Diplomarbeit

Ein fallbasiertes Informationssystem für fallbasierte Anwendungen und Systeme auf der Basis von INRECA

Alexandre Meissonnier

7. Oktober 1996

Betreuer:
Dr. Klaus-Dieter Althoff

Universität Kaiserslautern
Fachbereich Informatik
AG Künstliche Intelligenz – Expertensysteme

Prof. Dr. Michael M. Richter

ZUSAMMENFASSUNG

In dieser Arbeit wird ein fallbasiertes System entwickelt, das Angaben über existiertende fallbasierte Anwendungen und Werkzeuge verwaltet. Mit diesem System kann ein Entwickler von fallbasierten Systemen sich einen Überblick über den Stand der Technik verschaffen und vor allem Informationen über Systeme erhalten, die dem System, das er selbst entwickeln will, ähnlich sind.

Im Grundlagenteil dieser Arbeit werden zunächst die wichtigsten Grundbegriffe des fallbasierten Schließens zusammengefaßt. Da die Aufgabe des zu entwickelnden Informationssystems die Unterstützung der Wiederverwendung von Erfahrungswissen bezüglich der Entwicklung fallbasierter Software ist, wird auch ein Ansatz aus dem Software Engineering beschrieben und mit dem hier dargestellten Ansatz auf Basis des fallbasierten Schließens verglichen.

Die Falldaten für das System wurden mittels einer Umfrage direkt bei den Entwicklern fallbasierter Systeme erhoben. Neben der Aufnahme in die Fallbasis wurden diese Daten auch statistisch ausgewertet.

Die Modellierung der Falldaten für das System orientiert sich an einem Kriterienkatalog zur Beschreibung, Analyse und Entwicklung fallbasierter Systeme (aus Althoff, 1996). Die resultierende Struktur ist in dieser Arbeit ausführlich beschrieben.

Das so konzipierte System wurde schließlich auf dem Rechner implementiert. Die Grundlage dieser Implementierung ist das System CBR-Works, der Nachfolger des INRECA-Systems. Die Falldaten sind außerdem in der Fallrepräsentationssprache Casuel abgelegt und über das Internet der Öffentlichkeit zugänglich.

INHALTSVERZEICHNIS

1. EINLEITUNG

Die Technik des fallbasierten Schließens (Case-Based Reasoning, CBR) hat mittlerweile einen Stand erreicht, der ihr den Sprung von der Forschung in die betriebliche Praxis erlaubt. Dieser Sprung ist in einigen Anwendungsbereichen bereits erfolgt, in anderen steht er noch bevor. Die Anzahl betrieblicher fallbasierter Anwendungen ist inzwischen kaum noch zu überblicken. In manchen bereichen, wie z.B. bei Helpdesk-Anwendungen, gehören fallbasierte Methoden schnon zum Alltag.

Angesichts dieser Verbreitung wird ein Überblick über den derzeitigen Stand der Technik zunehmend schwieriger, sowohl aus der Sicht der Forschung als auch aus der Perspektive von kommerziellen CBR-Entwicklern. Die Vielzahl unterschiedlicher Ansätze macht es vor allem schwer, bei der Entwicklung eines neuen fallbasierten Systems die für die vorliegende Anwendungsaufgabe am besten geeigneten Methoden und Hilfsmittel aus der Menge existierender Systeme herauszusuchen.

Es ist also an der Zeit, die bei der Entwicklung fallbasierter Systeme vielerorts gemachten Erfahrungen zu sammeln und anderen zugänglich zu machen. Da die zu sammelnde Erfahrung in Form von Fällen, nämlich von existierenden fallbasierten Systemen vorliegt, empfiehlt sich für deren Aufbereitung wiederum das fallbasierte Schließen selbst.

1.1 Gegenstand der Arbeit

Das Ziel dieser Arbeit besteht nun darin, Erfahrungen aus der Entwicklung fallbasierter Systeme zu sammeln und in einem fallbasierten System zur Verfügung zu stellen. Das resultierende System soll soll einer möglichst breiten Öffentlichkeit zugänglich sein, also etwa auf einem ftp-Server oder im WWW.

Die Konsultation dieses Systems verläuft dann etwa nach dem folgenden Szenario: Ein Softwareentwickler, der eine fallbasierte Anwendung erstellen will, nennt dem System die Anforderungen an seine zu erstellende Anwendung. Dazu beschreibt er die Domäne und die Aufgabe, mit der sich seine Anwendung beschäftigen soll und nennt zusächlich technische und ergonomische Anforderungen, die sich aus seiner Arbeitsumgebung ergeben. Das System nennt ihm daraufhin die Anwendungen aus der Fallbasis, die der eingegebenen Beschreibung am ähnlichsten sind. Möglicherweise kann er aus der Beschreibung dieser

Fälle direkt für ihn nützliche Informationen entnehmen. Zusätzlich erhält er noch Literaturhinweise und eine Kontaktadresse, über die er sich weitere Informationen zu der in einem Fall beschriebenen Anwendung beschaffen kann. Eventuell kann er dabei sogar Falldaten oder Softwarekomponenten übernehmen und sich dadurch sehr viel Arbeit sparen.

Er kann aber auch gezielt nach einem fallbasierten Werkzeug[1] suchen, mit dem er seine Anwendung erstellen will. Auch hier erfolgt die Auswahl nach einem Ähnlichkeitskriterium.

Für die Implementierung dieses Systems eignet sich das System INRECA, bzw. dessen Nachfolger CBR-Works. CBR-Works wurde unter anderem an der Universität Kaiserslautern entwickelt und dort bereits zur Erstellung mehrerer fallbasierter Anwendungen benutzt. Ein wichtiger Vorteil von CBR-Works ist die Möglichkeit, Falldaten und Informationen über die Fallstruktur in der Fallbeschreibungssprache CASUEL zu verarbeiten und so mit anderen Systemen auszutauschen. Zukünftige Versionen von CBR-Works sollen eine Schnittstelle zum WWW haben, die es ermöglichen soll, eine Fallbasis über das Internet zu konsultieren. Damit bietet CBR-Works optimale Voraussetzungen für die Verbreitung des in dieser Arbeit zu erstellende Informationssystem.

1.2 Aufbau der Arbeit

Kapitel 2 faßt zunächst die Grundlagen zusammen, auf denen die übrige Arbeit aufbaut. Kapitel 3 beschreibt anschließend die Methode, mit der die Daten für das Informationssystem erhoben wurden und bietet einen statistischen Überblick über diese Daten. Die Modellierung dieser Daten für die Speicherung als Fälle wird in Kapitel 4 erläutert. Die eigentliche Implementierung beschreibt anschließend Kapitel 5. Kapitel 6 faßt schlließlich die Ergebnisse dieser Arbeit zusammen.

1.3 Terminologie

Im folgenden werden drei Begriffe für fallbasierte Software definiert, deren Unterscheidung für diese Arbeit wesentlich ist. Es sind dies die Begriffe *System*, *Anwendung* und *Werkzeug*.

- Der Begriff **System** wird im folgenden als Oberbegriff für jede Art von Softwareprodukten gebraucht. In der Regel ist dabei fallbasierte Software gemeint.

[1] Zur Klärung der Begriffe siehe weiter unten (Abschnitt 1.3).

- Eine **Anwendung** ist im Kontext dieser Arbeit ein CBR-System, das in dem Sinne vollständig ist, daß es die Modellierung einer konkreten Domäne inklusive einer Fallbasis enthält. Hier wird also die Technik des fallbasierten Schließens auf eine konkrete Situation angewandt.

- Ein fallbasiertes **Werkzeug** ist hingegen von der Domäne unabhängig. Es stellt allgemeine Funktionen zur Fallrepräsentation und Ähnlichkeitsbestimmung zur Verfügung, die auf unterschiedliche Domänen angewandt werden können. Solche Systeme werden auch häufig als *Shells* bezeichnet.

Ein Werkzeug bietet also eine Grundlage, auf der eine Anwendung erstellt werden kann. Es ist aber keine notwendige Voraussetzung. In vielen fallbasierten Anwendungen werden die Funktionen, die auch ein Werkzeug bietet, neu implementiert und dabei oft im Hinblick auf die konkrete Anwendung optimiert.

2. GRUNDLAGEN

2.1 Das Prozessmodell des fallbasierten Schließens

Die Grundidee des fallbasierten Schließens besteht darin, die Lösung eines gegebenen Problems auf die bereits bekannte Lösung eines ähnlichen früheren Problems zurückzuführen. Dieser Abschnitt beschreibt einige Grundbegriffe des fallbasierten Schließens. Weitere Erläuterungen hierzu finden sich beispielsweise in Althoff et al. (1995a) und Richter und Wendel (1990).

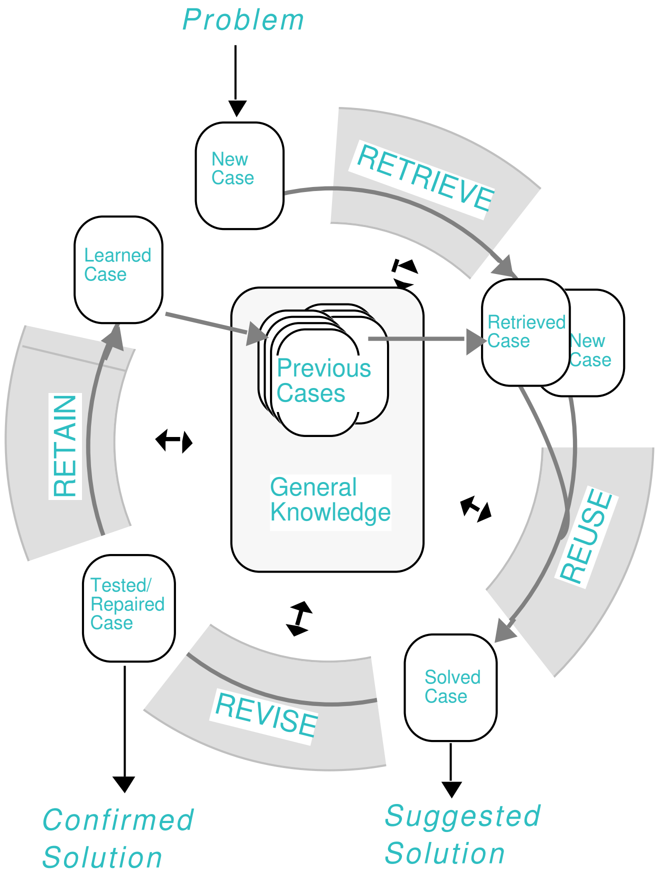

Abb. 2.1: Der CBR-Zyklus (nach Aamodt und Plaza, 1994)

Abbildung 2.1 zeigt das gängigste Prozessmodell des fallbasierten Schließens.

Es gliedert sich in die vier Phasen *Retrieve, Reuse, Revise* und *Retain.*

- **RETRIEVE:** Ausgehend von der Beschreibung des zu lösenden Problems, dem Anfragefall, werden zunächst ein oder mehrere ähnliche Fälle aus der Fallbasis ausgewählt. Der entscheidende Parameter ist hierbei die Präzisierung des Ähnlichkeitsbegriffs. Desweiteren ist aus Performanzgründen, gerade bei großen Fallbasen, eine gute Verwaltung des Fallspeichers nötig.

- **REUSE:** In diesem Schritt wird die Lösung des ähnlichsten Falls (oder mehrerer ähnlichster Fälle) auf den Anfragefall übertragen. Im einfachsten Fall geschieht dies durch bloßes Kopieren von Attributen. Oft muß jedoch eine mitunter sehr komplexe Anpassung (Adaption) vorgenommen werden.

- **REVISE:** Die gefundene Lösung wird nun in der realen Welt außerhalb des CBR-Systems angewendet. Als Rückmeldung erhält man dabei eine Bewertung der Lösung, mit der man diese nochmals korrigieren kann.

- **RETAIN:** In diesem letzten Schritt schließt sich der Kreis. Die bestätigte und/oder korrigierte Lösung wird nun in das Wissen des fallbasierten Systems aufgenommen. Dies bedeutet nicht unbedingt, daß jede Lösung als neuer Fall aufgenommen wird. Es können auch Änderungen am Ähnlichkeitsmaß oder an den zur Lösungsfindung herangezogenen Fällen vorgenommen werden.

Bei diesen vier Schritten reicht es oft nicht, sich ausschließlich auf Fallwissen zu beschränken. Um akzeptable Ergebnisse zu erzielen, muß bei vielen Anwendungsaufgaben zusätzliches generelles Wsissen. Gerade hier geht es darum, Elemente der Lösung zu finden, die über die gefundenen Fälle hinausgehen. Der Revise-Schritt wird wiederum, wenn er nicht außerhalb des CBR-Systems stattfindet, mit Hilfe von Domänenwissen durchgeführrt. In der Retain-Phase kann wiederum nicht nur der gerade gelöste Fall gelernt werden, sondern es kann auch das generelle Wissen modifiziert werden, um die in der Bearbeitung dieses Falls gewonnene Erfahrung zu speichern.

Die Art des verwendeten generellen Wissens ist nicht festgelegt. Prinzipiell kann jede wissensbasierte Methode eingesetzt werden, um das fallbasierte Schließen zu unterstützen. Das Fallwissen und das zusätzliche Wissen sind dabei nicht unbedingt strikt voneinander getrennt. Der Trend geht heutzutage immer mehr zu einer tiefen Integration von fallbasiertem Schließen mit anderen Ansätzen. Beispiele für verschiedene Arten von integriertem generellem Wissen sind Fehlerkontexte und Regeln in S^3+ (Priebisch, 1995), das simulationsfähige Modell in MoCAS (Pews, 1994), sowie die induktiven Methoden in INRECA und weitere Ansätze (Althoff et al., 1995b).

2.2 Aufgabentypen

Der Einsatzbereich fallbasierter Systeme gliedert sich in verschiedene Anwendungsszenarien. Diese werden, wie auch in anderen Teilgebieten der KI, zu Aufgabentypen zusammengefaßt, die dann als Problemklassen untersucht werden können. Im folgenden wird nun die Einteilung in Aufgabentypen beschrieben, die in dieser Arbeit benutzt wird (nach Althoff und Bartsch-Spörl, 1996; Weß, 1995).

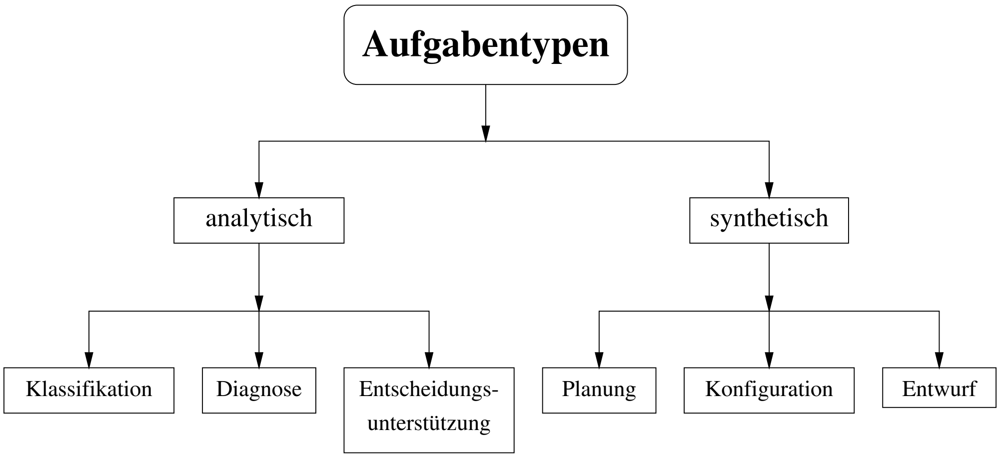

Abb. 2.2: Aufgabentypen

Bei der Einteilung der Aufgabentypen unterscheidet man zunächst *analytische* und *synthetische* Aufgaben. Bei analytischen Aufgaben besteht das Problem darin, einer gegebenen Situation bestimmte Daten aus der Fallbasis zuzuweisen. Hier unterscheidet man folgende Szenarien:

Klassifikation: Die gesuchte Lösung zu einem gegebenen Problem (Anfragefall) ist die Zuordnung zu einer bestimmten Klasse

Diagnose: Diese Situation stellt eine Erweiterung des Klassifikationsszenarios dar. Der Unterschied besteht darin, daß bei der Diagnose die Anfragesituation in der Regel unvollständig vorliegt. Hier muß das System bestimmen, welche Attribute zusätzlich erhoben werden müssen, um die Anfrage sinnvoll klassifizieren zu können.[1]

Entscheidungsunterstützung: Anders als bei der Klassifikation oder Diagnose ist hier kein Zielattribut vorgegeben. Welche im Fall gespeicherte Information[2] als Lösung zu betrachten ist, ist von Anfrage zu Anfrage unterschiedlich und entscheidet sich erst im Laufe der Konsultation des

[1] Vergleiche mit dem Vorgehen eines Arztes, der zunächst ausgehend von den bekannten Symptomen weitere Untersuchungen vornimmt, bevor er eine Diagnose stellt.

[2] Dies kann eine Kombination von mehreren Attributen sein, evtl. auch der ganze Fall.

Systems. Ein weiteres Merkmal der Entscheidungsunterstützung ist der verstärkte Einsatz von generellem Wissen.

Im Gegensatz zu analytischen Aufgaben bilden bei synthetischen Aufgaben die Falldaten erst die Grundlage, auf der die eigentliche Lösung generiert wird. Diese Aufgaben sind also komplexer als analytische Aufgaben. Man unterscheidet hier ebenfalls drei Aufgabentypen:

Planung: Das entscheidende Merkmal von Planungsaufgaben ist die Festlegung der zeitlichen Reihenfolge von auszuführenden Schritten. Daher spielt der Parameter *Zeit* hier eine wichtige Rolle.

Konfiguration: Charakteristisch für Konfigurationsaufgaben ist, daß vorgefertigte Teillösungen existieren, die geeignet zu einer Gesamtlösung zusammengesetzt werden.

Entwurf: Im Gegensatz zur Konfiguration gibt es bei Entwurfsproblemen zumindest in Teilbereichen keine vorgefertigten Komponenten, auf die man zurückgreifen kann. Die Lösungen müssen hier völlig neu entwickelt werden. Da die Kreativität von EDV-Systemen sehr eingeschränkt ist[3], erhält man von Entwurfssystemen meist nur Vorschläge für eine Lösung.

Die Einteilung in die genannten Aufgabentypen ist nicht eindeutig. Selbst zwischen analytischen und synthetischen Aufgaben läßt sich keine scharfe Grenze ziehen. So ist die in der Diagnose und Entscheidungunterstützung nötige Erhebung fehlender Information bereits eine synthetische Aufgabe (Althoff und Bartsch-Spörl, 1996), ebenso die bei allen Aufgabentypen vorkommende Fallanpassung (Adaption) (vgl. Voß et al., 1996). Andererseits kann bei Planungs- und Konfigurationsproblemen zunächst eine Klassifikation der Ausgangssituation vonnöten sein. Die Zuordnung zu einem der genannten Aufgabentypen ist meist dennoch möglich, wenn man den Schwerpunkt eines gegebenen Anwendungsproblems betrachtet (siehe auch Richter, 1992).

2.3 Wiederverwendung von Software

Im Anwendungsszenario für das in dieser Arbeit zu erstellende Informationssystem wird in zweifacher Hinsicht CBR betrieben. Einerseits ist der Gegenstand der Betrachtung fallbasierte Software, andererseits bedient sich die Implementierung des fallbasierten Schließens. Betrachtet man unseren Untersuchungsgegenstand genauer, so stellt man fest, daß es sich im Prinzip um ein spezielles Problem aus dem Software Engineering handelt. Auch im Software Engineering

[3] Zur Diskussion hierzu siehe Hofstadter (1985)

geht es darum, Erfahrung aus der Entwicklung von Software zu sammeln und bei späteren Projekten einzusetzen. Dieser Abschnitt beschreibt nun die diesbezüglichen Methoden des Software Engineering am Beispiel des Konzepts der *Experience Factory* (Basili et al., 1994b).

2.3.1 Die Experience Factory

Eine *Experience Factory* ist eine organisatorische und/oder physische Einheit, deren Aufgabe darin besteht, Erfahrungen aus Softwareprojekten zu verwalten. Diese Erfahrungen setzen sich zusammen aus dem Software-Endprodukt selbst sowie Informationen über den Erstellungs- und Wartungsprozess, über verwendete Modelle und Werkzeuge und weitere Angaben. Die *Experience Factory* stellt Projektteams diese Erfahrungen zur Verfügung und bereitet im Gegenzug die im Projekt gemachten Erfahrungen auf, so daß sie für weitere Projekte zur Verfügung stehen. Damit ist die *Experience Factory* kein Bestandteil eines Projektteams, sondern ein Bindeglied zwischen verschiedenen Projekten.

Das Zusammenspiel von einer Projektgruppe und der *Experience Factory* beruht auf dem Paradigma der Qualitätsverbesserung (*Quality Improvement Paradigm*). Dieses besteht aus folgenden 6 Schritten:

1. **Situration beschreiben** (*Characterize*): Die Augangssituation zu Beginn des Projekts muß mit vorhandenen Modellen und Daten sowie intuitiv erfaßt werden.

2. **Ziele festlegen** (*Set Goals*): Auf der Grundlage der Ausgangssituation müssen quantifizierbare Ziele definiert werden, an denen der Fortgang des Projekts gemessen werden kann.

3. **Prozess auswählen** (*Choose Process*): Aus dem vorhandenen Erfahrungsschatz werden Prozesse mit zugehörigen Methoden und Werkzeugen entnommen, mit denen die gesetzten Ziele erreicht werden sollen.

4. **Ausführen** (*Execute*): Das Projekt wird nach den Vorgaben der vorausgegangenen Schritte durchgeführt. Als Rückmeldung an die *Experience Factory* wird regelmäßig der Erfüllungsgrad der gesetzten Ziele gemessen.

5. **Untersuchen** (*Analyze*): Am Ende des Projekts werden die Ergebnisse ausgewertet. Dabei werden auch aufgetretene Probleme aufgezeigt und Verbesserungsvorschläge für zukünftige Projekte ausgearbeitet.

6. **Zusammenfassen** (*Package*): Die gewonnene Erfahrung wird in die Erfahrungsbasis eingearbeitet. Dabei werden bestehende Modelle ergänzt, neue hinzugefügt und weiteres Wissen in strukturierter Form abgelegt.

2.3.2 Der Goal/Question/Metric-Ansatz

Eine wichtige Voraussetzung für das gesamte Vorgehen sind quantifizierbare Ziele. Sie dienen zur Bewertung der Projektabläufe, und zwar sowohl während des Projekts als auch nach dessen Abschluß. Um solche Ziele zu definieren, gibt es verschiedene Techniken. Im vorliegenden Fall wird hiervon das *Goal/Question/Metric(GQM)*-Paradigma (Basili et al., 1994a) benutzt. Hier werden Ziele in einem Top-Down-Verfahren auf meßbare Größen zurückgeführt. Ein solches Modell besteht aus drei Ebenen:

- **Konzeptuelle Ebene** (*Goal*): Ein Ziel bezieht sich immer auf ein Bezugsobjekt. Dies kann ein Produkt, ein Prozess oder eine Ressource sein. Weitere Parameter eines Ziel sind: Der Zweck, dem es dienen soll, ein Qualitätsmodell sowie der Blickwinkel, aus dem es formuliert wird.
- **Operationale Ebene** (*Question*): Das Ziel wird in Fragen zerlegt, mit denen der Erfüllungsgrad des Zieles beschrieben werden kann. Sie beschreiben das Bezugsobjekt im Bezug auf einen gewählten Qualitätsbegriff und bewerten ihn aus dem gewählten Blickwinkel.
- **Quantitative Ebene** (*Metric*): Die Fragen werden schließlich auf meßbare Größen zurückgeführt. Die dabei gemessenen Daten können objektiv sein (nur vom Bezugsobjekt abhängig, nicht aber vom Standpunkt des Bewerters) oder aber subjektiv (auch vom Standpunkt des Betrachters abhängig, wie z.B. der Grad der Zufriedenheit). Ein Maß kann zur Beantwortung mehrerer Fragen herangezogen werden. Hierbei kann der gewählte Blickwinkel zwischen den Fragen auch unterschiedlich sein, so daß Messungen am gleichen Objekt unterschiedliche Werte liefern können.

2.3.3 Vergleich mit CBR

Der in der *Experience Factory* verwendete Ansatz der Qualitätsverbesserung weist einige Parallelen zum fallbasierten Schließen auf. Diese sollen nun aufgezeigt werden.

Das Prozessmodell des fallbasierten Schließens ist durch die Implementierung eines kognitiven Modells auf dem Rechner entstanden. Dieses Modell läßt sich aber auch auf andere Bereiche übertragen. Eine fallbasierte Vorgehensweise findet sich beispielsweise auch in organisatorischen Abläufen von Unternehmen. In allen Unternehmensbereichen werden Entscheidungen dort, wo nicht immer klare Handlungsanweisungen existieren, häufig auf der Grundlage von konkreten Erfahrungen getroffen. Hier findet man dieselben 4 Schritte des CBR-Zyklus wieder, d.h. in Ergänzung zu explizit geregelten Handlungsanweisungen wurd „erfahrungsbasiertes Schließen und Handeln“ verwendet.

Das explizit Modell der *Experience Factory* kommt dem fallbasierten Modell dabei sehr nahe. Vergleicht man die 6 Schritte des *Quality Improvement*-Ansatzes mit den 4 Phasen des fallbasierten Schließens, so ergibt sich eine starke Übereinstimmung der Schritte 3, 5 und 6 mit den Phasen *Retrieve*, *Revise* und *Retain*. Schritt 4 und die *Reuse*-Phase sind nicht direkt vergleichbar, was aber lediglich an einer unterschiedlichen Einteilung der Arbeitsschritte liegt. Die *Reuse*-Phase ist im *Quality Improvement*-Ansatz bereits in Schritt 3 (Prozess auswählen) enthalten, da die ausgewählten Prozesse bereits hier zu einem gesamten Projektplan zusammengesetzt werden. Schritt 4, die Durchführung des Projekts, steht im CBR-Modell zu Beginn der *Revise*-Phase, da auch dort die gefundene Lösung in der realen Situation angewendet wird.

Die Schritte 1 und 2 finden sich ebenfalls beim fallbasierten Schließen wieder. Die Beschreibung der Ausgangssituation (Schritt 1) ist mit der Eingabe eines Anfragefalls zu vergleichen. Schritt 2, das Setzen von quantifizierbaren Zielen, ist mit der Definition von Ähnlichkeitsmaßen vergleichbar. Dieser Schritt ist im Modell des fallbasierten Schließens nicht als eigene Phase ausgewiesen, weil die Auswahl von Bewertungsmaßstäben im allgemeinen Fall der Intuition überlassen wird und im speziellen Fall der Implementierung auf dem Rechner ein — mit eventuellen Freiheitsgraden ausgestattetes — Ähnlichkeitsmaß fest vorgegeben ist.

Das Modell der *Experience Factory* liegt gerade an der Nahtstelle zwischen einem informellen Vorgehen und einem festgelegten Algorithmus. Hier wird versucht, aus informellen Zielvorgaben, die meist subjektiv sind und sich von Projekt zu Projekt ändern, objektiv meßbare Größen abzuleiten. Dieser Schritt ist keinesfalls trivial und macht spezielle Methoden wie den *GQM*-Ansatz notwendig.

Insgesamt läßt sich also festhalten, daß die *Experience Factory* sowie das CBR-Prozessmodell einige Entsprechungen und Analogien beinhalten. Während das Modell der *Experience Factory* eine Infrastruktur zur systematischen Wiederverwendung von Erfahrungswissen in der Software-Entwicklung darstellt, ist das (organisatorische) Modell des fallbaierten Schließens in der Hinsicht allgemeiner, daß es erfahrungsbasiertes Schließen und Handeln für beliebige Problemstellungen (z.B. Geschäftsprozesse) modelliert. Die vorliegende Arbeit leistet einen Beitrag, die Nützlichkeit fallbasierten Schließens zur Wiederverwendung von Erfahrungswissen im Bereich der Software-Entwicklung zu demonstrieren.

3. DATENERHEBUNG

3.1 Die Erhebungsmethode

Als unabdingbare Voraussetzung für die Erstellung einer Fallbasis müssen zunächst vor allem die Falldaten vorliegen. Das Zusammenstellen der Falldaten ist im Fall der vorliegenden Arbeit keine triviale Aufgabe, da eine Sammlung von Daten über fallbasierte Systeme bisher nicht existiert. Die Daten liegen vielmehr verstreut bei den Entwicklern der jeweiligen Systeme, wobei nicht einmal bekannt ist, wie viele Systeme überhaupt existieren.

Eine Reihe von Systemen — hauptsächlich aus der Forschung — sind an verschiedenen Stellen in der Fachliteratur beschrieben. Solche Veröffentlichungen enthalten aber meist nur Teilaspekte eines Systems, und selbst wenn ein Überblick über ein ganzes System gegeben wird, so genügt der Detaillierungsgrad doch nicht den Anforderungen der Fallbasis. Zudem ist eines der Ziele dieser Arbeit, möglichst viele laufende Systeme aus der Praxis in die Fallbasis aufzunehmen, von denen nur ein sehr geringer Teil in der Fachliteratur beschrieben ist.

Diese Augangssituation machte es nötig, die Daten direkt bei den Entwicklern von fallbasierten Systemen zu erheben. Hierzu wurde ein Fragebogen entwickelt, der über verschiedene Informationskanäle dem Fachpublikum zugänglich gemacht wurde. Der Fragebogen wurde folgendermaßen verteilt:

- an alle Teilnehmer der letzten Umfrage über fallbasiertes Schließen im deutschsprachigen Raum (siehe Woltering und Weß, 1996).
- an einige persönlich bekannte Entwickler von fallbasierten Systemen
- an die Teilnehmer der ersten *ICCBR* (International Conference on Case-Based Reasoning, 1995)
- an die ca. 600 Abonnenten der elektronischen *CBR newsletter*
- Schließlich wurde der Bogen auch im WWW publiziert. Ein Verweis auf den Fragebogen wurde in die CBR-Homepage der Universität Kaiserslautern aufgenommen, weltweit eine der wichtigsten Übersichten zum fallbasieren Schließen im WWW.[1]

[1] URL: `http://wwwagr.informatik.uni-kl.de/~lsa/CBR/CBR-Homepage.html`

Bei der Ausarbeitung des Fragebogens ergab sich folgendes Problem: Der Fragebogen sollte dieselbe Struktur aufweisen wie die Beschreibung eines Systems als Fall in der endgültigen Fallbasis. Die Modellierung der Struktur der Fallbasis setzte aber zumindest einen Teil der Falldaten voraus, um auf deren Grundlage geeignete Strukturen definieren zu können. Eine ausreichende Menge an Ausgangsdaten war wiederum nur mit Hilfe des Fragebogens zu erhalten.

Um diesen Ringschluß, der an das „Henne-und-Ei"-Problem erinnert, aufzubrechen, wurde zunächst ein vorläufiger Fragebogen entwickelt und wie oben beschrieben verteilt. Dieser erste Fragebogen, der in Anhang A.1 wiedergegeben ist, enthielt Kriterien, die auf jeden Fall für die endgültige Modellierung wichtig sein würden, wie Angaben zur Größe und Struktur der Fallbasis und einige aufgabenbezogene Kriterien. Die Struktur der Modellierung war damit noch nicht festgelegt.

Der Rücklauf des ersten Fagebogens bildete die Grundlage für die eigentliche Modellierung der Fallbasis. Hierbei waren ausführliche Kommentare von einigen CBR-Entwicklern eine große Hilfe. So stellte sich beispielsweise heraus, daß einige Fragen zu ungenau formuliert waren oder daß vorgegebene Antworten von Multiple-Choice-Fragen das vorhandene Spektrum nicht voll abdeckten. Desweiteren wurden die Kriterien neu strukturiert. Eine ausführliche Beschreibung dieser Modellierung ist der Gegenstand von Kapitel 4. Ein auf der Grundlage dieser Struktur entwickelter zweiter Fragebogen wurde nun an alle Entwickler verteilt, die auf den ersten Bogen geantwortet hatten, und an die Stelle des ersten Bogens im WWW gesetzt.

3.2 Übersicht der Ergebnisse

Bei der Auswertung der Fragebögen bestand ein Schritt auch darin, die eingegangenen Daten statistisch zu untersuchen. Der folgende Abschnitt beschreibt die Verteilung der Daten, die mit dem ersten Fragebogen gewonnen wurden.[2] Die Gliederung der Auswertung orientiert sich an der Struktur des ersten Bogens. Da die Fragen des ersten Fragebogens alle — wenn auch in anderer Reihenfolge — in den zweiten Fragebogen eingearbeitet wurden, konnten die Ergebnisse des zweiten Bogens in diese Analyse eingearbeitet werden. Wenn für ein System beide Fragebögen ausgefüllt waren, wurden die beiden Bögen zu einem Datensatz zusammengefaßt.

3.2.1 Allgemeine Angaben

Bisher sind 50 Fragebögen von verschiedenen CBR-Entwicklern eingegangen, die sich wie folgt über den Erdball verteilen:

[2] Diese Auswertung ist auch in Bartsch-Spörl et al. (1996) veröffentlicht.

Deutschland	20
Großbritannien	7
USA	7
Frankreich	4
Schweiz	3
Österreich	1
Kanada	1
Finnland	1
Rußland	1
Spanien	1
Japan	1
Neuseeland	1
Singapur	1
Südkorea	1

Tab. 3.1: Herkunftsländer der eingesandten Fragebögen

Diese Verteilung zeigt einerseits, daß der Kontakt zu Entwicklern in Deutschland am besten ist und daß die Unterstützung der Umfrage hier am größten ist. Andererseits hat die erste internationale CBR-Konferenz gezeigt, daß Europa in diesem Feld eine starke Stellung einnimmt und daß insbesondere in Deutschland fallbasierte Systeme in allen Größen und Komplexitätsklassen vorliegen.

Die meisten Systeme (36 von 50) wurden an Universitäten oder anderen Forschungseinrichtungen entwickelt. Der hohe Anteil von Systemen aus der Wissenschaft ist nicht repräsentativ für die Verteilung fallbasiserter Systeme in der Praxis. Er zeigt vielmehr, daß insbesondere im Helpdesk-Bereich die Entwicklung fallbasierter Systeme keine wichtige Rolle mehr spielt.

Ein Großteil der Systeme (29) wurde lediglich als Prototyp eingestuft, während sich nur wenige Systeme im täglichen Einsatz befanden. Die Entwickler sind bei der Beurteilung des Erfolgs ihrer Systeme überaus vorsichtig. Lediglich etwa die Hälfte der Systeme wurde als erfolgreich bezeichnet. Für die meisten übrigen Systeme war eine solche Bewertung zum Zeitpunkt der Umfrage noch nicht möglich.

Die Summe der Zahlen in den folgenden Abschnitten erreicht nicht immer 50. In diesen Fällen war eine Frage auf einigen Fragebögen unbeantwortet geblieben.

3.2.2 Die Fälle

Die erste fallorientierte Frage befaßte sich mit der Anzahl der Fälle, die ein System verwaltet. Die Größen der Fallbasen verteilen sich wie folgt:

Abbildung 3.1 zeigt, daß die meisten Systeme mit recht wenig Fällen arbeiten.

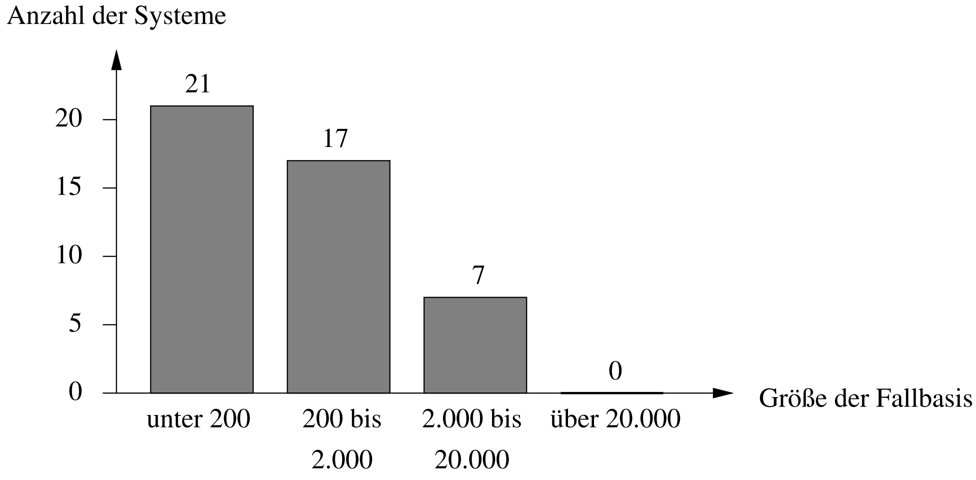

Abb. 3.1: Anzahl der Fälle im System

Insbesondere benutzen nur 6 Systeme mehr als 2.000 Fälle und keines mehr als 20.000 Fälle[3].

Die nächste Frage nach der Anzahl der Attribute pro Fall brachte folgendes Ergebnis:

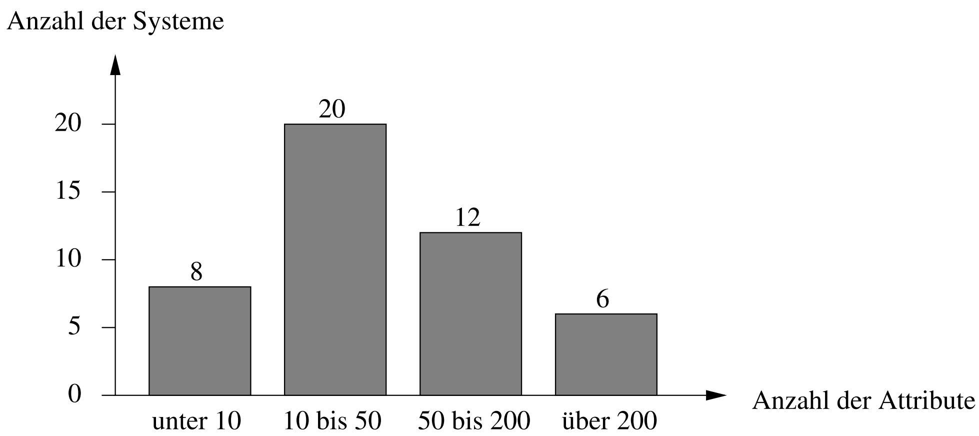

Abb. 3.2: Anzahl der Attribute pro Fall

Abbildung 3.2 ist zu entnehmen, daß ein typischer Fall zwischen 30 und 70 Attribute hat. Die Verteilung ist dabei jedoch gleichmäßiger als erwartet.

Die nächste Frage befaßte sich damit, ob diese Attribute hauptsächsich numerisch, symbolisch oder strukturiert waren.[4]

[3] Eines der Systeme arbeitete zwischenzeitlich mit bis zu 23.000 Fällen. Zum Zeitpunkt der Umfrage waren es nach einer Reorganisation der Fallbasis jedoch wieder unter 20.000

[4] Die Antwort „sowohl numerisch als auch symbolisch" war im Fragebogen nicht vorgesehen.

hauptsächlich symbolisch	16
hauptsächlich strukturiert	14
sowohl numerisch als auch symbolisch	9
hauptsächlich numerisch	6

Tab. 3.2: Struktur der Attribute

Die Antworten in Abbildung 3.2 zeigen, daß die meisten Systeme mit symbolischen Attributen arbeiten und daß strukturierte Attribute erstaunlich häufig vorkommen, während numerische Attribute nur eine untergeordnete Rolle spielen.

Dieser Eindruck bestätigt sich in der nächsten Frage, die sich mit der Art der Fallrepräsentation beschäftigt. Hier arbeiten nur 12 Systeme mit einer flachen Attribut-Wert-Liste, während 27 Systeme eine srukturierte Fallrepräsentation benötigen.

Etwa zwei Drittel der Entwickler erklären, daß ihre Fälle unabhängig vom Kontext der Anwendungsdomäne wiederverwendet werden können, während ein Drittel die Fälle als stark kontextabhängig bezeichnet. Die komplexeren Fallstrukturen fallen mit einer hohen Kontextabhängigkeit in der gewählten Anwendungsdomäne zusammen.

In der letzten fallorientierten Frage ging es darum, ob Fälle allein ausreichen, um ein aktuelles Problem zu lösen. Hier antworteten lediglich 9 Entwickler mit Ja, während 29 erklärten, daß in ihren Systemen neben den Fällen auch generelles Wissen existiert, das zur Problemlösung nötig ist. Diese Antworten zeigen, daß viele Entwickler, die den Fragebogen beantwortet haben, eher hybride auf Fallwissen und generellem Wissen basierende Systeme als reine fallbasierte Systeme entwickelt haben. Diese Tendenz korreliert wahrscheinlich stark mit der Komplexität der zu lösenden Probleme.

Insgesamt arbeiten die erfaßten Systeme mit weniger Fällen als erwartet, dafür aber mit einem größeren Anteil an komplexen Strukturen, Kontextabhängigkeiten und eingebundenem generellem Wissen. Ursache hierfür ist der überproportionale Anteil von Forschungssystemen.

3.2.3 Die Anwendungsaufgaben

Der nächste Teil des Fragebogens befaßte sich mit den unterschiedlichen Aufgabentypen, für die die fallbasierten Systeme eingesetzt werden. Die Antworten

Die Häufigkeit, mit der bei dieser Frage zwei Antworten angekreuzt wurden, rechtfertigt jedoch die Einführung einer zusätzlichen Tabellenzeile.

zeigen, daß keiner der Aufgabentypen vorherrschend ist und daß die eingegangenen Fragebögen das gesamte Feld gleichmäßig abdecken. Folgende Aufgabentypen traten auf:

Analytische Aufgaben	22	
unterteilt in:	Klassifikation	6
	Diagnose	7
	Entscheidungsunterstützung	8
	keine der genannten	1
Synthetische Aufgaben	22	
unterteilt in:	Planung	5
	Konfiguration	1
	Entwurf	16
Sonstige Aufgaben	6	

Tab. 3.3: Verteilung der Aufgaben

Die relativ einfachen Aufgaben wie Klassifikation und Diagnose sind hier schwächer vertreten als zu erwarten wäre, während die Anzahl der Entwurfssysteme sehr hoch ist — auch wenn man berücksichtigt, dass allein vom Projekt FABEL mehrere Module vertreten sind, die sich alle mit Entwurfsaufgaben beschäftigen. Nach der Studie von Mertens et al. (1993) war außerdem eine hohe Anzahl von Konfigurationssystemen zu erwarten. Diese Erwartung erfüllte sich allerdings nicht.

Eine Hälfte der „sonstigen Aufgaben" belegten CBR-Werkzeuge, die für mehr als einen Aufgabentyp geeignet sind. Die restlichen Systeme behandelten Aufgaben, die sich mit dem vorliegenden Raster nicht klassifizieren ließen.

Eine mögliche Erklärung für diese unerwartete Verteilung der Aufgabentypen besteht darin, daß der fallbasierte Ansatz tendenziell eher in solchen Bereichen eingesetzt wird, in denen andere Methoden entweder nicht existieren oder weniger geeignet sind als CBR. Ein Indiz für diese Begründung ist auch Abschnitt 3.2.2, in dem komplexe Fallrepräsentationen kombiniert mit generellem Wissen häufig auftreten.

Der Fragebogen enthielt noch weitere Fragen zur Anwendungsaufgabe (siehe Anhang A.1). Von diesen Fragen war — abhängig vom Aufgabentyp — nur ein Teil zu beantworten. Dieser Abschnitt war nur auf einem Teil der eingegangenen Fragebögen vollständig ausgefüllt, so daß die vorliegenden Daten für eine aussagekräftige Auswertung nicht ausreichen.

3.2.4 Implementierungsaspekte

Tabelle 3.4 zeigt, welche CBR-Werkzeuge zur Enwicklung von Anwendungen benutzt wurden. Die Tabelle enthält in der zweiten Spalte ein „x", wenn das

	Werkzeug	Anwendungen
CBR-Works	x	1
CBR2	x	1[a]
ESTEEM		1
KATE		2
Prodigy/Analogy	x	1
ReMind		1
Datenbanken		2

[a] Dieser Datensatz enthält Durchschnittswerte von mehreren CBR2-Anwendungen

Tab. 3.4: Benutzung von CBR-Werkzeugen

Werkzeug selbst in einem Fragebogen beschrieben ist. Die 3. Spalte zählt zusätzlich die Anzahl der beschriebenen Anwendungen, die mit diesem Werkzeug realisiert wurden.

Datenbanken lassen sich als eingeschränkte CBR-Werkzeuge betrachten und wurden deshalb ebenfalls in die Tabelle aufgenommen. Sie bieten Mechanismen für eine strukturierte Speicherung der Falldaten, Retrievalmöglichkeiten und meist auch eine Programmierschnittstelle. Das Retrieval ist hier auf relationale Abfragen beschränkt. Die Schritte *Reuse* und *Revise* müssen dabei entweder von Hand durchgeführt oder separat programmiert werden.

Die Häufigkeit der Benutzung von CBR-Werkzeugen ist deutlich geringer als erwartet. Ein Grund hierfür ist, daß viele der erfaßten Anwendungsdomänen zu komplex für die derzeit verfügbaren Werkzeuge sind. Ein anderer Grund hängt mit der hohen Anzahl von Forschungssystemen zusammen. Deren Zielsetzung ist oft geradedie Erprobung neuer Ansätze zur Weiterentwicklungn des fallbasierten Schließens, so daß sie nicht auf bestehenden Werkzeugen aufsetzen können.

Lisp	15
C, C++	11
Prolog	4
Smalltalk	3
Visual Basic	2
andere	7

Tab. 3.5: Benutzte Programmiersprachen

Die verwendeten Programmiersprachen sind in Tabelle 3.5 aufgeführt. Hier wird deutlich, daß sich prinzipiell jede Programmiersprache eignet, um ein fallbasiertes System zu implementieren. Ein Blick auf den Abschnitt „Erfahrungen“ zeigt jedoch, daß die Erfahrungen mit objektorientierten Sprachen, in Kombination mit grafischen Benutzeroberflächen, bei weitem am besten sind. Eine solche Programmierumgebung ist wohl zur Implementierung eines fallbasierten Systems am besten geeignet, sofern kein adäquates CBR-Werkzeug zur Verfügung steht.

Die meisten Systeme liefen auf Unix-Plattformen, gefolgt von PCs und Macintoshs. Nur 7 Systeme waren auf mehreren verschiedenen Plattformen lauffähig. Hier zeigt sich, daß viele Systeme Forschungsprototypen sind, die nicht für den Vertrieb an größere Benutzerkreise entwickelt wurden.

Einige Entwickler ließen wissen, daß sie an einer WWW-Version arbeiten, mit der ihr System einer großen Öffentlichkeit zugänglich sein würde. In Zukunft ist wohl mit weiteren solchen Anwendungen im WWW zu rechnen.

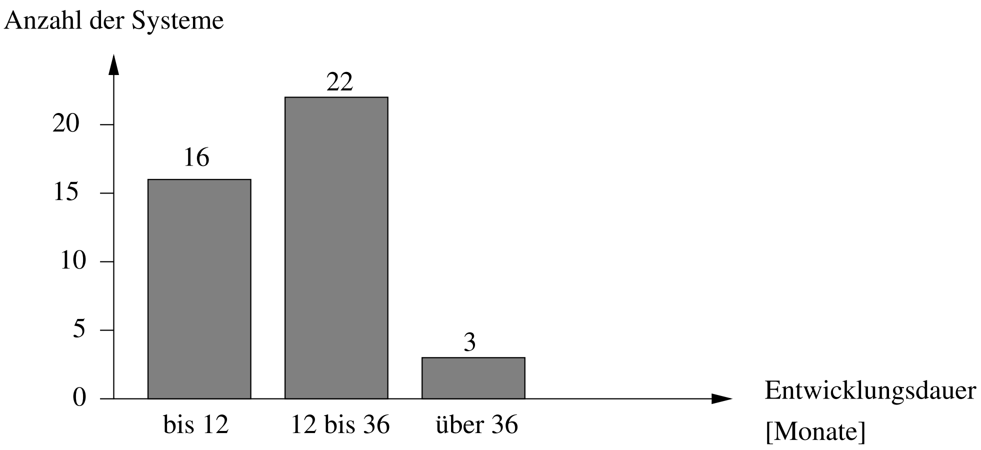

Abb. 3.3: Entwicklungsdauer

Abblidung 3.3 enthält die Antworten auf die Frage nach der Entwicklungsdauer. Hier ist zu erkennen, daß etwa ein Drittel der Systeme in weniger als 12 Monaten entwickelt wurde. Die meisten Systeme benötigten zwischen einem und drei Jahren für die Entwicklung, die bei 3 Systemen noch länger dauerte.

Eine ähnliche Verteilung zeigt der Entwlicklungsaufwand. Wie in Abbildung 3.4 zu erkennen ist, liegt er in den meisten Fällen bei bis zu 12 Mann-Monaten. Erstaunlicherweise sind die Systeme, die mit einem CBR-Werkzeug entwickelt wurden, nicht in der Kategorie mit dem geringsten Aufwand vertreten, sondern in der nächsthöheren. Hierfür ist möglicherweise ein nicht unerheblicher Aufwand zur Einarbeitung in das Werkzeug und in die Komplexität der Anwendung ausschlaggebend.

Ein weiterer interessanter Punkt ist die Anzahl der Endbenutzer. Diese Frage wurde lediglich auf der Hälfte der eingegangenen Fragebögen beantwortet, da sie für viele Forschungsprototypen und Systeme im Entwicklungsstadium noch nicht sinnvoll angegeben werden kann. Der hohe Anteil an Systemen mit nur

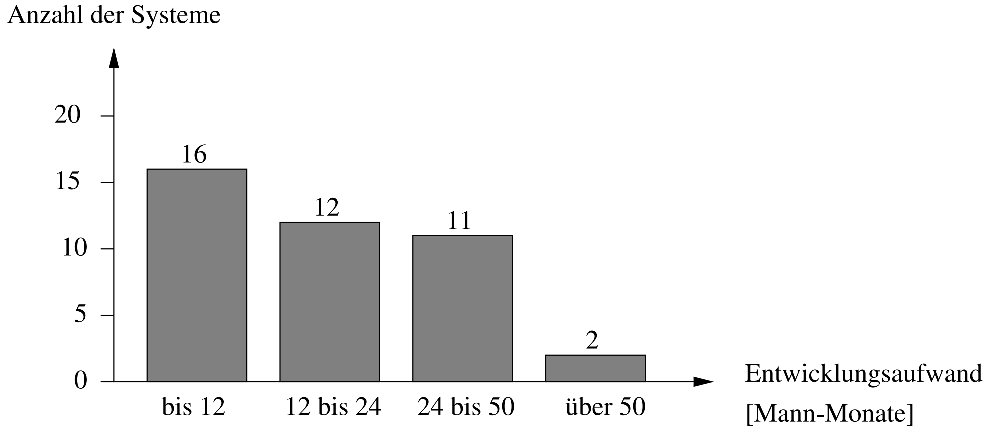

Abb. 3.4: Entwicklungsaufwand

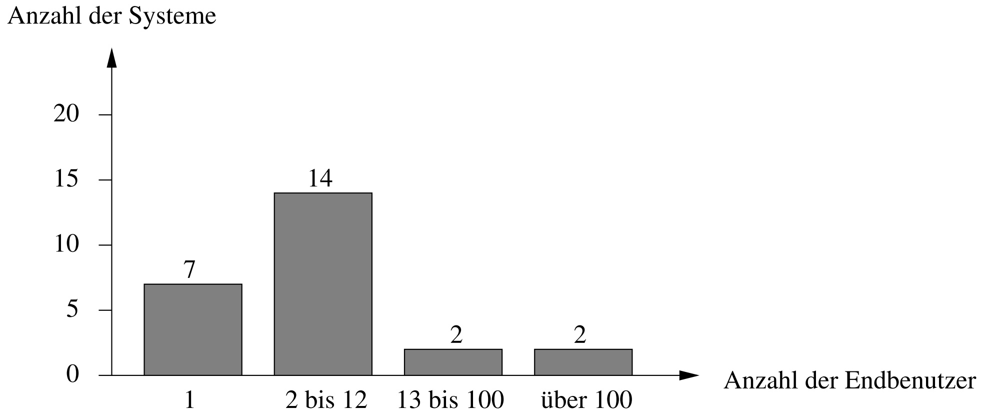

Abb. 3.5: Anzahl der Endbenutzer

einem oder sehr wenigen Benutzer in Abbildung 3.5 besteht ebenfalls großteils aus Prototypen.

3.3 Erfahrungen

Der interessanteste Teil des Fragebogens war der informelle Teil, der die Möglichkeit gab, Erfahrungen frei mitzuteilen. Die Antworten in diesem Teil zeigen kein gemeinsames Muster, doch gibt es einige Gedanken, die mehrfach geäußert wurden.

- CBR-Werkzeuge der ersten Generation wie ReMind oder ESTEEM waren nur für die Konstruktion von Prototypen geeignet, nicht aber für das endgültige System.

- Viele Entwickler berichteten von sehr guten Erfahrung mit objektorientierten Programmiersprachen und entsprechenden Entwicklungsumgebungen.
- Für besonders große Fallbasen empfiehlt sich der Einsatz von Datenbanksystemen für die Verwaltung der Fälle.
- Besonders wichtig ist die genaue Festlegung der Rolle und des Einsatzbereiches eines fallbasierten Systems, da hiervon viele weitere Entwurfsentscheidungen abhängen.
- Es ist von großer Bedeutung, möglichst frühzeitig Verantwortliche für den Einsatz und die Weiterentwicklung eines Systems zu bestimmen und in die Entwicklung einzubeziehen, insbesondere in Bereichen, die diese später übernehmen und weiterführen müssen.
- Die Schulung und Betreuung der Endbenutzer verdient hohe Aufmerksamkeit, besonders wenn die Endbenutzer noch nicht mit dem Einsatz von Softwaresystemen vertraut sind.
- Nicht wenige Entwickler berichteten, daß die Sammlung und Modellierung der Fälle nicht so einfach war wie erwartet. Gerade wenn Experten diese Daten zusammenstellen und die Qualität von Fällen beurteilen müssen, kann dieser Entwicklungsschritt äußerst zeitraubend sein.
- Während bei Klassifikations- und Diagnoseaufgaben der Ähnlichkeitsbegriff meist als Abschätzung der Wahrscheinlchkeit, die richtige Klasse bzw. Diagnose zu finden, benutzt wird, dient sie bei synthetischen Aufgaben und Entscheidungsunterstützungsaufgaben in der Regel der Abschätzung der Wiederverwendbarkeit bzw. Adaptierbarkeit eines Falles.
- Die Modellierung der Domäne ist ebenfalls eine nichttriviale Aufgabe, von deren Ergebnis der Erfolg des gesamten Systems abhängt. Dasselbe gilt auch für die Wahl von Gewichten bei gewichteten Ähnlichkeitsmaßen.
- Der gesamte Bereich der Reorganisierung und Pflege des generellen Wissens muß in der Entwicklungsphase und erst Recht während des Einsatzes des Systems beachtet werden.
- Die Entwickler von Systemen, die sich im täglichen Einsatz befinden, legen Wert auf:
  - die Integration des CBR-Systems in das Arbeitsumfeld des Endbenutzers, was manchmal eine Neustrukturierung von Geschäftsprozessen bedeutet, sowie
  - eine gut durchdachte und leicht bedienbare Benutzeroberfläche.

Gerade der letzte Punkt zeigt, daß die Technik des fallbasierten Schließens selbst für den Endbenutzer von untergeordneter Bedeutung ist. Er erwartet lediglich ein System, daß seine Anforderungen erfüllt und dabei leicht zu benutzen und zu warten ist.

4. MODELLIERUNG DER FÄLLE

4.1 Allgemeines zur Modellierung

Bei der Modellierung einer Domäne für eine CBR-Anwendung sind vor allem zwei Fragen von Bedeutung, die bestimmen, welche Angaben aus der Domäne überhaupt in die Fallrepräsentation aufgenommen werden sollen:

- Was weiß ein Anfragender?
- Was interessiert einen Anfragenden?

Die Antwort auf die erste Frage legt fest, welche Informationen eine Anfrage an das System enthalten kann. Die zweite Frage bestimmt, welche Angaben zusätzlich in der Fallbasis enthalten sein sollten. Eine Eingrenzung der zu modellierenden Informationen ergibt sich, wenn man die Fragen negiert, d.h. wenn man fragt, was ein Anfragender *nicht* weiß und was ihn *nicht* interessiert. Dadurch lassen sich überflüssige Aspekte eliminieren, die entweder bei Anfragen nie eingegeben werden können oder als Antworten keine für den Benutzer relevanten Informationen enthalten.

Die für Anfragen relevanten Kriterien stammen hauptsächlich aus den folgenden Bereichen, die man auch unter der Bezeichnung *Knowledge-Level-Beschreibung* (Althoff und Aamodt, 1996) zusammenfaßt:

- zu erreichende **Ziele**
- zu lösende **Aufgaben**
- **Methoden**, die verwendet werden sollen
- **Wissen über die Anwendungsdomäne**

Die zu verwendende Methode ist im Kontext dieser Arbeit grundätzlich das fallbasierte Schließen. Die Ziele sind indes nur schwer von den zu lösenden Aufgaben zu trennen. Es bleiben auf der Knowledge-Ebene also die zwei Bereiche *Domänen-* und *Aufgabenkriterien* übrig.

Hinzu kommen häufig noch Anforderungen an *technische* oder *ergonomische* Eigenschaften eines Systems. Insgesamt sind also gerade die vier Kriterienbereiche relevant, die in Althoff und Bartsch-Spörl (1996) und in Althoff et al. (1994, 1995b) beschrieben sind. Die hier beschriebene Modellierung basiert im wesentlichen auf diesen Kriterien.

Von den dort aufgezählten Kriterien ist jedoch nur ein Teil relevant, weil der Kontext, in dem diese Kriterien dort definiert werden, ein anderer ist als in dieser Arbeit. In Althoff et al. (1994, 1995b) dienen die Kriterien zur Evaluation bestehender Systeme, die im direkten Vergleich miteinander bewertet werden. Dabei kommt es darauf an, möglichst genau den Stand der Technik zu beschreiben, weshalb algorithmische Einzelheiten sehr detailliert beleuchtet werden.

Für die Fallrepräsentation in dieser Arbeit werden die Systeme jedoch aus der Sicht des Anwenders beschrieben. Für ihn sind viele technische Details des System, wie z.B. der verwendete Intexierungsmechanismus, überhaupt nicht sichtbar, während sie in den Evaluationsarbeiten von großer Bedeutung waren.[1] Auch Kriterien, für die in Althoff et al. (1994) umfangreiche Tests unter Laborbedingungen durchgeführt wurden, kommen für diese Modellierung nicht in Frage, da von den Entwicklern, die den Fragebogen ausfüllen, nicht verlangt werden kann, daß sie diese Tests durchführen. Statt dessen wird bei Anwendungen davon ausgegangen, daß die Systeme so konzipiert sind, daß sie die Anforderungen der Anwendungsdomäne und -Aufgabe erfüllen. Da dies auch im Interesse der Entwickler liegt, dürfte diese Annahme auch in den meisten Fällen zutreffen.

Die folgenden Abschnitte beschreiben die Modellierung, wie sie in den Fragebögen[2] für Anwendungen und für Werkzeuge festgelegt ist. Diese Modellierung liegt auch der Implementierung (siehe Kapitel 5) zugrunde. Dabei wird die Modellierung für fallbasierte Anwendungen und Werkzeuge zusammen beschrieben. Die Unterschiede zwischen den beiden Modellierungen bestehen im wesentlichen darin, daß bei Werkzeugen einige von der Anwendungsdomäne abhängige Kriterien entfallen. Dies betrifft den kompletten Bereich der Domänenkriterien aber auch einige Kriterien aus den anderen Abschnitten.

Die Fragen, die schon im ersten, provisorischen Fragebogen enthalten waren, wurden teilweise verändert. So wurden bei Fragen, bei denen Zahlenwerte erhoben werden, die vorgegebenen Intervalle, aus denen eines auszuwählen war, durch ein freies Eingabefeld ersetzt, in das die genaue Zahl einzugeben ist. Die so erhaltenen Zahlen sind bei einer Ähnlichkeitsberechnung leichter zu handhaben und eine Eingabe dieser Zahlen schien für die Adressaten des Fragebogens noch zumutbar.

[1] Selbst hier konnten nicht alle Inexierungsmechanismenverglichen werden, weil sie bei einigen Systemen nicht dokumentiert waren.

[2] Die kompletten Fragebögen sind im Anhang abgedruckt.

Bei Kriterien, die als Ja/Nein-Fragen formuliert sind, wurde fast durchgängig ein dritter Wert eingeführt, um die Nennung „teilweise“ zu ermöglichen.

Am Ende jedes Abschnitts enthalten die Fragebögen zusätzlich ein Feld, in das Kommentare zum gerade bearbeiteten Abschnitt frei eingetragen werden können. Diese Felder können Informationen aufnehmen, die nicht von den Kriterien abgedeckt werden, aber trotzdem aus der Sicht der ausfüllenden Person wichtig sind. Außerdem können hier Kommentare zur Struktur oder Gestaltung der Fragebögen selbst eingetragen werden, die es ermöglichen, die Fragebögen und damit die Modellierung weiter zu verbessern.

Zur Notation: Viele Kriterien sind in fließendem Text beschrieben. Um diese Kriterien leichter auffindbar zu machen, wurden die wichtigsten Stichworte, die sie beschreiben, in **Fettdruck** *gesetzt.*

4.2 Allgemeine Angaben

Neben der eigentlichen detaillierten Beschreibung eines Systems sind einige allgemeine Angaben wichtig, die das System zum einen identifizieren und zum anderen Informationen zum System als Ganzem bzw. zum zugehoerigen Projekt beinhalten.

Der Identifikationsteil setzt sich wie folgt zusammen:

An erster Stelle wird ein System über seinen **Namen** identifiziert. Gleichzeitig sollte in diesem Feld auch die *Versionsnummer* erscheinen. Diese ist wichtig, um den Entwicklungsstand eines Systems zu dokumentieren und erlaubt es, verschiedene Versionen eines Systems in der Fallbasis zu speichern. Wenn der Name des Systems ein Akronym ist, enthält die *Auflösung des Akronyms* oft wertvolle Informationen über den Anwendungsbereich oder den technischen Hintergrund seiner Implementierung.

Der oder die **Hersteller bzw. Entwickler** des Systems folgen unmittelbar danach und hat eine ähnliche Bedeutung wie der Autorenangabe eines Buches. Wurde das System von einem größeren Konsortium entwickelt, so sollte hier die komplette Liste der Projektpartner erscheinen.

Die **Kontaktadresse** dient als Anlaufstelle für Fragen von Benutzern der Fallbasis. Hier sollte möglichst eine konkrete Person als Ansprechpartener stehen.

Neben der Kontaktperson können auch weitere **Referenzen** angegeben werden. Hier ist Platz für bis zu drei *Literaturhinweise* sowie die Adresse einer *WWW-Seite*.

Die übrigen Angaben dieses Abschnitts betreffen allgemeine Merkmale des Systems und des Projektes, aus dem es hervorgegangen ist:

Zunächst ist der **Status** des System von Interesse. Folgende Einträge sind hier möglich:

- Prototyp
- System für den praktischen Einsatz befindet sich in der Entwicklungsphase — mit Angabe des voraussichtlichen Fertigstellungsdatums
- Pilotsystem
- System im praktischen Einsatz, zusammen mit dem Datum der Inbetriebnahme
- Der Betrieb des Systems wurde wieder eingestellt. Hier kann sowohl für die Inbetriebnahme als auch für das Ende der Betriebsdauer ein Datum angegeben werden.

Das nächste Feld enthält eine **Beurteilung des Systems** durch den Entwickler. Folgende Bewertungen können vergeben werden:

- Erfolg
- Mißerfolg
- ambivalente Beurteilung
- Beurteilung noch nicht erfolgt

In den nächsten Feldern können **konkrete Erfahrungen** beschrieben werden, die durch die Entwicklung und/oder den Betrieb des fallbasierten Systems gewonnen wurden. Dieser Abschnitt gliedert sich in Erfahrungen

- in Bezug auf den Entwicklungsprozess
- in Bezug auf die verwendeten Entwicklungswerkzeuge
- in Bezug auf das betriebliche Umfeld
- in Bezug auf sonstige Themen

Schließlich können CBR-System-Entwickler noch angeben, auf welche Art andere Entwickler von ihren Ergebnissen profitieren können. Dies kann auf folgende Art geschehen:

- durch persönliche Kontaktaufnahme
- durch Anforderung von Beschreibungen des Systems von der angegebenen Kontaktadresse
- durch Weitergabe der Fallbasis/Fallbasen
- durch Weitergabe von Softwarekomponenten.

4.3 Beschreibung der Domäne

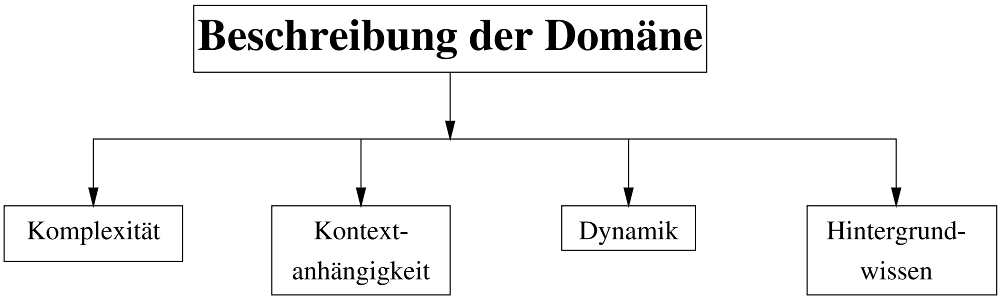

Abb. 4.1: Beschreibung der Domäne

Dieser Abschnitt beginnt zunächst mit einer informellen Beschreibung der Domäne. Diese setzt sich zusammen aus:

- dem **Namen** der Domäne bzw. einer kurzen stichwortartigen Beschreibung und
- einer ausführlichen **textuellen Beschreibung**.

Als Maß für die **Komplexität der Domäne** fungiert hier die Frage, ob sich die Domäne als flache Liste von Attribut-Wert-Paaren darstellen läßt. Auch wenn diese Einteilung sehr grob ist, hat dieses Kriterium eine Schlüsselfunktion: Es bestimmt die Komplexität der Fallrepräsentation und damit auch in hohem Maße die Komplexität des Retrievalmechanismus. Für flache Listen existieren bereits sehr effiziente Indexierungsmethoden, wie z.B. die k-d-Bäume in INRECA (Derwand, 1994; Weß, 1995). Solche Methoden lassen sich bei komplexen Fallrepräsentationen nur bedingt oder gar nicht anwenden, so daß hier in der Regel die gesamte Fallbasis linear durchsucht werden muß. Hinzu kommt noch, daß bereits die Bestimmung der Ähnlichkeit eine höhere Komplexität hat.

Die **Kontextabhängigkeit** spielt eine große Rolle bei der Wiederverwendung von Fällen. Wenn alle für die Wiederverwendung relevanten Kontextdaten in den Falltaten repräsentiert werden können, kann sich die Reuse-Funktion auf die Verarbeitung der Falldaten beschränken. Andernfalls muß sie Kontextwissen aus anderen Quellen hinzuziehen.

Die **Notwendigkeit von generellem Wissen** ist ein weiterer Faktor, der die Anforderungen einer Domäne an die Fähigkeiten eines fallbasierten Systems bestimmt. Diese Notwendigkeit kann auf einer dreiwertigen Skala mit folgenden Werten angegeben werden:

- irrelevant: Um solche Domänen zu handhaben, kommt ein CBR-System ohne Verwaltung von generellem Wissen aus.

- manchmal sinnvoll: Für solche Domänen eignen sich Systeme, die relativ einfache Mechanismen zur Verfügung stellen, mit denen das Ergebnis eines rein fallbasierten Vorgehens noch verbessert werden kann.
- notwendig: Hier führen rein fallbasierte Methoden nur zu unbefriedigenden Ergebnissen. Ein CBR-System für solche Domänen muß das vorhandene generelle Wissen in einer geeigneten Form verwalten können und es dort, wo es nötig ist, in die Berechnung der Ähnlichkeit oder in den Adaptionsvorgang einbeziehen.

Die **Dynamik der Fallbasis** bestimmt die Anforderung and die Funktionaliät, die ein CBR-System für die Verwaltung von Fallwissen bereithält. Sie läßt sich durch folgende zwei Parameter ausdrücken: Die Häufigkeit von Änderungen der Fallstruktur und die Aufnahmehäufigkeit neuer Fälle.

Änderungen der Fallstruktur könnten nötig sein, wenn in der behandelten Domäne selbst Änderungen auftreten. Dies kann beispielsweise der Fall sein, wenn an einer zu wartenden Maschine einige Komponenten durch andere ersetzt werden. Andere Domänen können als statisch angesehen werden. Hier sind Änderungen der Fallstruktur nur dann nötig, wenn die bisherige Modellierung korrigiert werden muß oder wenn zusätzliche Aspekte als wichtig erkannt wurden und in die Fallrepräsentation integriert werden müssen.

Wenn häufige Strukturänderungen zu erwarten sind, sollte die Software in der Lage sein, die Fallrepräsentation geeignet zu modifizieren, ohne daß dadurch die bisherigen Falldaten wertlos werden. Hierfür eignen sich besonders objektorientierte Repräsentationsmechanismen, die Strukturen, die sich nur in Teilaspekten unterscheiden, als Unterklassen verwalten können.

Das **Hinzufügen neuer Fälle** ist im Prozessmodell des fallbasierten Schließens explizit vorgesehen. Jedes CBR-System sollte also prinzipiell auf diese Aufgabe vorbereitet sein. Je nachdem, wie häufig dieser Vorgang auftritt, stellen sich jedoch unterschiedliche Anforderungen an die Verwaltung der Falldaten, insbesondere an die Verwaltung von Indizes. Wenn die Fallbasis annähernd vollständig ist, so daß kaum noch neue Fälle aufgenommen werden müssen, so lohnt es sich, auch mit hohem Aufwand erzeugte statische Indexstrukturen bei jeder Erweiterung der Fallbasis neu zu berechnen. Ist die Aufnahme neuer Fälle jedoch an der Tagesordnung, so sind Strukturen nötig, die neue Fälle ohne eine komplete Neuberechnung integrieren können oder solche, die erst zur Laufzeit erzeugt werden.

Schließlich ist noch die **Herkunft des Wissens** von Interesse. Diese läßt sich wie folgt klassifizieren:

- Expertenwissen, aufgeteilt in Fallwissen und generelles Wissen
- Technische Dokumentationen, wie z.B. Maschinenpläne

- Wartungsberichte
- andere wissensbasierte Systeme
- Datenbanken
- sonstige Quellen

Wenn das Wissen bereits in Datenbanken oder wissensbasierten Systemen elektronisch verfügbar ist, benötigt das CBR-System eine Schnittstelle für den Datenaustausch, die entweder die Daten im vorliegenden Format direkt lesen kann oder mit den anderen Systemen über ein spezielles Austauschformat kommuniziert.

4.4 Beschreibung der Aufgabe

Eine Reihe von Aspekten der Aufgabe, die ein fallbasiertes System erfüllt, läßt sich unabhängig von der Domäne beschreiben. Mit diesen Kriterien können Systeme verglichen werden, die zwar auf unterschiedlichen Domänen operieren, aber strukturell ähnliche Aufgaben bearbeiten.

Dieser Abschnitt fällt bei der Beschreibung von fallbasierten Werkzeugen nicht vollständig aus. Auch wenn viele Werkzeuge für mehrere Aufgabentypen benutzt werden können, gibt es immer noch bestimmte Typen, für die sie besser geeignet sind sowie andere, für die sie nicht geeignet sind. So sind beispielsweise einige Werkzeuge auf analytische Aufgaben spezialisert, während sie bei synthetischen Aufgaben versagen. Der aufgabenspezifische Teil der Beschreibung eines CBR-Werkzeuges besteht daher im wesentlichen aus einer **Liste der Aufgaben**, die das Werkzeut bearbeiten kann. Zusätzlich kann noch ein **Szenario** beschrieben werden, falls das Werkzeug für eine spezifische Anwendungssituation, wie z.B. Helpdesk-Anwendungen, entwickelt wurde.

4.4.1 Allgemeine Aufgabenkriterien

Bei der Modellierung von Anwendungen beginnt der Aufgabenteil ähnlich wie der Domänenteil mit dem **Namen** der Aufgabe und einer **textuellen Beschreibung**. Die Beschreibung der Aufgabe ist zwar oft nur schwer von der Domänenbeschreibung zu trennen, jedoch bewirkt die Trennung der beiden Teile eine Strukturierung der Beschreibung, die den Vergleich zweier Systeme einfacher macht. Außerdem wird damit sichergestellt, daß beide Aspekte in der Beschreibung des Systems abgedeckt werden.[3]

[3] Der erste Fragebogen (siehe Anhang A.1) enthielt nur ein einziges Feld zur textuellen Beschreibung des Systems. Hier wurden häufig Zusammenfassungen von Veröffentlichungen eingetragen, die sich mit Einzelaspekten der Systeme (meist technicher Art) befaßten, aber keinen Gesamtüberblick boten.

Die wichtigste Angabe im aufgabenbezogenen Teil ist der **Aufgabentyp**. Dieser kann wie in Abschnitt 2.2 allgemein als analytisch oder synthetisch angegeben werden oder genauer als Klassifikation, Diagnose oder Entscheidungsunterstützung bzw. als Planung, Konfiguration oder Entwurf. Für Aufgaben, die nicht in dieses Schema passen, oder solche, die sich nur als analytisch oder synthetich klassifizieren lassen, ist ein Textfeld zur Beschreibung des Aufgabentyps vortesehen.

Je nach Aufgabentyp sind in diesem Teil der Modellierung auch unterschiedliche Kriterien relevant (vgl. Abschnitte 4.4.2 und 4.4.3).

Die Kriterien, die alle Aufgabentypen betreffen, befassen sich mit der Frage, wie wichtig gewisse Funktionalitäten für die gegebene Aufgabe sind. Wird eine Funktionalität hier als unwichtig eingestuft, so kann man davon ausgehen, daß eine relativ einfache Implementierung dieser Funktionalität ausreicht, um die Aufgabe zufriedenstellend zu lösen. Im Gegenzug bedeutet die Einstufung „wichtig", daß eine mächtige und vor allem effiziente Implementierung erwartet wird.

Die in ihrer Wichtigkeit zu beurteilenden Funktionalitäten sind:

- Das fallbasierte Schließen an sich
- Fuzzy-matching
- Der Einsatz von gewichteten Ähnlichkeitsmaßen
- Das Lernen von Gewichten und
- Adaption.

4.4.2 Analytische Aufgaben

Die aufgabenspezifischen Kriterien beziehen sich hauptsächlich auf Eigenschaften der Falldaten. Da bei jedem Aufgabentyp mit den Fällen anders umgegangen wird, sind die Kriterien je nach Aufgabentyp unterschiedlich.

Bei analytischen Aufgaben, bei denen die Lösung meist direkt aus den Falldaten entnommen wird, ist die Qualität der Falldaten von großer Bedeutung. Diese läßt sich durch den **Verrauschungsgrad** und den **Anteil unvollständiger Information** ausdrücken. Da diese beiden Werte schwer zu ermitteln sind, genügt hier die Angabe eines Intervalls, in dem die Zahlen, ausgedrückt in Prozent, liegen.

Die Modellierung der Kriterien, die die Strukturierung der Fallbasis betreffen, ist nach den einzelenen Typen analytischer Aufgaben unterteilt. Teilweise werden in den getrennten Abschnitten gleiche Sachverhalte beschrieben. Da aber abhängig vom Aufgabentyp verschiedene Begriffe verwendet werden, und die Werte auch wegen der unterschiedlichen Bedeutung nur bedingt vergleichbar sind, werden diese Kriterien getrennt erfaßt.

Der erste Parameter dieser Art ist die Partitionierung der Fallbasis. Bei Klassifikations- und Diagnoseaufgaben ist dies die **Anzahl der Klassen bzw. Diagnosen**. Die analoge Angabe bei Entscheidungsunterstützungsaufgaben ist die **Anzahl von Attributkombinationen**, die als Lösungen in Frage kommen. In diesem Fall kann man allerdings nicht unbedingt von einer Partitionierung sprechen.

Dort, wo echte Partitionen vorliegen ist weiterhin die **Anzahl der Fälle pro Klasse bzw. Diagnose** von Interesse. Da diese Zahl in der Regel von einer Partition zur anderen unterschiedlich ist, genügt hier ein Durchschnittswert.

Eine wichtige Eigenschaft von Diagnoseaufgaben, die auch häufig in der Entscheidungsunterstützung auftritt ist die Selektion von Tests, die während der Systemkonsultation durchzuführen sind. Bei diesen beiden Aufgabentypen wird daher die durchschnittliche **Anzahl von Tests** erhoben.

Bei Entscheidungsunterstützungsaufgaben stellt sich schließlich noch die Frage, ob sich durch eine weitere Strukturierung der Domäne die Organisation der Fallbasis oder die Ähnlichkeitsberechnung verbessern läßt.

4.4.3 Synthetische Aufgaben

Da synthetische Aufgaben in der Regel deutlich komplexer sind als analytische Aufgaben, ist hier die **Zerlegbarkeit der Aufgabe** von Interesse. Wenn die Aufgabe derart in Teilaufgaben zerlegbar ist, daß man die Gesamtlösung aus den Lösungen dieser Teilaufgaben zusammensetzen kann, so bedeutet dies eine deutliche Herabsetzung der Komplexität des Lösungsprozesses.

Der Rest dieses Abschnittes ist wiederum in die drei Aufgabentypen Planung, Konfiguration und Entwurf unterteilt. Der Aufbau ist aber jeweils gleich. Diese Fragen befassen sich mit der Komplexität einer Lösung der jeweiligen Aufgabe. Dazu wird zunächst die **Anzahl der Teillösungen** erfragt. Diese Teillösungen können auch voneinander abhängen. Je nach Aufgabentyp werden diese Teillösungen als Teilaufgabe (Planung), Komponente (Konfiguration) oder als Entwurfs-Elemente bezeichnet.

Als Maß für die Komplexität einer solchen Teillösung dient schließlich die **Anzahl der Parameter pro Teillösung** (Teilaufgabe / Komponente / Entwurfs-Element). Diese Zahl macht die Angaben zur vorhergehenden Frage erst vergleichbar.

4.5 Technische Kriterien

Nach der Beschreibung der Anforderungen, die eine Domäne und eine Aufgabe an ein fallbasiertes Sysem stellen, folgt nun die technische Beschreibung des Systems, das diesen Ansprüchen gerecht werden soll. Die Kriterien dieses Abschnitts beziehen sich bis auf einige Ausnahmen sowohl auf Anwendungen als auch auf CBR-Werkzeuge.

Die Auflistung beginnt mit den Ausnahmen, die auf Werkzeuge nicht anwendbar sind, nämlich mit den Charakteristika der Fallbasis. Zunächst steht hier die **Anzahl der Fälle** der Anwendung. Ein Maß für das Datenvolumen, das beim Retrieval maximal durchsucht werden muß, erhält man unter Hinzunahme der **Anzahl der diskriminanten Attribute in einem Fall**. Nichtdiskriminante Attribute, die zusätzliche Informationen enthalten, bleiben unberücksichtigt. Sie erhöhen zwar den gesamten Speicherbedarf einer Anwendung, nicht aber den Aufwand des Retrievalprozesses.[4]

Ein weiteres wichtiges Charakteristikum der Fallbasis ist der vorherrschende **Typ der ähnlichkeitsrelevanten Attribute**. Hier wird zwischen symbolischen, numerischen und strukturierten Attributen unterschnieden. Sind symbolische und numerische in etwa gleichem Maß vorhanden, kann man hier auch „symbolisch und numerisch" als vierte Antwortmöglichkeit angeben.[5]

Die folgenden Kriterien befassen sich mit dem Einsatz von Hingergrundwissen, getrennt nach den Teilaufgaben *retrieve* und *reuse* aus dem Prozessmodell. Sie betreffen sowohl Anwendungen als auch Werkzeuge. Es besteht allerdings zwischen Anwendungen und Werkzeugen ein Unterschied in der Semantik dieser Kriterien. Bei Anwendungen wird festgestellt, welche Wissensarten tatsächlich benutzt werden, während bei sich Werkzeugen die Frage stellt, welche Möglichkeiten prinzipiell zur Verfügung stehen.

Für beide Teilaufgaben *retrieve* und *reuse* wird zunächst jeweils festgestellt, **ob generelles Wissen benutzt wird bzw. benutzt werden kann**. Ist dies der Fall, so stehen Textfelder zur Verfügung, in der — wiederum getrennt nach *retrieve* und *reuse* — die **Art des generellen Wissens** eingegeben werden kann.

Da mittlerweile im Bereich des fallbasierten Schließens eine vielzahl unterschiedlicher Algorithmen existiert, ist es kaum möglich, ein festes Raster zur Beschreibung algorithmischer Details anzugeben.[6] Um dennoch zumindest eine informelle Beschreibung zu ermöglichen, ist ein Feld vorgesehen, in dem die Entwickler

[4] Zumindest dann nicht, wenn das „Überspringen" der nicht ähnlichkeitsrelevanten Attribute keinen signifikanten Aufwand bedeutet.

[5] siehe auch Kapitel 3.2.1.

[6] Selbst beim Vergleich von nur wenigen CBR-Werkzeugen in Althoff et al. (1994) stellte die Einordnung der auftretenden Indexstrukturen und Retrievalalgorithmen in das Bewertungsraster eine der schwierigsten Hürden dar. Dieses Problem wurde durch eine uneinheitliche Terminologie noch verstärkt.

nach Belieben **besondere technische Eigenschaften** ihrer Systeme eintragen können, die sie in der Beschreibung ihres Systems für wichtig halten.

Die übrigen technischen Kriterien befassen sich mit der Implementierung des Systems. Bei Anwendungen interessiert hier zunächst ein eventuell **verwendetes Werkzeug**. Der Rest der Kriterien dient wiederum gleichermaßen der Beschreibung von Anwendungen und Werkzeugen. Sie wird fortgesetzt mit der Angabe der **verwendeten Programmiersprachen**.

Ein wichtiger Punkt ist die Angabe des **Betriebssystems**, unter dem ein fallbasiertes System abläuft. Nach den Erfahrungen mit dem ersten Fragebogen[7] sind die Antwortmöglichkeiten hier vorgegeben:

- Einfaches MS-DOS (ohne Windows)
- MS Windows
- Unix
- Macintosh
- Mainframe
- andere Betriebssysteme

Den Abschluß des Implementationsteils bilden die Angaben zum Implementierungsaufwand. Dieser wird einmal als **Entwicklungsdauer** in Monaten angegeben und zusätzlich als **Entwicklungsaufwand** in Mann-Monaten.

4.6 Ergonomische Kriterien

Die ergonomischen Kriterien befassen sich mit den Schnittstellen eines fallbasierten Systems zur Außenwelt. Dieser Abschnitt ist — mit Ausnahme der Anzahl der Endbenutzer — für Anwendungen und Werkzeuge gleichermaßen relevant. Am wichtigsten ist die Schnittstelle zum Benutzer, die als erste behandelt wird.

Die erste Frage zielt nun auf die Identifikation der **Klassen von Benutzern**, die mit dem System arbeiten. In den Ansprüchen an die Benutzerführung bestehen nämlich deutliche Unterschiede zuwischen den unterschiedlichen Benutzerklassen. Systemadministratoren sind hierbei ausgenommen, da sie in jedem Fall existieren und jedes System eine Art von Wartungsmodus haben muß. Folgende Klassen stehen zur Auswahl:

[7] Hier war die Frage offen gestellt. Unter den Antworten waren zum Teil Angaben zur Hardware („PC“) und einige exotische Betriebssysteme, deren Einordnung in die angegebene Liste (meist unter „Unix“) profunde Kenntnisse des Betriebssystemmarktes voraussetzten.

- Experten, die sich in der Domäne auskennen
- Weiteres Personal der Firma, die das System einsetzt
- Kunden
- andere Benutzer.

Bei Anwendungen kommt hier noch die **Anzahl der Endbenutzer** hinzu. Diese Zahl hat ebenfalls einen großen Einfluß auf den individuellen Anpassungsbedarf eines Systems sowie auf den Wartungsaufwand.

Die **Benutzerführung** während der Konsultation des Systems gehört zu den wichtigsten Aspekten bei der Beschreibung der Benutzerschnittstelle. Das entscheidende Kriterium ist die Freiheit des Benutzers, bei der Eingabe einer Anfrage die Reihenfolge der einzugebenden Attribute selbst zu bestimmen. Dabei kann man folgende Modi der Benutzerführung unterscheiden:

systemgesteuert: Die Reihenfolge der Attribute ist durch das System vollständig festgelegt. Bei Diagnose- oder Entscheidungsunterstützungsaufgaben kann dies beispielsweise über eine Testselektionsstrategie erfolgen.

benutzergesteuert: Hier hat der Benutzer die völlige Freiheit, die Reihenfolge, in der er Daten eingibt, selbst zu bestimmen.

Kombinationen aus den obengenannten Modi.

Nach der Benutzerschnittstelle folgt nun noch die Beschreibung der Möglichkeiten des Datenaustauschs mit anderen Systemen. Zunächst interessiert hier grundsätzlich, welche Daten einer Anwendung mit anderen Systemen ausgetauscht werden können. Diese Angaben können für **Export** und **Import** getrennt vorgenommen werden, da die Schnittstellen oft nur einseitig sind. Die Daten einer Anwendung werden hier unterteilt in:

- Falldaten
- Beschreibung der Fallstruktur
- Ähnlichkeitsmaße
- Generelles Wissen.

Schließlich wird noch festgestellt, mit welchen **Datenformaten** die oben angegebenen Schnittstellen arbeiten, um mit anderen Systemen zu kommunizieren. Folgende Möglichkeiten können dabei angegeben werden:

- ASCII-Formate

- Formate von Tabellenkalkulationsprogrammen
- Datenbankformate
- Fallrepräsentationssprachen
- andere Formate.

Dabei besteht jeweils die Möglichkeit, die benutzten Formate einzeln zu benennen.

5. IMPLEMENTIERUNG

Die in den vorangegangenen Kapiteln beschriebenen Konzepte wurden zusammen mit den aus der Umfrage erhaltenen Falldaten im Rahmen dieser Diplomarbeit in einer fallbasierten Anwendung implementiert. Diese Anwendung trägt den Namen *metaCBR* (*Modellierung Existierender Tools*[1] *und Anwendungen in CBR* oder englisch: *Model of Existing Tools and Applications in CBR*). Die Falldaten sind im CASUEL-Format (Manago et al., 1994) auf dem ftp-Server der AG Expertensysteme öffentlich zugänglich und können so mit jedem Werkzeug, das das CASUEL-Format unterstützt, benutzt werden. In Zukunft soll es auch möglich sein, das System direkt über das WWW zu konsultieren.

Dieses Kapitel beschreibt nun die Implementiertung von *metaCBR*. Es beginnt mit einer Beschreibung des verwendeten Werkzeugs CBR-Works. Anschließend wird die Umsetzung der in Kapitel 4 vorgestellten Modellierung erläutert. Der letzte Abschnitt beschreibt schließlich die Programme, die zur Automatisierung der Fallakquisition erstellt wurden.

5.1 Das System CBR-Works

Das fallbasierte Werkzeug CBR-Works ist der Nachfolger des Systems INRECA. Das aus dem Projekt INRECA hervorgegangene gleichnamige Werkzeug wurde im Rahmen des Projekts WiMo im Hinblick auf eine kommerzielle Nutzung reimplementiert. Das Resultat dieser Reimplementierung trägt den Namen CBR-Works. Es wird einerseits von der Firma tecInno in Kaiserslautern vermarktet und andererseits im Rahmen des Projekts INRECA II weiterentwickelt.

Die Reimplementierung von CBR-Works war bei der Implementierung von *metaCBR* noch nicht abgeschlossen. Die für diese Arbeit zur Verfügung stehende Version von CBR-Works war der erste Evaluationsprototyp (Version 1.0 beta), in dem einige Funktionen noch nicht voll implementiert waren. Daher konnte die Anwendung *metaCBR* bisher nur mit folgenden Einschränkungen erstellt werden:

- Ähnlichkeitsmaße lassen sich nur auf der Ebene einzelner Attribute definieren. Ähnlichkeitsmaße auf Konzeptebene, mit denen auch Gruppen

[1] Dieser Anglizismus ist ein Kompromiß zur Erhaltung des Akronyms.

von Attributen verglichen werden können, sind in der vorliegenden Version nicht möglich.

- Die Belegung von Attributen mit mehreren Werten gleichzeitig (*multi-valued slots*) ist ebenfalls noch nicht möglich. Dies betrifft auch die Definition von Intervallen für geordnete Attribute als Sonderfall von *multi-valued slots*. Ein temporärer Ausweg ist hier die Verteilung der Werte auf mehrere Attribute. Diese lassen sich allerdings nicht miteinander vergleichen, solange keine Ähnlichkeitsmaße für Konzepte definiert werden können.

- Gewichte können derzeit nur lokal für eine Anfrage definiert werden. Sie haben lediglich den Wert von individuellen Präferenzen des jeweiligen Benutzers und sind daher auch kein Bestandteil der Anwendung selbst. Globale, bei der Anwendungsentwicklung einzugebende Gewichte, die als Vorgaben dienen, sind zwar vorgesehen, aber derzeit noch nicht implementiert. Als einzig mögliche Vorgabe werden daher alle Attribute eines Konzepts gleich gewichtet.

Eine weitere Einschränkung, die auch bei der endgültigen Version von CBR-Works bestehen bleibt, betrifft die Benutzerschnittstelle. Diese läßt sich in der Programmiersprache Smalltalk, in der auch CBR-Works selbst implementiert ist, nach belieben Modifizieren und sollte auch nach Möglichkeit an die Bedürfnisse der jeweiligen Domäne angepaßt werden. Solche Modifikationen der Benutzerschnittstelle können aber nicht in der Sprache CASUEL ausgedrückt werden. Da eines der Ziele von *metaCBR* aber die Portierbarkeit (über CASUEL) ist, ist es nicht sinnvoll, Anpassungen an der Benutzerschnittstelle vorzunehmen, die nur auf der Instanz von CBR-Works wirksam sind, auf der sie programmiert wurden. Aus diesem Grund beschränkt sich *metaCBR* auf die Standard-Schnittstelle, die CBR-Works vorgibt.

5.2 Repräsentation der Falldaten

5.2.1 Die Objektstruktur

In *metaCBR* gibt es verschiedene Arten von Fällen. Die Modellierung (Kapitel 4) unterscheidet CBR-Werkzeuge und -Anwendungen, wobei letztere wiederum nach ihrem Aufgabentyp unterteilt sind. Dieser Unterteilung kommt die objektorientierte Fallrepräsentation von CBR-Works sehr entgegen. Die verschiedenen Arten von Fällen können als verschiedene Objektklassen, in CBR-Works *Konzepte* genannt, mit gemeinsamer Oberklasse (Oberkonzept) definiert werden. In diesem Fall gibt es ein Oberkonzept `CBRSystem` mit den Unterkonzepten `CBRTool` und `CBRApplication`. Letzteres hat weitere Unterkonzepte.

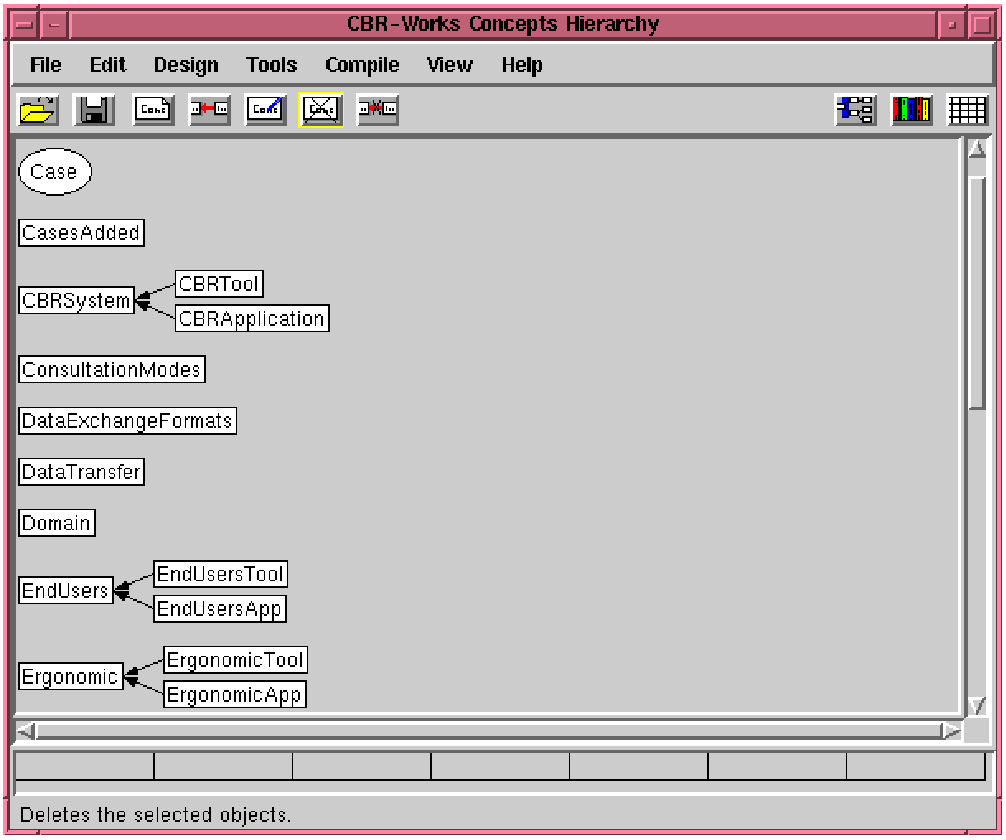

Abb. 5.1: Ausschnitt aus der Konzepthierarchie von *metaCBR*

Desweiteren besteht die Möglichkeit, Objekte mittels der `has-part`-Relation weiter zu strukturieren. Hiervon wurde bei *metaCBR* ebenfalls Gebrauch gemacht. Die Konzepte `CBRTool` und `CBRApplication` sind jeweils nach den Abschnitten der Fragebögen strukturiert. Diese Teilkonzepte machen sich wiederum die Vererbung von Ober- zu Unterkonzepten zunutze. So gibt es beispielsweise für die technischen Kriterien ein Oberkonzept `Technical`, das die gemeinsamen Attribute von Anwendungen und Werkzeugen enthält und dazu die Unterkonzepte `TechnicalApp` und `TechnicalTool` mit den jeweils spezifischen Attributen. Diese Teilkonzepte sind teilweise weiter strukturiert. Ein Ausschnitt aus der so entstehenden Konzepthierarchie ist in Abbildung 5.1 zu erkennen.

Ein Fall in *metaCBR*, d.h. eine Instanz des Konzeptes `Case` besteht nun aus den Attributen `date` und `cbrSystem`. Ersteres enthält das Datum der Erfassung des Falles. Letzteres enthält die eigentliche Beschreibung eines fallbasierten Systems. Dieses Attribut enthält eine Instanz des Konzeptes `CBRSystem`, das die beiden oben erwähnten Subkonzepte hat. Bei der Eingabe dieses Attributes kann man nun auswählen, ob man ein Werkzeug, eine Anwendung oder ein allgemeines fallbasiertes System beschreiben will[2]. Daraufhin ist der größte Teil der Struktur des einzugebenden Falles festgelegt. Die einzige weitere Auswahlmöglichkeit besteht bei der Eingabe der Aufgabenkriterien einer Anwendung. Hier muß abhängig vom Aufgabentyp eine weitere Auswahl zwischen dem Konzept `Task` und seinen Subkonzepten `AnalyticTask` und `SyntheticTask` getroffen werden.

5.2.2 Die Attributtypen

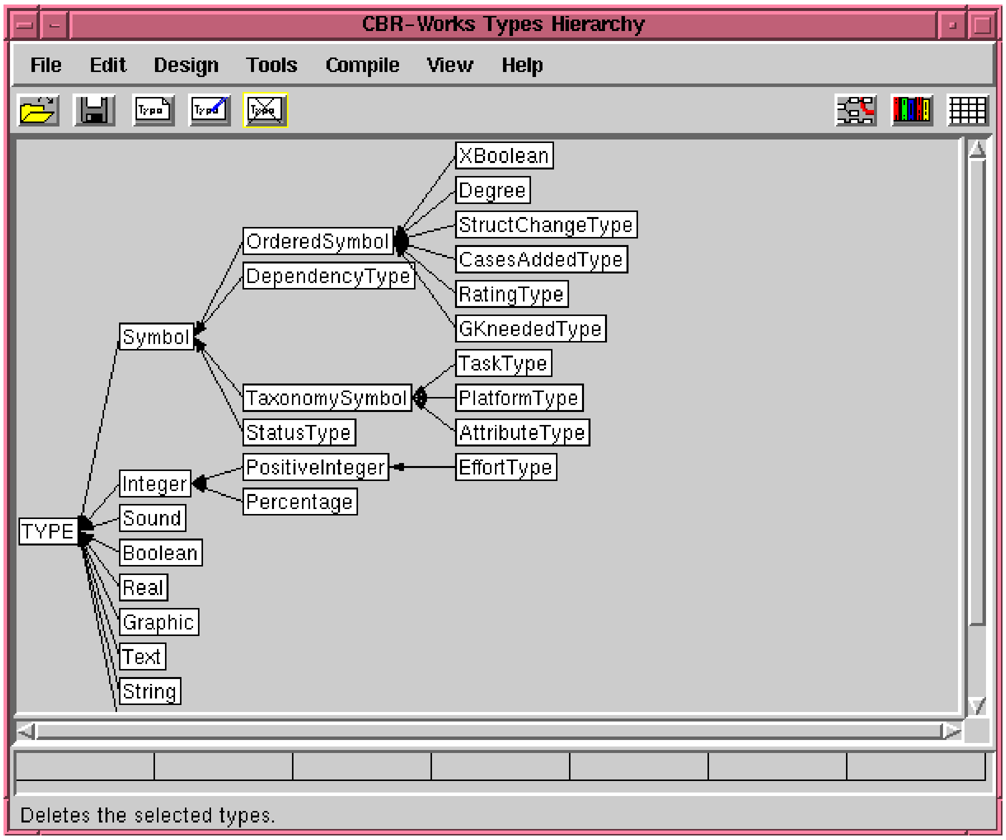

Abb. 5.2: Die Typenhierarchie von *metaCBR*

Da zu den meisten Fragen in der Modellierung die Antwortmöglichkeiten vorgegeben sind, sind die Attribute, mit denen sie repräsentiert werden größtenteils symbolischer Art. Hier wurde für jeden symbolischen Wertebereich ein eigener Typ definiert — je nach Anforderungen an das Ähnlichkeitsmaß als allgemeiner Symboltyp, geordneter Symboltyp oder Taxonomie.

Neben Symboltypen, die speziell auf einzelne Attribute zugeschnitten sind, gibt

[2] Diese Auswahlmöglichkeit ist für die „Wurzel" einer Fallbeschreibung nicht gegeben. Deshalb ist hier das für Anwendungen und Werkzeuge gleichermaßen gültige Konzept `Case` zwischengeschaltet.

es auch solche, die ganze Klassen von Fragen abdecken. So gibt es für Kriterien, die als Ja-Nein-Fragen formuliert sind, einen dreiwertigen Typ namens `XBoolean` („eXtended Boolean"), der neben den Antwortmöglichkeiten „Ja" und „Nein" zusätzlich noch die Angabe „Teilweise" erlaubt. Die gesamte Typenhierarchie zeigt Abblidung 5.2[3].

Den obengenannten Einschränkungen bezüglich mehrwertiger Attribute wurde auf zwei verschiedene Arten begegnet. Bei den taxonomischen Attributen Attributtyp, Aufgabentyp und Betriebssystem werden mehrfache Einträge durch ihren nächsten gemeinsamen Oberknoten in der Taxonomie repräsentiert.

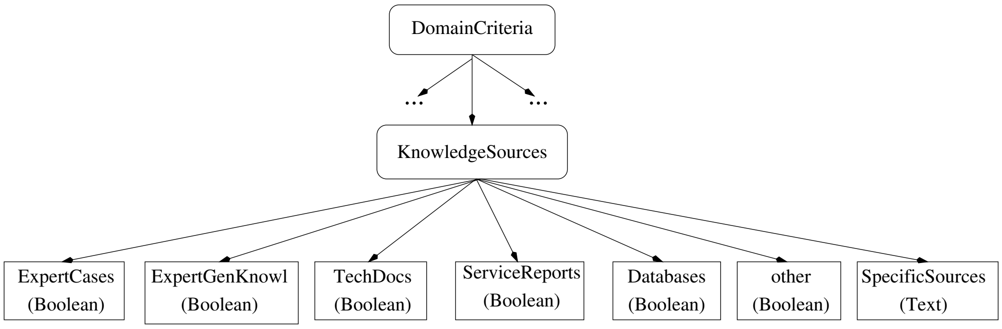

Abb. 5.3: Mehrere Boolesche Attribute als Ersatz für ein mehrwertiges Attribut

Die zweite Möglichkeit, mehrwertige Attribute zu emulieren, besteht darin, für jedes Element des Wertebereiches ein Boolesches Attribut zu definieren. Diese Variante wurde zur Repräsentation aller anderen Kriterien gewählt, bei denen mehrere Antworten gegeben werden können. Diese Booleschen Attribute sind pro Kriterium jeweils zu einem Konzept zusammengefaßt, das mit der `has-part`-Relation in die Gesamtstruktur eingebunden ist. Dieses Konzept enthält meist zusätzlich noch ein Textattribut, in dem alle Informationen, die zu dieser Frage textuell angegeben werden können[4], aufgenommen werden.

5.3 Ähnlichkeitsmaße

Ein Teil der Attribute von *metaCBR* ist für die Ähnlichkeitsbestimmung nicht relevant und daher als nicht diskriminant markiert. Dies betrifft vor allem den kompletten Abschnitt der allgemeinen Angaben (Teilkonzept `GeneralInformation`). Diese Angaben dienen lediglich dazu, einen Fall zu identifizieren und dem Benutzer mitzuteilen, wie er sich über das beschriebene System weiter informieren kann.

[3] Der Bildschirmabzug enthält auch die von CBR-Works bereitgestellten Systemtypen, die allerdings nicht alle in *metaCBR* benutzt werden.

[4] wie z.B. die geneue Spezifikaiton der Angabe „sonstiges".

Desweiteren werden alle Attribute vom Typ `String` oder `Text` nicht zur Ähnlichkeitsbestimmung herangezogen. Diese Felder dienen dazu, Informationen zu repräsentieren, die nicht in das Raster der numerischen und symbolischen Attribute passen. In den meisten Fällen steht in diesen Attributen die genauere Beschreibung dessen, was mit der Angabe „sonstiges“ gemeint ist. Ein rein syntakticher Vergleich dieser textuellen Angaben ist nicht sinnvoll.

Der Aufgabentyp „Entscheidungsunterstützung“, zu dem auch die Aufgabe von *metaCBR* gehört, zeichnet sich gerade dadurch aus, daß das Ziel einer Systemkonsultation nicht global festgelegt ist. Daher ist auch die Relevanz einzelner Attribute von den Präferenzen des Benutzers abhängig. Zu diesem Zweck nutzt *metaCBR* den `Query Editor` von CBR-Works, der es dem Benutzer erlaubt, die vorgegebenen Gewichte nach Belieben zu modifizieren.

Bei der Vorgabe von Gewichten geht CBR-Works davon aus, daß alle relevanten Attribute innerhalb eines Konzepts gleich wichtig sind. So werden auf der obersten Ebene eines Falls, der eine Anwendung repräsentiert, die vier Kriterienbereiche `DomainCriteria`, `TaskCriteria`, `TechnicalCriteria` und `Ergonomic Criteria` jeweils mit 25% gewichtet. Innerhalb dieser Kriterienbereiche werden die Attribute wiederum gleich gewichtet, so daß sich aus der Fallstruktur ein hierarchisches Ähnlichkeitsmaß ergibt. Modifikationen der Gewichte betreffen ebenfalls nur die lokale Gewichtung der Attribute innerhalb eines Konzepts.

5.4 Anwendungsbeispiele

Im folgenden soll anhand von zwei Beispielszenarien erläutert werden, welche Art von Abfragen möglich sind.

Szenario 1

Ein Entwickler möchte für seinen Kundendienst ein Helpdesk-System aufbauen. Er hat bereits einige Service-Berichte in einer Datenbank aufbereitet und möchte diese Daten auch für das CBR-System nutzen. Das System soll auf den in der Firma bereits vorhandenen Windows-PCs laufen.

In der Abfragemaske von *metaCBR* gibt er zunächst ein, daß er nach einer Anwendung sucht, und zwar mit einer analytischen Aufgabenstellung. In den Aufgabenkriterien trägt er dabei vorerst nur den Aufgabentyp „Diagnose“ ein. Bei den Domänenkriterien vermerkt er, daß ihm eine flache Fallrepräsentation genügt und das seine Quellen für Fallwissen die erwähnte Datenbank und weitere Serviceberichte sind. In der Rubrik „technische Kriterien“ markiert er die gewünschte Betriebssystemplattform. Da diese fest vorgegeben ist, erhöht er das vorgegebene Gewicht des Attributes `Platform` (siehe Abbildung 5.4.

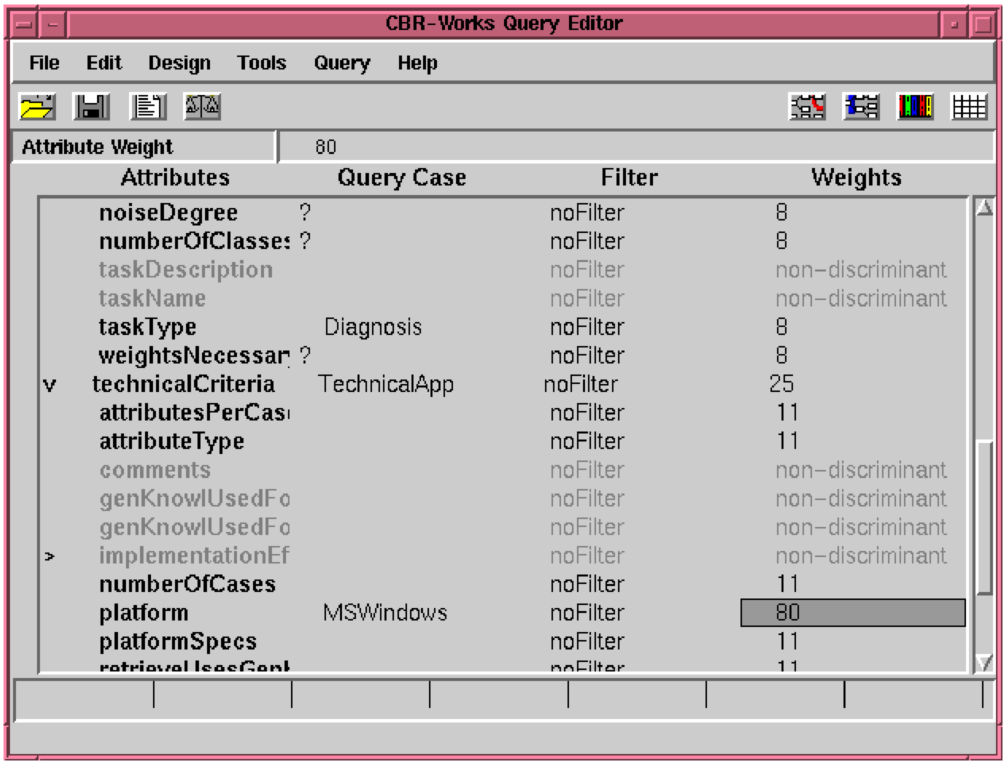

Abb. 5.4: Eine Anfrage in *metaCBR*

Mit diesen Angaben startet er das Retrieval. Da er mit den Fällen, die ihm *metaCBR* präsentiert noch nicht zufrieden ist, versuchter seine Anfrage weiter zu präzisieren. Er vermerkt nun zusätzlich bei den ergonomischen Kriterien, daß ihm eine Importschnittstelle für Datenbanken wichtig ist. Außerdem trägt er die ungefähre Anzahl der Endbenutzer ein. Die Systeme, die ihm *metaCBR* nach dieser erweiterten Anfgage als ähnlichste Fälle liefert, decken sich nun ausreichend mit den Plänen für sein System. Er konsultiert nun die zu diesen Fällen gespeicherten allgemeinen Angaben und kann sich nun anhand der angegebenen Referenzen gezielt weitere Informationen über diese Systeme beschaffen.

Szenario 2

Ein anderer Entwickler möchte gezielt nach einem Werkzeug suchen, mit dem er sein Entscheidungsunterstützungssystem erstellen will. In der Eingabemaske für Werkzeuge gibt er also zunächst den Aufgabentyp ein. Bei den technischen Kriterien vermerkt er, daß er sowohl für den *Retrieve*-Schritt als auch für den *Reuse*-Schritt generelles Wissen verwenden will. Nach einigen weiteren Angaben zu gewünschten Exportschnittstellen und zur Benutzerschnittstelle startet er seine Anfrage und erhält schließlich die Beschreibung von möglicherweise geeigneten Werkzeugen. Anhand der textuellen Zusatzinformationen entscheidet er sich, sich gezielt über ein bestimmtes Werkzeug im Detail zu informieren.

5.5 Fallakquisition

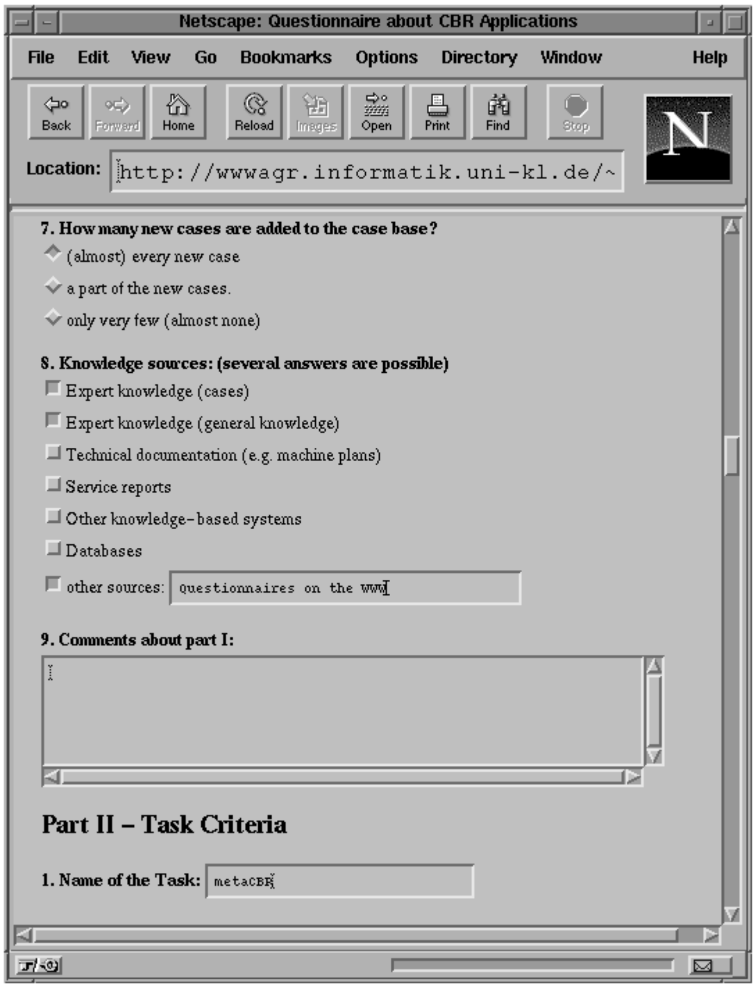

Abb. 5.5: Der Fragebogen im WWW

Die Fragebögen wurden bereits für die erste Datenerhebung als HTML-Formulare im WWW veröffentlicht. Das CGI-Programm[5], das die Formulardaten auswertete, schrieb die eingegebenen Daten einfach in eine Textdatei. Für die erste

[5] Common Gateway Interface

Auswertung reichte dies aus, da die Auswertung von Hand vorgenommen wurde und die endgültige Implementierung der Fallstruktur noch nicht feststand.

Nach der Festlegung der Fallstruktur wurde das CGI-Programm durch ein neues ersetzt. Es ist nun ein in der Programmiersprache Perl5 geschriebenes Programm, das die vom Benutzer über das WWW eingegebenen Daten direkt im CASUEL-Format ablegt. Dieses Programm wählt automatisch in Abhängigkeit von den eingegebenen Daten die richtigen Subkonzepte aus und wandelt die mehrwertigen Attribute in die richtige Form um. Die so erhaltenen CASUEL-Dateien können von CBR-Works direkt eingelesen werden. Vor der Aufnahme in die Fallbasis empfiehlt es sich allerdings, die Daten zu sichten und offensichtlich unsinnige Eingaben zu entfernen.[6] Eine erneute Konvertierung der Daten ist jedoch nicht mehr nötig.

[6] Bei Formularen im WWW ist immer damit zu rechnen, daß Netz-Surfer mit den Formularen „spielen“ und dabei völlig beliebige Daten eingeben.

6. ZUSAMMENFASSUNG UND AUSBLICK

6.1 Ergebnisse

Im Rahmen dieser Arbeit wurde eine fallbasierte Anwendung entwickelt, in der Daten von fallbasierten Systemen verwaltet werden. Auf der Grundlage von Arbeiten zur Beschreibung von fallbasierten Systemen wurden getrennte Fallstrukturen für Anwendungen und Werkzeuge modelliert. Falldaten wurden mit einer Umfrage erhoben und neben der Verwertung als Fälle auch statistisch ausgewertet. Das System wurde mit dem Werkzeug CBR-Works implementiert und ist als CASUEL-Fallbasis öffentlich zugänglich.

6.2 Bewertung

Eine Bewertung der Implementierung ist im Rahmen dieser Arbeit noch nicht möglich. Hierzu ist erst ein längerer Praxistest nötig, der Aufschluß über die Qualität des Systems geben kann.

Eine Unzulänglichkeit besteht allerdings darin, daß der Datenbestand der Fallbasis nicht repräsentativ ist. Dies betrifft einerseits die geographische Verteilung der eingegangenen Fragebögen mit einem überproportionalen Schwerpunkt auf Deutschland, vor allem aber die hohe Quote von Forschungssystemen zu lasten von Systemen aus der täglichen Praxis. Damit verbunden ist der hohe Anteil von synthetischen, insbesondere von Entwurfssystemen. Diese Verteilung war allerdings kaum zu beeinflussen.

6.3 Vergleich mit anderen Ansätzen

Einige Ansätze aus anderen Arbeiten wurden bereits kurz erwähnt. In diesem Abschnitt sollen sie nochmals vergleichend zusammengefaßt werden.

6.3.1 Andere Datenerhebungen

In den vergangenen Jahren wurden bereits mehrere Umfragen zu fallbasierten Systemen durchgeführt. Deren Ergebnisse sollen hier kurz vorgestellt werden.

Die Umfrage von Weß (1992) bot erstmals eine Übersicht von CBR-Projekten in Deutschland. Bei den 25 genannten Systemen handelte es sich fast ausschließlich um Forschungsprojekte. Lediglich 2 Systeme befanden sich im praktischen Einsatz. Fallbasierte Werkzeuge wurden hier noch überhaupt nicht benutzt, obwohl die ersten Werkzeuge bereits auf dem Markt waren.

In dieser Umfrage wurden lediglich textuelle Beschreibungen der Projekte bzw. Systeme erfaßt. Das einzige Kriterium, nach dem die Systeme klassifiziert wurden, war der Status des Projekts.

Eine Aktualisierung der Ergebnisse von 1992 bietet die Umfrage von Woltering und Weß (1996), die sich diesmal auf den gesamten deutschsprachigen Raum ausdehnt. Von den mittlerweile 33 erfaßten Systemen sind mittlerweile deutlich mehr im praktischen Einsatz als noch 1992.

Zum ersten Mal wurden hier auch verstärkt statistische Angaben erhoben. Im wesentlichen waren dies Informationen zum Einsatzbereich der Systeme sowie implementierungstechnische Aspekte. Im Vergleich mit der in dieser Arbeit durchgeführten Umfrage enthält diese Untersuchung deutlich mehr analytische als synthetische Systeme. Auch der Einsatz von CBR-Werkzeugen ist nach Angaben von Woltering und Weß (1996) häufiger. Da diese Umfrage nur kurze Zeit vor der vorliegenden Arbeit durchgeführt wurde, lassen sich die Unterschiede in den statistischen Ergebnissen nur dadurch erklären, daß beide Untersuchungen lediglich einen Teil der existierenden Systeme erfaßt haben. Dies gilt auch, wenn man von den in *metaCBR* erfaßten Systemen nur diejenigen aus dem deutschsprachigen Raum berücksichtigt.

Einen allgemeineren Untersuchungsgegenstand hat die Datenerhebung von Mertens et al. (1993). Hier wurden über mehrere Jahre Daten von über 3000 Expertensystemen gesammelt. Das Ziel dieser Sammlung ist — ähnlich wie bei *metaCBR* — einen Überblick über den Stand der Technik zu bieten und Entwicklern neuer Systeme Informationen über bereits existierende Systeme zur Verfügung zu stellen. Die einzigen Eigenschaften, nach denen diese Sammlung strukturiert ist, sind jedoch lediglich die Klassifikation der Aufgabe, der Projektstatus, Angaben zu Nutzeffekten sowie das Herkunftsland eines Systems. Ein Durchsuchen der Datenbank nach detaillierteren Kriterien ist im Gegensatz zu *metaCBR* nicht möglich. Außerdem handelt es sich hier bei den Abfragen um Datenbankabfragen, während *metaCBR* eine Ähnlchkeitssuche durchführt. Es bleibt zu hoffen, daß die Datensammlung von *metaCBR* in Zukunft einen ähnlichen Umfang erreicht wie die Expertensystem-Datenbank.

6.3.2 Die Experience Factory

In Kapitel 2.3 wurde bereits das fallbasierte Schließen, insbesondere das Szenario von *metaCBR* mit dem *Experience Factory*-Ansatz aus dem Software Engineering verglichen. An dieser Stelle sollen nun noch einige Unterschiede zwischen beiden Ansätzen aufgezeigt werden.

Zunächst ist anzuführen, daß bei der *Experience Factory* nicht primär an eine Implementierung gedacht ist. Es geht vielmehr darum, einen *Closed-Loop* in die betrieblichen Abläufe zu integrieren und hierfür die strukturellen Voraussetzungen zu schaffen. Das Ziel von *metaCBR* war jedoch von Anfang an die Implementierung. Daher mußten alle zu repräsentierenden Daten in ein formales Raster gebracht werden, während in einer *Experience Factroy* auch unstrukturierte Daten verwendet werden können.

Die in der *Experience Factory* verwalteten Daten bechreiben nicht nur Softwaresysteme selbst, sondern auch deren Entwicklungsprozesse über die gesamte Produktlebensdauer, einschließlich der diesbezüglichen organisatorischen Abläufe. Diese Angaben lassen sich nur im Kontext der jeweiligen Firma verwenden[1] und kommen somit für ein firmenübergreifendes System wie *metaCBR* nicht in Betracht.

6.4 Ausblick

Die Implementierung von *metaCBR* markiert erst den Anfang der Nutzungsphase des Systems. Im laufe der Praktischen Nutzung werden sich sicherlich durch Lerneffekte gewisse Feinkorrekturen am Ähnlichkeitsmaß ergeben, eventuell auch an der Modellierung der Fälle. Über die weiterhin im WWW verfügbaren Fragebögen ist die Akquisition weiterer Fälle möglich und auch durchaus wünschenswert.

Im Laufe der Weiterentwicklung von CBR-Works werden sich auch Erweituerungsmöglichkeiten für *metaCBR* ergeben. Sobald CBR-Works die volle Funktionalität von INRECA hat, empfiehlt es sich, auch *metaCBR* daran anzupassen, da hierdurch die Mächtigkeit des Ähnlichkeitsmaßes deutlich erweitert werden kann. Einen weiteren Fortschritt verspricht die geplante WWW-Schnittstelle von CBR-Works. Eine Anpassung von *metaCBR* an diese Schnittstelle würde die Akzeptanz in der Öffentlichkeit deutlich verbessern, da eine Konsultation über das WWW mit deutlich weniger Aufwand verbunden ist, als die CASUEL-Daten über das Internet zu beziehen und lokal zu installieren.

Neben der Implementierung birgt auch die Verbindung von CBR und Software Engineering Hoffnung auf weitere interessante Ergebnisse. Einen Rahmen hierfür bietet das Projekt INRECA II. Ein Schwerpunkt dieses Projektes beschäftigt

[1] Abgesehen von der Verwendbarkeit dieser Daten stellt sich die Frage, ob Firmen solche teilweise sehr sensiblen Daten überhaupt preisgeben würden.

sich mit Methoden zur Entwicklung fallbasierter Systeme, wozu die in dieser Arbeit durchgeführte Datensammlung bereits einen Beitrag geleistet hat.

6.5 Danksagung

Für das Gelingen dieser Arbeit bin ich Brigitte Bartsch-Spörl zu besonderem Dank verpflichtet. Sie war maßgeblich an der Entwicklung des ersten Fragebogens beteiligt und hat auch seine Verbreitung mit großem Einsatz vorangetrieben. Sie hat den Fragebogen nicht nur auf elektronischem Wege uber den *CBR newsletter* verbreitet, sondern vor allem auch durch persönliche Kontaktaufnahmen dafür gesorgt, daß für diese Arbeit genügend Fälle zur Verfügung standen. Desweiteren war sie an der statistischen Auswertung der Falldaten beteiligt (siehe Bartsch-Spörl et al., 1996) und hat einige wertvolle Kommentare zur Entwicklung des zweiten Fragebogens beigesteuert.

ANHANG

A. DIE FRAGEBÖGEN

A.1 Der erste Fragebogen

PART I: The Cases

1. **Approximately the number of different cases in your application:**
   - ∘ less than 200
   - ∘ between 200 and 2.000
   - ∘ between 2.000 and 20.000
   - ∘ more than 20.000
2. **On average the number of attributes per case:**
   - ∘ less than 10
   - ∘ between 10 and 50
   - ∘ between 50 and 200
   - ∘ more than 200
3. **Kind of attributes being relevant for similarity assessment:**
   - ∘ mainly symbolic
   - ∘ mainly numeric
   - ∘ mainly structured
4. **Structure of the cases:**
   - ∘ flat list of values
   - ∘ more complex structure
5. **Context dependency of the cases:**
   - ∘ cases can be reused independent from their respective context
   - ∘ cases can be only reused depending on their respective context
6. **General knowledge:**
   - ∘ exists and needs to be used
   - ∘ is not relevant

PART II: The Task

1. **Analytic Tasks**

1.1 Is case-specific knowledge the main knowledge source?

- yes
- no
- only in combination with others

1.2 Is the basic characteristic of case interpretation and the thereby offered flexibility of CBR important for your task?

- yes
- no
- only in combination with others

1.3 Is fuzzy matching important for your task?

- yes
- no
- only in combination with others

1.4 To which degree is the use of weights necessary for similarity assessment?

- none
- small degree
- high degree

1.5 To which degree is the automatic learning of weights necessary for similarity assessment?

- none
- small degree
- high degree

1.6 To which degree is case adaptation necessary?

- none
- small degree
- high degree

Sub-Classification:

1.7 CLASSIFICATION:
Does the task focus on pre-classified case-data?

If yes:

1.7.1 On average a case-base contains

- 2 classes
- up to 20 classes
- between 20 and 100 classes
- between 100 and 500 classes
- more than 500 classes

1.7.2 On average the degree of noise contained in a case is

- up to 10%
- between 10 and 20%
- between 20 and 30%
- between 30 and 40%
- more than 40%

1.7.3 On average a class can be described by

- 1 case
- between 1 and 3 cases
- between 3 and 7 cases
- more than 7 cases

1.8 DIAGNOSIS: Is a test selection strategy necessary to acquire missing information?

If yes:

1.8.1 On average a case-base contains

- up to 20 diagnoses
- between 20 and 100 diagnoses
- between 100 and 500 diagnoses
- more than 500 diagnoses

1.8.2 On average the degree of noise contained in a case is

- up to 10%
- between 10 and 20%
- between 20 and 30%
- between 30 and 40%
- more than 40%

1.8.3 On average a diagnosis can be described by

- ∘ 1 case
- ∘ between 1 and 3 cases
- ∘ between 3 and 7 cases
- ∘ more than 7 cases

1.8.4 On average the degree of incomplete information contained in a case is

- ∘ up to 10%
- ∘ between 10 and 20%
- ∘ between 20 and 40%
- ∘ between 40 and 60%
- ∘ more than 60%

1.8.5 On average the number of attributes to be additionally ascertained is

- ∘ up to 5
- ∘ between 5 and 10
- ∘ more than 10

1.9 DECISION SUPPORT:
Is the combination of case attributes, which are considered as the case solution, built dynamically?

If yes:

1.9.1 How many different combinations of case attributes can be of value as potential case solutions?

- ∘ up to 5
- ∘ between 5 and 10
- ∘ between 10 and 20
- ∘ more than 20

1.9.2 Can the organisation of the case-base and/or the similarity assessment process be improved through a further structuring/ clustering of the domain?

- ∘ no
- ∘ to a small degree
- ∘ to a high degree

1.9.3 On average the degree of incomplete information contained in a case is

- ∘ up to 10%
- ∘ between 10 and 20%
- ∘ between 20 and 40%
- ∘ between 40 and 60%
- ∘ more than 60%

1.9.4 On average the number of attributes to be additionally ascertained is

- ∘ up to 5
- ∘ between 5 and 10
- ∘ more than 10

2. Synthetic Tasks

2.1 Is the overall task decomposable into sub-tasks and can the overall solution be composed from partial solutions?

- ∘ yes
- ∘ no
- ∘ partially

2.2 Have solutions from similar problems to be adapted to the problem at hand?

- ∘ yes
- ∘ no
- ∘ only to a small degree

Sub-Classification:

2.3 PLANNING:
Does the task focus on the temporal structure of the sub-tasks that need to be processed?

If yes:

2.3.1 On average a solution contains

- ∘ up to 10 sub-tasks
- ∘ between 10 and 30 sub-tasks
- ∘ between 30 and 50 sub-tasks
- ∘ more than 50 sub-tasks

2.3.2 On average a sub-task can be described using

- ○ up to 3 parameters
- ○ between 3 and 7 parameters
- ○ more than 7 parameters

2.4 CONFIGURATION: Does the task focus on the configuration of technical devices?

If yes:

2.4.1 On average a solution contains

- ○ up to 10 different components
- ○ between 10 and 30 different components
- ○ between 30 and 50 different components
- ○ more than 50 different components

2.4.2 On average a component can be characterised using

- ○ up to 3 parameters
- ○ between 3 and 7 parameters
- ○ more than 7 parameters

2.5 DESIGN: Does the task focus on the design of individually created artefacts?

If yes:

2.5.1 On average a solutions contains

- ○ up to 10 different design elements
- ○ between 10 and 30 different design elements
- ○ between 30 and 50 different design elements
- ○ more than 50 different design elements

2.5.2 On average a design element can be characterised using

- ○ up to 3 parameters
- ○ between 3 and 7 parameters
- ○ more than 7 parameters

3. Other tasks (if your task does not fit into the given scheme)

PART III Additional Information

System name: ....

Application domain (may be several): ....

....

Short overview on the system (up to 20 lines): ....

....

....

....

....

Is the system

- ○ a success?
- ○ a failure?
- ○ we will see ...

Name and address of a contact person: ....

....

....

....

....

Up to 3 references to detailed descriptions in the literature:

- ....
  ....
- ....
  ....
- ....
  ....

Status of the system:

- ○ research prototype/feasibility study
- ○ real-life application is being developed
- ○ pilot system of a real life application in practical use
- ○ real-life application in practical use since: ....
- ○ real-life application in practical use from .... to ....

Time and effort:

- development time (in months): ....

- development effort (in person-months): ....
- number of involved end-users: ....

Used software technology and tools:

- used tools: ....
  ....
- internally developed tool based on the following programming languages: ....
  ....
- runs on the following platforms: ....
  ....

Lessons learned:

- concerning application development: ....
  ....
  ....
- concerning the used software-development tool: ....
  ....
  ....
- concerning the operational environment: ....
  ....
  ....
- concerning other topics: ....
  ....
  ....

Reusable parts:
Can other CBR system developers benefit from your work?

☐ no

☐ yes, by personal contact

☐ yes, by ordering descriptions of the system from the given contact address

☐ yes, by passing-on the case-base(s)

☐ yes, by passing-on software-components

A.2 Der zweite Fragebogen für Anwendungen

About this questionnaire: There are two types of questions in this questionnaire: open and closed questions. You can answer the open questions by filling text or numbers into the places marked with " .. ". In the closed questions, you can check one or more suggested answers. If the answers are marked with a small circle (○), please check *only one* answer. If they are marked with a box (□), you can check an arbitrary number of them.

At the end of each part of the questionnaire you will be asked for comments about that part. At these places you can tell us your opinion about the questions, describe problems you had while answering the questions make suggestions, etc. You can also use these fields to give us more information about your system which is not covered by the questionnaire. This information will help us improve the questionnaire for the future.

General Information

1. **Name of the system** : ....
   (including acronym expansion and version number)
2. **Developed by** (maybe several people/organisations):
   - ....
     ....
     ....
     ....
3. **Contact Address:**

   Name: ....
   Address: ....
   ....
   ....
   ....
   ....
   Phone: ....
   Fax: ....
   Email: ....
   WWW home page: ....
4. **References in literature (up to 3 entries)(*)**
   - ....
     ....

- ....
  ....
- ....
  ....

5. References on the WWW

- ....
  ....

6. Status of the system

- ○ Research prototype
- ○ Application for practical use being developed
  Date when probably finished: ....
- ○ Pilot system
- ○ Application in practical use since: ....
- ○ Application which was in practical use from: .... to: ....

7. Rating of the system: (*)

- ○ Success
- ○ Failure
- ○ Both positive and negative aspects
- ○ Not evaluated yet

8. Lessons learned: (*)

- Concerning the development process:
  ....
  ....
  ....
- Concerning the used software development tools:
  ....
  ....
  ....
- Concerning the organisational environment:
  ....
  ....
  ....
- Concerning other topics:
  ....
  ....
  ....

9. **Can other developers of CBR systems take benefit from your work? (*)**
   - ☐ By personal contact
   - ☐ By ordering a description of the system from the given contact address
   - ☐ By passing on case data
   - ☐ By passing on software components

10. **Comments about the "general information" part:**
    - ....
      ....
      ....
      ....

Part I - Domain Criteria

1. **Name of the domain:** ....
2. **Short description of the domain**
   - ....
     ....
     ....
     ....
     ....
3. **Can the domain be represented as a flat list of values?**
   - ○ yes
   - ○ no
4. **Environmental Dependency of the domain: (*)**
   - ○ Cases can only be reused depending on their respective context.
   - ○ All relevant context data can be coded into the case data.
5. **General knowledge**
   - ○ has to be used
   - ○ is sometimes useful
   - ○ is not relevant
6. **How often does the case structure change?**
   - ○ regularly
   - ○ from time to time
   - ○ only in case of previous modeling errors

7. **How many new cases are added to the case base?**
   - ○ (almost) every new case
   - ○ a part of the new cases.
   - ○ only very few (almost none)
8. **Knowledge sources:**
   - ☐ Expert knowledge (cases)
   - ☐ Expert knowledge (general knowledge)
   - ☐ Technical documentation (e.g. machine plans)
   - ☐ Service reports
   - ☐ Other knowledge-based systems
   - ☐ Databases
   - ☐ other sources: ....
9. **Comments about part I:**
   - ....
     ....
     ....
     ....
     ....

Part II - Task Criteria

1. **Name of the Task** ....
2. **Description of the Task**
   - ....
     ....
     ....
     ....
3. **Type of Task:**
   - **Analytic Tasks:**
     - ○ **Classification**
     - ○ **Diagnosis:** Parts of the knowledge for the query have to be acquired during the consultation process.
     - ○ **Decision Support:** The answer to a query is not concentrated in a single 'target attribute'.
     - ○ **other** analytic task: ....

- **Synthetic Tasks:**
  - ○ **Planning:** The main focus of the task is the temporal structure of subtasks.
  - ○ **Configuration** of technical devices
  - ○ **Design** of individual artefacts
  - ○ **other** synthetic task: ....
- ○ **other** task: ....

4. Is it necessary to use CBR for this task?

- ○ yes
- ○ partially
- ○ no

5. Is fuzzy matching important for the task? (*)

- ○ yes
- ○ partially
- ○ no

6. To which degree are weights necessary for similarity assessment? (*)

- ○ none
- ○ small degree
- ○ high degree

7. To which degree is the automatic learning of weights necessary for similarity assessment? (*)

- ○ none
- ○ small degree
- ○ high degree

8. To which degree is adaptation necessary? (*)

- ○ none
- ○ small degree
- ○ high degree

Please continue with the section that matches your type of task:

a) analytic (classification, diagnosis, decision support)

b) synthetic (planning, configuration, design)

a) Analytic Tasks:

a1. Average degree of noise contained in a case: (*)

- up to 10%
- between 10% and 20%
- between 20% and 30%
- between 30% and 40%
- more than 40%

a2. Average degree of incomplete information contained in a case: (*)

- up to 10%
- between 10% and 20%
- between 20% and 40%
- between 40% and 60%
- more than 60%

Please continue with the section that matches your type of task:

a.i) Classification

a.ii) Diagnosis

a.iii) Decision support

a.i) Classification Tasks:

a.i.1. Number of classes contained in a case base: .....

a.i.2. A class can be described by ..... **cases**

Please proceed with part III (p. 68).

a.ii) Diagnosis Tasks:

a.ii.1. Number of Diagnoses contained in a case base: .....

a.ii.2. A diagnosis can be described by ..... **cases.**

a.ii.3. Number of attributes to be tested during the consultation process: .....

Please proceed with part III (p. 68).

a.iii) Decision support tasks:

a.iii.1. How many different combinations of attributes can be considered as possible solutions? .....

a.iii.2. Number of attributes to be tested during the consultation process: .....

a.iii.3. Can the organisation of the case base and/or the similarity assessment process be improved by a further structuring or clustering of the domain? (*)

- ○ no
- ○ to a small degree
- ○ to a high degree

Please proceed with part III (p. 68).

b) Synthetic Tasks:

b.1. Can the task be decomposed into subtasks and can the solution be built from partial solutions? (*)

- ○ yes
- ○ partially
- ○ no

Please continue with the section that matches your type of task:

- Planning
- Configuration
- Design

b.i) Planning tasks:

b.i.1. A solution consists of ..... subtasks

b.i.2. A subtask can be described by ..... parameters

Please proceed with part III (p. 68).

b.ii) Configuration tasks:

b.ii.1. A solution consists of ..... components

b.ii.2. A component can be described by ..... parameters

Please proceed with part III (p. 68).

b.iii) Design tasks:

b.iii.1. A solution consists of ..... design elements

b.iii.2. A design element can be described by ..... parameters

9. Comments about part II:

- ....
....
....
....
....

Part III - Technical Criteria

1. Number of cases in the application: ....

2. Number of discriminant attributes per case: .....

3. The attributes which are relevant for the similarity assessment are mostly: (*)

- ○ symbolic
- ○ numeric
- ○ both symbolic and numeric
- ○ structured

4. Is general knowledge used in the "retrieve" step?

- ○ yes
- ○ no

5. What kind of general knowledge is used in the retrieve step?

- ....

6. **Is general knowledge used in the "reuse" step?**

   ◦ yes

   ◦ no

7. **What kind of general knowledge is used in the reuse step?**

   • ....

8. **Special technical features you would like to mention:**

   • ....
   ....

9. **Which tools were used to develop this application?**

   • ....
   ....

10. **Which programming languages were used to develop this application?**

    • ....
    ....

11. **On which platforms does the application run?**

    ☐ plain MS-DOS

    ☐ MS Windows ☐ 3.x ☐ 95 ☐ NT

    ☐ Unix ☐ all Unixes ☐ special Unix(es): ....

    ☐ Macintosh

    ☐ Mainframe

    ☐ other platforms: ....

12. **Implementation effort: (*)**

    • development time: .... months

    • development effort: .... person-months

13. **Comments about part III:**

    • ....
    ....
    ....
    ....
    ....

Part IV - Ergonomic Criteria

1. **Type of end-users** (apart from the system administrator):
   - ☐ experts
   - ☐ other company personnel
   - ☐ customers
   - ☐ other: ....
2. **Number of involved end-users: (*)** .....
3. **What consultation modes does the application offer?**
   - ☐ system-driven (the order of interactions is completely controlled by the system)
   - ☐ user-driven (the order of interactions is up to the user)
   - ☐ combination of both modes
4. **New cases are added to the case base**
   - ○ immediately
   - ○ after a separate validation process
5. **Which data can be exported to other applications?**
   - ☐ case data
   - ☐ case structure
   - ☐ similarity measures
   - ☐ general knowledge
6. **Which data can be imported from other applications?**
   - ☐ case data
   - ☐ case structure
   - ☐ similarity measures
   - ☐ general knowledge
7. **Which data formats can be used for communication with other applications?**
   - ☐ ASCII
   - ☐ Spreadsheet formats: ....
   - ☐ Database formats: ....
   - ☐ Case representation languages: ....
   - ☐ other: ....
8. **Comments about part IV:**

- ....
  ....
  ....
  ....
  ....

A.3 Der Fragebogen für Werkzeuge

About this questionnaire: There are two types of questions in this questionnaire: open and closed questions. You can answer the open questions by filling text or numbers into the places marked with " .. ". In the closed questions, you can check one or more suggested answers. If the answers are marked with a small circle (○), please check *only one* answer. If they are marked with a box (□), you can check an arbitrary number of them.

At the end of each part of the questionnaire you will be asked for comments about that part. At these places you can tell us your opinion about the questions, describe problems you had while answering the questions, make suggestions, etc. You can also use these fields to give us more information about your system which is not covered by the questionnaire. This information will help us improve the questionnaire for the future.

General Information

1. **Name of the system** : ....
   (including acronym expansion and version number)
2. **Developed by** (maybe several people/organisations):
   - ....
     ....
     ....
     ....
3. **Contact Address:**

   Name: ....
   Address: ....
   ....
   ....
   ....
   ....
   Phone: ....
   Fax: ....
   Email: ....
   WWW home page: ....
4. **References in literature (up to 3 entries)**
   - ....
     ....

- ....
  ....
- ....
  ....

5. References on the WWW

- ....
  ....

6. Status of the system

- ○ Research prototype
- ○ System for practical use being developed
  Date when probably finished: ....
- ○ Pilot system
- ○ In practical use since: ....
- ○ System which was in practical use from: .... to: ....

7. Rating of the system:

- ○ Success
- ○ Failure
- ○ Both positive and negative aspects
- ○ Not evaluated yet

8. Lessons learned:

- Concerning the development process:
  ....
  ....
  ....
- Concerning the used software development tools:
  ....
  ....
  ....
- Concerning the organisational environment:
  ....
  ....
  ....
- Concerning other topics:
  ....
  ....
  ....

9. **Can other developers of CBR systems take benefit from your work?**

   ☐ By personal contact

   ☐ By ordering a description of the system from the given contact address

   ☐ By passing on case data

   ☐ By passing on software components

10. **Comments about the "general information" part:**

   - ....
     ....
     ....
     ....

Part I - Task Criteria

1. **Which types of tasks can the system handle?**

   - **Analytic Tasks:**
     - ☐ **Classification**
     - ☐ **Diagnosis:** Parts of the knowledge for the query have to be acquired during the consultation process.
     - ☐ **Decision Support:** The answer to a query is not concentrated in a single 'target attribute'.
     - ☐ **other** analytic tasks: ....
   - **Synthetic Tasks:**
     - ☐ **Planning:** The main focus of the task is the temporal structure of subtasks.
     - ☐ **Configuration** of technical devices
     - ☐ **Design** of individual artefacts
     - ☐ **other** synthetic tasks: ....

   ☐ **other** tasks: ....

2. **Is there a specific task or scenario for which the tool was developed?**

   - ....

3. **Comments about part I:**

   - ....
     ....
     ....
     ....
     ....

Part II - Technical Criteria

1. **Is it possible to represent cases as complex structures?**
   - ◦ yes
   - ◦ no, only as flat tables
   - ◦ to a certain degree
2. **Is it possible to use structured attributes for similarity assessment?**
   - ◦ yes
   - ◦ no
3. **Is it possible to use general knowledge in the "retrieve" step?**
   - ◦ yes
   - ◦ no
4. **What kind of general knowledge can be used in the retrieve step?**
   - ....
5. **Is it possible to use general knowledge in the "reuse" step?**
   - ◦ yes
   - ◦ no
6. **What kind of general knowledge can be used in the reuse step?**
   - ....
7. **Special technical features you would like to mention:**
   - ....
     ....
8. **Which programming languages were used to develop this tool?**
   - ....
     ....
9. **On which platforms does the system run?**

   ☐ plain MS-DOS
   ☐ MS Windows ☐ 3.x ☐ 95 ☐ NT
   ☐ Unix ☐ all Unixes ☐ special Unix(es): ....
   ☐ Macintosh
   ☐ Mainframe
   ☐ other platforms: ....

10. **Implementation effort:** (Without effort for implementing specific applications)
    - development time: .... months
    - development effort: .... person-months
11. **Comments about part II:**
    - ....
      ....
      ....
      ....
      ....

Part III - Ergonomic Criteria

1. **Type of end-users** (apart from the system administrator):
   - ☐ experts
   - ☐ other company personnel
   - ☐ customers
   - ☐ other: ....
2. **What consultation modes does the application offer?**
   - ○ system-driven (the order of interactions is completely controlled by the system)
   - ○ user-driven (the order of interactions is up to the user)
   - ○ combination of both modes
3. **New cases can be added to the case base**
   - ☐ immediately
   - ☐ after a separate validation process
4. **Which data can be exported to other applications?**
   - ☐ case data
   - ☐ case structure
   - ☐ similarity measures
   - ☐ general knowledge
5. **Which data can be imported from other applications?**
   - ☐ case data
   - ☐ case structure
   - ☐ similarity measures
   - ☐ general knowledge

6. **Which data formats can be used for communication with other applications?**

   - ☐ ASCII
   - ☐ Spreadsheet formats: ....
   - ☐ Database formats: ....
   - ☐ Case representation languages: ....
   - ☐ other: ....

7. **Comments about part III:**

   - ....
     ....
     ....
     ....
     ....

LITERATURVERZEICHNIS

Agnar Aamodt und Enric Plaza (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59.

Klaus-Dieter Althoff (1996). *Evaluating Case-Based Reasoning Systems: The INRECA Case Study*. Habilitationsschrift, Universität Kaiserslautern.

Klaus-Dieter Althoff und Agnar Aamodt (1996). Zur Analyse fallbasierter Problemlöse- und Lernmethoden in Abhängigkeit von Charakteristika gegebener Aufgabenstellungen und Anwendungsdomänen. *Künstliche Intelligenz*, 10(1):10–15.

Klaus-Dieter Althoff und Brigitte Bartsch-Spörl (1996). Decision Support for case-based applications. *Wirtschaftsinformatik*, 38(1):8–16. Sonderheft über fallbasierte Entscheidungsunterstützung.

Klaus-Dieter Althoff, Eric Auriol, Harald Holz, Michel Manago, Alexandre Meissonnier, Carsten Priebisch, Stefan Wess und Wolfgang Wilke (1994). *An Evaluation of Commercial Case-Based Reasoning Tools*. INRECA Deliverable D5, ESPRIT Project 6322.

Klaus-Dieter Althoff, Eric Auriol, Ralph Barletta und Michel Manago (1995a). *A Review of Industrial Case-Based Reasoning Tools*. AI Perspectives. AI Intelligence, Oxford.

Klaus-Dieter Althoff, Stefan Wess, Karl-Heinz Weis, jun, Eric Auriol, Ralph Bergmann, Harald Holz, Roy Johnston, Michel Manago, Alexandre Meissonnier, Carsten Priebisch, Ralph Traphöner und Wolfgang Wilke (1995b). *An Evaluation of the Final Integrated System*. INRECA Deliverable D6, ESPRIT Project 6322.

Brigitte Bartsch-Spörl, Klaus-Dieter Althoff und Alexandre Meissonnier (1996). Learning From and Reasoning About Case-based Reasoning Systems. Eingereicht als Beitrag zur Expertensystemtagung XPS 97.

Victor R. Basili, Gianluigi Caldiera und H. Dieter Rombach (1994a). The Goal Question Metric Approach. In J.J. Marciniak (Hg.), *Encyclopedia of Software Engineering*, Band 1, S. 528–532. Wiley, New York.

Victor R. Basili, Gianluigi Caldiera und H. Dieter Rombach (1994b). The Experience Factory. In J.J. Marciniak (Hg.), *Encyclopedia of Software Engineering*, Band 1, S. 469–476. Wiley, New York.

Guido Derwand (1994). *Effizientes Retrieval und Extraktion von Entscheidungswissen mit k-d-Bäumen.* Diplomarbeit, Fachbereich Informatik, Universität Kaierslautern.

Douglas R. Hofstadter (1985). *Gödel, Escher, Bach: ein Endloses Geflochtenes Band.* dtv/Klett-Cotta, Stuttgart.

Michel Manago, Ralph Bergmann, Noël Conruyt, Ralph Traphöner, James Pasley, Jacques Le Renard, Frank Maurer, Stefan Wess, Klaus-Dieter Althoff und Sylvie Dumont (1994). *CASUEL: A Common Case Representation Language.* INRECA Deliverable D1, ESPRIT Projekt 6322.

Peter Mertens, Volker Borkowski und Wofgang Geis (1993). *Betriebliche Expertensystem-Anwendungen.* Springer-Verlag, Berlin, 3. überarbeitete und aktualisierte Auflage.

Gerhard Pews (1994). *MoCAS/2 - Fallbasierte Diagnose in technischen Domänen.* Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern.

Carsten Priebisch (1995). *Integration von fallbasierter und kontextbasierter Diagnose am Beispiel der Systeme INRECA und S3.* Diplomarbeit, Fachbereich Informatik, Universität Kaierslautern.

Michael M. Richter (1992). *Prinzipien der Künstlichen Intelligenz.* Teubner-Verlag, Stuttgart, 2. überarbeitete und erweiterte Auflage.

Michael M. Richter und Oliver Wendel (1990). *Lernende Systeme, Teil 1: Symbolische Methoden.* Vorlesungsskript, Universität Kaiserslautern.

Angi Voss, Brigitte Bartsch-Spörl und Rivka Oxman (1996). A Study of Case Adaptation Systems. In John S. Gero und Fay Sudweeks (Hg.), *Artificial Intelligence in Design '96*, S. 173–189, Dordrecht. Kluwer Academic Publishers.

Stefan Wess (1992). Fallbasiertes Schließen in Deutschland - Eine Übersicht. *Künstliche Intelligenz*, 6(4):46–51.

Stefan Wess (1995). *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik - Grundlagen, Systeme und Anwendungen.* Dissertation, Fachbereich Informatik, Universität Kaiserslautern. Auch erschienen im infix-Verlag, 1996.

Ansgar Woltering und Stefan Wess (1996). Entscheidungsunterstützungssysteme und fallbasiertes Schließen – Eine empirische Untersuchung im deutschsprachigen Raum. *Wirtschaftsinformatik*, 38(1):17–22.

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS