaft, dass die zu R inverse Relation R^{-1} eine Funktion ist, das heißt $[e', e], [e'', e] \in R \rightarrow e' = e''$.

In der Ontologie-Gemeinde haben sich bereits einzelne Konventionen bezüglich der Eigenschaften herausgebildet. Zum Beispiel ist es mittlerweile Konsens, die „IS_A“-Relation als Halbordnung zu betrachten, so dass man wegen der Transitivität die Darstellung mit Hilfe eines Hasse-Diagramms stark vereinfachen kann (siehe S. 108). Bei der „IS_PART“-Beziehung (Teil-Ganzes) gibt es Befürworter und Gegner der Transitivität. Ein prominentes Gegenbeispiel kommt aus einer Ontologie über Orthopädie, in der gilt (Darstellung in F-Logik) $finger[is\_part- >> hand]$ und $hand[is\_part- >> arm]$, aber nicht gilt $finger[is\_part- >> arm]$. Die Vergabe von Relationeneigenschaften kann auch die Modellierung eines Ähnlichkeitslexikons vereinfachen. Dabei bieten Ontologietools wie Protégé eine gute Hilfestellung. Das Ziel der Veränderungen und Erweiterungen ist es, die Retrievalergebnisse zu verbessern.

Leider fehlen empirische Werte für die Spezifikation der lokalen Ähnlichkeitsfunktion *sim*. Für erste Ansätze zu automatisch gewonnenen Werten verweise ich auf Kapitel 5.4, 6.4 und 10. In dieser Arbeit wird der größte Teil der Werte noch manuell eingepflegt, weshalb ich in Kapitel 4.3 das Hilfsmittel der Ähnlichkeitstypen eingeführt habe. So wird der numerische Ähnlichkeitswert, der einem Paar von IEs zugewiesen wird, empirisch motiviert durch die semantische Beziehung, in der die beiden IEs stehen. Kann man den Ähnlichkeitstypen Eigenschaften zuweisen, ist zu erwarten, dass diese zur Verbesserung der Retrieval-Ergebnisse beitragen. Eine systematische Überprüfung dieser Hypothese ist äußerst schwierig, da empirische Daten zur Güte von Retrievalergebnissen im laufenden Betrieb nur mit hohem manuellem Aufwand zu erzeugen sind (vgl. die Diskussion auf S. 79). Für ein Experiment im Labor verweise ich auf zukünftige Arbeiten. Es ist aber jetzt schon plausibel, dass die im Folgenden beschriebenen Veränderungen das Ähnlichkeitslexikon zumindest im Hinblick auf die Austauschbarkeit mit Domänenontologien verbessern.

Operatoren für die Relationen

Um die Eigenschaften von Ähnlichkeitstypen einzuarbeiten, bilde ich mit Hilfe von Operatoren aus dem originalen Ähnlichkeitslexikon $\mathfrak{S}$ ein neues Lexikon. Die Operatoren erweitern entweder einen bestehenden Ähnlichkeitstyp, so dass er bestimmte Eigenschaften (Transitivität, Symmetrie oder Inverse zu einer Partnerrelation) erfüllt, oder sie ergänzen das Lexikon um abgeleitete Relationen.

Definition 6.3.1 (Operatoren für Relationen)

Ein *Operator* $op : S \mapsto S'$ *für eine Relation* $S \subseteq E \times E$ erzeugt eine neue Relation S'.

Ich behandle im Folgenden nur einige Spezialfälle von Operatoren, für weitere verweise ich auf zukünftige Arbeiten:

Identität

$$id(S) = S$$

Inverse

$$inv(S) = S^{-1}$$

Transitive Hülle

Die transitive Hülle $< S >$ einer Relation S ist die kleinste transitive Relation, die S enthält.

$$t(S) =< S >$$

Symmetrische Hülle

$$s(S) = S \cup S^{-1}$$

Erweiterung um die Inverse einer Partnerrelation

Sei $f : \mathfrak{S} \rightarrow \mathfrak{S}$ eine Funktion, die einer Relation eine andere Relation als empirische Partnerrelation zuweist.

$$e(S) = S \cup f(S)^{-1}$$

Induzierte Geschwisterrelation

$$b(S) = (S^{-1} \circ S) \setminus I$$

I ist die Identitätsrelation; $b(S)$ hat also folgende Elemente:
$b(S) = \{[b, c] \mid b \neq c, \exists a \in E : [a, b], [a, c] \in S\}$

Das Ziel der Anwendungen von t, s und e ist es, Relationen mit bestimmten Eigenschaften zu erzeugen. t erweitert beispielsweise eine Relation mit dem Bezeichner „ABSTRACTION“ so, dass eine transitive Relation entsteht. e kann auf Paare von Relationen, zum Beispiel auf „IS_PART“ und „HAS_PART“, angewendet werden, um aus einem empirischen Paar ein tatsächliches Paar von Inversen zu erzeugen. Dazu reicht es, dass die Funktion f selbstreflexiv ist.

Definition 6.3.2 (Selbstreflexivität von f)

f heißt *selbstreflexiv*, wenn gilt:

$$f(S) = R \text{ gdw. } f(R) = S.$$

Ist f selbstreflexiv, entstehen durch Anwendung von e auf beide Partner S und $f(S)$ zwei zueinander inverse Relationen.

Folgerung 1 (Inversenbildung)

Ist f selbstreflexiv, so gilt:

$$e(S) = [e(f(S))]^{-1}$$

Beweis

$$\begin{aligned}
[e(f(S))]^{-1} &= [f(S) \cup [f(f(S))]^{-1}]^{-1} \text{ nach der Definition von } e \\
&= [f(S) \cup S^{-1}]^{-1} \text{ wegen der Selbstreflexivität von } f \\
&= [f(S)]^{-1} \cup S \\
&= e(S)
\end{aligned}$$

□

Es können mehrere Operatoren miteinander verkettet werden. Eine interessante Frage ist dabei, ob die erzeugten Eigenschaften einer Relation sich automatisch auch auf das Resultat einer Kette von Operatoren übertragen. Dazu untersuche ich im Folgenden einige Spezialfälle von Operatorenketten.

Definition 6.3.3 (Endliche Worte über Operatoren)

> Über dem Alphabet der Symbole $OP = \{\text{id, inv, t, s, e, b}\}$ werden *endliche Worte* w ϵ OP^* gebildet:
>
> $$\text{w} = a_1 a_2 a_3 ... a_n \quad \text{mit a}_i \ \epsilon \ OP$$
>
> Jedem Wort w ϵ OP^* ist ein Operator $\mu_w = \omega : \mathfrak{S} \rightarrow \mathfrak{S}$ zugeordnet. Die Operatoren eines Wortes werden von links miteinander verkettet und auf eine Relation angewendet:
>
> $$\omega(S) = a_1 \circ a_2 \circ ... \circ a_n(S) = a_n(...(a_1(S))...)$$

Der Benutzer legt fest, welche Ketten von Operatoren auf welche Relationen angewendet werden. Dazu stehen nur folgende acht Spezialfälle zur Auswahl, die ich im Folgenden näher untersuchen möchte:

$$\{te, tse, se, e, bte, btse, bse, be\}$$

Diese werden den Relationen eines Ähnlichkeitslexikons wie folgt zugeordnet und darauf angewendet:

Definition 6.3.4 (Erweiterung eines Ähnlichkeitslexikons)

> Seien $\mathfrak{S}$, $\mathfrak{S}'$ Ähnlichkeitslexika, das heißt Mengen von Relationen $S \subseteq E \times E$.
>
> **Spezifikation von f (für den Operator e, siehe S. 93)**
>
> > Sei $\mathfrak{S}_\mathcal{P} \subseteq \mathfrak{S}$ die Menge der Partnerrelationen und sei eine Partnerfunktion $pf : \mathfrak{S}_\mathcal{P} \rightarrow \mathfrak{S}_\mathcal{P}$ vom Benutzer angegeben. Damit wird die Funktion f wie folgt spezifiziert:
> >
> > $$f(S) = \begin{cases} pf(S), & falls\ S\ \epsilon\ \mathfrak{S}_\mathcal{P} \\ S^{-1}, & sonst. \end{cases}$$

Angabe der Operatorenkette ω_S

Sei $todo : \mathfrak{S} \rightarrow \{te, tse, se, e\}$ vom Benutzer angegeben. Abkürzend verwende ich $\omega_S = todo(S)$ und h_S für die Hüllenoperatoren im ersten Teil von ω_S:

$$h_S = \begin{cases} t\ , & falls\ \omega_S = te \\ ts, & falls\ \omega_S = tse \\ s\ , & falls\ \omega_S = se \\ id, & falls\ \omega_S = e. \end{cases}$$

Auswahl der Elternrelationen

Sei die Menge der Elternrelationen $\mathfrak{S}_\mathcal{E} \subseteq \mathfrak{S}$ vom Benutzer angegeben.

Anwendung der spezifizierten Operatorenketten

Der Operator $Erw : \mathfrak{S} \mapsto \mathfrak{S}'$ ist definiert als:

$$\begin{aligned} Erw(\mathfrak{S}) = \quad & \{\omega_S(S) \mid S\ \epsilon\ \mathfrak{S}\} \quad \cup \\ & \{b\omega_S(S) \mid S\ \epsilon\ \mathfrak{S}_\mathcal{E}\} \end{aligned}$$

Die Partnerfunktion pf weist jeder Relation aus der Menge $\mathfrak{S}_\mathcal{P}$ eine Partnerrelation zu. f ist mit pf und der eigenen Inversen für jede Relation $S\ \epsilon\ \mathfrak{S}$ definiert. ω_S verkettet e mit h_S. h_S gibt an, ob aus S eine transitive (t), symmetrische (s) oder transitive und symmetrische (ts) Relation entstehen soll. id bedeutet, dass die Relation unverändert bleibt. Dies schließt nicht aus, dass S bereits transitiv oder symmetrisch ist. Die Menge der Elternrelationen $\mathfrak{S}_\mathcal{E}$ bestimmt, für welche Relation der Operator b angewendet werden soll, um eine zweite Relation aus ihr zu erzeugen.

Für den Benutzer bedeutet dies, dass er eine Relation empirisch beispielsweise als transitiv betrachtet und aus seiner Sicht „fehlende“ Einträge im Ähnlichkeitslexikon ergänzen lassen möchte. Ein praktisches Beispiel für eine Spezifikation der Veränderug des Ähnlichkeitslexikons ist in Tabelle 6.1 auf S. 105 abgebildet.

Definition 6.3.5 (Konsistenz von ω_S mit f)

ω_S heißt *konsistent* mit f, wenn gilt:

$$\omega_S = \omega_{f(S)}$$

Im Folgenden zeige ich, dass Erw Relationen mit den in ω_S vorgegebenen Eigenschaften (Transitivität, Symmetrie oder Inverse zu einer Partnerrelation) erzeugt, wenn der Benutzer pf selbstreflexiv und ω_S konsistent mit f modelliert hat.

Satz (Eigenschaften der durch ω_S erzeugten Relationen)

(i) $\omega_S \in \{\boldsymbol{t}e, \boldsymbol{t}se\} \Rightarrow \omega_S(S)$ ist transitiv

(ii) $\omega_S \in \{t\boldsymbol{s}e, \boldsymbol{s}e\} \Rightarrow \omega_S(S)$ ist symmetrisch

(iii) Ist $S \in \mathfrak{S}_{\mathcal{P}}$, f selbstreflexiv und ω_S konsistent mit f, dann gilt:
$\omega_S(S) = (\omega_{f(S)}(f(S)))^{-1}$

Lemma 1

(i) $t \circ inv = inv \circ t$

(ii) $s \circ inv = inv \circ s$

(iii) $ts \circ inv = inv \circ ts$

Beweis

ad (i):

$$\begin{aligned}(x,y) \in t(R) &\\ &\Leftrightarrow \exists \text{ Pfad in } R \text{ von } x \text{ nach } y\\ &\Leftrightarrow \exists \text{ Pfad in } inv(R) \text{ von } y \text{ nach } x\\ &\Leftrightarrow (y,x) \in (t \circ inv)(R)\end{aligned}$$

ad (ii): wie man leicht sieht

ad (iii): folgt aus (i) und (ii)

$$ts \circ inv = t \circ (s \circ inv) = t \circ inv \circ s = inv \circ (t \circ s)$$

□

Beweis des Satzes

ad (i) und (ii): wie man leicht sieht

ad (iii):

$$\begin{aligned}
\omega_S(S) &= h_S \circ e(S) \\
&= h_S \circ (e(f(S)))^{-1} \text{ wegen Folgerung 1} \\
&= h_{f(S)} \circ inv \circ e(f(S)) \text{ weil } \omega_S \text{ konsistent mit } f \\
&= inv(h_{f(S)} \circ e(f(S))) \text{ wegen Lemma 1} \\
&= (\omega_{f(S)}(f(S)))^{-1}
\end{aligned}$$

□

Die gerade gezeigten Eigenschaften sind tatsächlich von der Reihenfolge der Operatoren abhängig:

- $e \circ t$ liefert möglicherweise eine kleinere Relation als $t \circ e$, die nicht transitiv ist. Ein Beispiel dafür liefern

  $S = \{[SCHUBLADE, PAPIERFACH]\}$
  $f(S) = \{[DRUCKER, PAPIERFACH]\}$
  mit
  $(e \circ t)(S) = \{[SCHUBLADE, PAPIERFACH], [PAPIERFACH, DRUCKER]\}$
  und
  $(t \circ e)(S) = \{[SCHUBLADE, PAPIERFACH], [PAPIERFACH, DRUCKER], [SCHUBLADE, DRUCKER]\}.$

- $s \circ t$ ist ein Spezialfall von $e \circ t$, bei dem $f(S)$ gerade S selbst ist. Also kann es auch hier vorkommen, dass $s \circ t$ ein kleineres Ergebnis liefert als $t \circ s$. Zur Illustration kann man sich eine Relation $R = S \cup f(S)$ aus obigem Beispiel konstruieren und auf R dann $s \circ t$ und $t \circ s$ anwenden.

- $e \circ s$ ist zwar semantisch unsinnig, weil eine symmetrische Relation keine andere Inverse als sich selbst benötigt, wäre aber rein syntaktisch natürlich möglich. Das Ergebnis dieser Kette hängt davon ab, wie $f(s(S))$ bestimmt wird. f ist ja nur für die Anwendung von e auf die Originalrelation konzipiert (vergleiche S. 101). Wird zum Beispiel das originale $f(S)$ als $f(s(S))$ verwendet, kann die Symmetrie des Ergebnisses dadurch verletzt werden, dass $s(S)$ zufällig in $\mathfrak{S}_{\mathcal{P}}$ ist.

Problem der Verschmelzung von Relationen

Bei der Anwendung von Erw auf eine Menge von Relationen kann es passieren, dass mehrere mathematisch identische Relationen aus verschiedenen Urbildern entstehen. In der Theorie verschmelzen sie zu einem Element von $Erw(\mathfrak{S})$. In der Praxis möchte ich sie aber nach wie vor unterscheiden, da eine so gravierende Modellierungs-Entscheidung wie das Löschen einer Relation nicht der Seiteneffekt eines Algorithmus sein soll, sondern vom Benutzer getroffen werden muss.

Eine einfache Lösung dieses Problems ist die Verwendung der Bezeichner der Ähnlichkeitstypen als persistente Namen. Ich nutze die Listenschreibweise für Mengen von Relationen. Der Name einer Relation ist dann durch ihren Positionsindex in der Liste repräsentiert.

Notation der Ähnlichkeitslexika als Listen

Die Elemente eines vom Benutzer angegebenen Ähnlichkeitslexikons $\mathfrak{S} = \{S_1, S_2, ..., S_m\}$ bilden eine Liste:

$$\mathfrak{S}_0 = [S_1, S_2, ..., S_n].$$

Das durch Anwendung von Operatorenketten erzeugte Ähnlichkeitslexikon $Erw(\mathfrak{S})$ wird ebenfalls in einer Liste dargestellt:

$$\mathfrak{S}_1 = [S'_1, S'_2, ..., S'_n, S_{n+1}, ..., S_m]$$
mit $n \leq m$ und $S'_i = \omega_{S_i}(S_i)$

Die Relationen, die durch Anwendung von ω_S entstanden sind, haben also dieselbe Position in $\mathfrak{S}_1$ wie ihr Urbild in $\mathfrak{S}_0$. Die Relationen, die durch Anwendung von b entstanden sind, bekommen einen neuen Index, also einen neuen Namen. Die mathematisch identischen Relationen in $\mathfrak{S}_1$ sind mit Hilfe ihres Positionsindexes unterscheidbar. Die Gewichtungsfunktion g arbeitet in der Praxis auch auf benannten Relationen. $\mathfrak{S}_1$ lässt sich leicht wieder in die „namenlose“ Mengendarstellung überführen:

Bemerkung

Sind α und β Listen von Relationen, dann kann man schreiben:

$$Erw(\mathfrak{S}) = \{S \mid \exists \alpha, \beta : \mathfrak{S}_1 = [\alpha S \beta]\}$$

Abgeschlossenheit

Ist der Operator Erw abgeschlossen, bedeutet das, dass nur bei der ersten Anwendung von Erw Veränderungen am Ähnlichkeitslexikon erzielt werden können. Eine nochmalige Anwendung desselben Operators erzeugt nur noch eine Kopie des Ähnlichkeitslexikons: $Erw(Erw(\mathfrak{S})) = Erw(\mathfrak{S})$. Diese Eigenschaft ist in der Praxis sehr erstrebenswert, weil der Algorithmus zur Veränderung des Ähnlichkeitslexikons dann terminiert. Er muss genau einmal durchlaufen werden.

Mit der derzeitigen Definition der Funktion f gilt dies nicht: Eine durch Erw neu erzeugte Relation kann zufällig identisch mit einer Relation aus dem originalen $\mathfrak{S}$ sein, die einen anderen Positionsindex hat: $\mathfrak{S}_1[i] = \mathfrak{S}_0[j]$ mit $i \neq j$. Diese Koinzidenz mit einer „fremden“ Relation führt dazu, dass $\mathfrak{S}_1[i]$ auch deren Partner zugewiesen bekommt, so dass $e(\mathfrak{S}_1[i])$ eventuell neue Ergebnisse liefert. Damit ist Erw nicht abgeschlossen.

Um die Abgeschlossenheit von Erw zu erreichen, verfeinere ich die Funktion f.

Definition 6.3.6 (Verfeinerung von f zu $\hat{f}$ unter Berücksichtigung des Positionsindexes)

Die Funktion $\hat{f} : \mathfrak{S} \times \mathbb{N} \rightarrow \mathfrak{S}$ ist definiert als:

$$\hat{f}(S,i) = \begin{cases} f(S) & , \quad falls\ \mathfrak{S}_0[i] = S \\ \omega_R(f(R)), & falls\ \mathfrak{S}_1[i] = S, i \leq m, \mathfrak{S}_0[i] = R \\ S^{-1} & , \quad sonst. \end{cases}$$

Diese neue Definition vermeidet Partnerwechsel und garantiert, dass die Ergebnisse des Operators b immer nur ihre eigene Inverse als Partnerrelation zugewiesen bekommen. Wie man leicht sieht, ist $\hat{f}$ selbstreflexiv, wenn f selbstreflexiv ist, da die Positionen der Partnerschaften sich von $\mathfrak{S}_0$ zu $\mathfrak{S}_1$ nicht verändern. ω_S ist aus demselben Grund auch konsistent mit $\hat{f}$.

Satz (Abgeschlossenheit von ω_S)

Benutzt der Operator e eine selbstreflexive Funktion $\hat{f}$ wie in Def. 6.3.6 und ist ω_S konsistent mit $\hat{f}$, so gilt die Abgeschlossenheit von ω_S:

$$\omega_S(\omega_S(S)) = \omega_S(S).$$

Beweis des Satzes

Der Beweis besteht aus zwei Teilen:

I $e(\omega_S(S)) = \omega_S S$

II $h_S(\omega_S(S)) = \omega_S(S)$

ad I:

$$\begin{aligned} eh_Se(S) &= h_Se(S) \cup (\hat{f}(h_Se(S)))^{-1} \\ &= h_Se(S) \cup (h_Se(\hat{f}(S)))^{-1} \text{ nach der Definition von } \hat{f} \\ &= h_Se(S) \cup h_S((e(\hat{f}(S)))^{-1}) \text{ wegen Lemma 1} \\ &= h_Se(S) \cup h_Se(S) \text{ wegen Folgerung 1} \end{aligned}$$

ad II: (nach der Definition von h_S in Def. 6.3.4)

(i) $t^2 = t$

(ii) $(ts)^2 = ts$

(iii) $s^2 = s$

(iv) $id^2 = id$

ad II (i), (iii) und (iv): wie man leicht sieht

ad II (ii) $ts = ts \circ ts$:

Wegen $tt = t$ genügt es zu zeigen, dass $sts = ts$.

$$\begin{aligned} (x, y) \,\epsilon\, t(s(R)) & \\ &\Leftrightarrow \exists \text{ Pfad in } s(R) \text{ von } x \text{ nach } y \\ &\Leftrightarrow \exists \text{ Pfad in } s(R) \text{ von } y \text{ nach } x \\ &\Leftrightarrow (y, x) \,\epsilon\, t(s(R)) \end{aligned}$$

□

Für die Abgeschlossenheit von Erw sind vier Fallunterscheidungen zu betrachten: (i) $\omega_S(\omega_S(S))$ habe ich bereits untersucht, (ii) $\omega_S(b \circ \omega_S(S))$ ist per definitionem von ω_S gleich $b \circ \omega_S(S)$. (iii) $b(b(S))$ diskutiere ich in Folgenden etwas ausführlicher, und (iv) $b(\omega_S(\omega_S(S))) = b(\omega_S(S))$ folgt aus der Abgeschlossenheit von ω_S.

Kaskaden von Geschwisterableitungen $b^k(S)$ schließe ich definitorisch aus, da $\mathfrak{S}_\mathcal{E}$ eine Teilmenge des originalen Ähnlichkeitslexikons $\mathfrak{S}$ ist und nicht während der Anwendung der Operatoren erweitert wird. Das heißt, dass bereits nach der ersten Anwendung von Erw alle Geschwisterrelationen abgeleitet sind. Dies ist empirisch motivitiert, da eine Geschwisterrelation „zweiter Generation“ nur Halbgeschwister bezüglich der ersten Generation miteinander verbindet. Die einer „dritten Generation“ erzeugt dann die Halbgeschwister der Halbgeschwister bezüglich der ersten Generation usw. Es entstehen also größere Familienverbünde, bei denen die Semantik nicht mehr klar ist. Ist ein Software-Modul beispielsweise Client eines Webservers und spielt gleichzeitig die Rolle eines Datenbank-Clients, so kann es bei bestimmten Problemen nützlich sein, andere Web-Clients oder Datenbank-Clients zu betrachten. Aber nicht jeder Datenbank-Client hat etwas mit Web-Clients gemeinsam und umgekehrt:

$$
\begin{aligned}
S = \quad & \{[MEIN\_MODUL, CGI\text{-}SKRIPT], \\
& [MEIN\_MODUL, DB\text{-}CLIENT], \\
& [GRUSSKARTE, CGI\text{-}SKRIPT], \\
& [PRODUKT\text{-}REPORT, DB\text{-}CLIENT]\} \\
b(S) = \quad & \{[MEIN\_MODUL, GRUSSKARTE], \\
& [MEIN\_MODUL, PRODUKT\text{-}REPORT]\} \\
b(b(S)) = \quad & \{[GRUSSKARTE, PRODUKT\text{-}REPORT]\}
\end{aligned}
$$

Praktische Umsetzung

Die Operatoren sind in Form von **Perl**-Skripten implementiert, die ein Ähnlichkeitslexikon in die **OWL**-Notation transformieren. Aus Implementierungsgründen wird bei der Rücktransformation nur der Teil der neuen Relationen abgespeichert, für den schon die aggregierende Funktion $aggr$ angewendet wurde: Bei Paaren von IEs, die Element mehrerer Relationen sind, wird eine Relation ausgewählt, aus der der Wert der lokalen Ähnlichkeitsfunktion sim bestimmt wird, die anderen Relationen werden ignoriert.

Dies implementiert die Spezifikation von *aggr* in dieser Arbeit als Selektions-Funktion über alte und neue Relationen (vgl. den Kommentar zu Def. 4.3.10 auf S. 47 f.):

$$aggr(sim_{S_1}(e',e''),...,sim_{S_n}(e',e''),sim_{S_{n+1}}(e',e''),...,sim_{S_{n+m}}(e',e'')) =$$

$$\begin{cases} 0 & , \quad sim_{S_j}(e',e'') = \perp, j = 1... \ n+m \\ sim_{S_i}(e',e''), & sim_{S_i} \neq \perp, S_i \succ S_j \ mit \ sim_{S_j} \neq \perp, j = 1... \ n+m. \end{cases}$$

Die Präferenzordnung der Relationen wird nach zwei Kriterien bestimmt:

1. nach der Herkunft des Eintrags im Ähnlichkeitslexikon und
2. nach der Priorität des zugeordneten Ähnlichkeitstyps.

Das zweite Kriterium ist dem ersten Kriterium bis auf eine Ausnahme nachgeordnet, das heißt dass alte Relationen normalerweise bevorzugt werden, weil sie manuell modelliert oder überprüft wurden. Neue Relationen sind hingegen automatisch durch Anwenden von Operatoren entstanden. Die Ausnahme bilden Einträge mit „MEDIUM_SIM“ als Ähnlichkeitstyp. „MEDIUM_SIM“ ist in der Praxis der Defaultwert für den Ähnlichkeitstyp, der einem Paar von IEs zugeordnet wird, wenn unter großem Zeitdruck modelliert wird. Die alte Relation „MEDIUM_SIM“ hat eine niedrigere Priorität als die neu erzeugten Relationen. Die Präferenzordnung der Relationen lässt sich erweitern, wenn andere automatische Quellen hinzukommen (siehe Kapitel 6.4 und 11).

Wie man leicht sieht, kann die Ontologie durch erneute Anwendung der Operatoren reproduziert werden und die Selektion durch *aggr* außer in Bezug auf die gelöschten „MEDIUM_SIM“-Einträge wieder rückgängig gemacht werden. Durch die Abgeschlossenheit der Operatoren entsteht auch bei mehrmaliger Transformation und Rücktransformation immer wieder das gleiche Ähnlichkeitslexikon bzw. die gleiche Ontologie. Manuelle Änderungen am Modell sind natürlich nach wie vor möglich und können auch die automatisch erzeugten Lexikoneinträge verändern.

Tabelle 6.1 zeigt eine Beispielspezifikation der Ähnlichkeitstypen eines Ähnlichkeitslexikons aus dem **ExperienceBook II**-Projekt (zur Beschreibung des Projekts siehe Kapitel 8.1). Die Gewichte der Ähnlichkeitstypen sind empirisch begründet. So sollen zum Beispiel zu einer Anfrage, die „WinOnCD“ enthält, auch Fälle über CD-Brenner gefunden werden, weil

„WINONCD [IS_MEANS_FOR] CD-BRENNER“

Tabelle 6.1: Spezifikation der Ähnlichkeitstypen im **ExperienceBook II**

Bezeichner	Gewicht g	h_S	pf	$\mathfrak{S}_\mathcal{E}$, Bez.
NO_SIMILARITY	0,0	id	-	-
ID	1,0	id	-	-
ABSTRACTION	0,3	t	SPECIFICATION	+, BROTH
SPECIFICATION	0,7	t	ABSTRACTION	-
HAS_PART	0,5	t	IS_PART	+, PBROTH
IS_PART	0,5	t	HAS_PART	-
USES_AS_MEANS	0,3	t	IS_MEANS_FOR	+, MBROTH
IS_MEANS_FOR	0,7	t	USES_AS_MEANS	-
STRONG_SIM	0,75	id	-	-
WEAK_SIM	0,25	id	-	-
MEDIUM_SIM	0,5	id	-	-
BROTH	0,5	id	-	-
PBROTH	0,5	id	-	-
MBROTH	0,5	id	-	-

gilt und der Benutzer „IS_MEANS_FOR“ hoch einschätzt. Die umgekehrte Richtung „USES_AS_MEANS“ wird vom Benutzer als weniger interessant bewertet, das heißt bei einer Anfrage zu CD-Brennern würden die WinOnCD-Fälle nur wenig Aktivierung erhalten durch

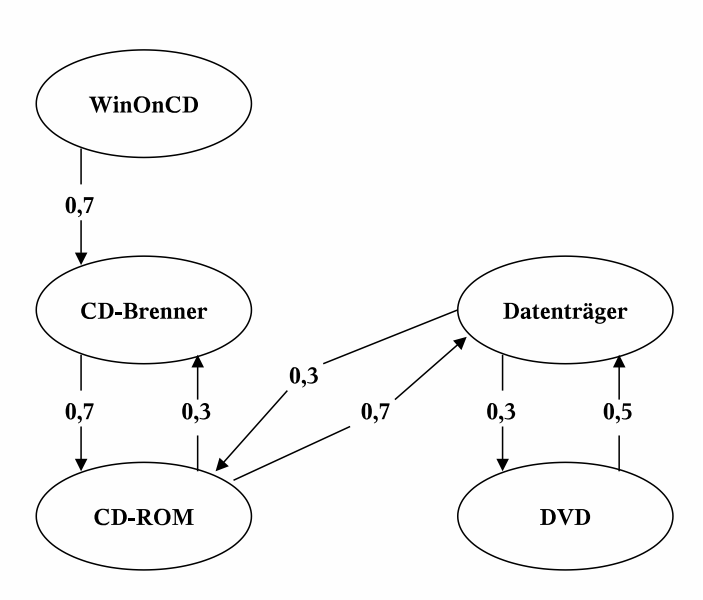
„CD-BRENNER [USES_AS_MEANS] WINONCD“.

Die Priorität der Ähnlichkeitstypen für das zweite Kriterium ist in absteigender Reihenfolge der Tabellenzeilen gegeben. So kann das höchstpriorisierte „NO_SIMILARITY“zum Beispiel bei fehlender Abwärtskompatibilität von Software die Übernahme einer „BROTH“-Beziehung in einer Richtung für ein Paar von IEs verhindern:

„NETSCAPE4 [NO_SIMILARITY] NETSCAPE3“ annulliert
„NETSCAPE4 [BROTH] NETSCAPE3“.

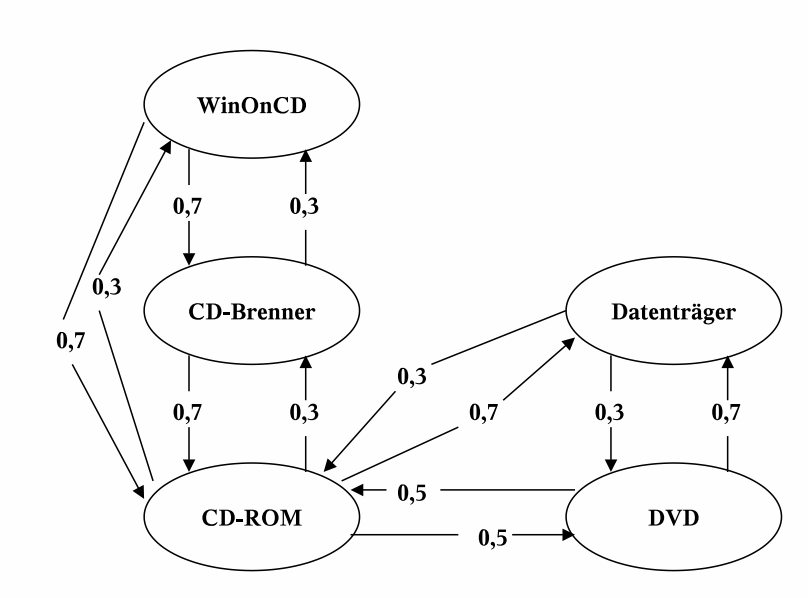
Theoretisch könnte „NO_SIMILARITY“ sogar die Identität „ID“ überschreiben, aber es ist semantisch sinnvoller, „NO_SIMILARITY“ irreflexiv zu modellieren.

Abbildung 6.4 zeigt ein einfaches Beispiel für ein Ähnlichkeitslexikon, das von Hand modelliert wurde. Im oberen Teil der Abbildung befindet sich das



___WINONCD___[IS_MEANS_FOR];___CD-BRENNER___
___CD-BRENNER___[IS_MEANS_FOR];___CD-ROM___
___CD-ROM___[USES_AS_MEANS];___CD-BRENNER___
___CD-ROM___[SPECIFICATION];___DATENTRAEGER___
___DATENTRAEGER___[ABSTRACTION];___CD-ROM___
___DATENTRAEGER___[ABSTRACTION];___DVD___
___DVD___[MEDIUM_SIM];___DATENTRAEGER___

Abbildung 6.4: Ausschnitt eines von Hand modellierten CRNs mit seinem Ähnlichkeitslexikon.



___WINONCD___[IS_MEANS_FOR];___CD-BRENNER___
___CDBRENNER___[USES_AS_MEANS];___WINONDCD___
___CD-BRENNER___[IS_MEANS_FOR];___CD-ROM___
___CD-ROM___[USES_AS_MEANS];___CD-BRENNER___
___WINONCD___[IS_MEANS_FOR];___CD-ROM___
___CD-ROM___[USES_AS_MEANS];___WINONCD___
___CD-ROM___[SPECIFICATION];___DATENTRAEGER___
___DATENTRAEGER___[ABSTRACTION];___CD-ROM___
___DATENTRAEGER___[ABSTRACTION];___DVD___
___DVD___[SPECIFICATION];___DATENTRAEGER___
___CD-ROM___[BROTH];___DVD___
___DVD___[BROTH];___CD-ROM___

Abbildung 6.5: Das automatisch veränderte CRN mit seinem Ähnlichkeitslexikon (neue und veränderte Einträge *kursiv*).

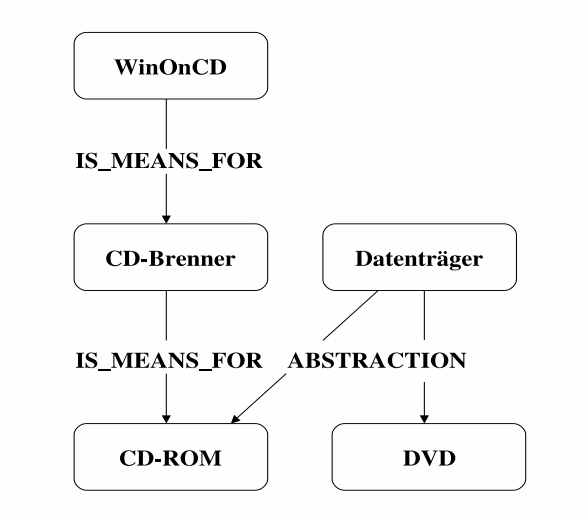
mit Hilfe von Tabelle 6.1 daraus erzeugte CRN. In Abbildung 6.5 ist das durch Anwendung der Operatoren und die Selektions-Funktion *aggr* veränderte Lexikon zu sehen. Durch die Anwendung des Operators e ist eine Kante mit einem Gewicht von 0,3 für „USES_AS_MEANS“ zwischen „CD-Brenner“ und „WinOnCD“ neu eingefügt worden. Außerdem hat e in Verbindung mit *aggr* die „mittlere Ähnlichkeit“ zwischen „DVD“ und „Datenträger“ durch eine „SPECIFICATION“-Ähnlichkeit ersetzt. t erzeugt im Beispiel die transitive Hülle für die „IS_MEANS_FOR“-Relation. Dies ergibt im CRN direkte Kanten für „WinOnCD“ und „CD-ROM“. Eine implizite „BROTH“-Relation wird von b aus der „ABSTRACTION“-Relation für die Kinder des Vaterknotens „Datenträger“ abgeleitet. Dies ergibt im CRN für „CDRom“ und „DVD“ in beiden Richtungen neue Kanten mit dem Gewicht 0,5.

Das Ontologietool Protégé eignet sich gut für die Visualisierung eines Ähnlichkeitslexikons. Die Eigenschaften von Ähnlichkeitstypen tragen dazu bei, die Darstellung zu vereinfachen: Bei Paaren von zueinander inversen Typen kann man einen Typ in der Darstellung weglassen, weil er aus der Gegenrichtung leicht erzeugt werden kann. In der OWL-Datei muss man aus Implementierungsgründen die Gegenrichtungen bei Inversen angeben. Bei symmetrischen Relationen genügt sowohl in der OWL-Datei als auch in der Visualisierung eine Richtung. Bei transitiven Relationen wird ein Hasse-Diagramm gebildet, das heißt es wird jede Kante für ein Paar $[a, c] \in R$ weggelassen, für die ein b mit $[a, b], [b, c] \in R$ existiert. Abgeleitete Geschwisterrelationen werden ebenfalls bei der Darstellung weggelassen, da sie leicht aus der zugehörigen Vaterrelation erzeugt werden können. Die Gewichte der Ähnlichkeitstypen können mit Hilfe der Spezifikationstabelle leicht aus den linguistischen Bezeichnern rekonstruiert werden.

Abbildung 6.6 zeigt die grafische Darstellung des kleinen Ähnlichkeitslexikons aus Abbildung 6.5 und einen Ausschnitt des dazugehörigen OWL-Codes. Wegen der übersichtlichen Darstellung empfiehlt es sich, Änderungen des Ähnlichkeitslexikons mit einem Tool wie Protégé zu machen.

6.4 Akquisition von Hintergrundwissen mit OntoDigger

Die von mir betreute Diplomarbeit von Peter Hammels [Ham03] beschreibt Konzepte und Verfahren, um Textkorpora zu analysieren und daraus semantische Erkenntnisse für die Modellierung von IE- und Ähnlichkeitslexika zu gewinnen.




```
<owl:Class rdf:ID="WINONCD">
   <rdfs:subClassOf rdf:resource="#IT-TERM"/>
   <rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty>
         <owl:ObjectProperty rdf:about="#IS_MEANS_FOR"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#CDBRENNER"/>
   </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
...
```

Abbildung 6.6: Darstellung von TCBR-Hintergrundwissen in einer Ontologie.

Die Verfahren sind alle halbautomatisch, das heißt ein Algorithmus liefert eine Liste von Vorschlägen für neue Einträge in das Ähnlichkeitslexikon, die dann manuell überprüft werden. Einfache statistische Methoden wie das Berechnen von Kookkurrenzen (gemeinsames Vorkommen) von Wörtern stehen neben komplizierteren Verfahren. Zum Beispiel wird ein *part-of-speech*-Tagger verwendet, um grammatikalische Informationen über einen Satz zu bekommen. Mit Hilfe dieser *tags* können spezielle Konstellationen (*pattern*) von Satzteilen gesucht werden, um Hypothesen für Beziehungen zwischen Wörtern zu generieren.

Die Ergebnisse der verschiedenen Verfahren sind am Beispiel einer Internet-Textsammlung aus dem Kulturjournalismus ausgewertet worden. Die in **OntoDigger** implementierte Syntaxanalyse mit Beschränkung auf einzelne Sätze ist nicht mächtig genug, um eine vollautomatische Analyse ohne menschliche Nachbearbeitung zu ermöglichen. Im verwandten Ansatz von Hahn et al. [HS04] für medizinische Ontologien stehen neben den Textkorpora bereits semi-formale Daten aus der **UMLS**-Ontologie [UML02] zur Verfügung. Mit Hilfe verschiedener Datenquellen kann dort eine Hypothese verifiziert werden. Da **OntoDigger** nur auf der Basis von Textkorpora arbeitet, wird der Mensch noch als Korrektiv benötigt. Die Erzeugung der Vorschlagslisten innerhalb des halbautomatischen Verfahrens ist aber eine große Hilfe bei der Modellierung eines Ähnlichkeitslexikons.

Kapitel 7

Wissensverteilung: Erfahrungswissen austauschen

Mit Hilfe von EM-Systemen lässt sich Wissen nach dem *Pull*-Prinzip verteilen. Wer etwas wissen möchte, stellt eine Anfrage an das System und bekommt so Zugriff auf dokumentiertes Erfahrungswissen. Die Akteure, die auf diese Art Wissen anfordern, sind normalerweise Menschen. Wir betrachten hier aber auch Szenarien mit mehreren EM-Systemen, die Wissen untereinander austauschen. Ein EM-System, das Wissen anfordert, ist ein Agent. Die Fragestellungen in diesem Kapitel sind:

- Sollten EM-Systeme miteinander kommunizieren?
- Wie können EM-Systeme Teile ihres Erfahrungsschatzes weitergeben?
- Wie kann ein EM-System fremdes Wissen möglichst konfliktarm integrieren?

In Kapitel 7.1 wird ein Multiagentensystem von persönlichen Assistenzagenten zur Benutzung und Installation von Software vorgestellt. Die Agenten kooperieren, indem sie Fälle und dazugehöriges Hintergrundwissen zur Erledigung ihrer Aufträge austauschen. Das Multiagentensystem wurde von Ralf Kühnel Ende der neunziger Jahre entwickelt (siehe [Küh99] und [Küh00]). In Kapitel 7.2 beschreibe ich die Erweiterung von Kühnels System um eine fallbasierte Komponente, die die Interaktion mit den Benutzern verbessern soll. Die fallbasierte Komponente stammt aus der von mir betreuten Diplomarbeit von Mike Wernicke [Wer03], siehe dazu auch den Konferenzbeitrag [MW03] und meinen Workshopbeitrag [Min05a]. Kapitel 7.3 behandelt den Austausch und die Integration von Wissen zwischen den Agenten.

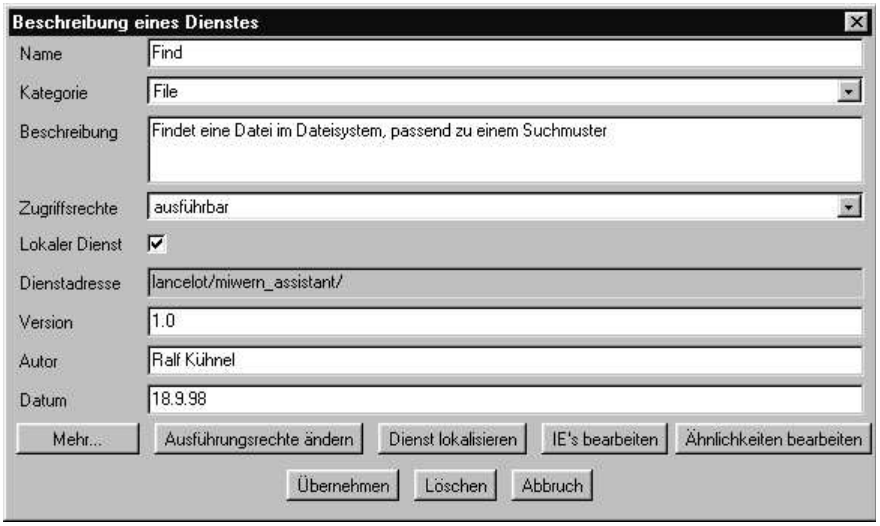


Abbildung 7.1: Beispiel einer Dienstbeschreibung zu „find“.

7.1 Persönliche Assistenzagenten

Jeder Kühnel-Assistenzagent hat eine Dienstbasis, einen eigenen Fundus an Diensten, die er im Auftrag seines Benutzers ausführen kann.

Definition 7.1.1 (Dienst)

> Ein *Dienst* ist ein Hilfsprogramm, zum Beispiel um eine Datei auszudrucken oder eine Webrecherche durchzuführen. Er besteht aus einer Dienstbeschreibung und einem ausführbaren Programmcode.

Der Programmcode der Dienste ist in Java implementiert. Abbildung 7.1 zeigt ein Beispiel einer Dienstbeschreibung zum UNIX-Befehl „find“. Sie ordnet den Dienst in eine Kategorie ein, im Beispiel ist dies die Arbeit mit Dateien. Desweiteren enthält sie einen Text zur Beschreibung der Funktionalität des Dienstes, die Zugriffsrechte, die zu seiner Ausführung benötigt werden, und einige weitere eher verwaltungstechnische Angaben. In einer zweiten Ebene kann der Entwickler des Dienstes Vorbedingungen für die Dienstausführung und Eingabe- und Ausgabeparameter in der Planungssprache STRIPS spezifizieren.

Eine Planungskomponente kann mit Hilfe dieser Parameter mehrere Dienste zu einem neuen, größeren Dienst kombinieren. Wenn nötig, können auch Dienstbasen anderer Assistenagenten hinzugezogen werden. Die Agenten kooperieren durch den Austausch von Dienstbeschreibungen oder sogar ausführbaren Codes. Der Agent, dem ein Dienst gehört, kann diesen auch im Auftrag des anderen Agenten ausführen. Wenn zum Beispiel ein Agent von seinem Benutzer den Auftrag bekommt, eine DVI-Datei auszudrucken, selbst aber nur weiß, wie PostScript gedruckt wird, kann er andere Agenten nach einer Umwandlung des Dateiformats von DVI in PostScript befragen. Kennt ein anderer Agent den UNIX-Befehl dvips, kann der unwissende Agent diesen Dienst entweder selbst lernen, oder dem anderen Agenten seine DVI-Datei zur Formatumwandlung zusenden. Wann ein Dienst gelernt werden kann und wann die Dienste anderer Agenten in Anspruch genommen werden müssen, wird über die Zugriffsrechte zu den Diensten und zu den dafür benötigten Ressourcen wie Drucker und Betriebssystem angegeben.

7.2 Erweiterung der Benutzerschnittstelle mit TCBR

In unserer Erweiterung von Kühnels Multiagentensystem um eine CBR-Komponente speichert jeder Agent seine Dienstbeschreibungen als Fälle in der persönlichen Fallbasis ab. Der Text der ersten Dienstbeschreibungsebene wird dafür auf Mengen von Informationseinheiten abgebildet, wie es in Kapitel 4.3 beschrieben ist. Die Benutzer können dann ihre Aufträge in natürlicher Sprache formulieren, zum Beispiel „drucke eine PDF-Datei für mich aus“. Ein fallbasiertes Retrieval (siehe ebenfalls Kapitel 4.3) liefert die Dienste aus der Fallbasis, die am besten zu dem Auftrag in der Anfrage passen. Dies hat den Vorteil, dass niemand sich - wie bisher - den exakten Wortlaut eines Dienstnamens merken muss, um den Dienst zu finden. Auch das in Java recht langsame Navigieren durch die Baumdarstellung nach Kategorien der Dienstbasis kann damit entfallen. Wir stellen nach wie vor den Navigationsbaum als Kontrollmöglichkeit oder Alternative zum Suchen zur Verfügung.

Die Einführung der TCBR-Komponente hat auch Auswirkungen auf die Entwicklung von neuen Diensten und den Austausch von Diensten. Bereits in Kühnels Implementation muss bei der Entwicklung eines Dienstes ja der ausführbare Code programmiert und die Beschreibung des Dienstes ausgefüllt werden. In unserer Variante sollte zusätzlich überprüft werden, ob das Hintergrundwissen für das Retrieval um neue Indexvokabeln und lokale Ähn-

lichkeitswerte erweitert werden muss. Auch beim Austausch von Diensten muss nun das Hintergrundwissen zur Dienstbeschreibung beachtet werden. Dies wird im folgenden Kapitel näher untersucht.

7.3 Persönliche Case Retrieval Nets

Das Multiagentensystem aus Kapitel 7.1 kann über mehrere Computer verteilt sein. Jeder Agent hat dort, wo er läuft, seine eigene Dienstbasis. Deshalb ist es empfehlenswert, dass jeder Agent sein persönliches CRN zum Retrieval in seiner Dienstbasis besitzt.

Natürlich wäre ein zentraler CRN-Server in der Art eines Yellow-Pages-Systems einfacher zu pflegen. Wenn aber jeder Agent die Benutzeranfragen zu diesem zentralen CRN schicken muss, um dazu passende Dienste zu ermitteln, ist dies eher die Implementierung einer Client-Server-Architektur als eines Agentenansatzes. Außerdem kann es vorkommen, dass Agenten sich auf bestimmte Aufgabenbereiche spezialisieren, zum Beispiel auf die Arbeit mit bestimmten Programmen, die ihre Benutzer verwenden. Zum Beispiel kann ein Agent viele Fälle zu Petrinetz-Software enthalten, während ein anderer Agent vielleicht kein Interesse an Petrinetzen hat, sich dafür aber bestens mit Hypertext-Entwurf auskennt. Das IE zum Begriff „Markierung“ hat in beiden Gebieten verschiedene Bedeutungen und wird mit unterschiedlichen anderen IEs vernetzt sein, zum Beispiel mit „Aktivierung“ bei den Petrinetzen und „Tag“ bei den Hypertexten.

Bei ontologiebasierten Systemen, in denen logische Schlussfolgerungen gezogen werden sollen, sind solche Doppeldeutigkeiten ein echtes Problem. Ähnliches gilt für die Auswirkungen von weiteren semantischen und syntaktischen Konflikten, die bei der Integration von Wortschätzen verschiedener Autoren entstehen. Das Fallbasierte Schließen ist zwar relativ robust gegen einzelne Inkonsistenzen, liefert aber verfälschte Ergebnisse, wenn diese sich wie in einem zentralen CRN-Server häufen (vergleiche S. 117).

Wir haben uns für individuelle CRNs entschieden, allerdings auf der Basis eines gemeinsam ausgelieferten Grundwortschatzes, der von den einzelnen Benutzern dann erweitert werden kann. Der Grundwortschatz reduziert den Modellierungsaufwand bei der Entwicklung von neuen Diensten. Für größere und wichtigere Anwendungen als unseren Prototypen sollte ein Maintenance-Konzept entwickelt werden, das die Besonderheiten von individuell gewachsenen CRNs berücksichtigt (vergleiche Kapitel 9 bzw. wieder die Diskussion auf S. 117).

```
__FINDEN__;FINDEN;FINDET;GEFUNDEN;GEFUNDENER;GEFUNDENE;
  GEFUNDENES;GEFUNDENEN;GEFUNDENEM
__DATEI__;DATEI;DATEIEN;FILE;FILES
__DATEISYSTEM__;DATEISYSTEM;DATEISYSTEME;DATEISYSTEMEN
__SUCHMUSTER__;SUCHMUSTER;SUCHMUSTERS;SUCHMUSTERN

__DATEI__[IS_PART];__DATEISYSTEM__
__DATEISYSTEM__[HAS_PART];__DATEI__
```

Abbildung 7.2: Hintergrundwissen des Falls aus Abbildung 7.1.

Integration von fremdem Hintergrundwissen

Persönliche CRNs sollten erweitert werden, wenn Fälle zwischen Agenten ausgetauscht werden. Eine neu empfangene Dienstbeschreibung kann wichtige Begriffe enthalten, die dem Empfänger-Agenten noch unbekannt sind. Um diese Dienstbeschreibungen in zukünftigen Retrieval-Prozessen auch wiederfinden zu können, haben wir uns dafür entschieden, dass Agenten beim Verschicken von Dienstbeschreibungen auch die dazugehörigen Teile des Hintergrundwissens mitschicken. Das sind die Lexikoneinträge aller IEs der internen Darstellung des Falls und der dazugehörigen Einträge im Ähnlichkeitslexikon. Abbildung 7.2 zeigt das Hintergrundwissen, das beim Austausch des Falls aus Abbildung 7.1 mitgeschickt wird.

Bei der Integration des Hintergrundwissens in das eigene CRN bzw. die eigenen Lexika, aus denen das CRN aufgebaut wird, treten folgende Typen von Konflikten auf, die negative Auswirkungen auf das Retrieval haben können:

1. Ein Duplikat eines bereits bekannten IE-Namens, das aber andere Wörter repräsentiert als das eigene IE mit demselben Namen.
2. Ein anderer Ähnlichkeitstyp für dasselbe Paar von IEs.
3. Ein Wort, das im eigenen Lexikon einem anderen IE zugeordnet ist als in den neu empfangenen IEs.

Ein Beipiel für einen Konflikt vom Typ 1 sind zwei IEs zum Begriff „CD“, wovon einer verschiedene Schreibweisen von „CD-Rom“ enthält und der andere die des UNIX-Befehls „cd“ zum Wechseln des Verzeichnisses:

```
__CD__;CD;CD-ROM;CD-ROMS
__CD__;CD;CD-BEFEHL
```

Im Gegensatz zu diesem Beispiel repräsentieren die meisten IEs mit gleichem Namen auch das gleiche Konzept, selbst wenn die Menge der zugeordneten Wörter sich leicht unterscheidet. Deshalb haben wir diesen Konflikttyp zu Gunsten der Mehrheit gelöst, indem wir einfach beide Wortmengen zu einem IE vereinigen. Dabei entstehen natürlich auch IEs mit doppeldeutiger Semantik, zum Beispiel:

```
__CD__;CD;CD-ROM;CD-ROMS;CD-BEFEHL
```

Eine Abwandlung von Konflikttyp 1 sind gleichnamige IEs mit unterschiedlichen Wortschatz-Kategorien. In unserem Prototypen haben wir diesen Konflikt durch eine Vereinfachung vermieden, nämlich dass es nur eine einzige Kategorie für alle IEs gibt. Will man verschiedene Wortschatz-Kategorien behandeln, lässt sich der Konflikt relativ einfach lösen, indem man zum Beispiel immer die eigene Wortschatz-Kategorie oder alternativ dazu immer die speziellste Wortschatz-Kategorie bevorzugt.

Konflikte vom Typ 2 werden ebenfalls ziemlich unkompliziert gelöst: Wir übernehmen immer den Ähnlichkeitstyp mit höherem Gewicht. Hier wäre auch zu überlegen, stattdessen immer die eigene Zuordnung zu bevorzugen. Dadurch entwickeln sich die individuellen CRNs weiter auseinander als bei unserer Lösung, dafür werden nie manuell modellierte Einträge überschrieben, sondern fremde Einträge dem eigenen Modell untergeordnet.

Ein Beispiel für Konflikte vom Typ 3 sind zwei IEs zu den Begriffen „print“ und „printer“, die beide „printer“ und „printers“ enthalten:

```
__PRINT__;PRINT;PRINTING;PRINTER;PRINTERS
__PRINTER__;PRINTER;PRINTERS;PRINT DEVICE
```

Dieser Konflikttyp ist also sozusagen die Umkehrung von Konflikttyp 1, bei dem ja derselbe IE-Name verschiedenen Wörtern zugeordnet ist. Bei Typ 3 sind dieselben Wörter verschiedenen IEs, also meist inhaltlich verschiedenen Konzepten zugeordnet. Selten kommt es auch vor, dass verschiedene Schreibweisen desselben Konzepts, zum Beipsiel „CD-Rom“ und „CDRom“ als IE-Namen verwendet wurden. Wir gehen davon aus, dass der IE-Name immer eine grammatikalische Grundform benutzt, so dass Abweichungen in der Schreibweise des IE-Namens eigentlich nur bei Sonderzeichen oder zusammengesetzten Begriffen vorkommen sollten. Leider gibt es in der Praxis auch gebeugte Formen als IE-Namen. In Umkehrung zu Konflikten von Typ 1 lösen wir Konflikte vom Typ 3 durch Nichtstun: Wir belassen die IEs in ihrer Originalform, selbst wenn dies zu Überschneidungen führt.

Potentielle Verschmutzung der individuellen CRNs durch Restkonflikte

Die drei Konflikttypen aus der obigen Aufzählung werden also rein syntaktisch gelöst, dabei kann es zu semantischen Veränderungen im CRN kommen, die sich auf das Retrieval auswirken können. In unseren zugegebenermaßen kleinen experimentellen Dienstbase mit insgesamt ca. 30 Diensten hat diese einfache Konfliktlöse-Strategie funktioniert, weil ein einzelner Ähnlichkeitswert oder ein IE mehr oder weniger geringen Einfluss auf das Gesamtergebnis des Retrievals hat. Ich bezeichne einen Ähnlichkeitswert, der semantisch nicht zum restlichen Modell passt, im Folgenden als „Fehler“, ebenso ein „falsch“ modelliertes IE. Die geringe Auswirkung dieser Fehler hat zwei Ursachen:

- Erstens ist die Ähnlichkeitsfunktion SIM robust gegen einzelne Fehler, weil sie kompositorisch ist, das heißt die anderen IEs und lokalen Ähnlichkeitswerte tragen ebenfalls zum Finden eines Falls bei. Ist die Kompositionsfunktion wie bei uns monoton bezüglich der Anzahl und der Werte ihrer Teilfunktionen, wird ein Fehler also nicht potenziert sondern mit hoffentlich korrekten Werten aggregiert.

- Zweitens besteht ein Retrieval-Ergebnis in unseren Anwendungsbeispielen immer aus einer Liste von Fällen. Falsche Ähnlichkeitswerte spielen bei der Anordnung der Fälle im Retrieval-Ergebnis die Rolle einer zu starken oder zu schwachen Erinnerung an einen Fall. Solange diese Über- oder Unterschätzung relevanter Fälle nicht zu stark ist, erscheinen diese ja nach wie vor in der Ergebnisliste eines Retrievalprozesses. Steht der beste Fall zum Beispiel an dritter statt an zweiter Position des Retrieval-Ergebnisses, ist dies nicht so schlimm, da der Benutzer ihn nach wie vor leicht findet (vgl. die Diskussion auf S. 79).

Problematisch wird es erst, wenn sich die Fehler so häufen, dass gute Fälle durch eine stark verfälschte Sortierung von den vorderen Positionen der Ergebnisliste verschwinden.

Dies kann passieren, wenn die individuellen CRNs zu stark durch Integration fremden Wissens verschmutzt sind. Dann wird eine diffizilere Konfliktlösung bei der Integration von CRNs wie im „ontology merging“ (siehe zum Beispiel [HPS04]) oder eine Strategie zur nachträglichen Beseitigung von semantischen Ungereimtheiten benötigt (zum Beispiel eine Erweiterung der Verfahren in Kapitel 6.3).

Ein Beispiel für die Integration zweier Fälle ist in Abbildung 7.3 zu sehen. In diesem Beispiel treten bei „finden“ und „snapshot“ Konflikte vom Typ 1 auf und werden durch Vereinigung der Original-IEs gelöst. Ein Konflikt vom

Typ 3 ist das Wort „pattern“, das in den Original-IEs zum IE „pattern“ gehört, in der neuen Menge aber zu „Suchmuster“. Deshalb tritt „pattern“ in der vereinigten Menge doppelt auf. Das heißt, dass das Wort „pattern“ in einer Anfrage zwei IEs aktiviert, im „worst case“ sogar ein Fall diese beiden IEs enthält und deshalb überbewertet wird.

Abbildung 7.4 zeigt die dazugehörige Integration von Einträgen des Ähnlichkeitslexikons. Die beiden Ähnlichkeitseinträge zwischen „Datei“ und „Dateisystem“ kommen neu hinzu. Die neuen Ähnlichkeitstypen bei „screenshot“ und „snapshot“ werden nicht übernommen, da zwischen diesen beiden IEs bereits Einträge mit dem Ähnlichkeitstyp „SYNONYM“ stehen, der ein höheres Gewicht als „HIGH_SIM“ hat.

Eigenes IE-Lexikon	neu empfangene IEs
___AUSSCHNITT___;AUSSCHNITT;AUSSCHNITTE; AUSSCHNITTEN	%aus einem Fall, wie man einen Bildschirmabzug macht:
___BILDSCHIRMABZUG___;BILDSCHIRMABZUG	___CLIPPING___;CLIPPING;CLIPPINGS
___BILDSCHIRM___;BILDSCHIRM;BILDSCHIRMS; BILDSCHIRME;BILDSCHIRMEN	___SCREENSHOT___;SCREENSHOT;SCREENSHOTS
___DATEISYSTEM___;DATEISYSTEM;DATEISYSTEME; DATEISYSTEMEN	___SNAPSHOT___;SNAPSHOT;SNAPSHOTS;SNAP-SHOT
___FINDEN___;FINDEN;FINDET	%aus einem Fall zum UNIX-Befehl 'find':
___PATTERN___;PATTERN;SUCHMUSTER	___DATEI___;DATEI;DATEIEN;FILE;FILES
___SCREENSHOT___;SCREENSHOT;SCREENSHOTS	___DATEISYSTEM___;DATEISYSTEM;DATEISYSTEME
___SNAPSHOT___;SNAPSHOTS	___FINDEN___;FINDEN;GEFUNDEN;GEFUNDENER; GEFUNDENE;GEFUNDENES;GEFUNDENEN; GEFUNDENEM
___SUCHE___;SUCHE;SUCHEN;SUCHT	___SUCHMUSTER___;SUCHMUSTER;SUCHMUSTERS; SUCHMUSTERN
___UNIX___;UNIX	

Erweitertes eigenes IE-Lexikon
___AUSSCHNITT___;AUSSCHNITT;AUSSCHNITTE;AUSSCHNITTEN
___BILDSCHIRMABZUG___;BILDSCHIRMABZUG
___BILDSCHIRM___;BILDSCHIRM;BILDSCHIRMS;BILDSCHIRME;BILDSCHIRMEN
___CLIPPING___;CLIPPING;CLIPPINGS
___DATEISYSTEM___;DATEISYSTEM;DATEISYSTEME;DATEISYSTEMEN
___FINDEN___;FINDEN;FINDET;GEFUNDEN;GEFUNDENER;GEFUNDENE;GEFUNDENES;GEFUNDENEN; GEFUNDENEM
___PATTERN___;PATTERN;SUCHMUSTER
___SCREENSHOT___;SCREENSHOT;SCREENSHOTS
___SNAPSHOT___;SNAPSHOT;SNAPSHOTS;SNAP-SHOT
___SUCHE___;SUCHE;SUCHEN;SUCHT
___SUCHMUSTER___;SUCHMUSTER;SUCHMUSTERS;SUCHMUSTERN
___UNIX___;UNIX

Abbildung 7.3: Integration der IEs aus zwei fremden Fällen.

Eigenes Ähnlichkeitslexikon	neu empfangene Werte
___BILDSCHIRMABZUG___[HIGH_SIM];___SNAPSHOT___	___DATEI___[IS_PART];___DATEISYSTEM___
___SCREENSHOT___[SYNONYM];___SNAPSHOT___	___DATEISYSTEM___[HAS_PART];___DATEI___
___SNAPSHOT___[HIGH_SIM];___BILDSCHIRMABZUG___	___SCREENSHOT___[HIGH_SIM];___SNAPSHOT___
___SNAPSHOT___[SYNONYM];___SCREENSHOT___	___SNAPSHOT___[HIGH_SIM];___SCREENSHOT___
Erweitertes eigenes Ähnlichkeitslexikon	
___BILDSCHIRMABZUG___[HIGH_SIM];___SNAPSHOT___	
___DATEI___[IS_PART];___DATEISYSTEM___	
___DATEISYSTEM___[HAS_PART];___DATEI___	
___SCREENSHOT___[SYNONYM];___SNAPSHOT___	
___SNAPSHOT___[HIGH_SIM];___BILDSCHIRMABZUG___	
___SNAPSHOT___[SYNONYM];___SCREENSHOT___	

Abbildung 7.4: Integration des Ähnlichkeitslexikons aus zwei fremden Fällen.

Kapitel 8

Wissensnutzung: Ein EM-System organisieren

Organisatorische und psychosoziale Aspekte sind sehr wichtig, um ein EM-System erfolgreich zu machen. Die Fragestellungen in diesem Kapitel sind:

- Wie vermeidet man organisatorische Barrieren durch Partizipation der Benutzer?
- Welche soziotechnischen und organisatorischen Maßnahmen sind erforderlich, um Benutzer in Entwicklungs- und Wissensakquisitionsprozesse einzubeziehen?

In Teil I der Arbeit habe ich bereits die drei Säulen des Wissensmanagements Organisation, Mensch und Technologien vorgestellt (siehe S. 11) und eine ganzheitliche Herangehensweise an Erfahrungsmanagement gefordert (siehe Kapitel 3.3 auf S. 32 ff.). Im Folgenden greife ich die Empfehlungen von Meyer und Scholl (siehe S. 33) auf, um mögliche Ursachen für das Scheitern von Wissensmangement-Systemen zu identifizieren und zu beseitigen. Ich setze die Empfehlungen an einem Anwendungsbeispiel, dem **ExperienceBook II**, in die konkrete Praxis um (siehe dazu meinen Konferenzbeitrag [Min05b]). Kapitel 8.1 beschreibt das **ExperienceBook II**. Die Umsetzung der Empfehlungen von Meyer und Scholl ist Gegenstand von Kapitel 8.2. Dies erfordert einige organisatorische und soziotechnische Maßnahmen, die ich wie folgt definiere:

Definition 8.0.1 (Organisatorische Maßnahme für ein EM-System)

> Eine *organisatorische Maßnahme für ein EM-System* ist eine Maßnahme, die nicht das System selbst, sondern seine Einbettung in die Organisation betrifft.

Definition 8.0.2 (Soziotechnische Maßnahme für ein EM-System)

> Eine *soziotechnische Maßnahme für ein EM-System* ist eine technische Veränderung des Systems, die neu gewonnene psychosoziale Erkenntnisse umsetzt.

Beide Arten von Aktivitäten gehen Hand in Hand und bilden zusammen mit den Empfehlungen aus Kapitel 8.2 quasi die „Implementierung“ des äußeren Kreislaufs des EM-Modells von Bergmann (siehe s. 30).

8.1 Das ExperienceBook II

Das **ExperienceBook II** ist ein fallbasiertes Informations- und Hilfesystem für Studierende der Informatik an der Humboldt-Universität, insbesondere für Neulinge in den ersten Semestern. Es setzt wie alle Anwendungsbeispiele in dieser Arbeit fallbasierte Techniken für Texte (siehe Kapitel 4.3) ein, um Erfahrungsmanagement zu unterstützen. Ein Apache-Webserver bedient Anfragen von CGI-Clients, indem er sie mit einem CRN-Server kommunizieren lässt. Die Webseite des **ExperienceBooks II** ist nur für Mitglieder des Instituts zugänglich. Der passwortfreie Zugriff ist durch eine `.htaccess`-Datei des Apache-Webservers auf IP-Adressen innerhalb des Instituts beschränkt. Von außen müssen die Benutzer sich beim ersten Zugriff auf die Webseite mit ihrem normalen UNIX-Passwort einloggen. Die Datenübertragung ist durch das sichere https-Protokoll vor fremden Augen geschützt.

Die Benutzer des **ExperienceBooks II** können mit Hilfe der Webseite auf das Erfahrungswissen ihrer Kommillitonen zugreifen. Die Fallbasis deckt eine große Bandbreite ab. Neben eher technischen Fällen, zum Beispiel zur Installation eines Prolog-Interpreters, stehen allgemeine Fragen und Probleme der Studierenden, beispielsweise, wie man eine Schlüsselkarte bekommt oder wo gute Kantinen auf dem Campus zu finden sind. Ein Teil der Falldaten bezieht sich direkt auf die Vorlesung „Praktische Informatik I“, die von fast allen Studierenden zu Beginn ihres Studiums gehört werden muss. Wegen dieser Themenvielfalt ist die Fallbasis in Unterthemen eingeteilt. Zur Zeit gibt es im **ExperienceBook II** Fälle zu folgenden Themen:

- UNIX-Probleme,

- Linux-Probleme,
- Prolog-Probleme,
- Probleme bei der Einwahl vom PC aus ins Institut,
- Fragen zum Aufgaben- und Punkteverwaltungssystem Goya,
- Fragen zur Vorlesung „Praktischen Informatik I“ und
- allgemeine Fragen und Probleme.

Der fallbasierte Teil des **ExperienceBooks II** ist mit einem Diskussionsforum gekoppelt, in dem die Studierenden aktuelle Fragen und Probleme diskutieren können. Man kann mit einem Mausklick von der fallbasierten Anfrageseite zum Diskussionsforum wechseln und umgekehrt. Im Diskussionsforum lässt sich auch ein Anfragefensterchen für die Fallbasis direkt benutzen. Leider gibt es noch keine automatische Transformation abgeschlossener Diskussionsbeiträge in Fälle. Es wäre eine interessante Forschungsfrage, einmal zu untersuchen, ob man in Vorbereitung dessen die Qualität eines Diskussionsbeitrags zum Beispiel durch syntaktische oder semantische Analyse der Antworten automatisch ermitteln oder zumindest abschätzen kann. Eine zweite offene Frage ist, wie ein guter Beitrag dann geschickt in das Fallformat überführt werden kann.

8.2 Handlungsempfehlungen für ein ganzheitliches EM

Die drei Handlungsempfehlungen von Meyer und Scholl aus dem Blickwinkel der Psychologie werden im **ExperienceBook II** in folgender Weise praktisch umgesetzt:

1. Wir *ermitteln den Bedarf* durch frühzeitige und langanhaltende Partizipation der Benutzer. Dies geschieht vor allem durch Gespräche vor, während und nach der Einführung des Systems und durch schriftliches Feedback aus Interviews. Zusätzlich liefert die Beobachtung des Anfrageverhaltens Erkenntnisse über den Wissensbedarf der Studierenden (siehe Ausführungen zu Punkt 3).
2. Die *Haltung der Benutzer* wird durch Informations- und Motivationsgespräche beeinflußt. Durch diese und andere Werbemaßnahmen hat sich die Zugriffsrate des Systems nachweislich erhöht (siehe unten), was auf

einen Abbau der organisatorischen Barriere „fehlende Motivation“ hinweist.

3. Weitere *organisatorische Barrieren* werden durch Beobachtung des Benutzerverhaltens und die oben genannten Interviews ermittelt. Die Ergebnisse führen zu organisatorischen und soziotechnischen Maßnahmen zur Verbesserung des Systems.

Das Grundprinzip unserer ganzheitlichen Herangehensweise ist zusammengefasst also die Partizipation der Benutzer zur gezielten Vermeidung potentieller und tatsächlicher Barrieren. Wir haben verschiedene Methoden eingesetzt, um den Handlungsempfehlungen zu folgen:

- Gespräche,
- Interviews,
- Vorträge,
- schriftliche Gruppenarbeit,
- Email-Kommunikation und
- Logfile-Analyse.

In den folgenden Unterkapiteln beschreibe ich chronologisch, wie die einzelnen Methoden in den jeweiligen Entwicklungsphasen zum Einsatz kommen und welche Handlungsempfehlungen von Meyer und Scholz jeweils damit befolgt werden.

Vor Einführung des Systems

Die organisatorischen Aktivitäten zur Förderung der Wissensnutzung folgen in dieser Phase vor Einführung des Systems hauptsächlich den Empfehlungen 1 und 2. Das heißt, wir ermitteln den Bedarf und versuchen, die Haltung der späteren Benutzer positiv zu beeinflussen.

Dazu ergreifen wir im Vorfeld des Systemstarts folgende Maßnahmen:

- Die Fallbasis wird initial mit einigen Fällen aus verschiedenen Quellen befüllt, so dass das System von Anfang an einen gewissen Nutzen bringen kann.
- Eine Werbekampagne informiert die potentiellen Benutzer über das neue System.

- Wir setzen ausschließlich intrinsische Motivierung ein und treffen strenge Datenschutzvorkehrungen, um Abwehrhaltungen vorzubeugen.
- Das System wird bewußt so gestaltet, dass es die Kommunikation innerhalb der Zielgruppe anregt.

Die Fälle für die initiale Fallbasis stammen aus dem Lehrmaterial vergangener Jahre und von den Webseiten der Rechnerbetriebsgruppe unseres Institutes, ergänzt durch Fälle aus einer schriftlichen Gruppenarbeit, die ich mit einer Seminargruppe von Studierenden aus dem dritten und aus höheren Semestern durchgeführt habe. Die Gruppensitzung hat noch einen zweiten Zweck erfüllt, nämlich den vermutlichen Wissensbedarf der Studierenden im ersten Semester etwas genauer zu bestimmen, da die Studierenden von ihren eigenen Erfahrungen zu Beginn des Studiums berichtet haben. Dabei stellte sich zum Beispiel heraus, dass viele Probleme bei der Einwahl ins Institut vom häuslichen PC aus entstehen und dieser Bereich als eigenes Unterthema ins System aufgenommen werden sollte.

Die Werbekampagne für das **ExperienceBook II** lief über verschiedene Kanäle, um die späteren Benutzer zu erreichen: Das System wurde mehrfach per Email und von der Webseite der Vorlesung „Praktische Informatik I“ aus angekündigt. Zusätzlich wurden Vorträge in Lehrveranstaltungen gehalten und Gespräche mit zwei studentischen Initiativen geführt. Die Auswertung der Web-Logfiles (siehe unten) hat ergeben, dass die Werbung mit persönlichem Kontakt am wirkungsvollsten war, weil danach die Zugriffszahlen in die Höhe schnellten. Alle Gespräche wurden wie die Schreibsitzung auch für die Bedarfsermittlung genutzt.

Wir haben uns trotz der Anbindung an die Vorlesung „Praktische Informatik I“ dagegen entschieden, die Benutzung und Befüllung des Systems extrinsisch zu belohnen, also zum Beispiel durch Zusatzpunkte für die Übungsaufgaben, die zu leisten sind. Die Vermutung, dass „unter Zwang“ geschriebene Fälle eine schlechte Qualität haben, hat sich bestätigt, als wir einem Studenten, der eine Zusatzleistung für einen Übungsschein erbringen musste, die Aufgabe gaben, zu vier vorgegebenen Themen Fälle zu schreiben. Diese Fälle sind im Gegensatz zu aus eigener Motivation geschriebenen kaum zu gebrauchen. Deshalb setzten wir auf intrinsische Motivierung, das heißt wir versuchen, die Benutzer vom Nutzen des Systems durch Argumente zu überzeugen. Neben dieser organisatorischen Maßnahme stehen soziotechnische Vorkehrungen zum Schutz der Daten und der persönlichen Freiheiten wie zum Beispiel dem anonymen Retrieval, der Möglichkeit, anonym Falldaten einzugeben, und die Verwendung von Passwortschutz und sicheren Protokollen (siehe oben). Diese soziotechnischen Maßnahmen sollen die Studierenden ermutigen, das System zu benutzen.

Der vierte Punkt, das kommunikationsfördernde Design des Systems, beeinflusst eher das Verhalten als die Haltung der Benutzer. Unser Ziel ist es, dass die Mitglieder der Nutzergemeinschaft mit Hilfe des Systems oder auch persönlich miteinander in Kontakt treten. Dazu haben wir folgende soziotechnische Maßnahmen ergriffen:

- Alle Benutzer haben Lese- und Schreibrechte an allen Falldaten.
- Bei der Anzeige eines Falls lädt ein Kommentarfenster dazu ein, den Fall um eigene Erfahrungen zu ergänzen.
- Jeder Fall kann von jedem Benutzer verändert werden. Von allen Revisionen der Fälle werden Sicherheitskopien angelegt, aus denen der Administrator bei versehentlicher oder mutwilliger Zerstörung eines Falls diesen wiederherstellen kann. Natürlich können veraltete oder sinnlose Fälle auch aktualisiert oder gelöscht werden.
- Im Normalfall gibt ein Autor eines Falls seine Emailadresse als Kontaktadresse mit an.
- Das System ist in ein Diskussionsforum eingebettet (siehe oben).

Die rege Benutzung des Diskussionsforums in der ersten Runde und die Verwendung des Kommentarfelds in der zweiten Runde haben gezeigt, dass die Kommunikation zumindest innerhalb des Systems angeregt wurde. Ob die Studierenden auch außerhalb des Systems Sozialisation nach Nonaka und Takeuchi (siehe S. 13) betreiben beziehungsweise durch Einsatz des **ExperienceBooks II** vermehrt betreiben, haben wir nicht untersucht. Es lässt sich aber auf den Fluren und in der Mensa beobachten, dass sie von sich aus rege über Fachthemen miteinander kommunizieren. Eine besondere Anregung zur Kommunikation ist bei Benutzergruppen, die auch außerhalb des Systems eine Gemeinschaft bilden, gar nicht nötig.

Nach Einführung des Systems (Erste Runde)

In dieser Phase überwiegt die Analyse des Benutzerverhaltens, um organisatorische Barrieren nach Handlungsempfehlung 3 zu entdecken und zu beheben.

Die Auswertung der Web-Logfiles wenige Wochen nach Einführung des Systems ergab, dass das System zwar häufig befragt wurde, aber kaum neue Fälle geschrieben wurden. Lediglich zwei Autoren lieferten von sich aus neue

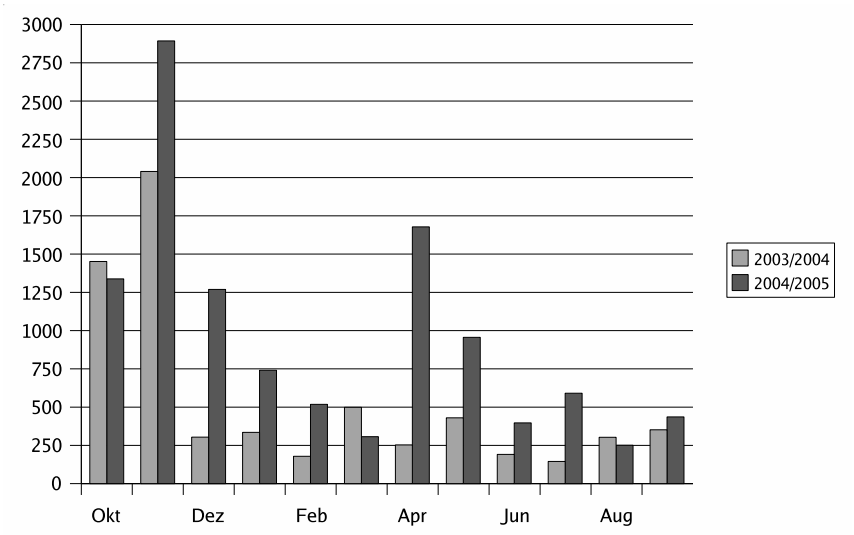


Abbildung 8.1: Ergebnisse der Weblog-Analyse für die beiden Runden des **ExperienceBook II**.

Fälle, alle anderen Fälle entstanden nur nach vorheriger Aufforderung dazu. Im Diskussionforum hingegen herrschte reger Betrieb zu verschiedenen Beiträgen.

Mündliches Feedback aus Gesprächen mit zwei studentischen Initiativen bestätigte diesen Trend. Die Studierenden empfanden es als schwierig, aus dem Bauch heraus neue Fälle zu schreiben. Infolgedessen wurde das **ExperienceBook II** um eine zusätzliche Webseite mit offenen Fragen ergänzt. Der Inhalt dieser Seite kommt aus regelmäßigen Analysen der anonym protokollierten Anfragen. Außerdem werden ab und zu bereits abgeschlossene Diskussionsbeiträge zu interessanten Themen in Fällen zusammengefasst. Beide Aktivitäten werden manuell getätigt, obwohl sie Potenzial zur Automatisierung bieten. Da sie bisher jedoch nicht erfolgreich waren, lohnt sich eine Automatisierung derzeit nicht.

Wenige Monate nach Einführung des Systems nahm die Benutzerrate rapide ab. Abbildung 8.1 zeigt die Zugriffsraten von Oktober 2003 bis September 2005. Die erste Runde des **ExperienceBook II** ist in hellgrau dargestellt, die zweite Runde (siehe unten) in dunkelgrau. Das Diskussionsforum der ersten Runde „überlebte“ bis Juli 2004 mit insgesamt 33 Diskussions-Strängen mit jeweils ein bis vierzehn Beiträgen. Über 60 Autorinnen und Autoren waren an den Diskussionen beteiligt.

Nach dem Scheitern des fallbasierten Teils des **ExperienceBooks II** wurde im Januar 2004 eine Fragebogenaktion durchgeführt, um den Hinter-

gründen auf die Spur zu kommen. Von den 298 per Email befragten Studierenden schickten zwar nur 15, also etwa 5%, den Fragebogen zurück, die Ergebnisse waren aber dennoch sehr aufschlußreich: Die ganze Community war auf ein Universitäts-externes System umgezogen. Ein paar findige Studierende hatten ein eigenes Diskussionsforum installiert und per Mund-zu-Mund-Propaganda bekannt gemacht. Die Gründe für den Umzug wurden in den Antworten zu den Fragebögen nicht genannt.

```
Bitte entsprechende Antworten durch [x] ankreuzen und ausgefuellten
Fragebogen bis 9. Februar 2004 per email an minor@informatik.hu-berlin.de
mit Stichwort "Umfrage" im Betreff!

1. Welche Webseiten haben Sie schon besucht?

[] das PI1-Forum
[] das ExperienceBook II
[] kein Interesse (danke fuer's Mitmachen, Fragebogen bitte jetzt
zurueckmailen)

2. Was gab den Anstoss fuer Ihren Besuch?

[] PI1-Webseite
[] Webseite des Lehrstuhls KI
[] Vorstellung des Projekts in der VL
[] Information per email im letzten Jahr
[] diese Umfrage-email
[] Mund-zu-Mund-Propaganda

3. Haben Sie etwas Nuetzliches im Forum gefunden?
(Frage bitte ueberspringen, wenn Sie die Seite noch nie besucht haben)

[] ja, was ich gesucht habe (zum Thema:   )
[] ja, beim Schmoekern (zum Thema:   )
[] nein, weil:
   [] Inhalte (zum Thema:   ) fehlen
   [] ich nicht wusste, wie die Seite zu benutzen ist
   [] das Angebot zu unuebersichtlich ist
   [] anderer Grund (bitte nennen:      )

4. Haben Sie etwas Nuetzliches im ExperienceBook gefunden?
(Frage bitte ueberspringen, wenn Sie die Seite noch nie besucht haben)

[] ja, was ich gesucht habe (zum Thema oder Fall-Nr.:   )
[] ja, beim Schmoekern (zum Thema oder Fall-Nr.:   )
[] nein, weil:
   [] Inhalte (zum Thema:   ) fehlen
```

```
   [] ich nicht wusste, wie die Seite zu benutzen ist
   [] das Angebot zu unuebersichtlich ist
   [] anderer Grund (bitte nennen:      )

5. Bei der (anonymen!) Auswertung der Log-Files hat sich gezeigt,
   dass an das ExperienceBook relativ viele Fragen gestellt wurden,
   aber relativ selten jemand einen Fall kommentiert hat bzw.
   einen neuen Fall eingetragen hat.
   Wie koennte man Ihrer Meinung nach dieses Problem am besten angehen?

[] durch Mixen der geloesten Faelle in den Antworten mit
   (gekennzeichneten) offenen Fragen, die von freiwilligen Studis
   beantwortet werden koennten
[] ich (meine mail:   ) waere bereit, mich an einer Patenschaft
   zum Beantworten offener Fragen zu beteiligen:
   [] fuer ein ExperienceBook zu PI2 im SoSe 04
   [] fuer eine zweite Runde zu PI1 im WiSe 04/05
[] ich wuensche mir Antworten von der Fachschaft

[] ich wuensche mir freiwilliges Engagement von Lehrkraeften
   (z.B. PI-Uebungsleiter)
[] andere Idee (bitte nennen:      )


weitere Kommentare zum Projekt (fakultativ):

Vielen Dank fuer Ihre Muehe! Fragebogen bitte bis 9. Februar zurueck an
minor@informatik.hu-berlin.de mit Stichwort "Umfrage" im Betreff!
```

Abbildung 8.2: Fragebogen mit fünf Fragen zum **ExperienceBook II**.

Die meisten Studierenden kannten das **ExperienceBook II** aus der Vorlesung, das heißt die persönliche Werbung war am effektivsten.

Die Möglichkeit zur Bewertung einzelner Fälle wurde nicht genutzt. [NAJ03] kommen in ihren Umfragen zu ähnlichen Ergebnissen. Für die Bewertung einzelner Fälle sind die in Kapitel 9 beschriebenen Methoden besser geeignet.

Mehrere Studierende hatten den Wunsch nach einer stärkeren Beteiligung von Lehrkräften bei der Erstellung von Falldaten.

Zweite Runde des Systems

Bei der zweiten Runde ab Oktober 2004 waren alle drei Handlungsempfehlungen im Gleichgewicht: Wir haben 1. erneut den Bedarf überprüft, 2. versucht die Haltung der Benutzer positiv zu beeinflussen und konnten 3. organisatorische Barrieren ausräumen, die für das Scheitern in der ersten Runde mit verantwortlich waren.

Wir haben nach dem Scheitern der ersten Runde zwei psychologische Barrieren vermutet und die Lehren daraus gezogen:

- *Die Machtverhältnisse*: Wir haben in der zweiten Runde immer wieder betont, dass die Lehrkräfte nicht die Fälle und Kommentare von Studierenden lesen.

- *Der Zugriff vom normalen Arbeitskontext aus*: Wir platzierten einen zusätzlichen Link auf das **ExperienceBook II** vom Punkte- und Kursverwaltungssystem 'GOYA' aus. Dieses System wird von allen Studierenden beinahe täglich benutzt, um Übungsaufgaben herunterzuladen, die Gruppenarbeit zu organisieren oder den eigenen Punktestand zu kontrollieren.

Im Wintersemester 2004/2005 startete dann mit den neuen Erstsemestlern eine zweite Runde des **E**xperienceBook II. Wieder wurde das System an die Vorlesung „Praktische Informatik I“ angebunden und dieselbe Werbekampagne wie in der ersten Runde durchgeführt. Auch die technische Gestaltung des Systems und der Benutzeroberfläche blieb gleich, da sie in den Fragebögen positiv bewertet worden war.

Da die Beiträge der Studierenden in der ersten Runde so spärlich gekommen waren, wurden für die zweite Runde weitere Falldaten von Lehrkräften zur Verfügung gestellt, so dass die Anzahl der Fälle inzwischen bei ca. 60 liegt. An der Entwicklung der Zugriffsszahlen in Abbildung 8.1 kann man deutlich sehen, wann eine intrinsische Motivation in Form einer Werbekampagne oder durch eine nahende Klausur gegeben war. Seit die Goya-Webseite mit einem Link auf das **ExperienceBook II** ausgestattet ist, kommen etwa

ein Drittel aller Zugriffe kommt dort aus. Insgesamt liegen die Zugriffszahlen 2004/2005 höher als im Jahr zuvor. Es lohnt sich also, psychosoziale Faktoren wie Vertrauen, Benutzerpartizipation, Motivation der Benutzer und bequeme Erreichbarkeit des Systems in die Gestaltung und Einführung von EM-Systemen einzubeziehen.

Kapitel 9

Wissensbewahrung: Erfahrungswissen aktuell halten

Erfahrungswissen veraltet, wenn sich der Kontext verändert, in dem es gesammelt wurde. Deshalb verliert der Inhalt eines EM-Systems ohne Maintenence mit der Zeit an Wert:

Definition 9.0.1 (Maintenance einer Fallbasis)

> Maintenance heisst, eine Fallbasis während ihrer Lebenszeit zu aktualisieren und neu zu organisieren [RY97].

Anstelle von Wartung, Pflege oder Instandhaltung hat sich auch im Deutschen der Begriff Maintenance etabliert, so dass im folgenden das englische Wort in der femininen Form verwendet wird. Die Fragestellungen in diesem Kapitel sind:

- Wie kann man die Qualität des Erfahrungswissens in einem EM-System sicherstellen?
- Wer soll sich um die Aktualisierung veralteten Wissens kümmern?
- Wann soll das geschehen?
- Gibt es leichtgewichtige Maintenance-Strategien für Assistenzsysteme?

In der Literatur wurden in letzter Zeit einige Rahmenwerke zum Thema Maintenance für fallbasierte Systeme veröffentlicht [Wil01] [RB03] [NAT01]. Umfangreiche Prozessmodelle und Checklisten gehören zu den darin vorgestellten Maintenance-Strategien. In der von mir betreuten Diplomarbeit von Alexandre Hanft [Han04] und in unserem Workshop-Beitrag [HM05] geht es

darum, eine leichtgewichtige Maintenance-Strategie zu entwickeln. Leichtgewichtig meint, dass der zeitliche Aufwand, der dafür betrieben werden muss, möglichst niedrig ist, und dass die Strategie von den beteiligten Personen sehr schnell verstanden werden kann, auch wenn sie keine Experten im Fallbasierten Schließen sind.

Im folgenden Kapitel wird eine leichtgewichtige Maintenance-Strategie für textbasierte Falldaten vorgestellt, bei der die Organisation der Maintenance automatisiert ist, die Durchführung hingegen manuell geschehen muss. Das Hintergrundwissen wird von dieser Strategie nicht erfasst, sondern muss nach wie vor von einem Administrator erweitert werden (siehe auch Kapitel 5.4 und 6.4). Kapitel 9.2 erläutert eine Fallstudie zu dieser Strategie am Anwendungsbeispiel **ExperienceBook I**.

9.1 Collaborative Maintenance

Collaborative Maintenance kann man übersetzen als „gemeinschaftliche Wissenspflege“. Die Grundidee ist ähnlich zum Community-Gedanken bei Wikipedia [Wik05], nämlich dass der Personenkreis der Nießnutzer des Systems auch verantwortlich ist für die Bereitstellung und Maintenance von Wissensinhalten.

Der Begriff Collaborative Maintenance wurde von Maria Angela Ferrario geprägt [FS00]:

Definition 9.1.1 (Collaborative Maintenance (CM) einer Fallbasis)

> *Collaborative Maintenance* ist eine verteilte Maintenance-Strategie, bei der die Benutzer selbst die Möglichkeit haben, Wissen in Fällen zu editieren. CM regelt die Begutachtung von neuem Fallwissen und bietet Verfahren zum Akzeptieren oder Verwerfen von Fällen.

Ferrario entwickelte als Anwendungsbeispiel das fallbasierte System Ulysses für eine Internet-Community. Ulysses gibt Empfehlungen für Restaurants, Pubs und andere Lokale in Dublin. Alle Benutzer dürfen Restaurant-Kritiken schreiben, ausgewählte Mitglieder korrigieren diese Kritiken. Erst wenn ein Konsens erzielt wurde — also mehrere Fassungen hinreichend übereinstimmen, wird die Empfehlung für die ganze Community zur Verfügung gestellt.

Zur kollaborativen Strategie gibt es zwei Alternativen: Erstens die traditionelle Maintenance durch Administratoren oder zweitens ein Patenschaftsmodell. Beim Patenschaftsmodell sind bestimmte Personen für einen festgelegten Teilbereich des Erfahrungswissens zuständig und organisieren Mainte-

nance-Prozesse dafür. Wikipedia [Wik05] arbeitet nach dem Patenschaftsmodell. Sowohl das Administratoren- als auch das Patenschaftsmodell haben den Nachteil, dass es für die Verantwortlichen zeitaufwändig ist. Wenn geeignete Personen nicht zur Verfügung stehen, bietet Collaborative Maintenance die Möglichkeit, dass dennoch Maintenance-Prozesse angestoßen und ausgeführt werden.

Ferrarios Modell lässt sich auch auf fallbasierte Assistenzsysteme übertragen. Allerdings müssen einige Anpassungen vorgenommen werden, da Ulysses mit sehr einfach strukturierten Falldaten arbeitet. Eine Empfehlung in Ulysses besteht nur aus fünf Attributen, die vordefinierte linguistische Werte annehmen können. Ein Gutachten bei Ferrario ist einfach eine neue Wertebelegung der fünf Attribute. Pro Gutachten wird die Abweichung der Werte vom Original berechnet. Diese Abweichungen werden dann zu einer Gesamtbewertung des Falls kombiniert. Weichen die Gutachten nicht zu stark voneinander ab, werden die Mittelwerte der vergebenen Attributwerte genommen und daraus der „korrigierte“ Fall gebildet. Eine Falldarstellung, die nur Attribute mit vorher festgelegten Skalen für die Werte kennt, eignet sich nicht zur Beschreibung von Erfahrungswissen in Assistenzsystemen. Dort benötigt man Textabschnitte und frei wählbare Attributwerte (siehe auch Kapitel 4.3).

Im folgenden wird ein eigenes Collaborative-Maintenance-Modell für textbasierte Falldaten vorgestellt (siehe [Han04]). Es arbeitet mit Lektoren, nie neue und geänderte Fälle begutachten. Sie benoten einen Fall mit Hilfe einer Punkteskala. Übersteigt die Durchschnittsnote einen bestimmten Schwellwert, wird die Einreichung genehmigt und ins System integriert.

Im folgenden werden die Aufgaben der Lektoren, die Zuordnung von Fällen und Lektoren, die Auslöser einer Begutachtung und die Steuerung des Begutachtungsprozesses diskutiert.

Tabelle 9.1: Punkteskala für die Benotung einer Fallrevision

5 Punkte	exzellente Lösung, erfolgreich ausprobiert, sehr gut erklärt
4 Punkte	gute Lösung, funktioniert, gut erklärt
3 Punkte	Lösung ausreichend, aber umständlich, Erklärung knapp
2 Punkte	nur teilweise Lösung / enthält Fehler
1 Punkt	unzureichend / funktioniert nicht
0 Punkte	Fall ist überflüssig, kann gelöscht werden

Aufgaben der Lektoren

Jeder neue Fall und jede neue Revision eines bereits integrierten Falls muss von mindestens einem Lektor positiv begutachtet werden. Dazu erzeugt er ein sogenanntes Gutachten.

Definition 9.1.2 (Gutachten (Review) für eine Fallrevision)

> Ein Gutachten bewertet den Inhalt einer Fallrevision. Es besteht aus einer Note der Punkteskala in Tabelle 9.1 und einem optionalen Textkommentar für den Autor des Falls.

Daneben kann der Lektor, der das Gutachten erstellt, den Fall auch selbst verbessern.

Der Schwellwert für die Akzeptanz einer Fallrevision liegt bei drei Punkten. Fallrevisionen, die im Durchschnitt aller Gutachten drei oder mehr Punkte erreicht haben, werden automatisch genehmigt. Alle anderen werden abgelehnt. Fälle mit Bewertung 0 werden sofort gelöscht. Hat ein Fall einen besseren Durchschnitt, kommt damit aber nicht auf drei Punkte, verbleibt er im Review-Prozess. Dort kann er zu einer neuen Revision weiterentwickelt werden oder auf bessere Gutachten warten.

Zuordnung der Fälle und Lektoren

Alle Fälle, die zur Begutachtung vorgesehen sind, müssen einem oder mehreren Lektoren zugeteilt werden.

In dieser Arbeit wird eine sehr einfache Zuordnung von Hand gewählt, die aber für grössere Datenbestände durch ein automatisches Verfahren ersetzt werden sollte. Hier haben die Lektoren Einsicht in eine Liste aller Fälle, die begutachtet werden müssen und wählen selbstständig Fälle zur Bearbeitung aus. Die Gutachten anderer Lektoren sind ebenfalls sichtbar. Die Vorteile dieses Auswahlverfahrens sind, dass es sehr einfach umzusetzen ist und dass

Lektoren nur Fälle begutachten müssen, die sie selbst ausgesucht haben. Diese Form der Selbstbestimmung hat aber auch gewisse Nachteile. Erstens muss wohl hin und wieder einmal jemand die Initiative ergreifen, um die Lektoren an die Gutachtertätigkeit zu erinnern. Zweitens könnten Fälle liegenbleiben, die zum Beispiel langweilig oder außerhalb des Kompetenzbereichs der Lektoren sind. Auch die Anzahl der Gutachten pro Fall ist damit variabel. Es muss mindestens ein Gutachten in die Gesamtwertung eingehen.

Eine Idee für eine automatische Lösung der Zuordnung ist eine Ähnlichkeitsberechnung zwischen einem zu begutachtenden Fall und der gesamten Fallbasis. Die Autoren mit den ähnlichsten Fällen können dann gebeten werden, Lektoren des Falls zu werden. Eine Portalsoftware kann solch eine Idee sinnvoll unterstützen und sogar potentielle Lektoren per Email informieren. Dieser Aufwand lohnt sich natürlich nur für eine grosse Anzahl von Autoren und Fällen. Die Fallstudie (siehe unten) hat gezeigt, dass ca. 80 Fälle, die gleichzeitig im Review-Prozess stecken, gerade noch zu überschauen sind. Eine Grenze für die Anzahl von Lektoren ist abhängig von der kulturellen und sozialen Situation, zum Beispiel davon, ob alle Lektoren in persönlichem Kontakt stehen und sich für die Wissensbestände verantwortlich fühlen oder davon, ob sie die Kompetenzen und das Engagement der Mit-Lektoren einschätzen können. Eine automatische Zuteilung bietet sich auch an, wenn verschiedene Zugriffsbeschränkungen für Teilbereiche der Wissensbestände gelten sollen.

Automatische Auslöser eines Review-Prozesses

Der Bedarf für eine Kontrolle oder Nachkontrolle wird auf der Seite der Fälle festgestellt. Kommt ein neuer Fall oder eine neue Revision eines Falls ins System, wird er automatisch in den Review-Prozess geleitet. Aber auch altgediente Fälle sollten regelmässig nachkontrolliert werden.

Es gibt verschiedene Möglichkeiten, wann ein alter Fall automatisch in den Review-Prozess geschickt wird. Man kann Zugriffsstatistiken auswerten und wenig benutzte Fälle kontrollieren lassen. Oder man gibt den Autoren die Aufgabe, ein „Mindesthaltbarkeitsdatum“ eines Falles anzugeben, damit er bei Ablauf erneut begutachtet wird. In dieser Arbeit gilt ein einfacher Zeitstempel. Jeder Fall, der länger als zwei Jahre nicht begutachtet wurde, wird aus dem laufenden Betrieb genommen und auf die Review-Liste gesetzt. Dies ähnelt dem TÜV für Automobile. Der gewählte Zeitraum hängt natürlich von der erwarteten Veränderung des Anwendungsgebiets ab. Dem TÜV-Vorbild folgend könnte man unsere Strategie erweitern und die Fälle schon etwas vor Ablauf des Zeitstempels in den Review-Prozess schicken, und parallel dazu noch im laufenden Betrieb lassen.

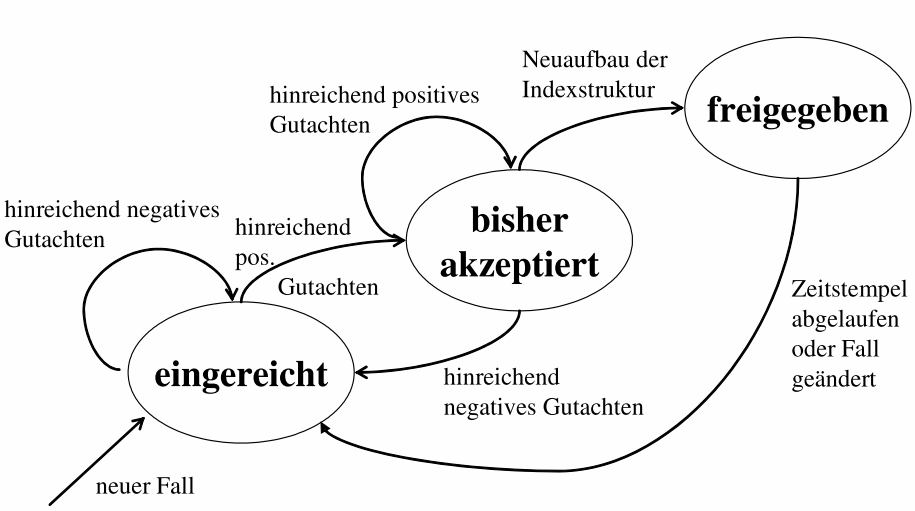


Abbildung 9.1: Statusangaben einer Fallrevision und Statusübergänge.

Damit die früheren Gutachten den Fall nicht sofort für eine Wiedereingliederung ins System qualifizieren, wird für den Begutachtungs-Prozess eine neue Revision des Falls erzeugt.

Steuerung der Reviewprozesse durch Statusinformationen

Die verschiedenen Revisionen eines Falls werden mit Hilfe von Statusinformationen verwaltet. Es sollen ja nur positiv bewertete Fälle im Retrieval Verwendung finden. Deshalb gibt es drei verschiedene Statusangaben für Fallrevisionen: *eingereicht*, *bisher akzeptiert* und *freigegeben* (siehe Abbildung 9.1).

Der initiale Status einer Fallrevision ist *eingereicht*. Den höheren Status *bisher akzeptiert* bekommt sie durch Gutachten mit positiver Bewertung. Der höchste Status ist *freigegeben* und wird erst bei einem Neuaufbau der Indexstruktur erreicht. Soll ein Fall erneut begutachtet werden (siehe oben), wird eine neue Revision mit dem Status *eingereicht* erzeugt, d.h. die ehemals positive Bewertung des Falls wird aufgehoben. Der Fall kann dann durch neue Gutachten wieder akzeptiert werden. *Bisher akzeptierte* Fälle können in den Status *eingereicht* zurückfallen, wenn ein negatives Gutachten den Durchschnitt hinreichend weit nach unten treibt.

Der Übergang vom Zustand *bisher akzeptiert* in *freigegeben* geschieht von selbst, wenn die Indexstruktur der Fallbasis neu aufgebaut wird. Dies wird in regelmässigen Zeitabständen automatisch angestossen und kann dazu führen, dass Fälle schon durch ein einziges Gutachten in den Zustand *freigegeben*

gelangen können. Die leichtgewichtige Strategie ohne menschlichen Organisator kann also nur gelingen, wenn die Lektoren konstruktiv und qualifiziert arbeiten.

Vier verschiedene Sichten benutzen die Statusinformationen:

1. Die Retrieval-Sicht verwendet jeweils die neueste Fallrevision der *freigegebenen* Revisionen.

2. Bei der Retrieval-Ergebnis-Sicht werden *freigegebene* Revisionen durch neuere *bisher akzeptierte* ersetzt.

3. In der Editier-Sicht wird statusunabhängig immer die neueste Revision eines Falls angezeigt.

4. Die Review-Sicht zeigt Fälle mit Status *eingereicht* oder *bisher akzeptiert*.

Sobald ein Fall verändert wird, wird eine neue Revision erzeugt, das heißt die alten Gutachten gelten nicht mehr und die neue Revision befindet sich im Status *eingereicht*. Die Zwischenrevisionen tauchen je nach Begutachtungsstand in keiner Sicht auf oder nur in der Retrieval-Ergebnis-Sicht.

Für kompliziertere Begutachtungsverfahren ist eine Erweiterung des Wertebereichs für Statusangaben möglich.

9.2 Ergebnisse einer Maintenance-Fallstudie am ExperienceBook I

Die oben beschriebene kollaborative Maintenance-Strategie wurde in einer Fallstudie zum **ExperienceBook I** ausprobiert. Das **E**xperienceBook I ist ein fallbasiertes Anwendungsbeispiel, das seit 1998 an unserem Lehrstuhl im Einsatz ist, um erfahrene Computeranwender und Systemadministratoren bei ihrer täglichen Arbeit zu unterstützen. Technisch ist es identisch mit dem **ExperienceBook II** (siehe S. 122). Der Inhalt der Falldaten ist ähnlich zu denen im **ExperienceBook II**, ist aber in den meisten Fällen etwas komplexer und mit mehr Fachbegriffen durchsetzt. Die Fälle geben zum Beispiel Tipps zur Installation einer bestimmten Software oder erläutern die Behebung eines Druckerproblems aus Anwendersicht.

Für die Fallstudie wurden 89 Fälle untersucht, die in den letzten sechs Jahren entstanden sind. In dieser langen Laufzeit sind viele Veränderungen passiert:

1. Der Personenkreis, der mit dem System zu tun hat, hat sich gewandelt. Die Autoren der Falldaten waren Mitglieder derselben Arbeitsgruppe, manche hatten zum Zeitpunkt der Fallstudie die Gruppe bereits verlassen. Dasselbe gilt für die Benutzer, die aus der gleichen Community stammen und von denen viele auch Autoren sind.

2. Selbstverständlich hat sich im Lauf der sechs Jahre auch die Anwendungsdomäne verändert. Es sind zum Beispiel neue Drucker und andere Geräte angeschafft worden. Natürlich hat sich die Software stark verändert, manche neuen Versionen von Anwendungssoftware haben alte ersetzt, ganze Programme wurden entfernt und neue sind hinzugekommen.

Sporadisch waren Fälle von ihrem Autor überarbeitet und an die neuen Gegebenheiten angepasst worden, die meisten Falldaten lagen aber noch in der originalen Revision vor.

Im Zug der Maintenance-Studie wurde das System selbst an einigen Stellen modernisiert, aber das ist nicht Gegenstand dieser Arbeit.

Zur Intensivierung der „normalen“ Maintenance-Strategie wurden alle Fälle, die älter als ein Jahr waren, aus dem Retrieval genommen und in den Review-Prozess geleitet. Fünf Lektoren wurden unter den Autoren ausgewählt und um Mitarbeit bei der Begutachtung gebeten. In der anschliessenden Review-Runde entstanden in zwei Wochen 120 Gutachten. Die Fallbasis war vollständig abgedeckt, jeder Fall bekam ein bis drei Gutachten. 46 mal wurden Kommentare für die Autoren verfasst.

Ergebnisse

Die Bewertung ergab 12 Löschempfehlungen, 69 akzeptierte Fälle und 8 nicht akzeptierte. Es war keine Korrelation zwischen dem Alter der Fälle und ihrer Bewertung zu erkennen. Also können auch altgediente Fälle nach wie vor aktuell sein.

Die Lektoren benötigten pro Review zwei bis vier Minuten, zum Beispiel bearbeitete eine Lektorin 50 Fälle in drei Stunden.

Fazit

Die Befragung der Lektoren hat ergeben, dass die oben beschriebene Maintenance-Strategie in dieser Fallstudie gut funktioniert hat. Die Begutachtung der gesamten Fallbasis auf freiwilliger Basis und häufige Vergabe von Textkommentaren bestätigt die Zuordnungsmethode von Fällen und Lektoren - jedenfalls für eine Autoren- und Lektorenschaft, die sich persönlich kennt und

kooperiert. Der Zeitaufwand war sehr niedrig und die Idee des leichtgewichtigen Prozessmodells liess sich somit durchhalten.

Der hohe Anteil nicht akzeptierter Fälle (22%) zeigt wie erwartet, dass systematische Maintenance-Aktivitäten dringend nötig sind.

Ob konzertierte Maintenance-Aktionen von Zeit zu Zeit oder eine ständige Maintenance besser funktionieren, bleibt abzuwarten. Ein Ergebnis der Befragung der Lektoren ergab, dass die Menge von 89 zu begutachtenden Fällen gerade noch überschaubar war, um sich selbst Fälle zum Review auszusuchen. Es wurde bezweifelt, dass dies bei grösseren Fallbasen noch möglich ist.

Zwei Empfehlungen lassen sich aus der Fallstudie ableiten:

1. Es hat sich bewährt, den Maintenance-Prozess akribisch zu erklären, um die Annahme einer solchen Collaborative-Maintenance-Strategie zu fördern.

2. Bei Fallbasen mit über hundert Fällen sollte ein automatisches Verfahren für die Zuordnung von Lektoren und Fallwissen zum Einsatz kommen. Eine übliche Portalsoftware ist dafür geeignet.

Teil III

Einordnung

Kapitel 10

Verwandte Arbeiten

In diesem Kapitel ordne ich meine eigene Arbeit in den wissenschaftlichen Kontext ein. Das Verhältnis von TCBR zu *Information Retrieval* und *Natural Language Processing* habe ich bereits auf S. 53 und 79 diskutiert. Einige Bemerkungen zum *Ontology Merging* stehen auf S. 114. Im Folgenden beschreibe ich zunächst alternative EM-Ansätze, dann gehe ich auf einige Techniken ein, die an der Grenze zwischen EM-Systemen und bloßen Hilfsmitteln zum Erfahrungsmanagement liegen.

Die **ExperienceFactory** [BCR94] ist einer der ersten Ansätze, die sich systematisch mit dem Wiederverwenden von Erfahrungswissen in einer Organisation beschäftigt. Aus der von Basili et al. entwickelten Vision einer „Learning software organization“ (LSO) sind viele Derivate der **E**xperienceFactory entstanden, die sowohl die Modellierung von Prozessen, Rollen und Verantwortlichkeiten beinhalten als auch technische Systeme zur Unterstützung des Lernens in einer Organisation. **COIN** (Corporate Information Network) [NAAD02, ADH+01] ist eins dieser Systeme, das nahe verwandt mit den in dieser Arbeit vorgestellten Beispielanwendungen ist. Zum einen setzt es Fallbasiertes Schließen ein, um das im System gespeicherte Erfahrungswissen wiederzufinden. Zu anderen verbindet es wie **S**imLex (siehe Kapitel 5.3) heterogene Wissensbestände miteinander. Die Repräsentationsformen des Wissens sind wesentlich komplexer als in meinen Anwendungsbeispielen, sie beinhalten zum Beispiel grafische Geschäftsprozess-Modelle oder strukturierte „Lessons Learned“-Beschreibungen. Infolgedessen kann die Vernetzung des Wissens in **COIN** nicht wie in **SimLex** automatisch geschehen.

INRECA [BBF+98, Ber02] ist eine Methodik zur Entwicklung fallbasierter EM-Systeme für den industriellen Bereich. Diese Methodik ist durch Prozessmodelle auf drei Ebenen beschrieben:

1. auf der generischen Ebene, die ein sehr großes Spektrum an Anwendungen umfasst,

2. auf der Kochrezept-Ebene, auf der bestimmte Klassen von Anwendungen, zum Beispiel Produktkataloge oder Help-Desk-Systeme, betrachtet werden, und

3. auf der Ebene spezifischer Projekte.

Ein Prozess hat eine Eingabe, eine Ausgabe und eventuell mehrere alternative Methoden, mit denen er ausgeführt werden kann. **INRECA** unterstützt den ISO 900x-Standard und ist damit ein sehr mächtiges Framework, das für große Softwareprojekte angemessen, für meine eher experimentellen Anwendungsbeispiele aber zu zeitintensiv ist.

FRODO (Framework for Distributed Organizational Memories) [Els05] ist ein Rahmenwerk für kooperierende Organizational-Memory-Systeme. Der Agentengedanke trägt dazu bei, dass verschiedenste lokale KM-Lösungen miteinander gekoppelt werden können. Dazu werden drei Typen von Agenten und zusätzlich Wrapper für Wissensquellen benötigt: Workflow-Agenten nutzen schwach-strukturierte Workflows zur Spezifikation eines Informationsbedarfs und des dazu gehörigen Kontexts. Informationsagenten stillen jeweils einen bestimmten Informationsbedarf. Ontologie-Agenten pflegen das domänenspezifische Vokabular. Im Nachfolgeprojekt **EPOS** werden formale, persönliche Informationsmodelle untersucht, um eine individuelle Herangehensweise an das Wissensmanagement zu erhalten (vergleiche dazu meine Diskussion der ontologischen Ebene von Nonaka und Takeuchi auf S. 14 ff.). **FRODO** und **EPOS** sind lose verwandt mit den persönlichen Assistenzagenten aus Kapitel 7. Der große Unterschied zwischen beiden Agentensystemen besteht darin, dass mein Anwendungsbeispiel fremdes Hintergrundwissen integriert, während **FRODO** und **EPOS** mit Mediatoren und Wrappern arbeiten.

SIAM (Setup, Initialization, Application, Maintenance) [RB03] ist ein Rahmenwerk für Fallbasierte Systeme, das die ursprünglich vier Prozesse aus dem Modell von Aamodt und Plaza [AP94] (siehe auch S. 38) um zwei Prozesse erweitert: Den Prozess der Anwendung des Systems und den Maintenance-Process. **S**IAM ist in die **INRECA**-Methodik eingebettet, so dass insbesondere die Maintenance-Aufgaben auf den oben beschriebenen drei Ebenen als Maintenance-Strategie, auf der Kochrezept-Ebene und für spezifische Projekte operationalisiert sind. **SIAM** bringt die organisatorische Perspektive systematisch in die Entwicklungsmethodik ein. In meiner Arbeit entspricht der Wissensbewahrungsprozess der Maintenance-Aufgabe, wird aber leichtgewichtig umgesetzt (vergleiche die Diskussion auf S. 132 f.) und in eine ganzheitliche, kollaborative Herangehensweise eingebettet.

Stellvertretend für einige Dutzend erfolgreiche EM-Projekte in der Industrie, die fallbasierte Techniken einsetzen, nenne ich zwei kommerzielle Ent-

wicklungsumgebungen für fallbasierte EM-Anwendungen, nämlich **Empolis' AIS** (vormals **orenge**) [Emp05] und den **Kaidara Advisor** [Kai05], sowie zwei aktuelle prestige-trächtige Projekte in Großunternehmen: Bei **DaimlerChrysler** läuft seit 1996 die fallbasierte Help-Desk-Applikation **S.T.A.R. online** [HW05] für Angestellte und Vertragshändler von Chrysler. **S.T.A.R. online** ist ein klassisches fallbasiertes System, das strukturierte Falldaten über eine Retrieval-Schnittstelle im Web bereitstellt. Der **General Motors VRA** [MCG+03, MCG+04] legt den Schwerpunkt auf die Integration des fall- und ontologiebasierten Systems mit einem Werkzeug zur direkten elektronischen Kommunikation zwischen Experten. Die Logfiles dieser Kommunikation bilden die Grundlage für Falldaten; dies ist vergleichbar zu meinem Ansatz in Kapitel 5.2, allerdings ohne unsere automatisch erzeugten Querverweise. Eine Beobachtung während des Projekts, dass nämlich soziale Aspekte äußerst wichtig für den Erfolg eines EM-Systems sind, und intrinsische Motivierung sehr gut funktionieren kann, stützt meine Ergebnisse in Kapitel 8.

Es gibt einige populäre Systeme, die ich an der Grenze zwischen „echten" EM-Systemen, die ja Assistenzsysteme sind, und simplen EM-Hilfsmitteln wie FAQ-Datenbanken oder Verzeichnisstrukturen ansiedle. **Wiki**s [Wik05] sind kollaborative Webseiten mit enzyklopädischen Artikeln, die wie ein Lexikon gestaltet sind. Neben der allgemeinen **Wikipedia** gibt es auch gruppenspezifische **Wiki**s, die zum Beispiel als Kommunikationsplattform für ein bestimmtes Projekt dienen. Jeder Benutzer kann auf sehr einfache Weise Artikel ins System einstellen, Artikel weiterschreiben oder über Artikel diskutieren. Die organisatorischen Aspekte bei einem **Wiki** sind sehr ähnlich zu den in Kapitel 8 und 9 beschriebenen. Auch dort können organisatorische Barrieren die Nutzung des Systems verhindern; auch dort müssen die Wissensinhalte gepflegt werden, wenn das System über einen längeren Zeitraum nützlich sein soll. Der Fokus bei einem **Wiki** liegt nicht auf einer intelligenten Unterstützung der Benutzer, sondern lediglich auf dem Speichern und Bereitstellen von Webseiten. Eine interessante Weiterentwicklung von **Wiki**s ist das ebenfalls kostenlose **SnipSnap**-System [JJS05, Sni05] von Fraunhofer FIRST. **SnipSnap** bietet ein automatisches Vorschlagswesen für eine Vernetzung von **SnipSnap**-Seiten auf der Basis einer Markup-Sprache. Im Gegensatz dazu kann unser System **SimLex** eine vollautomatische Vernetzung von Falldaten vornehmen. Auch „Business Intelligence"-Lösungen wie „Data Warehouses", die Erfahrungswissen abspeichern und es zum Beispiel mit „Data Mining"-Verfahren durchsuchen, sind entfernte Verwandte von EM-Systemen. Dort steht die Integration von Informationen im Vordergrund, oft auch von riesigen Mengen von Informationen. Diese Systeme müssen sehr gut strukturiert werden, um die Benutzer nicht mit einer Flut von Informationen zu überschütten. EM-Ansätze hingegen arbeiten situations-spezifisch, so dass

die Ergebnisse genauer fokussiert sind. Eine Kombination beider Ansätze, so dass ein EM-System auf einem Data Warehouse aufbauen kann, ist eine interessante Möglichkeit für die Zukunft, die Vorteile beider miteinander zu verbinden.

Kapitel 11

Resümee und Ausblick

In dieser Arbeit wurde ein Rahmenwerk für ganzheitliches Erfahrungsmanagement entwickelt, das durch Assistenzsysteme für verschiedene EM-Prozesse technisch unterstützt wird. Das Rahmenwerk ist nach einer beispielgetriebenen, praxisorientierten Entwicklungsmethodik entstanden. Es verbindet ein wirtschaftswissenschaftliches Prozessmodell von Gilbert Probst et al. (siehe Kapitel 2.3) mit Techniken des Fallbasierten Schließens, der Ontologiebasierten Systeme und der Agentenorientierten Programmierung. Jeder Kernprozess des Modells wird in der Arbeit auf die besonderen Anforderungen im Erfahrungsmanagement hin untersucht. Dabei wird überprüft, welche Teilprozesse sinnvoll durch ein EM-System unterstützt oder gar automatisiert werden können. Die fallbasierten Anwendungsbeispiele in Teil II der Arbeit liefern die empirische Bestätigung für diese Ansatzpunkte. Sie stammen aus Kooperationen mit Industriepartnern und aus kleinen, experimentellen Projekten mit Studierenden und Kollegen.

Der **wichtigste Beitrag der Arbeit** ist es, dass Entwickler von EM-Systemen das darin vorgestellte Rahmenwerk als Leitfaden für die Analyse möglicher Aufgaben des Systems nutzen können und damit beim Entwurf die ganzheitliche Perspektive systematisch verfolgen können. Die empirischen Untersuchungen innerhalb der Arbeit haben gezeigt, dass es sich durch höhere Benutzungsraten auszahlt, bei der Systementwicklung neben der technischen Sicht auch organisatorische und psycho-soziale Aspekte zu berücksichtigen.

Daneben liefert die Arbeit einen **neuen theoretischen Beitrag**, der sowohl für fallbasierte als auch für klassische ontologiebasierte Systeme von Interesse ist: Durch Operationen auf Relationen können automatisch Veränderungen an einer Ontologie oder einem Ähnlichkeitsmodell vorgenommen werden, die auf empirischen Eigenschaften der Relationen beruhen.

Außerdem trägt die Arbeit mit mehreren kleinen Neuerungen zur **Forschung im Fallbasierten Schließen** bei:

- Kapitel 5.2 beschreibt ein neues automatisches Verfahren, um heterogene Wissensquellen in ein fallbasiertes System zu integrieren und durch Querverweise miteinander zu verbinden.

- Kapitel 5.4 stellt mit **MultiLingual** einen Ansatz des Fallbasierten Schließens für Texte aus mehreren Sprachen vor.

- Das Life-Cycle-Modell für Falldaten aus Kapitel 6.1 eröffnet durch Statusinformationen über den Reifegrad eines Falls neue Möglichkeiten für den *authoring support* und die Maintenance (siehe auch Kapitel 9).

- Kapitel 7 beschreibt die Integration von Multi-Agenten-Systemen und Fallbasiertem Schließen dergestalt, dass Agenten persönliche Case Retrieval Nets besitzen und Teile davon untereinander austauschen können. In dieser Arbeit kann nur ein erster Ansatz dazu vorgestellt werden, aus dem sich viele neue Forschungsfragen ergeben haben (siehe unten).

- In Kapitel 8 werden die Auswirkungen bestimmter organisatorischer und soziotechnischer Maßnahmen auf die Akzeptanz des EM-Systems **ExperienceBook** untersucht. Auch hier öffnet sich ein weites Feld für weitere Untersuchungen in der interdisziplinären Forschung beispielsweise mit Psychologen und Wirtschaftswissenschaftlern.

An verschiedenen Stellen der Arbeit habe ich schon Hinweise auf mögliche weitere Forschungsarbeiten gegeben. Im folgenden Ausblick werden die wichtigsten davon gebündelt dargestellt und um Ideen für die fernere Zukunft ergänzt.

Die Open Source-Bewegung bietet interessante Ansatzpunkte für den Wissenserwerb, die Wissensentwicklung und die Wissensbewahrung. Es gibt bereits erste frei verfügbare CBR-Systeme, zum Beispiel **IUCBRF** (Indiana University Case-Based Reasoning Framework) [BL05]), empolis' **SIP** (SEKT Integration Platform, eine kleinere Variante des kommerziellen **IAS**) [Emp05] und **jCOLIBRI** [jCO05]. Eine gute Erweiterung dieser Ansätze wäre die Erarbeitung von frei verfügbarem, modular organisiertem und erweiterbarem Hintergrundwissen, aus dem die Benutzer sich wie aus einer Bibliothek mit den benötigten Modulen bedienen können. Die Entwicklung eines austauschbaren Formats oder entsprechender Wrapper reicht dafür nicht aus, sondern es müssen auch Algorithmen zur Integration mehrerer Module gefunden werden, die zum Beispiel Konflikte bei Überschneidungen lösen können. Ich betrachte die Arbeiten in Kapitel 6.3 als einen ersten Schritt in dieser Richtung.

Die Agenten mit persönlichen CRNs in Kapitel 7 bieten Stoff für weitere Arbeiten im Bereich der Integration von CRNs:

- Analyse der Übergänge von altem und neuem Teilnetz, um Relationen zu erweitern (vergleiche auch Kapitel 6.3),
- Integration von Verfeinerungstechniken für Ontologien wie bei Hahn et al. [HS04] oder aus dem „ontology learning“ [MS04],
- Bewertung fremden Wissens, zum Beispiel durch das Maß an Vertrauen in Partneragenten.

Beim Fallbasierten Schließen für Texte wurde bislang eine Adaption der Fälle immer ausgeklammert. Es gibt aber Anwendungsbereiche, in denen kleine Anpassungen von Texten schon ausreichend und nützlich wären, zum Beispiel die Erzeugung von Zusammenfassungen bei der Routenplanung mit Hilfe einer Fallbasis von Mustertexten oder zur Generierung von Staumeldungen oder Horoskopen. Eine Anfrage, zum Beispiel eine Liste von Attribut-Werte-Paaren zur Beschreibung eines Horoskops, könnte eine ähnliche Beschreibung aus der Fallbasis zu Tage liefern, für die es einen Text als Lösung gibt. Das Ziel ist, durch Anpassung dieses Textes eine Textbeschreibung für die Anfrage zu generieren. Flache Sprachgenerierungsmethoden wie in **TEMSIS** [Bus05] oder **D2S** [The03, Bos05], die mit Mustersätzen und einzelnen austauschbaren Satzteilen arbeiten, dienen als Vorbild für die Idee. Beispielsweise könnten Information-Extraction-Algorithmen so abgewandelt werden, dass sie Sätze an den Stellen leicht verändern, an denen ein Extraktions-Muster erkannt wurde, indem sie einen Begriff des Mustertextes ersetzen. Dieses Wissen zur Transformation der Lösung könnte beispielsweise in Form von Tags an die entsprechenden Satzteile der Falltexte annotiert werden.

Auch auf der praktischen Seite gibt es noch Bedarf an weiteren Arbeiten: Größere Fallstudien zum Erfahrungsmanagement mit return-on-invest-Erhebungen müssen in Zukunft entstehen, wenn das Forschungsgebiet sich weiterentwickeln und kommerziell erfolgreich werden soll.

Literaturverzeichnis

[ADH+01] Althoff, Klaus-Dieter ; Decker, Björn ; Hartkopf, Susanne ; Jedlitschka, Andreas ; Nick, Markus ; Rech, Jörg: Experience Management: The Fraunhofer IESE Experience Factory. In: Perner, Petra (Hrsg.): *Data Mining, Data Warehouse and Knowledge Management. Proc. Industrial Conference on Data Mining*. Leipzig, Germany : Institute for Computer Vision and applied Computer Sciences, 2001 (Institut für Bildverarbeitung und angewandte Informatik IBai Report). – ISSN 1431 – 2360, S. 12 – 29

[AH04] Antoniou, Grigoris ; Harmelen, Frank van: Web Ontology Language: OWL. In: Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.): *Handbook on Ontologies*. Heidelberg : Springer-Verlag, 2004, Kapitel 4, S. 67 – 92

[AL04] Angele, Jürgen ; Lausen, Georg: Ontologies in F-Logik. In: Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.): *Handbook on Ontologies*. Heidelberg : Springer-Verlag, 2004, Kapitel 2, S. 29 – 50

[AN95] Aamodt, Agnar ; Nygård, Mads: Different roles and mutual dependencies of data, information, and knowledge. In: *Data and Knowledge Engineering* 16 (1995), S. 191 – 222

[AP94] Aamodt, Agnar ; Plaza, Enric: Case-Based Reasoning: Foundational Issues, Methodological Variations , and System Approaches. In: *AI Communications* 7 (1994), Nr. 1, S. 39 – 59

[Arc04] Arcos, Josep L.: Improving the Quality of Solutions in Domain Evolving Environments. In: Funk, Peter (Hrsg.) ; González Calero, Pedro A. (Hrsg.): *Advances in Case-Based Reasoning: 7th European Conference, ECCBR 2004*. Berlin : Springer-Verlag, 2004 (LNAI 3155), S. 464 – 475

[BA01] Brüninghaus, Steffi ; Ashley, Kevin: The Role of Information Extraction for Textual CBR. In: Aha, David (Hrsg.) ; Watson, Ian (Hrsg.): *Case-Based Reasoning Research and Development, Proceedings of the ICCBR'01*, Springer-Verlag, 2001 (LNAI 2080), S. 74 – 89

[BBF+98] Bergmann, Ralph ; Bree, Sean ; Fayol, Emmanuelle ; Göker, Mehmet ; Manago, Michel ; Schmitt, Sascha ; Schumacher, Jürgen ; Stahl, Armin ; Wess, Stefan ; Wilke, Wolfgang: Collecting Experience on the Systematic Development of CBR Applications Using the INRECA Methodology. In: Barry Smyth, Pádraig C. (Hrsg.): *Advances in Case-Based Reasoning. Proceedings of the Fourth European Workshop on Case-Based Reasoning (EWCBR-98).* Berlin : Springer-Verlag, 1998 (LNAI 1488), S. 460 – 470

[BBG+99] Bergmann, Ralph ; Breen, Seen ; Göker, Mehmet ; Manago, Michel ; Wess, Stefan: *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology.* Berlin : Springer-Verlag, 1999 (LNAI 1612)

[BCR94] Basili, Victor R. ; Caldiera, Gianluigi ; Rombach, H. D.: Experience Factory. In: Marciniak, John J. (Hrsg.): *Encyclopedia of Software Engineering* Bd. 1. John Wiley & Sons, 1994, S. 469 – 476

[Bel04] Bell, Christina: *SimLex: Experience Management für die Simulationsliga des RoboCups.* Berlin, Institut für Informatik, Humboldt-Universität zu Berlin, Diplomarbeit, 2004

[Ber02] Bergmann, Ralph: *Experience Management: Foundations, Development Methodology, and Internet Based Applications.* Berlin : Springer-Verlag, 2002 (LNAI 2432)

[BHA+00] Braun, Hubert von ; Hesse, Wolfgang ; Andelfinger, Urs ; Kittlaus, Hans-Bernd ; Scheschonk, Gert: Conceptions are Social Constructs - towards a Solid Foundation of the FRISCO Approach. In: Falkenberg, Eckhard D. (Hrsg.) ; Lyytinen, Kalle (Hrsg.) ; Verrijn-Stuart, Alex A. (Hrsg.): *Information System Concepts: An Integrated Discipline Emerging, IFIP TC8/WG8.1 International Conference on Information System*

Concepts: An Integrated Discipline Emerging (ISCO-4), September 20-22, Univerity of Leiden, The Netherlands Bd. 164, Kluwer, 2000, S. 61–73

[BL05] Bogaert, Steven ; Leake, David B.: *IUCBRF – Indiana University Case-Based Reasoning Framework.* Internet: http://www.cs.indiana.edu/ sbogaert/CBR/index.html, 2005. – [Stand: November 2005]

[Bos05] Bosch, Antal van d.: Memory-based understanding of user utterances in a spoken dialogue system: Effects of feature selection an co-learning. In: Brüninghaus, Steffi (Hrsg.): *Workshop Proceedings of the 6th International Conference on Case-Based Reasoning, ICCBR-05.* Chicago, Illinois : DePaul University, 2005, S. 85 – 94

[BR01] Burkhard, Hans-Dieter ; Richter, Michael M.: On the Notion of Similarity in Case-Based Reasoning and Fuzzy Theory. In: Pal, Sankar K. (Hrsg.) ; Dillon, Tharam S. (Hrsg.) ; Yeung, Daniel S. (Hrsg.): *Soft Computing in Case-Based Reasoning.* Springer-Verlag, 2001, S. 29 – 46

[Bus05] Busemann, Stephan: *TEMSIS.* Internet: http://www.dfki.de/service/nlg-demo/, 2005. – [Stand: November 2005]

[Cox99] Cox, Michael T.: *AAAI-99 Workshop on Mixed-Initiative Intelligence.* Internet: http://www.cs.wright.edu/∼mcox/mii/, 1999. – [Stand: 8. Juli 2005]

[Cro82] Crocker, David H.: *RFC 822, Standard for the Format of ARPA Internet Text Messages.* Internet: http://www.ietf.org/rfc/rfc822.txt, 1982. – [Stand: 6. September 2005]

[DC04] Delany, Sarah J. ; Cunningham, Padraig: An Analysis of Case-Base Editing in a Spam Filtering System. In: Funk, Peter (Hrsg.) ; González-Calero, Pedro A. (Hrsg.): *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004, Proceedings*, Springer-Verlag, 2004 (LNCS 3155), S. 128 – 141

[Dub05] *Dublin Core Metadata Initiative.* Internet: http://www.dublincore.org, 2005. – [Stand: 5. September 2005]

[DWar] Dzida, Wolfgang ; Wandke, Hartmut: Software-Ergonomie – Gestalten und Bewerten interaktiver Systeme. In: Zimolong, B. (Hrsg.) ; Konradt, U. (Hrsg.): *Enzyklopädie der Psychologie* Bd. Ingenieurpsychologie D-III-2. Göttingen : Hogrefe, to appear

[Els05] Elst, Ludger van: *Distributed Enterprise Knowledge Management: Balancing Individual and Organizational Needs.* Internet: http://www.dfki.uni-kl.de/ elst/presentations/AMKM 2005 @ AAMAS.pdf, 2005. – [Stand: 5. September 2005]

[Emp05] *Empolis Arvato.* Internet: http://www.empolis.de, 2005. – [Stand: 6. September 2005]

[Epp05] Eppler, M.: *Definition von Wissensmanagement.* Internet: http://www.knowledgemedia.org, 2005. – [Stand: April 2005]

[Fel98] Fellbaum, Christiane ; Fellbaum, Christiane (Hrsg.): *Wordnet: An Electronic Lexical Database.* Cambridge, Mass. : The MIT Press, 1998

[Fen01] Fensel, Dieter: *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Heidelberg Berlin : Springer-Verlag, 2001

[FS00] Ferrario, Maria A. ; Smyth, Barry: Collaborative Maintenance - A Distributed, Interactive Case-Base Maintenance Strategy. In: Blanzieri, Enrico (Hrsg.) ; Portinale, Luigi (Hrsg.): *Advances in Case-Based Reasoning: 5th European Workshop, EWCBR 2000.* Heidelberg : Springer-Verlag, September 2000 (LNAI 1898), S. 393–405

[GA05] Gupta, Kalyan M. ; Aha, David W.: Knowledge Extraction for Conversational Case-Based Reasoning. In: Aha, David W. (Hrsg.) ; Gupta, Kalyan M. (Hrsg.) ; Pal, Sankar K. (Hrsg.): *Case-Based Reasoning in Knowledge Discovery and Data Mining*, John Wiley & Sons, 2005, S. to appear

[GMU+04] Gronau, Norbert ; Müller, Claudia ; Uslar, Matthias ; Weber, Edzard ; Winkler, Torsten: KMDL: Eine Sprache zur Modellierung wissensintensiver Prozesse nach dem SECI-Modell.

In: Abecker, Andreas (Hrsg.) ; Minor, Mirjam (Hrsg.) ; Stojanovic, Ljiljana (Hrsg.): *Lernen Wissen Adaptivität, Workshop Proceedings*, 2004, S. 238 – 244

[Gru93] Gruber, Tom R.: A translation approach to portable ontology specifications. In: *Knowledge Acquisition* 2 (1993), Nr. 5, S. 199–220

[GS03] Geiger, Daniel ; Schreiögg, Georg: Wenn alles Wissen ist, ist Wissen am Ende nichts ?! In: *DBW* 63. Jhrg. (2003)

[GW99] Ganter, Bernhard ; Wille, Rudolf: *Formal Concept Analysis. Mathematical Foundations*. Berlin : Springer-Verlag, 1999

[Ham03] Hammels, Peter: *OntoDigger: Erfassen von Ontologien aus deutschen Texten mit Hilfe von Wortanalyse und regulären Ausdrücken*. Berlin, Institut für Informatik, Humboldt-Universität zu Berlin, Diplomarbeit, 2003

[Han04] Hanft, Alexandre: *Collaborative Maintenance in einem FBS System*, Institut für Informatik, Humboldt-Universität zu Berlin, Diplomarbeit, 2004

[HM05] Hanft, Alexandre ; Minor, Mirjam: A Low-Effort, Collaborative Maintenance Model for Textual CBR. In: Weber, Rosina (Hrsg.) ; Branting, Karl (Hrsg.): *Workshop Proceedings of the 6th International Conference on Case-Based Reasoning*. Chicago, Illinois : DePaul University, 2005, S. 138 – 149

[How05] Howe, Denis: *The Free On-line Dictionary of Computing*. Internet: http://foldoc.doc.ic.ac.uk/foldoc/, 2005. – [Stand: 5. September 2005]

[HPS04] Hameed, Adil ; Preece, Alun ; Sleeman, Derek: Ontology Reconciliation. In: Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.): *Handbook on Ontologies*. Heidelberg : Springer-Verlag, 2004, Kapitel 12, S. 231 – 250

[HS04] Hahn, Udo ; Schulz, Stefan: Building a Very Large Ontology from Medical Thesauri. In: Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.): *Handbook on Ontologies*. Heidelberg : Springer-Verlag, 2004, Kapitel 7, S. 133 – 150

[HW05] Hedin, Priscilla ; White, Larry: Daimler Chrysler Global Service S.T.A.R. Center - Star Online. In: Göker, Mehmet (Hrsg.) ; Cheetham, William (Hrsg.): *Industry Day Proceedings of the 6th International Conference on Case-Based Reasoning, ICCBR-05*. Chicago, Illinois : DePaul University, 2005, S. 35 – 41

[jCO05] *jCOLIBRI*. Internet: http://sourceforge.net/projects/jcolibri-cbr, 2005. – [Stand: November 2005]

[JJS05] John, Michael ; Jugel, Matthias ; Schmidt, Stephan: SnipSnap - Erfahrungen mit kollaborativem Wissensmanagement. In: *wissensmanagement - das Magazin für Führungskräfte* Heft 2/2005 (2005), S. 40 – 41. – ISSN 1438–4426 50745

[Kai05] *Kaidara Software*. Internet: http://www.kaidara.com, 2005. – [Stand: 6. September 2005]

[KH98] Kunze, Mirjam ; Hübner, Andre: CBR on Semi-structured Documents: The ExperienceBook and the FAllQ Project. In: Gierl, Lothar (Hrsg.) ; Lenz, Mario (Hrsg.): *6th German Workshop on CBR*. Rostock : Universität Rostock, 1998, S. 77 – 85. – IMIB Series Vol. 7

[Kli92] Klix, Friedhart: *Die Natur des Verstandes*. Göttingen : Hogrefe, 1992

[Kol84] Kolodner, Janet L.: *Retrieval and Organizational Strategies in Conceptual Memory*. Hillsdale, New Jersey : Lawrence Erlbaum, 1984

[Küh99] Kühnel, Ralf ; Wagner, Gerd (Hrsg.) ; Yu, Eric (Hrsg.): *Assistant Agents that Distribute How-To-Knowledge*. Internet: http://www.aois.org/99/, 1999. – [Stand: 6. September 2005]

[Küh00] Kühnel, Ralf: *Eine Methode zur Entwicklung agenten-basierter Software*, Humboldt-Universität zu Berlin, Diss., 2000

[Kun98] Kunze, M.: *Das ExperienceBook – Dokumentation eines fallbasierten Systems zur Unterstützung der Systemadministration*, Humboldt-Universität zu Berlin, Diplomarbeit, 1998

[LB96] Lenz, Mario ; Burkhard, Hans-Dieter: Lazy Propagation in Case Retrieval Nets. In: Wahlster, Wolfgang (Hrsg.): *12th European Conf. on Artificial Intelligence (ECAI96)*, John Wiley & Sons, 1996, S. 127 – 131

[Lee00] Lee, Laurence L.: Knowledge Sharing Metrics for Large Organisations. In: Morey, Daryl (Hrsg.) ; Maybury, Mary (Hrsg.) ; Thuraisingham, Bhavani (Hrsg.): *Knowledge Management: Classic and Contemporary Works*. The MIT Press, 2000, S. 403 – 419

[Leh00] Lehner, Franz: *Organisational Memory. Konzepte und Systeme für das organisatorische Lernen und das Wissensmanagement*. München : Carl-Hanser-Verlag, 2000

[Len98] Lenz, Mario: Defining Knowledge Layers for Textual Case-Based Reasoning. In: Barry Smyth, Pádraig C. (Hrsg.): *Advances in Case-Based Reasoning. Proceedings of the Fourth European Workshop on Case-Based Reasoning (EWCBR-98)*. Berlin : Springer-Verlag, 1998 (LNAI 1488), S. 298 – 309

[Len99] Lenz, Mario: *Case Retrieval Nets as a Model for Building Flexible Information Systems*, Humboldt-Universität zu Berlin, Diss., 1999

[Leo05] *Link Everything Online*. Internet: http://dict.leo.org/, 2005. – [Stand: 5. September 2005]

[LHK98a] Lenz, Mario ; Hübner, André ; Kunze, Mirjam: Question Answering with Textual CBR. In: Andreasen, Troels (Hrsg.) ; Christiansen, Henning (Hrsg.) ; Larsen, Henrik L. (Hrsg.): *Flexible Query Answering Systems*. Berlin : Springer-Verlag, 1998 (LNAI 1495), S. 236 – 247

[LHK98b] Lenz, Mario ; Hübner, André ; Kunze, Mirjam: Textual CBR. In: Lenz, Mario (Hrsg.) ; Burkhard, Hans-Dieter (Hrsg.) ; Bartsch-Spörl, Brigitte (Hrsg.) ; Wess, Stefan (Hrsg.): *Case-Based Reasoning Technology — From Foundations to Applications*. Berlin : Springer-Verlag, 1998 (LNAI 1400)

[MB05] Minor, Mirjam ; Biermann, Christina: Case Acquisition and Semantic Cross-Linking for Case-Based Experience Management Systems. In: Zhang, Du (Hrsg.) ; Khoshgoftaa, Taghi M. (Hrsg.) ; Shyu, Mei-Ling (Hrsg.): *Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration (IRI-2005)*. Las Vegas : IEEE Systems, Man, and Cybernetics Society, 2005, S. 433 – 438

[McB04] McBride, Brian: The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS. In: Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.): *Handbook on Ontologies*. Heidelberg : Springer-Verlag, 2004, Kapitel 3, S. 51 – 65

[MCG+03] Morgan, Alexander P. ; Cafeo, John A. ; Gibbons, Diane I. ; Lesperance, Ronald M. ; Sengir, Gulcin H. ; Simon, Andrea M.: The General Motors Variation-Reduction Adviser: Evolution of a CBR System. In: Ashley, Kevin D. (Hrsg.) ; Bridge, Derek G. (Hrsg.): *Case-Based Reasoning Research and Development. Proceedings of the 5th International Conference on Case-Based Reasoning ICCBR-03*, Springer-Verlag, 2003 (LNAI 2689), S. 306 – 318

[MCG+04] Morgan, Alexander P. ; Cafeo, John A. ; Godden, Kurt ; Lesperance, Ronald M. ; Simon, Andrea M. ; McGuinness, Deborah L. ; Benedict, James L.: The General Motors Variation-Reduction Adviser: Deployment Issues for an AI Application. In: McGuinness, Deborah L. (Hrsg.) ; Ferguson, George (Hrsg.): *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, AAAI Press / The MIT Press, 2004, S. 777 – 784

[MDP00] Minor, Mirjam ; Del Prado, José C.: Multilingual Textual Case-Based Reasoning. In: Burkhard, Hans-Dieter (Hrsg.) ; Czaja, Ludwik (Hrsg.) ; Skowron, Andrzej (Hrsg.) ; Starke, Peter (Hrsg.): *Proceedings of the Workshop Concurrency, Specification and Programming (CS&P 2000)*, Humboldt-Universität zu Berlin, 2000 (Informatik-Bericht Nr. 140, Vol. I), S. 143 – 148

[MDP01] Minor, Mirjam ; Del Prado, José C.: Multilingual Textual Case-Based Reasoning. In: Schnurr, Hans-Peter (Hrsg.) ; Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.) ; Stumme, Gerd (Hrsg.) ; Sure, York (Hrsg.): *Professionelles Wissensmanagement: Erfahrungen und Visionen*. Aachen : Shaker-Verlag, 2001, S. 281 – 282

[MH00] Minor, Mirjam ; Hanft, Alexandre: The Life Cycle of Test Cases in a CBR System. In: Blanzieri, Enrico (Hrsg.) ; Portinale, Luigi (Hrsg.): *Advances in Case-Based Reasoning: 5th European Workshop, EWCBR 2000*. Berlin : Springer-Verlag, 2000 (LNAI 1898), S. 455 – 466

[Min05a] Minor, Mirjam: Assistant Agents with Personal Ontologies. In: Diggelen, Juriaan van (Hrsg.) ; Dignum, Virginia (Hrsg.) ; Elst, Ludger van (Hrsg.) ; Abecker, Andreas (Hrsg.): *Agent Mediated Knowledge Management Workshop*. Utrecht : Universiteit Utrecht, 2005, S. 41 – 51

[Min05b] Minor, Mirjam: Introduction Strategy and Feedback from an Experience Management Project. In: Althoff, Klaus-Dieter (Hrsg.) ; Dengel, Andreas (Hrsg.) ; Bergmann, Ralph (Hrsg.) ; Nick, Markus (Hrsg.) ; Roth-Berghofer, Thomas (Hrsg.): *Professional Knowledge Management. WM 2005 post-conference proceedings.* Heidelberg : Springer-Verlag, 2005 (LNAI 3782), S. 284 – 292

[ML05] Myritz, Helmut ; Lindemann, Gabriela: Monitoring Patient Behaviour in Clinical Studies by Multi-Agent Techniques. In: Czaja, Ludwik (Hrsg.): *Proceedings of the Workshop Concurrency, Specification and Programming (CS&P 2005)*, Warzaw University, Poland, 2005, S. 342 – 352

[MLB05] Mott, Bradford ; Lester, James ; Branting, Karl: The Role of Syntactic Analysis in Textual Case Retrieval. In: Brüninghaus, Steffi (Hrsg.): *Workshop Proceedings of the 6th International Conference on Case-Based Reasoning, ICCBR-05*. Chicago, Illinois : DePaul University, 2005, S. 120 – 127

[MS04] Maedche, Alexander ; Staab, Steffen: Ontology Learning. In: Staab, Steffen (Hrsg.) ; Studer, Rudi (Hrsg.): *Handbook on Ontologies*. Heidelberg : Springer-Verlag, 2004, Kapitel 9, S. 173 – 190

[MS05] Meyer, Bertolt ; Scholl, Wolfgang: A Comparison of Paradigmatic Views in Knowledge Management: An Empirical Case Study on Shortcomings in KM. In: Ferstl, Otto K. (Hrsg.) ; Sinz, Elmar J. (Hrsg.) ; Eckert, Sven (Hrsg.) ; Isselhorst, Tilman (Hrsg.): *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety, 7. Internationale Tagung Wirtschaftsinformatik 2005, Bamberg, 23.2.2005 - 25.2.2005*, Physica-Verlag, 2005, S. 1003–1022

[Mül05] Müller, Fabian: *Integration von TFBS-Systemen und Ontologien*. Berlin, Institut für Informatik, Humboldt-Universität zu Berlin, Diplomarbeit, 2005

[MW03] Minor, Mirjam ; Wernicke, Mike: Experience Management with Knowledge Sharing Agents. In: McGrath, Fergal (Hrsg.) ; Remenyi, Dan (Hrsg.): *Forth European Conference on Knowledge Management*. Reading : MCIL, 2003, S. 669–675

[NAAD02] Nick, Marcus ; Althoff, Klaus-Dieter ; Avieny, Thomas ; Decker, Björn: How Experience Management Can Benefit from Relationships among Different Types of Knowledge. In: Minor, Mirjam (Hrsg.) ; Staab, Steffen (Hrsg.): *Proceedings of the 1st German Workshop on Experience Management* Bd. P-10. Bonn : Gesellschaft für Informatik, 2002, S. 95 – 106

[NAJ03] Nick, Markus ; Althoff, Klaus-Dieter ; Jedlitschka, Andreas ; Rombach, Dieter (Hrsg.) ; Verlage, Martin (Hrsg.): *Acquiring Knowledge for Linking Software Engineering Experience Maintenance with Evaluation*. Internet: http://www.old.netobjectdays.org/node03/CDindex.html, 2003. – [Stand: 6. September 2005]

[NAT01] Nick, Markus ; Althoff, Klaus-Dieter ; Tautz, Carsten: Systematic Maintenance of Corporate Experience Repositories. In: *Computational Intelligence Journal - Special Issue on Maintaining Case-Based Reasoning Systems* 17 (2001), Nr. 2, S. 364 – 386

[Nic05] Nick, Markus: *Experience Maintenance through Closed-Loop Feedback*, Technische Universität Kaiserslautern, Diss., 2005

[NSS59] Newell, A. ; Shaw, J.C. ; Simon, H. A.: A general problem-solving program for computer. In: *Computers and Automation* 8 (1959), Nr. 7, S. 10 – 16

[NT95] Nonaka, Ikujiro ; Takeuchi, Hirotaka: *The knowledge creating company*. Oxford University Press, 1995

[NT97] Nonaka, Ikujiro ; Takeuchi, Hirotaka: *Die Organisation des Wissens*. Frankfurt : Campus-Verlag, 1997

[NT00] Nonaka, Ikujiro ; Takeuchi, Hirotaka: Reflection on Knowledge Management from Japan. In: Morey, Daryl (Hrsg.) ; Maybury, Mark (Hrsg.) ; Thuraisingham, Bhavani (Hrsg.): *Knowledge Management: Classic and Contemporary Works*. The MIT Press, 2000, Kapitel 7

[PHW94] Pagel, Bernd-Uwe ; Hans-Werner, Six: *Software Engineering: Die Phasen der Softwareentwicklung.* Bonn : Addison-Wesley, 1994

[Pla63] Platon ; Killy, Walther (Hrsg.): *Phaidros.* Frankfurt : Fischer Bücherei, 1963

[Pla88] Platon: *Der Staat.* Leipzig : Reclam, 1988. – Die Ausgabe beruht auf der 1923 in der Philosophischen Bibliothek"des Verlages Felix Meiner als Band 80 erschienenen Übersetzung von Otto Apelt (6. Auflage)

[Pla05] Plato: *Politeia.* Internet: http://gutenberg.spiegel.de/platon/politeia/politeia.htm, 2005. – [Stand: Juli 2005]

[Pol85] Polanyi, Michael: *Implizites Wissen.* Frankfurt/Main : Suhrkamp, 1985

[Por80] Porter, Martin: An algorithm for suffix stripping. In: *Program* 14 (1980), Nr. 3, S. 130 – 137

[Pro03] Probst, Gilbert: *Ganzheitliches Wissensmanagement: Trends und kritische Reflexionen, Eingeladener Vortrag auf der WM 2003.* Internet: http://wm2003.aifb.uni-karlsruhe.de/invitedtalk_probst.pdf, 2003. – [Stand: November 2004]

[Pro05] *Projekt Deutscher Wortschatz.* Internet: http://wortschatz.uni-leipzig.de/, 2005. – [Stand: 5. September 2005]

[PRR99] Probst, Gilbert ; Raub, Steffen ; Romhardt, Kai: *Wissen managen: Wie Unternehmen ihre wertvollste Ressource optimal nutzen.* Wiesbaden : Gabler-Verlag, 1999

[Qui92] Quinn, James B.: *Intelligent enterprise: a knowledge and service based paradigm for industry.* New York : Free Press, 1992

[Qui93] Quinn, James B.: Managing the intelligent enterprise: Knowledge and service-based strategies. In: *Planning Review* 21 (1993), Nr. 5, S. 13–16

[RB03] Roth-Berghofer, Thomas: *Knowledge Maintenance of Case-Based Reasoning Systems - The SIAM Methodology*, Universität Kaiserslautern, Diss., 2003. – DISKI 262, infix-Verlag

[Ric98] Richter, Michael M.: Introduction. In: Lenz, M. (Hrsg.) ; Burkhard, Hans-Dieter (Hrsg.) ; Bartsch-Spörl, Brigitte (Hrsg.) ; Wess, Stefan (Hrsg.): *Case-Based Reasoning Technology — From Foundations to Applications*. Berlin : Springer-Verlag, 1998 (LNAI 1400)

[Ric00] Richter, Michael M.: Fallbasiertes Schließen. In: Görz, Günther (Hrsg.) ; Rollinger, Claus-Rainer (Hrsg.) ; Schneeberger, Josef (Hrsg.): *Handbuch der Künstlichen Intelligenz*. München : Oldenbourg-Verlag, 2000, Kapitel 11, S. 407 – 430

[Rij79] Rijsbergen, C. J.: *Information Retrieval*. London : Butterworths, 1979

[Rol00] Rollett, Herwig: *Aspekte des Wissensmanagements*. Graz, Institut für Informationsverarbeitung und computergestützte neue Medien, Technische Universität Graz, Diploma Thesis, 2000

[Ros89] Ross, B H.: Some Psychological Results on Case-Based Reasoning. In: Hammond, Kristian J. (Hrsg.): *Proceedings: Case-Based Reasoning Workshop*, Morgan Kaufmann Publishers, 1989, S. 144 – 147

[RSS02] *Kapitel* Technik und Handeln. Wenn soziales Handeln sich auf menschliches Verhalten und technische Abläufe verteilt. In: Rammert, Werner ; Schulz-Schäfer, Ingo: *Können Maschinen handeln? Soziologische Beiträge zum Verhältnis von Mensch und Technik*. Frankfurt/Main u.a. : Campus-Verlag, 2002, S. 11 – 64

[RY97] Racine, Kirsti ; Yang, Qiang: Maintaining Unstructured Case Bases. In: Leake, David B. (Hrsg.) ; Plaza, Enric (Hrsg.): *Case-Based Reasoning Research and Development, Proceedings of the 2nd International Conference on Case-Based Reasoning ICCBR-97* Bd. 1266. Heidelberg : Springer-Verlag, 1997, S. 553 – 564

[Sch82] Schank, Roger C.: *Dynamic Memory: A Theory of Learning in Computers and People*. New York : Cambridge University Press, 1982

[SKMH04] Scholl, Wolfgang ; König, Christine ; Meyer, Bertolt ; Heisig, Peter: The Future of Knowledge Management – an inter-

national Delphi Study. In: *Journal of Knowledge Management* 8 (2004)

[SMS00] Sure, York ; Maedche, Alexander ; Staab, Steffen: Leveraging Corporate Skill Knowledge - From ProPer to OntoProPer. In: Reimer, Ulrich (Hrsg.): *Proceedings of the Third International Conference on Practical Aspect s of Knowledge Management (PAKM 2000)* Bd. 34. Basel, Switzerland : CEUR-WS.org, 2000

[Sni05] *SnipSnap - the easy Weblog and Wiki Software*. Internet: http://www.snipsnap.org, 2005. – [Stand: 23. September 2005]

[Sta02] Staab, Steffen: *Wissensmanagement mit Ontologien und Metadaten*, Universität Karlsruhe (TU), Habilitationsschrift, 2002

[SW00] Stumme, Gerd ; Wille, Rudolf: *Begriffliche Wissensverarbeitung - Methoden und Anwendungen*. Heidelberg : Springer-Verlag, 2000

[TAB+03] Tecuci, Gheorghe ; Aha, David W. ; Boicu, Mihai ; Cox, Michael T. ; Ferguson, George ; Tate, Austin: *IJCAI-03 Workshop on Mixed-Initiative Intelligent Systems*. Internet: http://lalab.gmu.edu/miis, 2003. – [Stand: 8. Juli 2005]

[The03] Theune, Mariet: From monologue to dialogue: natural language generation in OVIS. In: Freedman, Reva (Hrsg.) ; Callaway, Charles (Hrsg.): *AAAI 2003 Spring Symposium on Natural Language Generation in Written and Spoken Dialogue*. Palo Alto : AAAI, 2003, S. 141 – 150

[UML02] *UMLS (Unified Medical Language System), United States National Library of Medicine*. Internet: http://www.nlm.nih.gov/research/umls, 2002. – [Stand: 23. September 2005]

[Vor02] Vorländer, Karl: *Geschichte der Philosophie*. Internet: http//www.textlog.de/6133.html, 1902. – [Stand: Juli 2005]

[WDA99] Wolf, T. ; Decker, S. ; Abecker, A.: Unterstützung des Wissensmanagements durch Informations- und Kommunikationstechnologie. In: Scheer, A.-W. (Hrsg.) ; Nüttgens, M. (Hrsg.): *Electronic Business Engineering, 4. Internationale Tagung Wirtschaftsinformatik*. Heidelberg : Physica-Verlag, 1999, S. 745 – 766

[Wer03] Wernicke, Mike: *Textuelles Fallbasiertes Schließen für die Benutzerschnittstelle von Agenten.* Berlin, Institut für Informatik, Humboldt-Universität zu Berlin, Diplomarbeit, 2003

[Weß95] Wess, Stefan: *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*, Universität Kaiserslautern, Diss., 1995. – DISKI 126, infix-Verlag

[Wik05] *Internet-Enzyklopädie Wikipedia.* http://www.wikipedia.org, 2005. – Januar 2005

[Wil01] Wilson, David C.: *Case-Based Maintenance: the Husbandry of Experience*, Indiana University, Diss., 2001

[WKM04] Wiratunga, Nirmalie ; Koychev, Ivan ; Massie, Stewart: Feature Selection and Generalisation for Retrieval of Textual Cases. In: Funk, Peter (Hrsg.) ; González Calero, Pedro A. (Hrsg.): *Advances in Case-Based Reasoning: 7th European Conference, ECCBR 2004.* Berlin : Springer-Verlag, 2004 (LNAI 3155), S. 809 – 820

[Wor05] *WordNet.* Internet: http://wordnet.princeton.edu/, 2005. – [Stand: 5. September 2005]

[Wur05] Wurst, Michael: Evaluating Knowledge Management in Open Multiagent Environments. In: Diggelen, Juriaan van (Hrsg.) ; Dignum, Virginia (Hrsg.) ; Elst, Ludger van (Hrsg.) ; Abecker, Andreas (Hrsg.): *Agent Mediated Knowledge Management Workshop.* Utrecht : Universiteit Utrecht, 2005, S. 82 – 96

Glossar

Ähnlichkeit wird im Fallbasierten Schließen eingesetzt *S. 35*

Ähnlichkeitsfunktion, globale Ähnlichkeitsfunktion Funktion, die den Grad der Ähnlichkeit eines Falls zu einer Anfrage bezeichnet, benutzt oft eine lokale Ähnlichkeitsfunktion *S. 41*

Ähnlichkeitsfunktion, lokale s. lokale Ähnlichkeitsfunktion

Ähnlichkeitslexikon enthält Paare von IEs mit Ähnlichkeitstyp *S. 46*

Ähnlichkeitstyp zum Beispiel WEAK_SIM, binäre Relation für IEs im Ähnlichkeitslexikon, aus der Werte für die lokale Ähnlichkeitsfunktion abgeleitet werden können *S. 46*

Agent, Software-Agent, agentenorientiertes Programm Computerprogramm, das als Akteur auftritt *S. 3*

agentenorientiertes Programm s. Agent

aggregierende Funktion fasst mehrere partielle Ähnlichkeitsfunktionen zusammen *S. 47*

Akteur Mensch oder technisches Artefakt *S. 3*

Aktivierungsfunktion wird im CRN benutzt, um das Retrieval zu implementieren *S. 49*

Anfrage Beschreibung einer aktuellen Problemsituation *S. 37*

Assistenzsystem Computersystem, das Menschen oder andere Computersysteme unterstützt *S. 3*

Attribut-IE repräsentiert ein Attribut-Werte-Paar *S. 45*

Aufgaben des EM *S. 27 und S. 28*

Bausteine des Wissensmanagements Modell mit den Kernprozessen Wissensidentifikation, Wissenserwerb, Wissententwicklung, Wissens(ver)-teilung, Wissensnutzung und Wissensbewahrung *S. 15*

Case-Based Reasoning s. Fallbasiertes Schließen

Case Retrieval Net, CRN Speicherstruktur für Falldaten, die ein effizientes Retrieval ermöglicht *S. 42*

CBR s. Fallbasiertes Schließen

Collaborative Maintenance einer Fallbasis verteilte Maintenance-Strategie unter Mitwirkung der Benutzer *S. 133*

CRN s. Case Retrieval Net

CRN-Server Implementierung eines CRNs in einer Client-Server-Architektur *S. 71*

Dienst eines Agenten Hilfsprogramm, das der Agent für seinen Benutzer ausführen kann *S. 112*

Domäne Aufgabenbereich oder Weltausschnitt *S. 2*

Domänenontologie Ontologie zur Beschreibung einer Domäne *S. 74*

EM s. Erfahrungsmanagement

EM-Aufgaben s. Aufgaben des EM

EM-Modell Prozessmodell des EM *S. 30*

EM-System, EM-Assistenzsystem unterstützt Menschen bei der Bearbeitung von EM-Aufgaben *S. 27*

epistemologische Dimension der Wissensschaffung implizit oder explizit *S. 12*

Erfahrungsmanagement, EM Spezialform des WM, die sich mit Erfahrungswissen befasst *S. 26*

Erfahrungswissen spezielles Wissen, das ein Akteur in einem Problemlösungskontext erworben hat *S. 25*

ExperienceBook I fallbasiertes EM-System für IT-Mitarbeiter und Administratoren *S. 138*

ExperienceBook II fallbasiertes EM-System für Erstsemestler, Werbestrategie und kommunikationsfördernde Elemente *S. 122*

explizites Wissen lässt sich in formaler, systematischer Sprache weitergeben *S. 12*

Externalisierung von Wissen Prozess der Wissensschaffung *S. 13 und 21*

Fall, Fallbeispiel Beschreibung einer Problemsituation plus Lösungserfahrungen *S. 36*

Fallbasiertes Schließen, FBS, Case-Based Reasoning, CBR Ansatz zur Wiederverwendung von spezifischem Problemlösungswissen im Kontext eines aktuell zu lösenden Problems *S. 36*

Fallbasiertes Schließen für Texte, Textual CBR, TCBR Fallbasiertes Schließen für Falldaten, die Texte enthalten *S. 43*

Fallbasiertes System, FBS-System implementiert Fallbasiertes Schließen: verwaltet Problemlösungswissen in der Form von Fällen, beantwortet Anfragen durch dazu „ähnliche“ Fälle *S. 37*

Fallbasis Sammlung von Fällen *S. 37*

Fallrevision, Revision eines Falls Momentaufnahme eines Falls zu einem bestimmten Zeitpunkt *S. 84*

Fallstruktur im TCBR besteht aus einem Fallbezeichner, Textabschnitten und Abschnitten mit Attribut-Werte-Paaren *S. 43*

FBS s. Fallbasiertes Schließen

F-Logik zur Darstellung von Relationen in Ontologien *S. 73*

Ganzheitliches EM integriert technische, organisatorische, psychologische und kulturelle Aspekte *S. 32*

Gewichtungsfunktion für Ähnlichkeitstypen wird für die Bestimmung der Werte der lokalen Ähnlichkeitsfunktion benötigt *S. 46*

Globale Ähnlichkeitsfunktion s. Ähnlichkeitsfunktion

Gutachten, Review für eine Fallrevision bewertet den Inhalt einer Fallrevision *S. 135*

Hasse-Diagramm zur Darstellung einer transitiven Relation *S. 108*

Hintergrundwissen eines EM-Systems systeminternes Wissen zur Bewältigung der Aufgaben des Systems *S. 28 und S. 52*

IE s. Informationseinheit

IE-Name eindeutiger Bezeichner für ein IE *S. 40*

implizites Wissen persönlich, kontextspezifisch, daher schwer kommunizierbar *S. 12 und 15*

Indexlexikon Lexikon aller Text-IEs, jeder Eintrag enthält IE-Name, Wortschatz-Kategorie und zugehörige Wörter *S. 45*

Indexvokabular alle Wörter, die auf Text-IEs abgebildet werden *S. 44*

Informationseinheit, IE atomarer Bestandteil eines Falls *S. 40*

„Intelligentes“ Computersystem setzt Methoden der Künstlichen Intelligenz ein *S. 2*

Internalisierung von Wissen Prozess der Wissensschaffung *S. 13 und 21*

Kombination von Wissen Prozess der Wissensschaffung *S. 13 und 21*

Lebenszyklus eines Falls, Life-Cycle wird beschrieben durch alle Fallrevisionen von der Entstehung bis zum Löschen des Falls *S. 84*

Life-Cycle eines Falls s. Lebenszyklus

lokale Ähnlichkeitsfunktion Funktion, die den Grad der Ähnlichkeit eines IEs zu einem anderen bezeichnet *S. 41*

Maintenance einer Fallbasis Aktualisieren und neu organisieren *S. 132*

MultiLingual fallbasiertes Anwendungsbeispiel für den Wissenserwerb, multilinguales Fußball-Informationssystem *S. 77*

OntoCBR fallbasiertes Anwendungsbeispiel für den Wissenserwerb, Transformation von Domänenontologien in TCBR-Hintergrundwissen und umgekehrt *S. 74 und 91*

OntoDigger Tool zur Unterstützung bei der Modellierung von Hintergrundwissen *S. 108*

Ontologie Konzepte und Beziehungen *S. 73*

Ontologietypen Domänenontologien und andere *S. 74*

ontologische Dimension der Wissensschaffung beim Individuum, in der Gruppe, im gesamten Unternehmen oder bei der Interaktion zwischen Unternehmen *S. 12*

organisatorische Barrieren des WM zum Beispiel Zeitmangel, schlechte Motivation der Mitarbeiter, zu kleine Budgets oder einschränkende Paradigmen und Traditionen *S. 33*

organisatorische Maßnahme zur Einbettung eines EM-Systems in eine Organisation, um dessen Nutzung sicherzustellen *S. 122*

Operator für eine Relation $op : S \mapsto S'$ für $S \subseteq E \times E$, $S' \subseteq E \times E$ *S. 93*

OWL, Web Ontology Language Ontologiesprache *S. 73*

Partielle Ähnlichkeitsfunktion bestimmt den Grad der Ähnlichkeit für ein Paar von IEs, die aus dessen Zugehörigkeit zu einem Ähnlichkeitstyp resultiert, geht in die aggregierende Funktion *aggr* ein *S. 47*

Query s. Anfrage

Retrieval, Retrieve Prozess des Wiederfindens passender Fälle, bildet das menschliche Erinnern nach *S. 38 und S. 47*

Review einer Fallrevision s. Gutachten einer Fallrevision

Revision eines Falls s. Fallrevision

Seki-Modell Wissensschaffung durch **S**ozialisation, **E**xternalisierung, **K**ombination und **I**nternalisierung *S. 13*

SimLex fallbasiertes Anwendungsbeispiel für den Wissenserwerb, Web-Lexikon für die RoboCup-Community *S. 66*

Software-Agent s. Agent

Sozialisation von Wissen Prozess der Wissensschaffung *S. 13 und 21*

soziotechnische Maßnahme technische Veränderung eines EM-System zur Umsetzung psychosozialer Erkenntnisse *S. 122*

TCBR, Textual CBR s. Fallbasiertes Schließen für Texte

Text-Informationseinheit, Text-IE IE, das eine Menge von Wörtern repräsentiert, die für ein Konzept stehen *S. 44*

transitive Hülle $< R >$ kleinste transitive Relation, die R enthält *S. 93*

Typen von Ontologien s. Ontologietypen

Web Ontology Language s. OWL

WM s. Wissensmanagement

Wissen aus Assistenzsystem-Sicht Wissensbegriff dieser Arbeit, wie Wissen aus technischer Sicht ergänzt um subjektive Komponente *S. 9*

Wissen aus psychologischer Sicht lernorientierter Wissensbegriff, Wissensstrukturen im menschlichen Gedächtnis als Resultat von Lernprozessen *S. 7*

Wissen aus technischer Sicht losgelöst vom Individuum, Wissen als Menge verknüpfter Informationen *S. 9*

Wissen aus wirtschaftswissenschaftlicher Sicht prozessorientierter Wissensbegriff, Wissen als Prozess der Erklärung *S. 8*

Wissen, explizites s. explizites Wissen

Wissen, implizites s. implizites Wissen

Wissensbewahrung Kernprozess des WM *S. 17, S. 22 und Kap. 9*

Wissenscontainer eines fallbasierten Systems Vokabular, Ähnlichkeitsmaß, Fallbasis und Lösungstransformation *S. 39*

Wissensentwicklung Kernprozess des WM *S. 16, S. 20 und Kap. 6*

Wissenserwerb Kernprozess des WM *S. 16, S. 19 und Kap. 5*

Wissensidentifikation Kernprozess des WM *S. 16 und S. 18*

Wissensmanagement, WM systematischer Ansatz mit dem Ziel, den Umgang mit Wissen zu verbessern *S. 10*

Wissensnutzung Kernprozess des WM *S. 17, S. 22 und Kap. 8*

Wissensquelle Quelle, aus der Wissen für ein Assistenzsystem gewonnen werden kann *S. 18*

Wissens(ver)teilung Kernprozess des WM *S. 17, S. 21 und Kap. 7*

Wortschatz-Kategorie Teilmenge der IEs, zum Beispiel allgemeinsprachlicher Grundwortschatz oder Computerbegriffe *S. 44*

Danksagung

Ich danke meinen engagierten Doktorvätern Hans-Dieter Burkhard, Ralph Bergmann und Gerd Stumme, die mich alle drei während der Entstehung der Arbeit mit Rat und Tat begleitet haben. Es ist eine große Ehre für mich, dass drei so hochrangige Wissenschaftler mich unterstützt haben.

Ohne meine Diplomanden wäre nichts aus dieser Arbeit geworden: Alexandre Hanft, Christina Biermann (vormals Bell), José Carlos Del Prado, Mike Wernicke, Peter Hammels, Fabian Müller, ich danke euch sehr herzlich für eure Flexibilität, euren Enthusiasmus und die wertvolle Arbeit, die ihr geleistet habt. Auch den in der Arbeit unerwähnten Diplomanden Sandro Köppen, Gordon Kramer, Alexander Moczko und Martin Stiel sei herzlich gedankt: Ihre Arbeiten lagen außerhalb des Fokus dieser Arbeit und haben mich inspiriert.

Der nächste Dank gilt meinen mathematischen Schutzengeln Thomas Noll und Karsten Schmidt, die viel Geduld mit mir aufgebracht haben und neben den vielen anderen Aufgaben Zeit gefunden haben, meine Formeln mit mir zu polieren. Sollten jetzt noch Ungereimtheiten darin zu finden sein, liegt es ganz bestimmt nicht an euch, vielen Dank für eure Hilfe.

Interdisziplinäre „Schützenhilfe“ bekam ich von Hartmut Wandke, Wolfgang Scholl, Bertolt Meyer, Peter Heisig und Bettina Berendt. Vielen Dank für die anregenden Diskussionen, die Literaturhinweise und vieles mehr.

Wertvolle Anregungen und immer wieder persönliche Ermutigungen verdanke ich Brigitte Bartsch-Spörl, Mehmet Göker, Thomas-Roth-Berghofer, David Wilson, Steffen Staab, Klaus-Dieter Althoff, Markus Nick, Ulrich Reimer, Virginia Dignum, Alexander Glintschert, Peter Bittner, Katinka Wolter und nicht zu vergessen meinen Kolleginnen und Kollegen am Lehrstuhl, besonders Dagmar Monett-Diaz, Gabi Lindemann, Olga Schiemangk, Kay Schröter, Diemo Urbig und Renate Zirkelbach. Ich danke euch für die praktische und moralische Unterstützung.

Meine Familie, meine Freunde, meine Musikerkollegen: Ohne euch wäre ich zur Fachidiotin verkommen. Danke, dass ihr mir manchmal den nötigen Abstand verschafft habt, um wieder klar denken zu können. Es ist nicht leicht,

die Launen einer Doktorandin, die mitten im Schreiben steckt, auszuhalten. Zu eurem Beitrag zu dieser Arbeit gäbe es noch so viel zu sagen, dass ich es ganz kurz mache: Danke!

Selbstständigkeitserklärung

Hiermit erkläre ich, dass

- ich die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und nur die angegebene Literatur und Hilfsmittel verwendet habe,
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze und
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist.