STUDIENARBEIT

Diskussionssysteme

von

Clemente Laboreo, Daniel

eingereicht am **29.9.2007** beim
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
der Universität Karlsruhe

Referent: Prof. Dr. Rudi Studer
Betreuer: Dr. York Sure

Die Arbeit wurde innerhalb eines ERASMUS-Austauschprogramms verfasst.
Meine Heimatsuniversität ist die *Universitat Politècnica de Catalunya* (UPC),
Institut: *Facultat d'Informàtica de Barcelona* (FIB), in Barcelona, Spanien.
Die Arbeit kann von `http://www.danielclemente.com/disk/`
heruntergeladen werden. Diese ist nicht die abgegebene Version, sondern
enthält auch etliche sprachliche Korrekturen (und ich korrigiere sie weiter!).
Version vom 2.11.2007.

Zusammenfassung

Argumentation ist sowohl im täglichen Leben als auch in der Wissenschaft anwesend, in Form von Problemlösung, Entscheidungsfindung, oder Austausch und Begründung von Meinungen. Jedoch sind die Computerprogramme, die üblicherweise zur Diskussion genutzt werden, nicht an die Behandlung und Verwaltung von Argumenten angepasst. Diese Studienarbeit umschließt sowohl eine Analyse- als auch eine Designphase. Zuerst werden die Bereiche beschrieben, in denen Diskussionen mit Hilfe des Computers geführt werden können: Verschiedene Anwendungsbeispiele werden zeigen, wie umfassend dieses Thema ist, denn viele typische Abläufe können auch als Diskussion verstanden werden. Dazu folgt eine Beschreibung der heutzutage für diesen Zweck verwandten Ansätze; spezialisierte Programme und deren theoretische Grundlagen wie z. B. Notationen und Methodologien werden beschrieben, aber auch die grundsätzlichen Internet-Kommunikationsmittel werden berücksichtigt, denn sie sind in der Praxis diejenigen die zum Diskutieren dienen. Die Designphase der Arbeit zielt auf eine Verbesserung der Argumentationsbetrachtung der aktuellen Programme ab; einerseits werden Zusammenstellungen schon existierender Technologien vorgeschlagen, die einfach zu verwirklichen sind, andererseits werden neue und begründete Ideen vorgestellt, um ein einzelnes und grundsätzliches Diskussionsprogramm zu gestalten, das alle Anwendungsbeispiele erfüllt.

Inhaltsverzeichnis

1 Einleitung **5**
- 1.1 Definitionen . . . 5
- 1.2 Zielsetzung . . . 6
- 1.3 Relevanz eines solchen Programms . . . 7
- 1.4 Aufbau der Arbeit . . . 8

2 Anwendungsbeispiele **9**
- 2.1 Persönliches Meinungsbild . . . 9
- 2.2 Kollaboratives Lernen . . . 9
- 2.3 Annotation eines Textes . . . 10
- 2.4 Beweis der Wahrhaftigkeit eines Textes . . . 10
- 2.5 Problemlösen . . . 11
  - 2.5.1 Schwierigkeitsniveau der Probleme . . . 11
  - 2.5.2 Hilfe durch ein Diskussionssystem . . . 12
  - 2.5.3 Zusammenarbeit beim Problemlösen . . . 13
- 2.6 Speicherung von logischen Argumenten . . . 13
- 2.7 Debatte . . . 13
- 2.8 Entscheidungsfindung . . . 14
- 2.9 Zusammenfassung und Anforderungen . . . 14

3 Verwandte Ansätze **16**
3.1 Geläufige Systeme für die Diskussion 16
3.2 Neue Lösungen in diesem Bereich 18
3.2.1 Diskussionsforen 18
3.2.2 Mailing-Listen 19
3.2.3 Erweiterte Wikis 19
3.2.4 Allgemeine Groupware-Systeme 20
3.2.5 Systeme für allgemeines Wissensmanagement 21
3.2.6 Gedankenkarten-Programme 21
3.2.7 Annotationsprogramme 21
3.2.8 Ontologien 23
3.2.9 Methodologien und deren Implementierungen 24
3.2.10 Spezielle Programme für die Argumentation 25
3.3 Formale Notationen 26
3.3.1 Frühere Ansätze 26
3.3.2 Einfache Notationen 27
3.3.3 Andere Notationen 28
3.3.4 Argumentendarstellung durch Programme 28
3.3.5 Zusammenfassung der Notationen 29
3.4 Zusammenfassung der verwandten Ansätze 29

4 Neue Ideen für bessere Systeme **30**
4.1 Erste Beschreibung der Anforderungen 30
4.1.1 Abhängigkeit von der Arbeitsweise 30
4.1.2 Erwartungen an das Programm 31
4.1.3 Gestalt des Programms 31
4.2 Implementierungen mittels gegenwärtiger Technologien 33
4.2.1 Text mit einer Begründung für jeden Satz 33
4.2.2 Aussagenwiki 35
4.2.3 Gedankenkarte 36
4.2.4 Semantisches Aussagenforum 37
4.2.5 Zusammenfassung und gemeinsame Merkmale 38
4.3 Notation 39
4.3.1 Gewünschter Notationstil 39
4.3.2 Prosa vs. Graphen 40
4.3.3 Betrachtung der Begriffe 41
4.3.4 Granularität 42
4.3.5 Anzahl und Typen der Beziehungen 43
4.3.6 Informationseingabe 44
4.4 Semantische Informationen jeder Aussage 45
4.4.1 Wahrheitswert 45
4.4.2 Gültigkeit der Argumente 46
4.5 Wissensnutzung 47
4.5.1 Berechnung des Wahrheitswertes 47
4.5.2 Zusammenstellung ähnlicher Argumente 47
4.5.3 Neue Formulierungen 48

4.5.4 Beimischung von Themen 49
4.5.5 Suche nach Fehlern und Fehlschlüsseln 49
4.5.6 Archivierung 50
4.5.7 Begründungen 50
4.5.8 Mögliche Erweiterungen 51
4.6 Argumentensuche 51
4.7 Kollaboration zwischen Benutzern 52
4.7.1 Kollaborationsarten 52
4.7.2 Prozesse, Probleme und Lösungen 53
4.8 Graphische Schnittstelle und Visualisierung 56
4.8.1 Benutzung der Schnittstelle 56
4.8.2 Mehrere Schnittstellen 56
4.8.3 Genauigkeit 57
4.8.4 Integrierte Suche 57
4.8.5 Ausgabe 58

5 Ergebnisse **58**
5.1 Analyse 58
5.2 Synthese 59
5.3 Schlussfolgerung 60

6 Nächste Schritte **61**

7 Literatur **62**

1 Einleitung

1.1 Definitionen

Unser Thema betrifft mehrere Begriffe mit verschiedenen Namen, da es viele ähnliche Ansätze gibt, um eine so triviale Sache wie „Diskutieren“ zu tun. Zum Beispiel nennen es die Philosophen „argumentieren“, die Wirtschaftwissenschaftler aber „Entscheidungsprobleme“, die Politiker „Debatte“, und die Laien einfach „Diskutieren“. All diesen Ansätze haben jedoch gemeinsame Punkte, und zwar:

- Es gibt ein Hauptthema, z. B. eine Frage zu beantworten, eine Wahl zwischen Alternativen zu treffen, oder nur die Absicht, die eigene Meinung besser zu kennen oder mehr über das Thema herauszufinden. Themaablenkung kann dennoch vorkommen und sie ist meisten unerwünscht.

- Jede Diskussion besteht aus Argumenten, und zwar aus Pro- oder Contra-Argumenten. Diese können wahr oder falsch sein. Die Beziehungen zwischen ihnen ist besonders wichtig, denn falsche Verbindungen verursachen häufig Fehlschlüsse.

Trotzdem gibt es auch Unterschiede zwischen den verschiedenen Ansätzen. Beispielsweise ist die Anzahl der Teilnehmer nicht immer gleich: Eine einzige Person kann zwar an die Pro- und Contra-Argumente allein denken (z. B. bei Entscheidungsproblemen), aber üblicherweise nimmt mindestens eine Person pro Seite teil. Es ist auch möglich, eine Diskussion zwischen hunderten Personen zu führen, z. B. in Internetforen. Außerdem ist die Gestalt und das Höflichkeitsniveau in Diskussionen sehr vielfältig.

Unter „System“ verstehen wir sowohl Computerprogramme als auch Methodologien (theoretische Verfahren), die helfen, eine beliebige Art von Argumentation auf eine teilweise logische Weise zu behandeln. Häufig werden Programme und Methodologien zusammen studiert, da Programme oft die Arbeitsweise beeinflussen und begrenzen.

1.2 Zielsetzung

Diese Arbeit beschreibt und verfolgt die vorherige Untersuchung in diesem Bereich (computerunterstützer Visualisierung und Behandlung der Argumente), schon in Büchern wie (KSE03) betrachtet. Es gibt allerdings verschiedene Anwendungsfälle und Varianten der Diskussionen; wir werden zwar alle berücksichtigen, der bevorzugte Ansatz wird aber vor allem die Programme und Methodologien betreffen, die die normalen Computerbenutzer benötigen. Die Arbeit will somit das persönliche Wissensmanagement erleichtern.

Das Problem, das wir lösen wollen, ist folgendes: Wenn die Menschen miteinander diskutieren wollen, verzichtet man noch auf die Hilfe, die Computer liefern können; meistens werden alle Argumente und Beziehungen im Kopf des Benutzers behandelt, und die einzige Leistung des Computers, die genutzt wird, ist lediglich ein Textverarbeitungsprogramm. Das hat natürlich sowohl Vor- als auch Nachteile.

Mit diesem Gedanke als Ziel wollen wir den Weg besser verstehen und planen, und zwar:

- Die aktuellen Systeme analysieren, sowohl als theoretisches Verfahren als auch als Computerprogramm.
- Ihre Probleme kennen, und verstehen, warum die Leute noch ungeeignete Werkzeuge benutzen (wie E-Mail oder bloße Wikis).
- Die verschiedenen Arten und Bedürfnisse der Diskussionssysteme untersuchen
- Versuchen, ein besseres System zu entwickeln, indem die typischen Probleme gemieden und trotzdem die Erfordernisse erfüllt werden (falls möglich).
- Studieren, wie die Methoden des Semantischen Webs am besten dafür verwandt werden können, sowohl für Repräsentation als auch für die Wissensverarbeitung.
- Feststellen, welche die nächsten zu verfolgenden Schritte sind.

Wir werden leider das perfekte System nicht entwickeln (das wird seit Jahrzenten versucht), aber wir werden lernen, wie das versucht wurde und inwiefern jeder Ansatz funktioniert hat. Auf diese Weise können wir vielleicht dieselben Fehler vermeiden, und auf einen besseren Ansatz abzielen.

1.3 Relevanz eines solchen Programms

Ein Kernpunkt, der zur Motivation für diese Arbeit führt, ist folgender: Viele Probleme können in eine Diskussion umgewandelt werden.

Jeder kennt z. B. Wikipedia, und, obwohl ihre Artikel interessant oder manchmal nützlich sein können, sollte jeder wissen, dass ihr als Quelle nicht vertraut werden kann, denn die Wahrhaftigkeit der enthaltenen Informationen wird nicht immer kontrolliert. Man kann aber jeden Artikel als eine Diskussion betrachten, in der jede Aussage ein Argument ist. Wenn alle Argumente (also Behauptungen) gut begründet werden, aus vertrauenswürdigen Quellen entstammen, und es keine wichtigen Gegenargumente gibt, dann kann man einen Artikel komplett vertrauen.

Man kann dieses Beispiel auf das ganze World-Wide-Web verallgemeinern: Es gibt nämlich Millionen von Webseiten, deren Inhalt auf keine Weise kommentiert werden kann, und deshalb kann man nicht wissen, ob stimmt, was sie sagen. Man sollte in der Lage sein, jeden Inhalt zu diskutieren, obwohl man nicht dessen Autor ist; zum Beispiel durch die Ergänzung einer Webseite mit Notizen, die Argumente für oder wider jede Aussage darstellen. Auf diese Weise wird der Wert des Internets erhöht.

Man kennt auch die üblichen Systeme, um im Internet mit anderen Leuten über ein Thema zu diskutieren, und zwar Wikis, Mailing-Listen, Foren, und eigene Webseiten wie Blogs. Wer schon versucht hat, über verzwickte Themen zu reden, kennt wahrscheinlich die auch üblichen Probleme, z. B.: schwierige Wiedernutzung der früheren Informationen, schnelle Themaablenkung, Provozieren oder Beleidigung seitens mancher Benutzer, Kommentare, die nicht beantwortet werden, usw. Wir sind der Meinung, dass die Software verbessert werden kann, um diese Probleme mit weniger Aufwand vermeiden zu können.

Diskussionen werden auch in wichtigen Gebieten angewandt, beispielsweise in Jura, weil der Richter alle Argumente zusammenfassen muss, damit die möglichen Fehlschlüsse oder Widersprüche ermittelt werden können und durch Gesetze (d.h. Argumente) eine Beurteilung entschieden wird. Es gibt viel Interesse an einem solchen Programm, das dieses Verfahren erleichtern kann.

In großen Unternehmen werden auch Entscheidungsunterstützungssysteme benutzt. Das Entscheiden ist auch ein umfangreiches Gebiet, in dem Argumentation viel helfen kann, denn man strebt nach Rationalität in der Entscheidung.

Am wichtigsten aber ist der Rahmen des allgemeinen Problemlösens. Beispielsweise ist in der Wissenschaft die Diskussion durch korrekte Argumente die Basis für den Weg zu der Lösung eines Problems. In der Praxis gibt es auch Situationen, in denen man diskutieren muss, um ein Problem zu korrigieren. Z. B. die Arbeit mit einem Bug-Tracker-Programm fordert Diskussion zwischen Alternativen für die Lösung eines Programmfehlers heraus. Wenn man aber z.

B. Bugzilla (einer der wichtigsten Bug-Tracker) erwägt, dann konstatiert man, dass er gar nicht geeignet für die zielorientierte Diskussion ist, denn er hat noch mehr Probleme als z. B. eine Mailing-Liste. Deshalb wird auch in diesem Gebiet ein besseres Diskussionssystem benötigt.

Auch persönliche Gegenstände finden eine Hilfe bei Diskussionssystemen. Man kann nämlich Argumentation benutzen, um mehr über ein Thema zu erfahren, um die eigene Meinung zu sortieren, um bessere Texte zu schreiben, oder um eine kleine Debatte zusammenzufassen. Manchmal kann man nicht alles im Kopf entscheiden, und Werkzeuge wie Gedankenkarten oder Wikis helfen uns, große Mengen an Informationen einzuordnen.

Deshalb bilden Diskussionssysteme ein Thema, das viele unserer Interessen betrifft: als Benutzer des Internets, als Teilnehmer in Foren, als Arbeiter die ein Problem lösen müssen, und als Personen, die manchmal große Entscheidungen treffen müssen. Ein solches Programm wäre nützlich für viele Bereiche und Erfordernisse, und deshalb ist es sowohl vormals als auch heutzutage ein versprechender Untersuchungsstoff, den auch diese Arbeit studiert.

1.4 Aufbau der Arbeit

Kapitel 2 beschreibt 8 bestimmte Beispiele von Bereichen und Anwendungen, bei denen ein argumentativer Ansatz hilfreich sein kann; anschließend werden sie verglichen und zusammengefasst. Dies wird helfen, das Problem und dessen Anforderungen zu erfahren.

Kapitel 3 befasst sich mit dem Stand der Technik, und studiert die Programme, die für die Argumentation nützlich sind. Erstens (3.1) werden die weit verbreiteten Kommunikationsmittel betrachtet, zweitens (3.2) die spezialisierten Programme, und drittens (3.3) manche theoretische Notationen, die für eine Formalisierung der Argumente nötig sind.

Während Kap. 2 und 3 eine *Analyse* darstellen, beschäftigt sich Kap. 4 mit dem *Design* besserer Diskussionswerkzeuge. Die wichtigen technischen Eigenschaften des Programms werden grob geplant (4.1). In 4.2 werden 4 Modelle vorgestellt, um typische Programme auf die Diskussion anzupassen: Text mit Begründungen (4.2.1), Aussagenwiki (4.2.2), Gedankenkarte (4.2.3) semantisches Forum (4.2.4); aus diesen Modellen werden die gemeinsamen Merkmale herausgefunden (4.2.5). Anschließlich wird jedes der Merkmale ausführlich betrachtet: Notation (4.3), semantische Informationen (4.4), Wissensnutzung (4.5), Argumentensuche (4.6), Kollaboration (4.7), Schnittstelle (4.8). Für jeden Punkt werden neue Ideen für die Entwicklung eines besseren Diskussionsprogramms vorgeschlagen.

Die Arbeit endet mit einer Zusammenfassung der Ergebnisse (Kap. 5), und stellt neue Ideen für weitere Arbeiten vor (Kap. 6).

Herzlichen Dank an alle, die mir mit den Schwierigkeiten der deutschen Sprache geholfen haben, und an meinen Betreuer York Sure für seine Zustimmung, mich die Arbeit auf Deutsch verfassen zu lassen, obwohl ich Deutsch erst seit einigen Monaten gelernt hatte.

Diese Arbeit wurde anfangs mit LyX, darauf mit `org-mode` (einem Modus des Texteditors `GNU Emacs`) verfasst; der Inhalt wurde schließlich auf LaTeX mittels der LISP-Funktion `org-export-latex` exportiert.

2 Anwendungsbeispiele

Programme für MindMapping, Wikis, und andere Werkzeuge für Argumentationen haben umfangreiche Nutzungsmöglichkeiten. In diesem Kapitel werden einige Anwendungsbeispiele betrachtet und verglichen, um Ähnlichkeiten und Unterschiede herauszuarbeiten. Zusätzlich werden verschiedene Arten von Systemen vorgestellt, um das Problem und die Anforderungen genauer zu beschreiben.

Da es so viele verschiedene Situationen gibt, in denen ein Diskussionssystem benötigt wird, bleibt die Frage offen, ob ein einzelnes Programm alle Anforderungen erfüllen kann. Nach dem Vergleich wird es entschieden, ob das möglich ist, und welche Merkmale ein universelles System aufweisen sollte.

2.1 Persönliches Meinungsbild

Das einfachste „persönliche Wissensmanagement“ am Computer wird durch einen Texteditor erreicht, indem man lockere Gedanken schreibt. Das Ziel hierbei ist nicht das Problemlösen oder das Treffen einer Entscheidung, sondern nur die eigenen Ideen zu beschreiben, zu organisieren und nachzusehen, ob alles einen Sinn hat.[1]

Dafür gibt es passendere Werkzeuge als einen Editor, z. B. Programme, die sich auf Gedankenkarten basieren (Mind Map, Concept Map, und andere). Sie unterstützen die Argumentation aber nicht sehr, da z. B. der Zustand jeder Aussage manuell kontrolliert werden muss. Da Argumentationen über verzwickte Themen sehr üblich sind, wäre ein Diskussionssystem hilfreicher als die geläufigen Systeme – vorausgesetzt, dass es das schnelle Ideenfinden des Menschen nicht beeinträchtigt.

Ein argumentativer Ansatz zur Ideenfindung ist vorteilhaft, da ein Thema besser verständlich ist, wenn über alle möglichen Zusammenhänge nachgedacht wird, und zwar unter Berücksichtigung der Vor- und Nachteile (Pro- bzw. Gegenargumente) und der logischen Verbindung der Behauptungen.

Ein besonders interessanter Fall ist *argumentatives Schreiben*, bei dem ein Text verfasst wird, der Argumente, Thesen und Begründungen ohne Fehlschlüsse vorstellt.

2.2 Kollaboratives Lernen

Der soeben erklärte Ansatz kann auch bei gleichzeitiger Nutzung mehrerer Personen angewandt werden. Es kommt häufig zu Diskussionen, wenn mehrere Menschen zusammenarbeiten, aber dabei können neue Gedanken entstehen, auf

[1] Diese Versinnlichung wird auch etwa „sense making“ auf Englisch genannt.

die der Einzelne nicht käme. Beim Aufbau eines Lernprogramms wird es empfohlen, auf die Argumentation zu zielen, denn die ausführliche Sammlung und Beschreibung von möglichen Zusammenhängen sollte das Lernen verbessern. Für diese Themen interessiert sich das Gebiet der *computervermittelten Kommunikation* (CVK; englisch CMC); weitere Informationen darüber gibt es in (KSE03; VAK99).

Ein wichtiger Ansatz zur Zusammenarbeit ist bislang das *kollaborative Schreiben*: Mit Hilfe eines speziellen Editors können viele Personen denselben Text bearbeiten; dieser Ansatz wird auch in Wikis benutzt.

2.3 Annotation eines Textes

Statt einen neuen Text allein oder kollektiv zu schreiben, kann man einen schon geschriebenen Text (oder andere Werke) kritisieren und verbessern. Eine gute Art und Weise, sich dem zu verbessernden Text zu nähern, wird durch Anhängen der Anmerkungen an Abschnitte erreicht, z. B. um mitzuteilen, dass man mit einer Behauptung nicht einverstanden ist, oder dass ein Satz mit anderen Worten besser formuliert wäre. Nach einer kurzen Diskussion und dem Austausch von Argumenten kann der Text vielleicht verbessert werden.

Man könnte diesen Ansatz sogar bei Wikipedia benutzen, da Wikipedia eine große Sammlung von Texten darstellt, die ständig verbessert werden müssen. Eine Verbesserung muss anhand der schon entschiedenen Rechtlinien und der verfügbaren Informationsquellen erfolgen – nicht aber anhand persönlicher Meinungen, die nicht begründet werden können. So wäre auch am einfachsten festzustellen, welche Abschnitte die provozierendsten sind und welche das Einverständnis der Benutzer haben.

Das Beste ist, dass diese Anmerkungen an beliebige Webseiten des Internets angehängt werden können, auch wenn deren Autoren eine Bearbeitung nicht gestatten. Dabei wird die Webseite nicht verändert, sondern nur werden ihr zusätzliche Informationen hinzugefügt, die in einem anderen Server gespeichert werden können.

Die Verbesserung anderer Arbeiten neben Webseiten (z. B. Bilder, Musik oder Ontologien) ist ebenfalls beachtenswert.

2.4 Beweis der Wahrhaftigkeit eines Textes

Dies ist die Fortführung des vorherigen Anwendungsbeispieles. Wenn viele Leute einen Text reichlich diskutieren und verbessern, soll am Ende nachgewiesen werden können, dass der Text vollständig der Wahrheit entspricht.

Man kann auch anzuhängende Notizen dafür verwenden, aber nicht um Meinungskonflikte zu äußern, sondern um die Behauptungen zu begründen, natürlich auch anhand richtiger Argumente. Ziel ist es den Text lediglich zu korrigieren, nicht zu erweitern oder unzählige z. T. persönliche Anmerkungen zu erhalten. Dabei soll die Wahrhaftigkeit jedes Textabschnitts und folglich des gesamten Textes nachgewiesen werden.

Die gleiche Methode kann auf Webseiten des Internets angewandt werden, um den Wahrheitsgehalt einer Webseite zu überprüfen, insbesondere bei Nachrichtendiensten, Firmeninternetpräsenzen oder wissenschaftlichen Publikationen. Das funktioniert natürlich nur, wenn die Benutzer sich Mühe geben, den z. T. wechselnden Inhalt zu diskutieren.

Dieser Ansatz lässt sich auch auf Wikipedia anwenden, um eine seiner größten Schwachstellen (Ungewissheit über die Wahrhaftigkeit des Inhaltes) zu verbessern. Zumindest bei fast abgeschlossenen Artikeln, die sich nicht mehr ständig ändern.

Andere vom Computer darstellbare Objekte (wie Ontologien oder Programme) sind so ebenfalls beweisbar.

2.5 Problemlösen

Der Themenbereich des Problemlösens ist ein Sonderfall bei Diskussionen, denn das Ziel ist nicht das Diskutieren, sondern die Lösung einer Aufgabe. Diskussion ist hier eher unerwünscht: je weniger diskutiert wird, desto schneller wird das Problem gelöst.

Ein Beispiel dafür ist das Verwalten von Programmfehlerbeschreibungen bzw. -lösungen durch ein *Bug-Tracker-System* wie Bugzilla. Auch die Beantwortung schwieriger Fragen kann als Problemlösen betrachtet werden.

2.5.1 Schwierigkeitsniveau der Probleme

Nicht alle Probleme sind genauso schwer zu lösen. Zu den Einfachsten gehören **gut strukturierte Probleme**, z. B. die Aufgaben aus Schulbüchern oder Logikrätsel. Sie sind relativ einfach und lassen sich in kleinere Teilprobleme zerlegen. Diese Dekomposition kann auch benutzt werden, um komplexere Probleme zu vereinfachen, vorausgesetzt, das Problem ist gut verständlich.

Das ist aber nicht immer der Fall: Manche Probleme sind so schlecht definiert, dass man nicht wissen kann, welches genau das zu lösende Problem ist. Beispielsweise stellt die Frage „Wie kann der Erfolg eines Unternehmens garantiert werden?“ eigentlich mehr als nur dieses Problem dar. Denn diese Aufgabe kann weder eindeutig beschrieben noch aufgeteilt werden, und alle vorgeschlagenen Lösungen wären ebenfalls problematisch. Da keine Lösung *a priori* angewendet werden kann, sind sie weder „richtig“ noch „falsch“, sondern höchstens „vielleicht besser“, „vielleicht schlechter“, oder „ungefähr genauso gut“.

Man nennt diese Art Probleme *unstrukturierte* oder **verzwickte Probleme**; diese Benennung entspricht dem englischen Begriff „ill-structured problems“. Verzwickte Probleme sind sehr typisch in allen Bereichen, v. a. in Wissenschaft und Wirtschaft.

Noch problematischer sind die so genannten „**bösartigen Probleme**“ die 1973 von Horst Rittel unter dem englischen Begriff „wicked problems“ definiert wurden. Bösartige Probleme sind ja verzwickte Probleme, bei denen außerdem jede Person, die an der Lösung des Problems arbeitet, eine andere Ansicht des Problems hat. Das erschwert natürlich die Zusammenarbeit an der Lösung und

verursacht Diskussionen zwischen den Mitarbeitern. Deshalb ist das Studium bösartiger Probleme wichtig für die Erschaffung eines effektiven Diskussionssystems. (KSE03, Kap. 2)

Beispiele für bösartige Probleme sind Fragestellungen wie „Können Maschinen denken?“ oder „Wie kann man Terrorismus am besten kämpfen?“. Solche Formulierungen werden für immer und ewig als schlecht formuliert gelten, dennoch wurden und werden sie das Thema vieler Diskussionen.[2] Themenbereiche wie Politik oder Philosophie, sowie kreative Aufgaben wie Musikkomposition, Zeichnung oder Design stellen auch gute Beispiele dar.

Einige typische Eigenschaften bösartiger Probleme sind folgende:

- Die Problemdefinition wird erst verstanden wenn eine Lösung erfolgt, und ändert sich während die Lösung entwickelt wird.
- Es gibt keinen Anhaltspunkt, der die Lösung des Problems anzeigt. Beendet wird die Lösung, sobald keine Ressourcen mehr vorhanden sind.
- Eine Lösung ist nicht „richtig“ oder „falsch“, sondern „besser“, „schlechter“, „gut genug“ oder „nicht gut genug“.
- Es gibt keine eindeutige Ansicht über das Problem, da es viele verschiedene Informationsquellen gibt.
- Jedes bösartige Problem ist anders, sodass eine spezifische Lösung für jedes Einzelne gefunden werden muss.
- Jeder Versuch, eine Lösung auszuprobieren, hat zur Folge, dass das Problem komplizierter wird, oder dass noch mehr bösartige Probleme erzeugt werden.
- Es gibt keine alternative Lösung, die das Problem umgeht.

Die Behandlung bösartiger und anderer verzwickter Probleme ist mit vielen Schwierigkeiten behaftet. Die Tatsache, dass das Problem sich während der Lösung ständig ändert, lässt die Frage aufkommen, ob diese Art von Problemen vielleicht für eine Bearbeitung mit dem Computer nicht geeignet seien.

2.5.2 Hilfe durch ein Diskussionssystem

Manche Leute (z. B. Horst Rittel) glauben aber, Argumentation sei der Kernpunkt für die Lösung bösartiger Probleme. Diskussionen zwischen den Teilnehmern seien also nicht zu vermeiden, sondern sogar wünschenswert da sie zur Lösung beitragen. Dieser Gedanke war die Grundlage der ersten Ansätze der Diskussionssysteme. (KSE03, S. 26)

Wenn ein Programm beim Problemlösen verwendet wird, kann es eine Hilfe oder ein Hindernis darstellen. Bei bösartigen Problemen ist es oftmals von Hilfe, da man viel Aufwand betreiben muss, um das Problem zu verstehen.

[2] Gleich wie andere interessante Fragen, die den Philosophen und Theologen für Jahrhunderte eingefallen haben, z. B. „Wieviele Engel können auf einer Nadelspitze tanzen?“.

Ein Diskussionssystem kann dennoch die tatsächliche Arbeit verhindern. Bei der Problemlösung kann es zu dem seltsamen Fall kommen, dass mit der Verwaltung der Lösung mehr Zeit verbracht wird als mit der Lösung selbst. Beispielsweise wird viel Zeit dafür verwendet, Programmfehler in einem Bug-Tracker-System richtig zu kategorisieren und zu beschreiben, neue Beiträge zu lesen und sie mit anderen Berichten zu verknüpfen. Das ähnelt dem Papierkrieg des realen Lebens und sollte vermieden werden, da mit diesem Verhalten die Benutzer getäuscht werden: sie werden im Glauben gelassen, dass sie etwas Nützliches beisteuern, tragen aber tatsächlich nichts oder wenig zur Lösung bei.

2.5.3 Zusammenarbeit beim Problemlösen

Es gibt verschiedene Art und Weisen, sich einem bestimmten Problem gegenüberzustehen:

- Eine einzelne Person kann das Problem von alleine lösen. Bei einem einfachen Problem ist eine Lösung im Kopf möglich, bei verzwickten Problemen kann ein Computerprogramm von Nutzen sein.
- Verschiedene Personen gliedern die Aufgabe und jeder beschäftigt sich mit einem Teilgebiet. Dieser Vorgang wird „Kooperation“ genannt (laut (KSE03, S. 25)) und ist nur möglich, wenn sich das Problem gut aufteilen lässt.
- Oder mehrere Personen arbeiten zusammen, indem sie dasselbe Ziel und Aufgabe haben. Diese Zusammenarbeit wird auch „Kollaboration“ genannt und ist am schwersten umzusetzen.

2.6 Speicherung von logischen Argumenten

Ein weiteres Anwendungsbeispiel ist die Speicherung von Informationen in Form von Diskussionsschemata. Diese Daten können dann von anderen Programmen benutzt werden; dazu wird nur eine Notation benötigt, die alle Wissensbestandteile richtig wiedergeben kann. Schnittstellen oder Kollaboration der Benutzer sind aber nicht erforderlich.

Ein solches Ziel entspricht dem der Ontologien (die in der Regel Konzepte speichern), aber mit Rücksicht auf typische Argumentationsstrukturen: Behauptungen, Für- und Gegenargumente, Wahrheitswerte, Verknüpfungen usw.

2.7 Debatte

Debatten sind Diskussionen in reinster Form und deshalb ein sehr vielfältiges Feld. Sie sind zum Teil streng geregelt, beispielsweise in der Politik oder im juristischen Bereich, da von den Ergebnissen wichtige Entscheidungen abhängen. In anderen Fällen wird zwar nur zum Spaß debattiert, aber ebenfalls mit strengen Regeln, z. B. in Debattierklubs. Die Rhetorik ist hier wichtiger als das Vorhandensein eines Computerprogramms, das die Argumente speichert; trotzdem

kann es nützlich sein, falls der Verlauf der Debatte beschrieben oder gespeichert werden muss.

Manche Debatten werden mit Hilfe von Computern geführt, z. B. in Diskussionsforen oder Mailing-Listen. Hier ist das Förmlichskeitsniveau geringer, da normalerweise keine Diskussionsregeln existieren und auch keine Absichten, eine Diskussion zu „gewinnen“ oder zu „verlieren“. Hauptzweck der Foren ist die Kommunikation und der Austausch von Gedanken.

In einem Forum gibt es aber nicht nur Gerede. Gelegentlich sind die gegebenen Informationen interessant und hochwertig, und bieten sich zum Gebrauch für weitere Anwendungen an: Bestätigung der Information, Speicherung von Teildiskussionen um sie künftig wieder zu nutzen (vielleicht in anderen Zusammenhängen), Unterscheidung von Unfug und guten Beiträgen, und weitere semantische Funktionen, die ein auf Diskussion gezieltes System anbieten kann.

2.8 Entscheidungsfindung

Eine Diskussion kann sich um eine Entscheidung drehen. Vor allem Unternehmen müssen ihre Geschäftstrategien für die Zukunft vergleichen und eine auswählen; sie müssen z. B. entscheiden, in welches Produkt sie investieren wollen, in welchem Land sie produzieren wollen oder welche Zeitplanung am günstigen ist. Dafür gibt es die so genannten Entscheidungsunterstützungssysteme (EUS), Methodologien und Programme, die seit den 50er Jahren üblich sind.

Entscheidungen können auch als Probleme angesehen werden, meistens als verzwickte Probleme. Es gibt jedoch andere Merkmale, die die zwei Ansätze unterscheiden. Eine Entscheidung ist nämlich keine offene Frage, wie z. B. „Wie kann die Konkurrenz reagieren?“, sondern wird mit einer Reihe möglicher Alternativen versehen. Es kann auch eine subjektive Abstimmung zwischen Alternativen erfolgen. Bei einer Entscheidung ist außerdem notwendig, eine Wahl zu treffen; sie ist also *präskriptiv*, im Gegensatz zum *deskriptiven* Ansatz, der benutzt wird, um durch Diskussion mehr über ein Thema zu entdecken.

Entscheidungen weisen Merkmale und Schwierigkeiten der bösartigen Probleme auf. Die Lösung kann nämlich ebenfalls nicht als „gut“ oder „schlecht“ bezeichnet werden (falls möglich würde man die schlechten Alternativen einfach ausschließen), sondern nur als „besser“ oder „schlechter“. Auch hier können die Alternativen nicht ausprobiert werden, da jede wohl ungünstige Folgen aufweist.

Da die Entscheidungsfindung nicht komplett zum Problemlösen passt, gibt es andere Ansätze neben Argumentation, z. B. die Nutzung von Statistik und der Wahrscheinlichkeitstheorie. Oft werden allerdings Entscheidungen durch Intuition getroffen, da die formalen Beschreibungsverfahren den Menschen ungeeignet sind. (EW03, Kap. 1)

2.9 Zusammenfassung und Anforderungen

Etliche Eigenschaften der Anwendungsbeispiele sind schon besprochen worden. In der folgenden Tabelle wird subjektiv dargestellt, inwiefern jedes Merkmal auf die jeweiligen Beispiele zutrifft.

Die verwendeten Abkürzungen für die *Anwendungsbeispiele* sind: 1 (Meinungsbild), 2 (kollaboratives Lernen), 3 (Textannotation), 4 (Wahrheitsbeweis), 5 (Problemlösen), 6 (Speicherung), 7 (Debatte), 8 (Entscheidungsfindung). Jede *Zelle* enthält entweder j für „klares Ja“, n für „klares Nein“, oder nichts wenn keine klare Antwort gefunden wurde.

Eigenschaft	1	2	3	4	5	6	7	8
Mehrere Benutzer	n	j				n	j	
Keine Anfangsinformationen vorhanden	j	j	n	n	j		j	n
Ziel: Mehr über ein Thema zu erfahren	j	j	n	n	j	n	n	n
Ziel: Diskussion an sich	n	j		n	n	n	j	n
Ziel: Verbesserung einer Arbeit			j	j	n	n	n	n
Argumentenwahrheit ist sehr wichtig	n	j	j	j		n	j	j
Eine Entscheidung wird getroffen	n	n	n	n	n	n		j
Bösartige Probleme kommen oft vor	n	n	n		j	n	n	j
Berücksichtigung vieler Regeln	n	n	n		j	n	j	j

Man bemerkt, dass es wenige Anwendungsfälle gibt, die immer allein oder immer kollektiv durchgeführt werden. Normalerweise darf eine Diskussion sowohl allein als auch mit vielen Benutzern geführt werden.

Wenn eine einzelne Person das Problem bearbeitet, ist der Wahrheitswert der Argumente eher subjektiv und hängt von dem persönlichen Standpunkt ab; so können bösartige Probleme vermieden werden, indem nur eine Ansicht des Problems ausgewählt wird. Mehr Eigenschaften bösartiger Probleme (also starke Kooperationsprobleme zwischen Benutzern) sind zu sehen:

- Bösartige Probleme kommen vor allem vor, wenn mehr Wissen über ein Thema nötig ist, oder wenn Entscheidungen getroffen werden müssen.
- Sie werden vermieden, wenn man Diskussion als Ziel hat, weil dann die Teilnehmer etwas Nützliches aus der Diskussion herausziehen wollen, anstatt nur zu streiten. Vielleicht ist deshalb Argumentation doch ein Hauptinstrument um bösartige Probleme zu behandeln.
- Manche Regeln schränken die Ablenkung ein (z. B. Redezeitbeschränkungen in einer Debatte), wobei auch bösartige Probleme vermieden werden. Jedoch können auch Regeln und Beschränkungen oft Probleme verursachen, z. B. beim Problemlösen oder bei der Entscheidungsfindung.

Die Anforderungen für ein Diskussionssystem sind sehr unterschiedlich: das Treffen von Entscheidungen kommt fast nur bei Entscheidungsproblemen vor, die Verbesserung eines Werkes nun wenn dies das Ziel war, usw. Eine Speicherung der Argumente ist aber wohl der einfachste Fall, der behandelt werden kann, da er keine zusätzlichen Ziele anstrebt; daher ist das ein guter Ansatz für den Beginn eines Argumentationsprogramms.

Die Anwendungsfälle können anhand des Schwierierigkeitgrads der Bearbeitung so subjektiv geordnet werden: Speicherung, Meinungsbild, Textverbesserung, Wahrheitsbeweis, kollaboratives Lernen, Debatte, Entscheiden, Problemlösen.

Da alle auf Argumentation aufbauen, ist es anscheinend möglich, mit einem grundsätzlichen Programm alle Anwendungsbeispiele unterstützen zu können.

3 Verwandte Ansätze

Im Folgenden werden Programme und Methodologien für die Computerargumentation beschrieben; erstens die am weitesten Verbreiteten und zweitens diejenigen, die Argumentation explizit als Ziel haben. Auch einige theoretische Notationen werden anschließend erklärt.

3.1 Geläufige Systeme für die Diskussion

Obwohl es manche Programme gibt, die sich nur mit Argumentation bzw. Wissensmanagement beschäftigen, kennen die normalen Computerbenutzer sie nicht. Stattdessen werden etliche einfache Werkzeuge schon seit langem und überall benutzt, um sowohl einfache als auch komplexe Aufgaben zu erledigen.

Webseiten, E-Mail, Foren, Wikis usw. *können* dafür benutzt werden, Diskussionen zu halten, aber sie könnten noch besser sein. Hierzu vergleiche man: ein Blatt Papier und ein Bleistift ermöglichen es auch, Argumente zu visualisieren und zu verknüpfen, aber sie bieten keine Hilfe an. Man *braucht* also keine Werkzeuge, und deshalb benutzt man gerne für Diskussionen sogar Programme, die nicht dafür geeignet sind.

Ein zwar alter aber immer noch verwandter Ansatz zur Speicherung der Behauptungen über ein Thema ist die Nutzung von **statischen Systemen**, wie Büchern und anderen Dokumenten. Man benutzt z. B. noch wissenschaftliche Artikel, um Beiträge gebündelt über ein bestimmtes Problem vorzustellen, indem man andere Werke zitiert, bezweifelt oder benutzt, um das Eigene zu unterstützen. Diese Arbeitsweise ist heutzutage immer noch zu berücksichtigen, denn statische **Webseiten** funktionieren auf die selbe Weise. Man darf eine normale Webseite nicht verändern, und deshalb werden Diskussionen theoretisch vermieden. Man muss sich die große Mühe machen, eine neue Webseite zu erzeugen, um die Ursprüngliche zu kommentieren. Dennoch ist das Verknüpfen von Webseiten ziemlich trivial, und deshalb gibt es Ideen für das Ergänzen statischer Informationen seit dem Entstehen des Hypertextes, z. B. der Memex[3] oder Programme, die irgendwelchen Webseiten Anmerkungen hinzufügen können.

E-Mail ist ein allgemeines Kommunikationsmittel, Informationen auszutauschen, und deshalb wird sie trotz ihrer Probleme für alle Zwecke verwendet. E-Mail leidet gemeinhin unter einem Mangel an Struktur, denn die Kontrolle der Gesprächsfäden ist zu schwach und geht oft verloren, sobald mehr als zwei Personen sich einander schreiben; für diesen Zweck wird noch eine andere Software –eine **Mailingliste**– benötigt. Darüber hinaus ist es nur bequem, die *vorherige* Nachricht des Schreibers zu zitieren, aber nicht die Früheren; die übertragenen

[3] Vannevar Bush, 1945, *As we may think*.

Informationen sind eher vorläufig und nicht zum Speichern oder zur Wiederbearbeitung gedacht. Es gibt außerdem keinen Standard, Worte zu zitieren, sondern man muss den ganzen Text nochmals kopieren und die Kommentare entweder oben, unten oder dazwischen einfügen.

Seit langem gibt es auch Diskussionsforen, jedoch anfangs unter dem Namen **„Newsgroups“** bekannt. 1979 entstand USENET als Sammlung dieser Foren, 10 Jahre vor der Gestaltung des WWWs. Diese zwar alten aber dennoch immer noch benutzen Foren unterstützen die Diskussion besser als E-Mail. Die Kontrolle der Gesprächsfäden ist hier nämlich besser, und die Daten werden auf einem Server gespeichert. Man hat eine Hierarchie von Themen (z. B. de.etc.fahrzeug.auto), die jedoch nicht gut genug ist, da der Versand einer Nachricht an viele Foren (*crossposting*) nicht gern gesehen wird, obwohl eine Nachricht doch mehrere Themen betreffen könnte.

Darüber hinaus gibt es mehrere Arten von **Diskussionsforen**, vor allem als Webseiten im WWW. Die meisten bieten die grundsätzlichen Funktionen an, und typische Erweiterungen sind eher Smileys und tolle Stile, als eine bessere Beschreibung und Nutzung der Information. Die großen Foren (z. B. Slashdot) wissen bereits um einige der Probleme dieses Gebiets (z. B. Informationüberflutung und Vandalismus), und benutzen deshalb Methoden, um die Diskussion angenehmer zu machen, z. B. durch Moderation und durch Benutzerränge (jeder Benutzer bekommt bzw. verliert Punkte durch gute bzw. schlechte Beiträge). Man hat sowieso die argumentative Struktur nicht, die wir für unsere Anwendungsfälle möchten. Die Beiträge kann man schwerlich einordnen, ergänzen, befürworten, widerlegen, oder irgendwie verknüpfen, und zudem enthält jede Nachricht normalerweise viele Aussagen und Themen. Obwohl diese Foren noch fern unserer Idee von einem logischen Diskussionssystem liegen, kann man daraus viele Schwierigkeiten und deren möglichen Lösungen ableiten.

Eine für alle Ziele benutzte Lösung sind die **Wikis**, da sie auch Diskussionen zwischen Benutzern erlauben (Wikipedia hat nämlich eine Diskussionsseite pro Artikel). In einem Wiki gibt es aber kaum Regeln den Inhalt betreffend, und stattdessen gibt es die Möglichkeit, den Inhalt später von anderen Personen verbessern zu lassen. Da dies nicht immer geschieht, kommen hier alle möglichen Probleme der anderen Systeme vor.

Genauso wie in der „normalen“ (alltäglichen) Kommunikation besteht auch die Möglichkeit, sich mit Hilfe des Computers informell zu unterhalten, nämlich per **Chat** (z. B. IRC) oder per **sofortiger Nachrichten** zwischen mehreren Benutzern. Ähnlich wie die „normale“ Rede profitiert ein solcher Ansatz kaum von den Möglichkeiten der Computerunterstützung: man hat keine Notation, kein Interesse an der Beibehaltung des Gesprochenen, keine Kontrolle des Gesprächsfadens, man vermag nicht einfach zu zitieren, usw. Diese normale Rede sollte aber nicht geringgeschätzt werden, denn auch seriöse Diskussionen werden auf diese Weise durchgeführt (z. B. Debatte).

Andere Programme, wie **Bug-Tracking-Systeme**, zielen auf einige unserer Anwendungsfälle ab, aber die Argumentation wird oft verachtet. Ein Beispiel

dafür ist Bug 18574[4] im Mozillas Bugzilla: Es gibt zwar keine Entscheidung über die Problemlösung, aber mehr als 700 Kommentare, unter denen man alles finden kann: Code, persönliche Angriffe, Zitate über Demokratie und Diktatur, gültige technische Argumente, Fehlschlüsse, Zusammenfassungen, und Klagen gegen die Nutzlosigkeit des Bug-Berichts. Oft wird sogar vorgeschlagen[5], die Diskussion irgendwo anders zu führen. Ein weiteres Beispiel mit erheblichem Argumentenaustausch und Strukturverworrenheit ist Bug 25537[6], insbesondere Kommentar 231.

3.2 Neue Lösungen in diesem Bereich

Die vorher beschriebenen Ansätze werden für das Diskutieren tatsächlich benutzt, obwohl sie nicht dafür geschafft wurden. Nun beschreiben wir die Systeme, die die Argumentation als Hauptziel unterstützen. Darunter findet man verbesserte Grundwerkzeuge (wie Editoren oder Wikis), theoretische Methodologien und Programme, die ausschließlich für die logische Diskussion programmiert wurden.

3.2.1 Diskussionsforen

Alle Diskussionsforen und Mailing-Listen besitzen eine gute Basis für die Argumentation, denn sie unterstützen Gesprächsfäden, mehrere Benutzer, Speichern und Zitieren, und sie verlangen keine reglementierte Arbeitsweise. Um die vorher erwähnten Probleme zu vermeiden, gibt es Erweiterungen:

Ein einfacher Ansatz, Ergebnisse aus einer Diskussion zu erzeugen, besteht in der Möglichkeit, einem Beitrag nicht nur antworten sondern auch *zustimmen* bzw. ihn *ablehnen* zu können. Auf diese Weise lassen sich Statistiken erzeugen, sowie auch Abschätzungen über den Wahrheitswert jeder Aussage. Dieser Ansatz wird z. B. bei Debate Point[7] verwendet, und ist sehr einfach zu nutzen, aber ebenfalls zu missbrauchen, denn es zählt nicht der Inhalt der Argumente, sondern die Anzahl an Leuten, die sie unterstützen.

Geeigneter für unseren Zweck ist TruthMapping[8], eine Webseite die ein besseres Diskussionssystem als die Ursprünglichen sein will. Sie speichert viele unter Kategorien stehende Aussagen und deren Kritiken, aber wichtiger sind die Ergebnisse jeder Diskussion. Man kann einer Aussage entweder zustimmen oder sie widerlegen, indem man markiert, welche Art von Fehlschluss sie beinhaltet. Jede Aussage entspricht aber vielen Paragraphen und wird mit eigenen Worten verfasst; deshalb ist es möglich, zu viel zu schreiben bzw. abzuschweifen.

Noch mehr auf die Argumentation gezielt sind Foren wie DebateMapper[9],

[4] Wiedereinführung des MNG-Formats in Firefox. `https://bugzilla.mozilla.org/show_bug.cgi?id=18574`

[5] z. B. Kommentar 135: `https://bugzilla.mozilla.org/show_bug.cgi?id=52746#c135`

[6] „Alt text is not displayed as a tooltip over <img>“. `https://bugzilla.mozilla.org/show_bug.cgi?id=25537`

[7] `http://debatepoint.com/`

[8] `http://www.truthmapping.com/`

[9] `http://debatemapper.com/`

eine Webseite, wo verzwickte Themen (insbesondere Politik) formalisiert, graphisch dargestellt und diskutiert werden können. Sie dient dazu, verschiedene Ansichten zusammen zu bringen und die üblichen Fehlschlüsse so zu vermeiden. Alle benötigten Operationen zur Diskussion (wie Abstimmungen, Verfolgung, Verlinkung, Struktur, ...) sind vorhanden. Die Anwendung ist aber nur auf einige Webbrowser beschränkt und erfolgt ausschließlich in einer einzelnen Webseite mit propietären Programmen; man darf deshalb kein grundsätzliches Diskussionsprogram daraus erweitern. Sowieso bleibt DebateMapper der beste Webseite-Ansatz bis jetzt.

3.2.2 Mailing-Listen

Mailing-Listen können auch erweitert werden, aber nicht auf die selbe Weise wie Foren, denn das Format der E-Mail kann nicht einfach geändert werden. Stattdessen werden zusätzliche Programme hinzugefügt, die Informationen aus den Nachrichten prozessieren und Semantik im Nachhinein erzeugen können. Ein solcher Ansatz ist Semalan (Gö06), der ein strukturiertes Diskussionsmodell vorschlägt, um E-Mail-Diskussionen mit feiner Granularität darzustellen. Eine Schnittstelle zum Datenmodell, eine Benutzeroberfläche und der Rest der Architektur wurden auch programmiert. Das Modell wird in RDF gespeichert, damit andere Programme diese Informationen nutzen können.

Semalan zerteilt jede Nachricht in Abschnitte (wie z. B. Zitate), die sich anschließend miteinander verknüpfen lassen. Die wichtigen Beziehungen für E-Mail („antwortet“ und „zitiert“) werden automatisch erzeugt, aber die Verknüpfung von Argumenten in einer Diskussion bedarf menschlicher Arbeit. Semalan selbst unterstützt die Argumentation nicht, aber es gibt doch Ansätze, E-Mail-Antworten mit Annotationen wie „für“ und „gegen“ zu markieren. Das ist mit den herkömmlichen WWW-Links nicht möglich, aber andere Modelle von Hypertext-Systemen, z. B. „Open Hypertext System“ (OHS, von Douglas Engelbart entwickelt) erlauben es, typisierte Links zu haben. Es gibt z. B. einen Ansatz[10], um Annotationen aus der Argumentationsmethodologie IBIS[11] mittels OHS auf Mailing-Listen anzuwenden.

Das Diskussionsmodell aus Semalan unterstützt auch Abstimmungen mittels semantischer Annotationen. Auf diese Weise kann die Meinung der Teilnehmer explizit erfasst werden.

3.2.3 Erweiterte Wikis

Es gibt viele Ansätze, um semantische Informationen an den sonst unsemantischen Wikis zu verwenden. Beispielsweise gibt es Semantic Mediawiki (eine Erweiterung zu Wikipedias Software). Dessen Syntax erlaubt eine bessere Beschreibung der Aussagen (z. B. „Die Stadt hat [[Einwohnerzahl:=3393933]] Einwohner“) und der Verknüpfungen zwischen Artikeln (z. B. „Berlin ist Hauptstadt

[10] Eugene E. Kim, Ken Holman. *Interoperability Between Collaborative Knowledge Applications*. EML 2002.

[11] IBIS ist eine argumentative Problemlösen-Methodologie und wird später erklärt.

von [[Hauptstadt von::Deutschland]]“). Das taugt für die tabellarischen Daten (Personen, Städte, Bücher usw.) und ermöglicht bessere Suchanfragen, aber die Rede der Argumentation ist erheblich komplexer; sie enthält nämlich ganze Sätze und Nebensätze, Negation, abstrakte Ideen (statt Weltobjekte), und mehr. Die Anforderungen sind zu viel für die Wiki-Syntax (die man per Hand schreiben muss).

Deshalb verzichtet man in einem Wiki darauf, alles zu formalisieren, und zieht vor, nur die Grundlagen der Argumentation zu schreiben, z. B. „Pro“, „Contra“, und Verfolgungsinformation wie „Erledigt“. In den Wikimedia-Projekten (Wikipedia, Wiktionary usw.) gibt es Vorlagen dafür, die man vorwiegend in Diskussionsseiten oder Abstimmungen benutzt. Die Vorlagen schreiben übrigens nur den Text, aber speichern keine semantische Information. Das Übliche ist, in Diskussionsseiten wie gewöhnlich zu schreiben, d. h. ohne Notation und ohne Gestalt.

Es kann aber auch helfen, in einem Wiki bestimmte Konventionen zu haben, um die Argumentation damit besser zu kontrollieren. Ein Beispiel ist ibaw[12], eine Empfehlung, nur „Ausgabe, Idee, Pro-Arg., Contra-Arg., Entscheidung, Siehe auch“ im Wiki zu schreiben, und zwar in entsprechenden eingerückten Listen.

Man braucht nicht so viele Konventionen wenn die Benutzer die Absicht haben, zu kooperieren statt zu kämpfen. Für kollaboratives Schreiben ist nämlich ein Wiki geeignet, aber es gibt auch Programme, wie Gobby oder MoonEdit, die die Bearbeitungen echtzeit koordinieren. Sie wurden für Anwendungsfälle wie gemeinsame Programmierung oder Dokumentation geschafft, nicht aber für den Austausch oder Speicherung logischer Argumente.

Es gibt allerdings viele Wikis im Internet, die auf Argumentation basieren, und die Absicht haben, die „Wahrheit“ zu finden. Manche haben sogar ehrgeiziger Ziele, wie „den Bürgern die Demokratie zu bringen“, und tragen noch keine oder wenige Neuerungen zur Wikitechnik bei. Einige Beispiele sind: WikiReason, Demosphere Project, DebatePedia, Chains of Reason. Wie andere Wikis sind sie nur nützlich wenn es jemanden gibt, der die Fehler schnell korrigieren kann. Und die Fehler sind versichert, denn die Wiki-Syntax plus die zusätzlichen Regeln der Diskussion verursachen eine schwierig zu benutzende Schnittstelle zum Inhalt.

3.2.4 Allgemeine Groupware-Systeme

Unter der Bennenung „Groupware“ findet man viele komplexe Systeme, um Aufgaben in einem Team zu verwalten. Sie zielen nicht genau auf Argumentation ab, aber oft enthalten Foren, Wikis, Bug-Tracker und andere Werkzeuge, die normalerweise besser als die Üblichen geplant sind.

Beispiele davon sind eGroupWare, Citadel/UX, Exchange Server (von Microsoft), GroupWise (von Novell), Kolab, Launchpad (von Canonical), Lotus Notes, Open-Xchange, PHProjekt, und viele mehr. Mitsamt Inhaltsverwaltungssystemen werden sie häufig in Unternehmen und Gemeinschaften benutzt, auch wenn

[12] ibaw: issue based argumentation in a wiki. `http://xam.de/2006/02-ibaw.html`

das Lernen sämtlicher Funktionen einen erheblichen Aufwand benötigt.

3.2.5 Systeme für allgemeines Wissensmanagement

Es gibt viele Lösungen für das sowohl persönliche als auch kollaborative Wissensmanagement. Manche Programme bieten verschiedene Techniken an, um Informationen zu verwalten, z. B. durch Gedankenkarten, Notizen, Wikis, E-Mail, Foren usw. Es ist sinnvoll, ein solches integriertes System zu haben, denn alle Gebiete überschneiden sich stark.

Ein komplettes Beispiel dafür ist das Projekt NEPOMUK[13]. Inhalte (wie Notizen, Ideen, Sätze, Webseiten, Dateien usw.) sowie auch deren Beziehungen können semantisch und auf jedem Granuläritatsniveau beschrieben werden, und die Anwendungen können diese Daten nutzen. Verschiedene Werkzeuge, wie ein Wiki und eine graphische Schnittstelle mit Knoten, sind vorhanden, aber jedes andere Programm ist möglich; auch eines nur für Argumentation.

Neben zahlreichen derartigen Programmen existieren auch viele HyperText-Systeme (wie Xanadu, ENQUIRE, NoteCards, WWW und andere), die sehr gute Ideen für die Verknüpfung von Ideen beitragen (bidirektionale und typisierte Links, Transklusion, URIs, ...).

3.2.6 Gedankenkarten-Programme

Mind Mapping und Concept Mapping sind Ansätze, um Ideen durch Knoten und Kanten graphisch darzustellen. Es gibt viele Programme, die eine solche graphische Schnittstelle anbieten, z. B. vym, MindManager, Decision Explorer, kdissert oder FreeMind. Sie dienen zum Brainstorming oder zur Ideenorganisierung, gemeinhin aber nicht zur Diskussion. Üblicherweise legen sie mehr Wert auf die graphische Darstellung (also auf Farben, runde Ecke, Emoticons, ...) als auf die semantische Nutzung der Informationen.

Manche Ansätze (z. B. Concept Mapping) erlauben es, typisierte Beziehungen zu benutzen, d. h. sie stellen Tripel (gleich wie RDF) dar; dies ermöglicht die Formalisierung von Argumenten. Danach (im Punkt 3.3.4) werden einige Gedankenkarte-ähnliche Programme dafür beschrieben.

3.2.7 Annotationsprogramme

Da man eine beliebige Webseite nicht direkt modifizieren darf, wird ein zusätzliches System benötigt: entweder eine andere Webseite, ein Programm oder ein Webbrowser-Plugin.

Ein Webseite-Ansatz ist z. B. in e-marked[14] zu finden, obwohl noch mit unausreichenden Funkionalitäten. Man darf Text auswählen und kleine Kommentare hinzufügen, aber nicht antworten oder ausführlich diskutieren. Diese

[13] NEPOMUK: `http://nepomuk.semanticdesktop.org/`
[14] e-marked: `http://www.e-marked.com/`

Webseite muss natürlich als Vermittler zwischen dem Benutzer und dem Internet dienen, was nicht sehr effektiv ist. Andere Probleme gibt es wenn sich die ursprüngliche Webseite schnell ändert.

Eine praktische und erfolgreiche Anwendung eines Annotationssystems war die Diskussion über die Entwürfe der freien Softwarelizenz GPL in ihrer dritten Version[15]. Erfolgreich verfolgt wurde die Verbesserung des Lizenztextes, die Lösung der Unklarheiten und das Nachdenken über die Konfliktspunkte. Im Programm werden die Textabschnitte je nach Anzahl an Kommentaren in verschiedenen Farben hervorhoben, damit die provozierendsten Ideen schnell identifiziert werden können. Um zu vermeiden, dass die Benutzer gleiche Kommentare hinzufügen, gibt es den Link „zustimmen“. Es ist aber momentan unmöglich, größere Diskussion zu führen, denn Notizen dürfen anderen Notizen nicht antworten. Dieses auf Javascript geschriebene System war dann bzw. ist nützlich, aber es muss jedem Werk angepasst werden.

Annotea ist ein W3C-Standard für denselben Zweck; deshalb unterstützt er URIs, RDF, Dublin Core, und andere semantische Standards durchaus. Durch ein Programm (z. B. den Editor/Browser Amaya), einen Webbrowser-Plugin oder eine Webseite auf einem spezialen Webserver kann man beliebige Annotationen in einem Annotationserver speichern, unter anderen Notizen, Erklärungen oder Weblesezeichen.

Die typische Anwendung besteht darin, eine statische Webseite zu haben, die kurze Annotationen nur als Ergänzung hat. Für die Diskussion wird aber ein Modell gebracht, das mehr Wert auf die Notizen legt (anstatt auf den ursprünglichen Inhalt), und sie semantisch beschreibt und mit den anderen verknüpfen lässt. Das verhindert Annotea sowieso nicht, denn die Annotationen können beliebig komplex sein: eine Annotation kann sogar andere Annotationen annotieren, weswegen z. B. die Operationen der E-Mail (wie das Antworten) auch möglich sind.

Andere Anwendungen, die Annotea benutzen, stellen z. B. Kommentare aus einem Blog als Annotationen dar, und bessere Programme sind auch möglich. Leider sind die meisten Anwendungen noch Tests, und es gibt kein großes allgemeines Programm um beliebige Inhalte zu annotieren; erst recht weniger um logische Argumentation auf diese Weise zu führen.

Ein anderer Ansatz namens ClaimSpotter (SSM04) erlaubt die semantische Beschreibung des Inhaltes durch dessen Annotierung, vor allem für Forscher, die die Ideen aus wissenschaftlichen Arbeiten formalisieren wollen. Annotationen werden als Tripel (wie in RDF) betrachtet, und es gibt eine Ontologie mit festgestellten Beziehungen wie „supports“, „is consistent with“ oder „proves“. ClaimSpotter hilft bei der Identifizierung von Begriffen und Aussagen in Werken, was übrigens auch teilweise automatisch stattfinden kann. Andere Programme derselben Autoren (wie ClaiMaker und ClaimFinder; siehe Punkt 3.2.10 unten) helfen dabei, die Aussagen zu diskutieren.

[15] Diskussion über GPLv3: `http://gplv3.fsf.org/comments/`

3.2.8 Ontologien

Das Ziel vieler Diskussionen ist Wissen zu erzeugen und zu speichern. Ontologien sind ja ein Mittel, um dieses Wissen strukturiert zu speichern; deshalb könnte das Diskussionsverfahren zugleich mit dem Aufbau der Ontologie erfolgen.

DILIGENT (VPST05) bietet eine solche Infrastruktur für die kollaborative Arbeit zwischen Experten; wo nämlich viele Konflikte auftauchen, denn die Experten im Gebiet und die Informatiker beschreiben die Sachen anders. Die an einer lokalen Ontologie vorgenommenen Änderungen werden durch Argumente begründet, damit sie in die gemeinsame und verteilte Ontologie eingehen können. Auf diese Weise werden verschiedene begründete Sichten der Welt gegenübergestellt und anschließend ein Konsens angestrebt.

In diesem Zusammenhang wurden Tests durchgeführt um zu studieren, inwiefern Argumentation dabei helfen kann, und wie man die Diskussion am besten gestaltet. In einem ersten Experiment wurden die Teilnehmer erlaubt, in einem Chat frei zu diskutieren, wobei abschließend viele Probleme festgestellt wurden: sie gingen auf zu viele Teildiskussionen ein und verloren oft das Hauptthema; die Diskussion war zu schnell und nicht alle Leute konnten ihr folgen; es gab zwar einen Moderator, aber er konnte schwerlich teilnehmen; man durfte keine Abstimmung machen.

Das zweite Experiment wurde strenger: Die Grundbasis der Ontologie war schon am Anfang fertig, um philosophische Diskussionen zu vermeiden; es gab zwei Gesprächskanäle (einen für Themenvorschläge, Abstimmungen und um das Wort zu bitten, und anderen für Argumentenaustausch); die Argumente wurden auch auf das Grundsätzliche beschränkt (und zwar auf die im ersten Test bemerkten Argumentationstrukturen). Die Ergebnisse waren besser und wurden schneller erreicht, und die Diskussion verlor den Fokus nicht.

Es gibt aber wenige Werkzeuge, um Ontologien zu diskutieren. Ein erster Ansatz ist SWOOP, ein Ontologieneditor, der Annotea benutzt (siehe 3.2.7), um Kritiken und Kommentare als Annotationen zu zeigen. Die Aussagen der Benutzer werden nicht logisch formalisiert, aber Inkonsistenzen der Daten in der Ontologie werden dagegen automatisch mittels Logik begründet.

Die DILIGENT-Methodologie erweitert den Ansatz in SWOOP mit einem eigenen Werkzeug, das auf OntoStudie basiert und der Argumentation dient.[16] Es unterstützt das Anhängen von Für- und Gegenargumenten an den vorgeschlagenen Änderungen, und auch Abstimmungen. Die Notation betrachtet Aufgaben, Ideen und Argumente (und zwar Alternative, Gegenbeispiel, Beispiel, Evaluation/Begründung), aber jede Idee ist eine mögliche Änderung der Ontologie, d.h. eine genau beschriebene primitive Operation, die die Ontologie ändert. Das Programm (EvA, Evolution Annotater) kann daher die Argumente „verstehen“, denn es versteht Ontologien. Die Visualisierung hebt die widersprüchlichen Varianten hervor und zeigt auch die eigene Meinungen über die abgestimmten Ideen. Das Programm EvA befindet sich noch in der Entwicklung.

[16] Beschreibung des Prozesses in (EVS06); Details über die Ontologie in (TPSS05).

3.2.9 Methodologien und deren Implementierungen

Eine der bekanntesten Methoden, Diskussionen auf nützliche Art und Weisen zu behandeln, ist IBIS (Issue-Based Information System). IBIS wurde 1970 von Horst Rittel angefangen, um die sog. „bösartige Probleme“ diskutieren zu können. Dieses theoretische Verfahren beschreibt Probleme durch Fragen, Ideen und Argumente, wobei die Fragen das Wichtigste sind. Deshalb muss man fähig sein, sinnvolle Fragen zu stellen, und sich treffende Für- und Gegenargumente auszudenken.

Die Idee, eine Diskussion in Fragen, Ideen und Argumenten zu zergliedern, ist sehr grundsätzlich und deshalb wird IBIS (oder ähnliche) bei vielen Computerprogrammen unterstützt. Z. B. das Werkzeug QuestMap (früher gIBIS) unterstützt es explizit durch eine Schnittstelle, die jedes Element graphisch darstellt und sie mittels Beziehungen verknüpfen lässt. QuestMap erweitert außerdem das Modell mit anderen Beziehungen, wie *spezialisiert* (ein Element ist Teil eines anderes), *begegnet* (ein Thema will ein anderes widerlegen), und *erweitert* (eine Frage dient dazu, mehr über ein anderes Element zu erfahren). Es gibt auch zusätzliche Arten von Knoten: *Notiz*, *Referenz*, *Entscheidung*, *Nachricht*, *Sicht*.

IBIS beschreibt nicht, wie die Lösung eines Problems am Ende entschieden werden muss, aber QuestMap gliedert das Diskussionsverfahren in drei Teilen: *Divergenz* (freie Ideenfindung), *Konvergenz* (Einigung und gezielte Konservation), und *Entscheidung* (Wahl der Lösung). In der graphischen Darstellung werden die ausgenommenen Alternativen mit „(Namen)“ statt „Namen“ markiert, und die ausgewählten Lösungen einfach mit „*Namen“.

Der Entwickler von gIBIS (Vorgänger von QuestMap), Jeff Conklin, schrieb auch eine andere Methode zum Problemlösen, *Dialog Mapping*, die auch IBIS verwendet. (KSE03, Kap. 5) Diese und andere Methodologien sind aber Sammlungen von Ratschlägen für die Menschen, die sich mit bösartigen Problemen begegnen müssen, nicht jedoch für die Programmierer, die dieses Verfahren automatisieren wollen.

Eine andere Methodologie ist *Conversational Modeling* (Sel99), die IBIS erweitert, um kollaboratives Design und zentrales Informationsmanagement durch Argumentation und Hypertext zu erleichtern. Daraus wurde das Projekt *Compendium* gegründet und ein neues Programm (auch *Compendium* genannt) entwickelt[17], das durch eine Gedankenkarte-ähnliche Schnittstelle viele Verfahren unterstützt: vor allem Unternehmensprozesse, aber auch kollaboratives Schreiben, Konfliktlösung und andere.

Man kann Dokumente (z. B. E-Mails) in *Compendium* laden, und sie werden in Knoten und Verknüpfungen umgewandelt. Dann können sie mit anderen Ideen der Datenbank verbunden werden, damit neues Wissen aus diesen Daten durch die menschliche Zusammenarbeit erzeugt wird. Conversational Modeling beachtet die vorläufige Mischung zwischen formalem und nicht-formalem Wissen, die am Anfang der Ideenfindung stattfindet.

[17] Anscheinend basiert es auf das Programm *Mifflin*, von Verizon, aber man kann kaum Informationen über Mifflin im Internet finden.

Das Programm *Compendium* zeigt den Stand der Technik im Bereich des kollaborativen Problemlösens, denn es ist relativ neu und schließt die früheren Ansätze ein, indem es deren Probleme vermeidet. Sein *Conversational Modeling* enthält aber viele Weltmodelle und Verfahren, die für die anderen Anwendungsfälle nicht relevant sind.

3.2.10 Spezielle Programme für die Argumentation

Die vorher beschriebenen Ansätze unterstützen ja Diskussion, aber immer unter anderen Zielen. Es gibt leider sehr wenige Programme, die auf Diskussion per se abzielen, ohne sich auf anderen Kategorien (z. B. Wiki, Problemlösen, Entscheidungen, Debatte, Ontologien) zu beschränken. Das Programm, das wir suchen, hat *alles* rund um die logische Diskussion.

Ein sehr guter Ansatz dafür ist ClaiMaker+ClaimFinder[18]. Der Entwickler ist Simon Buckingham Shum, ein Autor des Buches „Visualizing Argumentation“ (KSE03), dessen Kapitel 9 übrigens diese Programme beschreibt[19]. Das Programm ClaiMaker vermag, Argumente in semantischen Netzen zu organisieren und dann ein Diskussionsthema als Gedankenkarte oder Tabelle in einer Webseite zu zeigen. Die Suche der Argumente ist vorhanden[20], und man kann sogar Suchen wie „Argumente gegen Schwangerschaftsabbruch“ ausführen. Jedes Argument hat eine Quelle (z. B. in der Literatur), die man mit einem Klick im Knoten sieht.

Die in ClaiMaker benutzte Ontologie verfügt über viele Arten von Beziehungen, von denen manche nur Synonyme mit kleinen Unterschieden sind, denn jede Person zieht eine andere Benennung vor. Obwohl die Nutzung von ClaiMaker deutlich schwieriger als z. B. diejenige eines Forums ist, ist es wohl bis jetzt der beste Ansatz für unsere Zwecke.

Andere interessante Systeme sind:

- Belvédère[21], das Diagramme aufbauen lässt und über einen Chat verfügt (VAK99).
- Rationale (früher „Reason!Able“), mit einfacher Schnittstelle.
- SIBYL[22], um das Entscheidensverfahren zu beschreiben, und zwar mit einer Notation namens DRL, Decision Representation Language.
- CROCODILE[23], eine ausführliche Arbeit über problembasiertes Lernen und ein Programm, das u. a. Konflikte aufspüren kann.

[18] ClaiMaker: `http://claimaker.open.ac.uk/`

[19] Kapitel 9 ist auch hier verfügbar: `http://kmi.open.ac.uk/projects/scholonto/docs/VizNetArg2002.pdf`

[20] Man kann die Suche (ClaimFinder) hier ausprobieren: `http://claimfinder.open.ac.uk/`

[21] Belvédère: `http://acm.org/sigchi/chi95/proceedings/intpost/dds_bdy.htm` . Die eigentliche Webseite (`http://advlearn.lrdc.pitt.edu/belvedere/`) ist tot aber die Version aus 2001 kann in `http://web.archive.org/` immer noch nachgelesen werden.

[22] SIBYL: `http://doi.acm.org/10.1145/99332.99344`

[23] CROCODILE: `http://elib.tu-darmstadt.de/diss/000086/`

- ACT, und viele mehr; siehe (KSE03), Seite 40.

Auch sehr beachtenswert ist „Convince me“[24], das auch Argumente graphisch darstellt (und in XML speichert), aber auch über eine feine Kontrolle der Plausibilität der Argumente verfügt. Man kann nicht nur Abschätzungen angeben (wie z. B. „schwer zu glauben“, „viele Leute sind dagegen“, „ich glaube das nur 30%“, „das ist nur ein Standpunkt“ usw.), sondern es benutzt diese Werte, um mitzurechnen, inwiefern jedes Argument stimmt. Dafür verwendet es ein Model für den Gedankengang, namens ECHO, das auf Grundregeln basiert (z. B. je mehr Evidenzen für ein Argument es gibt, desto plausibeler ist es) und auch Methoden aus der Statistik benutzt.

Es gibt eine Simulationsphase, und danach kann gesehen werden, was ECHO „denkt“. Wenn das unseren Ideen nicht entspricht, dann kann man die Gewichtsverteilung ändern, indem man z. B. wählt, wie „skeptisch“ oder „tolerant“ das Model sein muss.

Dieses Java-Programm hat GPL als Lizenz, ist einfach herunterzuladen und zu bedienen, ist gut dokumentiert, und enthält auch viele genaue Ideen und Ratschläge über die logisch richtige Nutzung der Hypothesen (sogar auch viele Übungen für das Umschreiben eines Textes in eine Liste von Hypothesen). „Convince me“ hat viele Methoden für die Wissensverarbeitung, es ist aber kaum mehrbenutzerfähig, hat wenige Darstellungsmöglichkeiten und hat wenige Optionen für die ausführliche Suche in großen Argumentennetzen; dies sind aber Gebiete, die z. B. ClaiMaker/ClaimFinder sehr gut abdeckt. Eine Kombination beider Ansätze wäre deshalb sehr interessant.

3.3 Formale Notationen

Auf vielen Diskussionssystemen ist notwendig, die argumentativen Strukturen mit Hilfe einer Notation zu formalisieren. Diese Formalisierung beschreibt die *Grundelemente* der Diskussion, z. B. Ideen, Für- und Gegenargumente, Verweise oder Verknüpfungen, sowie eventuell deren Semantik (Regeln, anwendbare Operationen usw.).

Ein solches formales Datenmodell hilft dabei, den Menschen die Daten auf eine einfache Weise darzustellen. Die Tatsache, dass die Vorschläge und Alternativen explizit dargestellt werden, vereinfacht den Konsens und verringert die Absichtsunterschiede zwischen Teilnehmern. (KSE03, S. 14) Darüber hinaus ermöglicht eine Notation, dass ein Rechner das Argumentationsschema teilsweise verstehen kann.

3.3.1 Frühere Ansätze

Das Darstellen von Diskussionen mittels Diagramme entstand weit früher als die Computer-Ära, was davon ausgehen lässt, dass keine Computer dafür notwendig sind. Sowohl die Rechtswissenschaft als auch die Philosophie verfügen

[24] „Convince me“: `http://www.soe.berkeley.edu/~schank/convinceme/`

bereits über verschiedene Instrumente um Argumente zu beschreiben und zu analysieren. Beispielsweise:

- Wigmore-Diagramme, 1913.[25]
- Toulmin-Diagramme, 1958.[26]
- Bayes'sche Netze.

Ähnliche Ideen für die Darstellung von Wissen und Begründungen sind diejenigen von Vannevar Bush für den Memex (1945), oder von Douglas Engelbart (1962); eher als Notationssysteme haben sie uns ihre Weitblick über Hypertextsysteme vorgestellt. Da diese Ideen den Wurzeln der Diskussionssysteme gehören, können mehr Informationen darüber in (KSE03), Kap. 1 (*The Roots of Computer Supported Argument Visualization*) gefunden werden, sowie auch Diagramme und Zitate.

3.3.2 Einfache Notationen

Argumente und Beziehungen können sehr präzise beschrieben werden, was aber die Notation erheblich erschwert. Die von vielen Programmen gewählte Lösung besteht in der Nutzung einer sehr einfachen Syntax, die die Ideen typischerweise folgendermaßen aufteilt:

- **Aufgabe** zu erledigen, eventuell in Form einer Frage.
- **Idee**: mögliche Problemlösung.
- Argument: **Fürargument** oder **Gegenargument**.
- (eventuell) **externe Quelle** mit mehr Informationen.

Mit einer solchen Syntax lassen sich z. B. Knoten in einer Gedankenkarte einen Typ zuordnen, der die Rolle jeder Idee beschreibt.

Dieses System wird z. B. von der Methodologie **IBIS** (bereits in 3.2.9 beschrieben) benutzt, sowie auch von deren Varianten (wie gIBIS oder rIBIS). Der IBIS-Wortschatz[27] enthält nämlich die Typen **Aufgabe**, **Einstellung** und **Argument**, und verfügt außerdem über etliche Beziehungen (um Knoten zu verknüpfen): *generalises*, *specialises*, *replaces*, *questions*, *is_suggested_by*, *responds_to*, *objects_to*, *supports*.

ibaw[28] ist eine noch vereinfachtere Notation, die die gleichen Begriffe wie IBIS anbietet, jedoch keine Beziehungen zwischen Ideen beschreibt.

Eine Syntax mit wenigen Datentypen ist für den Benutzer einfach und vorteilhaft, denn Benutzer haben Probleme mit Programmen, die komplexen Syntaxen benutzen: Nicht nur müssen sie die Syntax lernen, sondern die Syntax lenkt sie auch vom Thema ab. (KSE03, S. 38)

[25] Beschreibung auf (KSE03), S. 6.
[26] Beschreibung auf (KSE03), S. 9 und genauere Beispiele auf (KSE03), S. 76.
[27] Eine RDF-Version des IBIS-Wortschatzes kann auf `http://dannyayers.com:88/xmlns/ibis/index.htm` geholt werden.
[28] ibaw: `http://xam.de/2006/02-ibaw.html`

3.3.3 Andere Notationen

Es gibt sehr viele Notationen für die Einteilung von Argumenten. Es ist nicht das Ziel dieser Arbeit, solche Notationen zu beschreiben oder zu vergleichen, denn das wäre zu schwierig und ergäbe keine nützlichen Ergebnisse. Alle sind nämlich subjektiv; daher ist es sehr schwierig zu begründen, ob eine besser oder schechter ist als die andere. Lediglich einige Merkmale werden erwähnt; mehr Informationen darüber stehen z. B. in (KSE03), S. 18 zur Verfügung.

Beispiele für Notationen sind: CDSCMC (*Collaborative Discourse Structures in Computer Mediated Group Communications*), *Rhetorical Structure Theory*, *argumentative design*, *gIBIS*, *Design Space Analysis*, *Decision Representation Language*, und diejenige jedes Programms bzw. jeder Methodologie.

Beispiele für Typen von Ideen: *Widerstand*, *Für*, *Gegen*, *Alternative*, *Abstimmung*, *Detail*, *Spezifikation*, *Sammlung*, *Zusammenfassung*, *Aufgabe*, *Beobachtung*, *Glied*, *Verknüpfung*, *Weg*, *Inferenz*, *Ausarbeitung*, *Ausschluss*, *Vermutung*, *Extrapolation*, *Gegensatz*, *Hintergrund*, *Vorbereitung*, *Datum*, *Bejahung*, *Zusicherung*, *Widerlegung*, *Unterstützung*, *Abschluss*, ...

Zusätzlich gibt es verschiedene *Werte*, die dargestellt werden könnten: *Wichtigkeit*, *Gültigkeit*, *Wunsch*, *Durchführbarkeit* u. v. a.

Beispiele von Beziehungen: *hat Bezug*, *beeinflusst*, *ist Teil von*, *ist ähnlich*, *abhängig von*, *ist Beobachtung von*, *Antithese*, *unbeabsichtigter Grund*, *beabsichtiger Grund*, *Begleitumstand*, *Bedingung für*, *Bewertung*, *ist Evidenz für*, *begründet*, *motiviert*, *ist Absicht für*, *neue Formulation*, *unbeabsichtigtes Ergebnis von*, *beabsichtiges Ergebnis von*, *fasst zusammen*, *löst*, *benutzt*, *verbessert*, *unvereinbar mit*, *sagt voraus*, *teilt Gründe mit*, *ist Beispiel von*, ...

Diese präzisen Notationen sind zwar nicht abzulehnen, sie gehören jedoch nicht in das *grundsätzliche* Diskussionsprogramm, das wir in dieser Arbeit betrachten, sondern in ein spezialisierte Programm auf einer höheren Komplexitätsebene.

3.3.4 Argumentendarstellung durch Programme

Es gibt mehrere Programme für die Darstellung von Argumentationsdiagrammen (ein Netz von verbundenen Argumenten). Sie sind nützlich um verschiedene Notationen auszuprobieren und um deren Ausdrucksmöglichkeiten zu vergleichen, aber zusätzlich helfen sie dabei, ein Thema besser zu verstehen (also Anwendungsbeispiele 1 und 2).

Araucaria (RMRW06) ist eines der Systeme mit dem reichsten Angebot an Eigenschaften. Es nimmt Argumente aus einem Text heraus (mit Hilfe des Benutzers) und speichert sie in einem XML-Dialekt, AML: *Argument Markup Language*, der einfach und erweiterbar ist (es gibt nämlich Programme, die Araucaria erweitert haben). Die Argumente werden in Baumform gespeichert (nicht Netz) und sowohl Beziehungen als auch Widersprüche zwischen ihnen können gespeichert werden.

Araucaria unterstützt mehrere Diagrammsysteme (auch Wigmore und Toulsin) und speichert Informationen über das benutzte Schema und die Arten der

Argumente, mit Hilfe von Begriffen aus der Rhetorik (*ad hominem*-Angriff, Expertenmeinung, das Ganze für den Teil, ...). Das macht von Araucaria ein mächtiges Programm zur Argumentenanalyse. Auf seiner Webseite[29] gibt es zusätzlich eine große Datenbank von schon beschriebenen Argumenten, die gesucht werden können. Darüber hinaus ist Araucaria kostenlos, plattformunabhängig und seine Lizenz ist die GPL.

Andere Programme dienen zur Argumentendarstellung. Auf (Har05) gibt es eine genaue Kritik und Beschreibung von Araucaria, Argutect, Athena Standard, Inspiration und Reason!Able, und zwar unter Berücksichtigung der Benutzerfreundlichkeit und der Ausdrucksmöglichkeiten für die Darstellung von philosophischer Gespräche.

3.3.5 Zusammenfassung der Notationen

- Es gibt viele Notationen (typischerweise erstellt jedes Programm eine Neue). Das liegt wohl an folgenden Punkten:
  - Fast jede Technologie kann benutzt werden, um Argumente zu beschreiben, z. B. Gedankenkarte.
  - Viele Vorschläge sind subjektiv, denn sie umfassen nicht nur eine Art und Weise, Argumente zu beschreiben, sondern auch Methoden um Diskussionen zu halten, um Probleme zu lösen, um Personen Rollen zuzuordnen, etc.
  - Allein die Art und Weise, Argumente zu beschreiben, ist nicht einzigartig.
- Die von mehreren Programmen geteilte Notation besteht darin, Ideen in *Aufgabe*, *Pro*, *Contra* einzuteilen. Eventuell auch *externe Quelle*.
- Eine genaue Notation ist kompliziert, schwierig zu nutzen, und schreckt die Benutzer ab. Sie kann dennoch die Information sehr gut beschreiben.

Auf Grund dieser Subjektivität glauben wir, dass das Thema „Notation“ intrinsisch schwierig ist, und dass es nicht vermeiden lässt, dass neue Syntaxen auftauchen.

3.4 Zusammenfassung der verwandten Ansätze

Es gibt grundsätzliche und weit benutzte Kommunikationsmittel (nämlich E-Mail, Forum, Chat und Wiki), auf die man nicht verzichten will. Diskussionssysteme sollten sich einigermaßen auf diese Mittel anpassen.

Für E-Mail haben wir den Ansatz Semalan gefunden. Als Forum für Argumentation wählen wir DebateMapper.

[29] Araucaria: `http://babbage.computing.dundee.ac.uk/`

Für Wikis und Chat haben wir aber kein effektives Programm gefunden. Da diese Mittel sehr locker in Bezug auf die für Diskussion wichtigen Eigenschaften (wie Gesprächsfäden) sind, gelingt diese Aufgabe nur, wenn man sich auf Konventionen beschränkt; das Programm kann jedoch dasselbe sein.

Diese Konventionen sind manchmal sehr komplex, da die Argumentation ebenfalls komplex ist. Den Benutzern gefällt es jedoch nicht, komplexe und ausführliche Notationen zu lernen bzw. zu benutzen.

Als Programm um das Argumentationsschema zu speichern und graphisch darzustellen haben wir Araucaria gefunden (zugleich mit seiner Darstellungssprache AML, seinen Argumentationsschemen, seiner Argumentendatenbank und Suchmaschine). Viele andere Programme verwenden eine Gedankenkarten-ähnliche Darstellung, dennoch reicht ein bloßes Gedankenkartenprogramm nicht aus.

Praktische Anwendung der Annotationstechnik haben wir auf dem Kommentarprozess der GPL3-Lizenz gefunden. Das Programm ClaimSpotter benutzt sie, um Texte mit ihren Aussagen semantisch zu annotieren.

Wir haben auch mehrere Projekte gefunden, die Argumentation auf allen Mitteln ermöglichen wollen. Z. B. Compendium, oder die Sammlung der Projekte ClaiMaker+ClaimFinder+ClaimSpotter. Diese Systeme können aber noch erweitert oder kombiniert werden, was im nächsten Kapitel dargestellt werden soll.

4 Neue Ideen für bessere Systeme

Die meisten beschriebenen Systeme sind nicht auf die Diskussion angepasst. Hier wird beschrieben, welche Verbesserungen vorgenommen werden könnten, und wie das ideale System sein sollte.

4.1 Erste Beschreibung der Anforderungen

4.1.1 Abhängigkeit von der Arbeitsweise

Manche Methodologien erzwingen oder schlagen bestimmte Arbeitsweisen vor, beispielsweise:

- Die Lösung eines Problems besteht aus 7 Phasen: *Situationsanalyse*, *Problemeingrenzung*, *Alternativen aufzeigen*, *Lösungsauswahl*, *Chancen und Risiken abschätzen*, *Einführung und Umsetzung*, *Nachbearbeitung und Lernen*.

- Entscheidungsprobleme lassen sich in einzelne Komponenten zerlegen, z. B. *Handlungsalternativen*, *Umwelteinflüsse*, *Konsequenzen von Handlungsalternativen und Umwelteinflüssen*, *Ziele und Präferenzen des Entscheiders*. (EW03, Kap. 2)

Es gibt viele solcher Verfahren, vor allem in der Wirtschaftswissenschaft. Nur über „Entscheiden“ oder „Problemlösen“ kann man tausende Bücher finden. Das bedeutet, diese Verfahren sind subjektiv.

Manche Diskussionssysteme wurden zwecks einer bestimmten Methodologie programmiert. Ein grundsätzliches Diskussionssystem sollte aber nicht auf diese Weise entwickelt werden, denn Argumentation ist die Grundlage zu vielen unterschiedlichen Anwendungen (siehe Kapitel 2), die nicht durch eine einzelne Methode betrachtet werden können.

Vergleiche man diese Situation mit den grundsätzlichen Systemen, die üblicherweise zur Diskussion benutzt werden: Wiki, E-Mail, Chat, Forum. Diese erzwingen keine bestimmte Arbeitsweise, sondern sie sind offen zu allerlei Nutzungsmethoden.

Natürlich sind einige der Arbeitsweisen besser oder produktiver als andere. Aber alle sind subjektiv, und deshalb sollten sie nur als Empfehlungen, Anleitungen oder Richtlinien bleiben. Denn um ein grundsätzliches Diskussionssystem zu schaffen, das an allen Anwendungsszenarien taugt, müssen die Arbeitsweise und Methoden des Benutzers am wenigstens geändert werden.

4.1.2 Erwartungen an das Programm

Hier folgt eine grobe Liste von Anforderungen, die wir (als Benutzer) für ein besseres Programm für die Argumentation möchten.

- Beiträge der Benutzer aufnehmen, und zwar in irgendeiner Form (sowohl grob als auch genau, sowohl wahr als auch falsch, etc.).
- Erlauben, diese Beiträge in „Aussagen“ zu zerteilen.
- Den „Wahrheitswert“ jeder Aussage kennen, berechnen und zeigen.
- Die Beziehungen zwischen Aussagen kennen und „verstehen“.
- Ähnliche Argumente suchen und sie zusammen zeigen (z. B.: *für* und *gegen*, Widersprüche, ...).
- Auch den Zustand der ganzen Diskussion zeigen (schon entschieden, fertig, noch aktiv, ...).
- Bequem für den Nutzer sein: Erlauben, Diskussionen zu folgen; sehen, was andere Benutzer machen; Auflistungen und Statistiken sehen, ...
- Argumente in verschiedenen Formaten speichern, importieren und exportieren.

4.1.3 Gestalt des Programms

Ein Diskussionsprogramm kann auf verschiedenen Ebenen laufen: nur eine graphische Benutzeroberfläche, nur eine Webseite, nur eine Inferenzmaschine, ... Hier wird es grob entschieden, *wie* das Programm aussehen sollte.

Ein wichtiges Merkmal ist, dass das Programm **mehrbenutzerfähig** sein muss. Die Vorteile sind offensichtlich: Ein mehrbenutzerfähiges Programm kann auch von einem Einzelnen benutzt werden. Es ist außerdem was unseren Anwendungsbeispielen entspricht, denn die meisten Diskussionen werden zwischen mehreren Personen durchgeführt. Es gibt Ansätze, in denen ein so genannter *Moderator* die Debatte befolgt, beschreibt und kontrolliert (z. B. „Debate Maping"), aber dies ist auf noch mehr Benutzer nicht gut skalierbar. Wir entscheiden daher, dass die Benutzer selbst ihre Beiträge im Programm schreiben sollten.

Neben Mehrbenutzerfähigkeit sollte das Programm **Internet** benutzen, das heißt, es sollte nicht nur für mehrere Benutzer desselben Rechners verfügbar sein (wie z. B. ein Programm für die Darstellung von Diagrammen). Heutzutage ist es sehr wichtig, das Potential des Internets, des semantischen Webs und der sozialen Netze zu profitieren, unter anderen weil sie geographisch verteilt sind; siehe (DSW06) oder andere Studien (z. B. (EVS06)) für mehr Argumente. Ein Diskussionssystem ist mit den Ideen der Hypertextsysteme wie Memex oder WWW sehr stark verbunden (alle bestehen darin, Informationen zu verlinken und semantisch zu beschreiben).[30]

Ein solches Programm muss über eine *oder mehrere* **graphische Schnittstelle** für die Benutzer verfügen, d. h., es muss mehr als nur eine Infrastruktur, Notationssprache oder Inferenzmaschine sein. Die Schnittstelle spielt eine große Rolle am Programm, und sollte enifach zu bedienen sein, denn der typische Benutzer kennt nur die Schnittstellen aus den typischen Programmen (E-Mail, IRC, Forum, Wiki, ...), die nicht sehr komplex sind.

Das Programm muss außerdem die existierenden **Technologien kombinieren**. Denke man an die Themen, die betrachtet werden müssen: semantische Beschreibung der Argumente, Wissensverarbeitung, Suche, Schnittstelle, Kollaboration usw. Und nebenbei wollen wir effektive und besonders bequeme Methoden. Angesichts dessen können wir davon ausgehen, dass es sehr schwierig wäre, ein einzelnes Programm zu schaffen, das alles zugleich erfolgreich kann. Es gibt bereits spezialisierte Programme (siehe Kapitel 2), deshalb ist es eine gute Idee, sie zu kombinieren. Ein modularer Ansatz ist sehr wirksam, denn z. B. mehrere graphische Schnittstellen (Text, Webseite, Gedankenkarte, Baum, ...) können benutzt werden, und alle werden Zugriff zu den selben Informationen haben.

Andere gewünschte Eigenschaften des Programms sind:

- freie Lizenz (wie GPL oder BSD). Es ist sehr wichtig, dass irgendjemand das Programm korrigieren bzw. erweitern *darf*, sogar im Fall, dass seine Entwicklung aufgegeben werde.
- Kompatibilität mit Standards (wie XML), denn die Kooperation zwischen Programmen ist von großer Bedeutung.

[30] Übrigens hatte Tim Berners-Lee (Erfinder des WWWs) schon 1989 ein Diskussionssystem programmiert, um die Informationsverlust zu vermeiden, die durch Themaabkommen und nutzlose persönliche Angriffe in Diskussionen stattfindet. „Ein Gesprächswerkzeug, das die Logik einer Diskussion aufnimmt". (BL99, S. 160)

- Plattformunabhängigkeit, d. h., dass es sich nicht nur zu einem Betriebssystem einschränkt.
- einfach zu finden im Internet. Dies mag unwichtig klingen, aber es gab mehrere Programme, die wir nicht testen konnten, weil sie im Internet nicht stehen oder weil es kaum Informationen darüber im Internet gibt.

4.2 Implementierungen mittels gegenwärtiger Technologien

Ein so weites Ziel wie dieses kann auf mehrere Arten verstanden werden, und hat natürlich Bezug zu vielen existierenden Technologien. Beispielsweise sind unter unseren Wünschen:

- freies Verknüpfen beliebiger Ideen, wie in einem Gedankenkarten-Programm.
- beliebige Inhalte bearbeiten zu dürfen, wie in einem Wiki.
- verschiedene Themen und Gespräche zugleich führen zu können, wie in einem Forum.
- Gespräche als Nachrichten und Antworten zwischen Benutzern zu betrachten, wie in einer Mailing-Liste.
- Informationen strukturiert zu speichern, wie in einer Datenbank oder Ontologie.

Zunächst beschreiben wir einige Ideen, wie ein Diskussionssystem durch Kombination existierender Ansätze verstanden werden kann.

4.2.1 Text mit einer Begründung für jeden Satz

Ein auf viele Mittel (Bücher, Wikis, Foren, Webseiten usw.) anwendbarer Ansatz besteht darin, einen Text als eine Menge von Aussagen (Sätzen, die falsch oder wahr sein können) zu verstehen, und dann jedem Satz eine Begründung hinzuzufügen. Das heißt, für jeden Satz wird erklärt, warum er wahr ist.

Dieses System wird teilweise bei wissenschaftlichen Arbeiten und bei Wikipedia benutzt, nämlich in Form von *Quellen*. Diese kommen auf zwei Arten vor: entweder als Fußnoten (z. B. [1]), oder in einem gesonderten Abschnitt „Referenzen“ am Ende (falls sie sich auf den ganzen Artikel beziehen).

Jedoch gibt es einige Probleme damit:

- Umfang: Wenn [1] am Ende eines Satzes steht, ist nicht klar, was genau durch die Quelle 1 begründet wird: nur der letzte Satz, der ganze Absatz, oder einige der zuletzt erwähnten Ideen.
- Genauigkeit: Manchmal wird ein Buch zitiert, ohne zu sagen, wo genau (auf welcher Seite) die Information sich befindet, die als Begründung benutzt werden muss.

- Verfügbarkeit: Oft werden Ressourcen zitiert, auf die nicht jeder Zugriff hat; daher ist es sehr schwierig, eine Begründung zu bestätigen.
- Eignung: Jemand kann ein Werk zitieren, das die Aussage eigentlich nicht unterstützt, oder das sie lediglich erwähnt ohne sie zu begründen.

Ein System, das diese Punkte verbessert, muss deshalb:

- ermöglichen, einen Text in Sätze zu trennen, und jedem eine oder mehrere Referenzen zuzuordnen, und zwar auf eine *eindeutige* Art und Weise.
- sicherstellen, dass jede Referenz genügend Informationen der Ressource angibt: Werk, Seite, Abschnitt, Satz, ...
- sicherstellen, dass sowohl eine URI, als auch mindestens eine gültige URL der Ressource angegeben werden; entweder zum Inhalt oder zu einer anderen Ressource, die den Inhalt beschreibt.
- ermöglichen, bestimmten Referenzen Etiketten oder Kommentare anzuhängen, z. B. „bestätigen“, „geprüft“, „fehlende Begründung“ usw. So lassen sie sich außerdem die Referenzen, die *begründen*, und diejenigen, die nur auf *zusätzliche Informationen* verweisen, einfach unterscheiden.

Obwohl z. B. auf Wikipedia Referenzen oft angegeben werden, wäre es jedoch nicht sehr geeignet, eine Referenz per Satz anzugeben, denn dann wären die Artikel sehr lang und mit Referenznummern beladen. Ein besseres System muss die Lesbarkeit und das Aussehen des Textes beachten und gleichzeitig das semantische und eindeutige Annotieren irgendeines Satzes erlauben, sogar der Sätze aus den schon mit anderen Systemen geschriebenen Texten.

Um dies zu ermöglichen ist es nötig, beliebige Sätze im Text referenzieren zu können. Es gibt Auszeichnungssprachen, die Referenzsysteme anbieten; das Referenzsystem SiSU[31] beispielsweise, ordnet jedem Absatz eine Nummer zu, die immer beibehalten wird, unabhängig vom Format, in welchem publiziert wird (z. B. Buch oder Webseite). Außerdem ermöglicht es die semantische Beschreibung des Inhaltes (mit Dublin Core und RDF) und die Suche von Abschnitten durch SQL und sowohl mittels semantischer als auch textueller Kriterien.

Das Zitiersystem (mit dem Absatz als Einheit) zusammen mit der relationalen Datenbank machen SiSU zu einer mächtigen Sprache, die sogar in Wikipedia benutzt werden könnte, um Aussagen (anstatt nur Artikel) verknüpfen zu können. Das Problem dieses Systems ist, dass in einem Wiki der Inhalt sich schnell ändert, und damit auch die Reihenfolge und Einordnung der Absätze. Um Aussagen zu referenzieren wird deshalb ein stabiles Referenzsystem benötigt.[32]

Wenn ein Satz anstatt eines Absatzes benutzt wird, wird ein System mit genaueren Beziehungen geschaffen (die dennoch in natürlicher Sprache bleiben). Aber da das Zerlegen jeder Begründung in Sätze sehr aufwändig ist, ziehen

[31] SiSU: `http://www.jus.uio.no/sisu/`

[32] Wie z. B. der von Giuseppe Peano vorgeschlagene Stil, den Alfred North Whitehead und Bertrand Russell im Werk *Principia Mathematica* benutzt haben.

wir vor, *Absätze* als Einheiten zu benutzen. Die andere Möglichkeit ist noch vorhanden, weil der Benutzer immerhin Absätze schreiben kann, die nur einen Satz beinhalten.

Ein einfaches Wiki mit einem guten Referenzsystem, das erlaubt, jedem Satz eine Begründung zuzuordnen, wäre schon eine erhebliche Verbesserung, vor allem um die Wahrhaftigkeit eines Textes zu begründen (Anwendungsbeispiele 3 und 4).

4.2.2 Aussagenwiki

Im letzten Abschnitt wurde ein System betrachtet, bei dem ein Artikel (z. B. eine Wikiseite) Sätze hat, die mit Quellen begründet werden müssen. Wenn alle Sätze richtig begründet werden, dann kann der Artikel im Ganzen als wahr angesehen werden.

Daraus ergibt sich, dass ein Artikel einen *Wahrheitswert* hat, und deshalb kommt er der Rolle einer Aussage gleich. Eine neue Idee ist, jeden Satz einer Wikiseite nicht mit einer Quelle zu begründen (wie auf Wikipedia), sondern mit einer anderen Seite desselben Wikis, die die Begründung enthält.

Dafür reicht es, ein einfaches Wiki zu benutzen, aber mit der Konvention, dass *jede Seite eine Aussage darstellt*. Deshalb sollten nicht Seiten wie „Euthanasie“ oder „Kritiken gegen Euthanasie“ angelegt werden, sondern z. B. „Euthanasie muss in Deutschland legalisiert werden“, „Euthanasie ist natürlich“, „Das Christentum erlaubt die Euthanasie“, „Das Christentum erlaubt die Euthanasie nicht“ usw.

Jede Seite behauptet etwas, und zwar mit dem Zweck, diese Behauptung zu begründen und zu unterstützen. Dabei kann auf beliebige Aussagen verwiesen werden (mit den entsprechenden Diskussionen), oder sogar auf Definitionen, Quellen und externe Seiten, alles noch mit der üblichen Freiheit eines Wikis.

Dieses System legt aber manche Probleme offen, wenn es mit einem herkömmlichen Wikiprogramm programmiert wird:

- Die Seitennamen wären zu lang.
- Die Inhalte werden wiederholt, d. h.: Dieselbe Information befände sich unter verschiedenen Titeln, da es verschiedene Formulierungen für dieselbe Aussage geben kann.
- Um zu erfahren, ob ein angegebenes Argument wahr ist, muss man dem Link folgen und eine neue Seite lesen, eventuell rekursiv oder in Schleifen (z. B. A benutzt B als Begründung, und B benutzt A, aber keine gibt mehr Quellen).
- Vandalismus.

Als Lösungen schlagen wir folgende Erweiterungen eines Standardwikis vor:

- Jeder Seite eine Identifikationsnummer oder einen Beinamen zuweisen, um die Zwischenverlinkung zu vereinfachen.

- Die Suchmaschine so verbessern, dass sie mit dem Verlinkungsprozess integriert ist; eine Suche vor dem Einfügen jedes neuen Arguments hilft dabei, Wiederholungen zu vermeiden
- Jeder Seite semantische Information hinzufügen, die den Stand der Diskussion beschreibt (z. B.: Entscheidung getroffen, zweifelhaft, vertraute Quellen, ...); diese Informationen müssen verbreitet werden, damit andere Seiten diese Daten kennen.
- Das Kollaborationsmodell zwischen Benutzern so ändern, damit die nutzlosen Argumente unbeachtet bleiben.

Diese Probleme und Ideen werden später erklärt.

Das „Aussagenwiki“ ist eine Idee, die noch zu entwickeln ist, und wir rechnen damit, dass neue Wikis in diesem Stil auftauchen werden. Projekte wie Wikipedia bieten Definitionen einzelner Begriffe an, aber ein Aussagenwiki beschreibt –auf einer höheren Abstraktionsebene– die *Beziehungen zwischen Begriffen*. Die Tatsache, dass es sehr einfach auszuführen ist (es ist nämlich ein bloßes wiki mit Konventionen), macht es für viele Gebiete nützlich, vor allem wenn es wenige Benutzer gibt (z. B. in einem persönlichen Wiki).

4.2.3 Gedankenkarte

Eine Gedankenkarte ist eine sehr gute Darstellungsform, denn sie vereinfacht das Verknüpfen beliebiger Elemente, und kann mittels Farben und Formen die Arten der Elemente verdeutlichen. Damit sie aber als Basis eines Diskussionssystems benutzt werden kann, werden Konventionen und Erweiterungen benötigt.

Ein Knoten speichert ein Argument. Viele Systeme erlauben, sowohl Für- (normalerweise grüne Knoten) als auch Gegenargumente (rote) zu benutzen; das trägt zu einem besseren Verständnis bei, es ist aber eine Einschränkung, denn eine Aussage (z. B. „Ohne Kernel kann man einen Rechner nicht benutzen“) ist nicht „für“ oder „gegen“ *per se*, sondern diese Einordnung ist abhängig des diskutierten Themas. Wenn das Ziel ist, Argumente wieder nutzen zu können, sollten wir deshalb diese Einteilungen vermeiden, und stattdessen nur eine lokale *Rolle* der Argumente erlauben.[33]

Die Struktur der Gedankenkarte ermöglicht, ein beliebiges Argument mit beliebigen anderen zu verbinden. Damit jede Verbindung nützliche Information enthält, müssen sie unterschieden werden; beispielsweise wäre es möglich zwischen A und B eine Beziehung wie „A weil B“, „A wenn nicht B“ oder „A gleich B“ zu haben. Aber eine Verknüpfung ist auch etwas Diskutierbares, und deshalb sollte das System erlauben, auch Kanten mit anderen Knoten zu verknüpfen, z. B. indem jede Verbindung auch als Knoten dargestellt wird.[34]

Genauso wie das Aussagenwiki, erfordert eine Aussagenkarte sowohl semantische Informationen auf jedem Knoten als auch die Verbreitung der neuen Daten

[33] Das System ClaiMaker kann das; wie früher im Punkt 4.3.5 erklärt wurde.

[34] Die Sprache *Conceptual Data Structures* (CDS, Teil des Projekts NEPOMUK) funktioniert auf diese Weise.

auf die jeweils verknüpften Knoten. In diesem Sinne funktionieren Wiki und Gedankenkarte gleich. Später wird erklärt, welche Informationen gespeichert werden sollten.

Die Darstellung mittels Graphen benötigt ausreichend Platz. Viele Gedankenkarten-Programme versagen oder werden unangenehm, wenn man mit großen Datenmengen arbeitet, denn diese sammeln sich unübersichtlich am Bildschirm an. Ein Diskussionssystem muss das *Zoomen* implementieren, d. h.: eine Teilsicht des Ganzen ermöglichen, in der neue Elemente erst allmählich auftauchen, während man das Zoomsniveau erhöht.[35]

Die Funktionsweise einer Gedankenkarte ist für die Benutzerkollaboration nicht besonders geeignet, denn das Programm ist schwieriger zu nutzen, als die typischen Systeme im Internet, und darüber hinaus ist es schwierig, sich ständig ändernde Informationen visuell zu verfolgen.

Von daher ist es besser, dass die Gedankenkarte nur eine Darstellungsform der Daten ist, aber dass die Informationen auch auf anderen Formaten gespeichert und verfügbar sind.

Zusammenfassend schlagen wir eine Gedankenkarte vor, wo die Argumente allgemein gültig sind (d. h. ohne Themen-Einteilung), die Kanten mit anderen Knoten verknüpft werden können, die Knoten semantische Informationen haben, und wo Zoomen und Suchen möglich sind. Da dieses System jedoch nicht alle unserer Anwendungsbeispiele erfüllen kann, suchen wir weitere Ansätze.

4.2.4 Semantisches Aussagenforum

Eine andere Art und Weise, ein Diskussionssytem zu verstehen, ist, es als ein typisches Diskussionsforum zu betrachten (mit Benutzern, Themen, Fragen, Antworten, festen Kategorien etc), in welchem die Benutzer beliebige Kommentare schreiben können, nebst einer formalen Beschreibung des Inhaltes des Kommentars.

Dies ist zwar eine aufwändige Aufgabe für die Benutzer, ergibt aber viele Informationen, die von dem Rechner verarbeitet werden können. Natürlich ist die Nützlichkeit der Informationen durch die Fähigkeiten des Programms bezüglich der Texterkennung und künstlicher Inteligenz eingeschränkt. Hier spielt nämlich die Inferenzmaschine die entscheidende Rolle. Das Programm könnte –aus dieser Informationsbasis– mögliche Widersprüche suchen, Argumentketten bilden, Fehlschlüsse aufspüren, oder die Wichtigkeit jedes Arguments berechnen.

Der entscheidende Faktor, der die Schwierigkeit der Methode bestimmt, ist die Präzisionsebene der Sprache, die zum Formalisieren dient. Ein einfacher Ansatz erfolgt durch eine einfache Argumentationsontologie, die grundsätzliche Beschreibungen erlaubt, z. B. „für 12, gegen 13, 13=>14“ um die Beziehungen zwischen nummerierten Kommentaren zu zeigen. Diese Hinweise können auch zwischen im Text mit näturlicher Sprache stehen, beispielsweise zwischen Etiketten oder am Kopfzeile; genauso wie auf Webseiten oder Wikiseiten, nur in

[35] Innerhalb des Projekts NEPOMUK befindet sich das Gedankenkartenprogramm iMapping, das Zoom weitgehend benutzt. Siehe `http://imapping.info/`

kleineren Bereichen: Die Metainformation gilt nämlich für jeden Kommentar anstatt für jede Seite.

Eine andere Variante dieses Ansatzes wäre, dass der analysierte Text direkt die natürliche Sprache wäre. Nach statischen Textanalysenmethoden kann der Rechner Stichwörter und Begriffe einfach aufspüren, denn sie kommen in einer gezielten Diskussion wiederholt vor.[36] Andere Fachwörter der Argumentation, wie „nicht“, „weil“, „ich stimme zu“ usw. können auch ohne große Probleme erkannt werden und können dabei helfen, einen Kommentar in genauere Teile zu zerlegen.

Das Verstehen der natürlichen Sprache ist jedoch sehr schwierig und die Ergebnisse sind zu ungenau, was für eine Diskussion nicht geeignet ist, denn die kleinen Details sind wichtig. Damit der Rechner die Eingaben besser verstehen kann, kann auch eine konstruierte Sprache benutzt werden: z. B. Esperanto hat eine Grammatik, die sehr einfach zu analysieren ist, und trotzdem wird sie von Menschen gesprochen.

Eine Sprache wie Esperanto ist aber genauso mehrdeutig wie natürliche Sprachen. Eine bessere Plansprache, um Aussagen zu beschreiben, wäre Lojban: Sie basiert auf Prädikatenlogik, sie hat eine eindeutige Grammatik, die durch Rechner analysierbar ist[37], und sie berücksichtigt alle Probleme und Nuancen der Logik, beispielsweise: Verknüpfungen zwischen Sätzen und Elementen (*und*, exklusives und inklusives *oder*, verschiedene Arten von *weil*, ...), verschiedene Methoden zur *Negation*, mehrere Systeme um Sätze oder Wörter genau zu referenzieren, Fehlerkorrigieren, ... Außerdem ist jeder Satz eine Aussage per se, und hat deshalb einen Wahrheitswert.

Das Verstehen der Plansprachen ist ohnehin kompliziert; außerdem besteht das Problem darin, dass die Leute solche Sprachen erst lernen müssen (z. B. Lojban-Sprecher zu finden ist zwar möglich aber schwierig). Deshalb ist es besser für den Ansatz „Forum mit semantisch beschriebenen Kommentaren“, eine reduzierte und einfache Sprache zu wählen, die nur die Hauptmerkmale beschreibt; welche zu wählen, wird im Punkt 4.3 erklärt.

4.2.5 Zusammenfassung und gemeinsame Merkmale

Wir haben einige Ideen vorgeschlagen, wie ein Diskussionsystem durch die Verbesserung und Zusammenstellung existierender Ansätze implementiert werden könnte. Damit wurden einige geteilte Punkte betrachtet, die die Programme brauchten:

- eine Notation um Aussagen zu beschreiben. Einschließlich:
  - ein Referenzsystem, das eine beliebige Idee genau zitieren kann.
  - eine oder mehrere Methoden, um Informationen einzugeben.

[36] Programme wie ClaimSpotter wenden diese Methode an, um dem Benutzer das Annotieren von Aussagen aus einem Text zu erleichtern. (SSM04, S. 6)

[37] Es stehen sowohl BNF-Grammatik als auch Verifizierung- und Übersetzung-Werkzeuge zur Verfügung.

- semantische formale Informationen jedes Arguments (z. B. Wahrheitswert).
- Informationsverarbeitungsregeln (z. B. für Deduktion).
- eine Suchmaschine, die besonders für die Argumentensuche angepasst ist.
- ein gutes Kollaborationsmodell zwischen Benutzern.
- eine oder mehrere Methoden, die gespeicherten Daten darzustellen.

Diese Arbeit will ein einzelnes und grundsätzliches Diskussionsprogramm gestalten, das alle Anwendungsbeispiele aus Kapitel 2 erfüllt. Von daher sagen wir aus, dass ein universelles Diskussionssystem eine Lösung zu jedem Punkt auf der vorherigen Liste aufweisen soll. Zunächst wird jeder Punkt betrachtet und neue Ideen angegeben, um jeden der Punkte effektiv zu behandeln.

4.3 Notation

4.3.1 Gewünschter Notationstil

Wie schon in der Sektion 3.3 erwähnt wurde, gibt es viele subjektive Notationen. Gewünscht in dieser Arbeit ist aber ein System, das bei vielen Anwendungsfällen funktioniert (siehe Kap. 2) und deshalb haben wir entschlossen, keine bestimmte Arbeitsweise zu erzwingen (Sektion 4.1.1). Aus diesen Gründen wählen wir eine einfache Notation, die möglicherweise vielen Fällen gemeinsam ist.

Deren Genauigkeitsebene ist ein schwieriges Thema:

- Eine schwierige Notation verursacht dem Benutzer Probleme, denn er muss sich Mühe geben, um zu erfahren ob eine Idee z. B. *Ursache*, *Begründung*, *Anlass* oder *Bemerkung* ist.
- Eine zu einfache Notation ergibt wenig für das Programm nützliche Informationen. Obwohl z. B. „ist Fehlschluss“, „schlecht definiert“, „unbegründet“, „durch_Experten_widerlegt“ alle *Gegenargumente* sind, wäre es nützlich, jedes anders behandeln zu können.

Da keine Option perfekt ist, ist es notwendig, beide zu unterstützen. Das wird durch ein erweiterbares Datenmodell geschafft, d. h.: Die Benutzer verfügen über einfache Ideen und Beziehungen, die spezialisiert werden können, ohne dass die frühere Semantik verloren geht. Diese Idee wird z. B. in CDS (VH06)[38] benutzt, und ist für ein Diskussionssytem sehr geeignet.

Deshalb wählen wir eine einfache und erweiterbare Notation.

[38] Ein Beispiel aus CDS: eine fundamentale Beziehung ist „ist ähnlich wie“, aber es gibt zwei abgeleitete Beziehungen: „ist gleich wie“ und „ist Alias von“. Ebenfalls hat „ist Alias von“ eine Unterbeziehung „ist Alias mit Ersatz von“. Der Benutzer kann neue abgeleitete Beziehungen an irgendeiner Stelle der Hierarchie anlegen, z. B. „ist kommplett gleich“, „ist gleichartige Kopie“, „ist visuell ähnlich“, „ist Referenz auf“, ...

4.3.2 Prosa vs. Graphen

Der Prosatext ist die natürliche Art und Weise, Argumente darzustellen, aber es ist für die computerunterstütze Visualisierung nicht sehr geeignet. Argumente aus einem Text herauszuziehen ist etwas sehr Schwieriges und Subjektives, denn es gibt viele richtige Methoden, es zu machen. Eine Argumentenkarte zu verstehen ist dagegen einfacher, weil alles schon aufgeklärt wurde. (KSE03, S. 100)

Die Notation muss einfach, schnelle und verlässlich sein; was die Prosa nicht erfüllt. Andere Nachteile der Prosa sind: Sie ist aufeinanderfolgend (die Reihenfolge der Sätze ist schon festgestellt), hat keine semantischen Informationen (z. B. „die Meinung, die ich hiermit erläutere, ist ein Fürargument“), und sie profitiert nicht von den Möglichkeiten wie Formen und Farben, mit welchen das menschliche Gehirn sehr gut umgehen kann. (KSE03, S. 102)

Gleichwohl ist es hilfreich, dass die Notation die Muster und Strukturen aus der natürlichen Sprache nachahmt, denn auf diese Weise wird sie einfacher zu nutzen sein (KSE03, S. 187, Punkt 1). Die Notation muss eine schnelle Handlung der Daten ermöglichen, ohne auf jede Kleinigkeit achten zu müssen. (KSE03, S. 146) In Prosa aber ist es so, dass sogar kleine Änderungen wie z. B. an der Reihenfolge der Sätze oder an den Benennungen der Begriffe sehr aufwändig und fehleranfällig sind.

Man kann immerhin nicht darauf verzichten, argumentativen Prosatext zu behandeln, deshalb ist eine Formalisierung (ein Prozess für das Herausziehen von Argumenten) notwendig. Wir schlagen diesen Prozess vor:

1. Aus einem Prosatext werden Argumente herausgezogen und in einer Notation dargestellt
2. Die Arbeit erfolgt ausschließlich mittels dieser Notation
3. Falls benötigt, werden die verarbeiteten Daten nochmals in Prosa umgeschrieben

Punkt 1 können Programme zur semantischen Annotierung (wie ClaimSpotter) ausführen; es ist immer subjektiv und aufwändig. Natürlich kann man diesen Schritt weglassen, wenn man vermag, Argumente direkt in der gewählten Notation zu schreiben. Punkt 2 ist am schwierigsten auszuführen; er wird heutzutage von Gedankenkartenprogrammen unterstützt, die alleinig mit ihrer graphischen Notation arbeiten. Punkt 3 wird manuell in die Tat umgesetzt, weil der Mensch besser als der Computer ist, um richtigen und kohärenten Prosatext zu erzeugen (in der Regel; es gibt immer Ausnahmen).

Jedenfalls hängt man noch von natürlicher Sprache ab, wenn man eine formale Sprache benutzt. Jedes Diskussionssystem muss deswegen sämtliche Probleme der natürlichen Sprache kennen (z. B. mehrdeutige Definitionen, Synonyme um dasselbe anzudeuten, wenig Präzision, zu viele Informationen zusammen, ...), und damit entsprechende Verfahren anbieten, um diese Probleme zu umgehen.

Die in einem Diskussionssystem benutzte semantische Notation muss also sehr unterschiedlich zu Prosa sein (teilweise um die Prosa zu ergänzen), aber dennoch muss es einen Übersetzungsprozess geben.

4.3.3 Betrachtung der Begriffe

In jedem Diskussionssystem oder jedem Szenarium, wo man mit Aussagen arbeitet, gibt es zwei Hauptdatentypen: Begriffe und Beziehungen zwischen ihnen. Die **Begriffe** sind *Sachen* oder statische *Ideen*, z. B. mein Haus, der beste Etat, die von einem Benutzer geschriebene Antwort, die Wichtigkeit, die Existenz, das Jahr 2007, ich, ... Die **Beziehungen** sagen eine Verknüpfung zwischen mehreren Begriffen aus, z. B. „Dieses Buch ist interessant“, „UFOs existieren“, ... Es ist außerdem immer möglich, eine Aussage als Begriff zu verwenden: „Die Tatsache, dass UFOs existieren“.

In jeder Aussage werden also Begriffe verknüpft. Eines der Ziele eines Diskussionssystems sollte es sein, die in einer Diskussion erwähnten Begriffe zu verdeutlichen. Das ist äußerst wichtig, denn üblicherweise werden die Probleme zwischen Benutzern nur dadurch verursacht, dass sie unterschiedliche Ansichten über dieselben – mehrdeutigen – Begriffe haben.[39]

Ein Diskussionsprogramm sollte fähig sein,

- die in einer Aussage benutzten *Begriffe* zu erkennen und markieren.
- einen Definition- und Aufklärungsprozess der Begriffe anfangen.

Dieser zweite Schritt ist eine Designphase: Aus einigen ungenauen Wörtern wird sinngebende Information erzeugt. Wir schlagen vor, dass ein gutes Diskussionssystem *den Definitionsprozess auch als Diskussion betrachten soll*, d. h. mit Aussagen, Benutzerkollaboration, Themaabkommen wo benötigt, etc. Das Diskussionssystem erlaubte deshalb nicht nur über Beziehungen zwischen Begriffen diskutieren, sondern auch über den Sinn der Begriffe.

Das betrifft alle Fälle; zum Beispiel:

- wenn man ein sog. „bösartiges Problem“ betrachtet, wobei das diskutierte Thema unklar ist.
- wenn man Alternativen vergleicht (z. B. Texteditor), denn man muss erstens formalisieren, *was* ein Texteditor ist, welche Merkmale er hat, usw.
- wenn man subjektive Adjektive benutzt, wie z. B. „Dieser Film ist gut“; dort ist es nämlich möglich, über „Wann ist ein Film “gut”?“ zu diskutieren.

[39] Zum Beispiel kann jemand behaupten: „Eine Arbeit vor dem Studium an der Universität ist sehr wichtig“ und eine andere Person: „Es ist ein Nachteil, ohne Universitätsausbildung eine Arbeit anzufangen“. Obwohl beide eine „Arbeit“ erwähnen, vielleicht beziehen sie sich auf unterschiedlichen Begriffe; z. B. der Erste auf „eine kurze und dem Studium begleitende Arbeit“ und der Zweite auf „eine langfristige und hauptamtliche Arbeit mit Blick auf die Zukunft“.

Dieser Ansatz (Diskussionen über Begriffe zu erlauben) ist besser als den Benutzer verpflichten, seine Aussagen eindeutig zu formalisieren. Beispielsweise ist die Aussage

- „Eine Arbeit vor dem Studium ist vorteilhaft“.

sehr mehrdeutig. Man könnte diese Idee ziemlich genauer beschreiben:

- „Eine Arbeit von 3 bis 10 Monaten, mit höherem Gehalt als der Mindestlohn, im selben Fach wie das Studienfach, mit weniger als 20 Stunden pro Woche, und zugleich mit einem Universitätsstudium von nicht mehr als 30 Stunden wöchentlich... verursacht, dass 95% der Studenten der deutschen Universitäten eine Arbeitstelle in weniger als 3 Monate bekommen, an der man mehr Geld verdienen kann als ein Student, der diese Bedingungen nicht erfüllt.“

Aber anstatt von fordern, dass die Aussagen in diesem „Gesetzbuch“-Stil geschrieben werden, ziehen wir vor, dass noch mehr zwar mehrdeutig aber nützliche Diskussionen entstehen, sogar wenn die Themen sich vermehren. Nützlich wäre Diskussionen um die Begriffe „Arbeit vor dem Studium“ und „vorteilhaft“ aus dem ursprünglichen Satz zu definieren, aber andere interessante Themen könnten auftauchen, z. B. „Was versteht man unter “guter Arbeit”“, „Welche Arten von Arbeiten vor der Universität gibt es“, ...

Da Definitionen zwangsläufig in jeder Diskussion vorkommen, ist es äußert wichtig, sie gut zu behandeln. Definitionen durch den Austausch von Argumenten zu ermöglichen ist ein einfacher und mächtiger Ansatz.

4.3.4 Granularität

Systeme wie IBIS verlangen, die **Aufgaben** in Form von Fragen zu formulieren, z. B. „Was sollten wir tun, um die Gewinne der Firma zu erhöhen?“, und **Einstellungen** in Form von Behauptungen über mögliche Alternativen, z. B. „Die Preise steigern“, „Kosten erniedrigen“, „Den Chef entlassen“, ...

Die Nutzung von Fragen hat Vorteile: Die Diskussion ist auf ein bestimmtes Thema gezielt (nämlich die Frage zu beantworten), und wenn man sich eine Frage stellt, tauchen einige Möglichkeiten auf. Es hat aber auch Nachteile:

- Keine Antwort wird erzeugt: Es ist möglich, eine sehr große Karte mit Informationen und Ideen über die Frage „Können Computer denken“[40] zu haben, und dennoch bleibt die ursprüngliche Frage offen.

- Es ist sehr schwierig, gute Fragen zu stellen. Die IBIS-Anleitungsbücher erklären oft, welche Arten von Fragen nicht passend sind (z. B. „Sollten wir die Preise steigern?“) und wie sie umgeschrieben werden müssen („Was sollten wir tun?“ „Die Preise steigern“).

[40] Dies ist eine typische Debatte, für die es umfangreiche Argumentenkarten schon gibt. Auf `http://www.macrovu.com/CCTGeneralInfo.html` ist die Karte zu finden, und auf (KSE03), Kap. 8 gibt es eine lange und detailierte Erklärung über das Verfahren, eine solche große Karte zu bauen, und über die Probleme, die unterdessen auftauchten.

- Eine Frage ist keine Aussage. Damit eine Debatte aktiv läuft muss man Meinungen und klare Argumente beitragen, nicht aber Fragen. Unter Fragen können viele Arten von Fehlschlüsseln liegen; typisch sind Beispiele wie „Sollte das Ministerium damit aufhören, Geld für den Krieg auszugeben?“ wenn das Ministerium niemals Geld dazu zugewiesen hat. In diesem Fall ist es falsch, an die Antwort zu denken.

Wir wollen deshalb mit **Aussagen** als Hauptelement arbeiten, wobei die Fragen nur als bloße **Titel** der Diskussionen verstanden werden können. Es ist aber nicht nötig, dass sie in Form von Frage stehen; ein Titel wie „Ideen um die Gewinne zu steigern“ reicht schon aus.

Ein schwieriger Punkt sind **mehrfache Aussagen** wie „Das Klonen ist ethisch richtig, aber es kann schlimme emotionale Folgen haben“. Es ist besser; das in zwei Sätze zu zerlegen. Da wir aber nicht erwarten können, dass jeder Benutzer irgendeinen Text in Aussagen zerteilen kann, muss ein gutes Diskussionssystem ermöglicht, eine schon eingetragene mehrfache Aussage in Teile zu zerlegen. Das ist wichtig um Paradoxien zu vermeiden, wie z. B. „Der König Frankreichs ist kahlköpfig“, das übrigens zwei Sachen impliziert: dass es einen König Frankreichs gibt, und dass er kahlköpfig ist.[41]

Außerdem muss die Nutzung anderer Informationsformate außer Text in Betracht gezogen werden: Wenn man z. B. über Verbesserungen an Bildern diskutiert (Anwendungsbeispiel 3), dann ist vielleicht jedes Argument nicht ein Satz, sondern eine Änderung des Bildes. Andere Gebiete haben Fachnotationen (Formeln, Tabellen, Datenstrukturen, ...). Dieses Problem sollte aber in spezialisierten nachkommenden Studien gelöst werden.

Was entschlossen wurde, ist, zwei Sachen zu speichern: Aussagen in Form von Text und mit entscheidbaren Wahrheitswert (zwischen wahr und falsch), und eventuell Titel (aber nicht nur Fragen).

4.3.5 Anzahl und Typen der Beziehungen

Wenn Argument A mit Argument B irgendwie verknüpft werden, ist es möglich, den Beziehungstyp anzugeben. Die einfachen Diskussionssysteme benutzen eine kleine Wahl an Typen:

- In einem Wiki oder einer Webseite einfach „A linkt auf B“.
- In einem Forum oder einem E-Mail-System wird auch „A antwortet auf B“ gespeichert.
- In einfachen Argumentendarstellungssystemen „A ist Fürargument für B“, „A widerspricht B“ und wenige mehr.

Andere Systeme haben mehr:

[41] Dieses Paradoxon und dessen Wahrheitswert und Verneigung wurde u. a. von Bertrand Russell vorgeschlagen und analysiert.

- Die Notation ClaiMakers hat z. B. 36 Arten von Beziehungen, und bleibt sowieso generisch.[42]

- Andere Programme benutzen präzise Notationen, die feine Unterschiede darstellen können, z. B. zwischen „A ist der natürliche Grund von B“, „A ist die Motivation für B“, „A impliziert logisch B“, „A ist die Justifizierung von B“, ...

Wir glauben, dass *die Beziehungsart nicht anzugeben* schon gut genug ist; eigentlich funktioniert das ganze World-Wide-Web so. Wie schon im Punkt 4.3.1 vorgeschlagen, wäre am besten eine erweiterbare Notation, die erlaubt, „A hat einen Bezug mit B“ zu sagen, und danach diese Information zu spezialisieren.

Darüber hinaus ist zu berücksichtigen ((KSE03), S. 37), dass jede Beziehung auch Attribute aufweisen kann (z. B. *Glaubwürdigkeit*), und dass der Wortschatz keine bestimmte Benennung erzwingen sollte, sondern verschiedene *Dialekte* anbieten (manchen Benutzern ist z. B. „widerlegt“ bequemer, anderen dagegen „ist unvereinbar mit“). (KSE03, S. 188, Punkt 6).

Die bestimmte Argumentationsontologie eines Programms wäre noch etwas Subjektives und geht sowieso über die Vorhaben dieser Studienarbeit.

4.3.6 Informationseingabe

Häufig haben Benutzer viel Inhalt, der schon in anderen Formaten steht; deshalb muss ein Programm die Möglichkeit anbieten, um diesen Inhalt in die Notation zu verwandeln. (KSE03, S. 139). Der von Programmen wie ClaimSpotter gefolgte Ansatz um Argumente aus einem Text manuell herauszuziehen ist schon dazu geeignet.

Die Eingabe neuer Informationen wird typischerweise durch graphische Schnittstellen durchgeführt, die ähnlich den Gedankenkartenprogrammen sind: Man wählt einen Knoten (sei er Idee, Argument oder andere) aus, dann einen anderen Knoten, und durch die Maus zeichnet man eine Linie zwischen den beiden. Wenn die Notation einfache ist, können aber noch andere Eingabemethoden benutzt werden (z. B. ein bloßer Texteditor mit Autovervollständigung) um Argumente zu beschreiben.

Mit einem einfachen **Referenzsystem** (z. B.: Jeder Satz hat eine ihm zugeordnete Nummer) könnte der Benutzer diesem Vorgehen folgen, um eine Aussage hinzufügen:

- Die Argumente und Begriffe, die in der neuen Aussagen vorkommen müssen, suchen.

- Eine Liste all dieser Elemente im Bildschirm sehen, zusammen mit einer Identifizierungsnummer für jedes.

[42] Man kann die Liste in (KSE03), S. 190 finden. Es wird empfohlen, diese Syntax zu studieren, denn sie beschreibt alle üblichen Beziehungen auf eine natürliche Weise, indem sie den in früheren Ansätzen gelernten Kompromiss zwischen Benutzerfreundlichkeit und Genauigkeit respektiert.

- Einen Satz wie „<24> aber nicht <28>, weil <3> und <2>“ schreiben, wo sich jede Nummer auf eines der Elemente des letzten Schritts bezieht, und die Stichwörter Teil der gewählten Notation sind.
- Gegebenenfalls einen noch beschreibenderen Text hinzufügen.
- Diese Information wird hinzugefügt und wird auch eine Identifizierungsnummer bekommen.

Immerhin ist es sehr wichtig, über mehrere Methoden zur Informationseingabe zu verfügen. In diesem Punkt sind viele der typischen Systeme für Wissensmanagent schon weit entwickelt, und deshalb wäre es nützlich, eines als Schnittstelle für ein Diskussionssystem zu benutzen.

4.4 Semantische Informationen jeder Aussage

In Wikipedia sind einige Seiten mit einer Bewertung versehen (z. B. gut, exzellent, ohne Quellen, subjektiv, ...). Ähnlicherweise ist es wichtig, dass jede Information im Diskussionssystem einige Attribute enthält. Im Unterschied zu Wikipedia muss aber die Nutzung der Attribute standarisiert werden, um zu ermöglichen, dass die Programme auf immer die selbe Art und Weise die Informationen nutzen können.

4.4.1 Wahrheitswert

Ein Argument muss zumindest über einen **Wahrheitswert** verfügen, der zeigt, ob die Information als falsch oder wahr angesehen werden sollte. Diese Idee ist aber sehr breit und kann auf mehrere Art und Weisen implementiert werden. Einige Möglichkeiten sind:

- Der Ww. (*Wahrheitswert*) könnte eine Nummer zwischen 0 und 1 sein.
- Der Ww. kann einfach „ohne Gegenargumente“ oder „mit Gegenargumenten“ sein.
- Der Ww. kann bzw. sollte trinär sein, im Sinne dass ein Argument „wahr“, „falsch“ oder „unentschieden“ sein kann (und nicht nur „wahr“/„falsch“).
- Gegebenfalls könnte der Ww. einfach ein vom Benutzer eingegebener subjektiver Wert sein, damit der Ww. eines Arguments vom bestimmten Benutzer abhängt.
- Bzw. könnte der Ww. dem Konsens der Benutzer entsprechen, indem alle Benutzer genau denselben Wert sehen.
- Der Ww. kann auch durch den Rechner aus den anderen Daten entschieden werden.

- Es gibt viele Methoden, einen „wahr“-Zustand für ein Argument zu erhalten, z. B.: eine anerkannte Quelle anzugeben, das Argument durch Deduktion mittels anderer wahrer Argumente zu begründen, das Argument trotz Versuche nicht wiederlegen zu können, usw.

- Ebenfalls kann ein Argument vielerlei verfälscht werden, z. B.: durch ein Gegenbeispiel, nach seiner Feststellung als Fehlschluss, durch viele Gegenstimmen, usw.

- Ein unentschiedenes Argument kann sich gleichfalls in mehreren Zuständen befinden: zweifelhaft, wahrscheinlich wahr, schlecht formuliert, es fehlen Quellen, veraltet, es hängt von nicht mehr wahren Prämissen ab, es kann weder als wahr noch als falsch bewiesen werden, usw.

Ein guter Ansatz sollte all diese möglichen Begriffe *zugleich* anbieten, und zwar mittels einer erweiterbaren Hierarchie, auf die selbe Weise wie die Methode für die im Punkt 4.3.1 erklärte Einordnung der Beziehungen.

Die Idee, jedem Argument Informationen zuzuordnen, ist nützlich um andere Daten in anderen Argumenten auszurechnen. Zum Beispiel: Wenn sich das Argument A auf die Wahrheit von B und von C stützt, dann sollte eine Änderung des Wahrheitswertes Bs den Wahrheitswert As verändern: A könnte z. B. als „zweifelhaft“/„überprüfen“ markiert werden, oder sogar einen neuen Wert berechnet bekommen.

Diese automatische Verbreitung muss schließlich das Hauptthema (etwa den Titel) der Diskussion erreichen. Der Wahrheitswert einer beliebigen abstrakten Idee im Programm sollte daher kurz zeigen, was die Leute darüber denken. Die Verbreitungsregeln können dann genauer zeigen, **warum** dieser Wahrheitswert erhalten wurde.

4.4.2 Gültigkeit der Argumente

In einem Diskussionssystem muss es entschieden werden, ob ein bestimmtes Argument zu einem bestimmten Thema beschränkt ist, oder ob es dagegen allgemeingültig (d. h. bei allen möglichen Themen nutzbar) ist.

Beispielsweise betrachte man eine Diskussion über moderne Kommunikationsmittel, in der u. a. über Handys besprochen wird; jemand kritisiert sie mit der Aussage „Die Mobilfunkantennen verursachen Krebs“, was die Diskussion zu diesem Thema lenkt. Tage später, in einer anderen technischen Diskussion, die durch dasselbe System unterstützt wird, diskutieren Ärzte über die möglichen Ursachen des Krebses. Jemand behauptet: „Die Mobilfunkantennen verursachen Krebs“. Sollte das Programm erlauben, genau dasselbe Argument in beiden Zusammenhängen zu benutzen?

Bei manchen Programmen muss eine Idee in jedem neuen Zusammenhang neu formuliert werden, wobei auch Links zwischen Diskussionen möglich sind. Andere führen die Idee *Rolle* eines Argumentes ein: anstatt eine Idee in „für“, „gegen“, „Fehlschluss“ usw. einzuteilen, ist diese Benennung nur eine vom aktuellen Zussamenhang abhängige Eigenschaft. Beispielsweise könnte ein Argument

in einer Diskussion als „Problem zu lösen“ angesehen wird, und dasselbe in einer anderen Diskussion nur als „Vermutung“. (KSE03, S. 188, Punkt 5).

Wir glauben, dass der geeignete Ansatz darin besteht, jedes Argument als allgemeingültig zu betrachten. Das heißt, der Satz „Die Mobilfunkantennen verursachen Krebs“ besitzt einen Wahrheitswert, der unabhängig des Zussamenhangs diskutiert werden kann. Er **ist** kein Fürargument, keine Bemerkung, keine Widerlegung, etc., sondern nur eine Aussage. Es ist ohnehin sehr nützlich, die **lokale Nutzung** dieser Aussage, d. h. ihre Rolle, beschreiben zu können.

Deshalb schlagen wir vor, im Gegensatz zu einem Baum-Ansatz (in dem jedes Argument abhängig des aktuellen Themas ist) einen Netz-Ansatz (in dem jedes Argument mit jedem anderen verknüpft werden kann) vorzuziehen.

4.5 Wissensnutzung

4.5.1 Berechnung des Wahrheitswertes

Der Wahrheitswert jedes Arguments muss automatisch berechnet werden, und zwar unter Berücksichtigung der Beziehungen, die es mit anderen Argumenten hat.

Werde beispielsweise das Argument A durch G begründet (G impliziert A), und B ebenfalls durch die drei X, Y und Z. Dann müssen viele Regeln eingehalten werden, z. B.:

- Wenn G falsch ist, dann kann A nicht wahr sein.
- Wenn G wahr war aber als falsch bewiesen wurde, dann sollte sich der Wahrheitswert As zu „zweifelhaft“ verändern.
- Damit B wahr sein kann, müssen all seine Grundlagen (X, Y, Z) wahr sein, und diese liste muss vollständig („suffizient“) sein.
- Wenn M das Gegenteil Ns ist, dann müssen sie gegenseitige Wahrheitswerte haben.

Es gibt auch die Mögchlichkeit, dem Benutzer zu erlauben, diese Werte manuell zu verändern. Desgleichen könnten auch nur einige numerische Werte eingegeben werden, damit das Programm den Rest berechnet.[43]

Die Tatsache, dass der Wahrheitswert automatisch von dem Programm berechnet wird, ist für den Benutzer eine Hilfe, denn dies spart die Mühe, nachzuschlagen, ob jeder Satz wahr oder falsch ist.

4.5.2 Zusammenstellung ähnlicher Argumente

In einer komplexen Diskussion wird es oft sehr ähnliche Beiträge geben. Neben einer Aussage wie z. B. „Hühner können fliegen“ gibt es zahlreiche Varianten:

- Neuformulierung: „Das Huhn ist flugfähig“.

[43] Gleich wie die Funktionsweise des Programms „Convince me“.

- Verneinung: „Hühner können nicht fliegen“.
- Steigerung: „Einige Hühner können fliegen“, „Viele Hühner können fliegen“, „Fast alle Hühner können fliegen“, ...
- Andere taxonomisch bezogene Argumente: „Alle Vögel können fliegen“ (Abstraktion), „Die Hühner aus meinem Dorf können fliegen“ (Spezialisierung), ...
- und viele mehr.

Ein Diskussionssystem muss sich mit der schwierigen Aufgabe auseinandersetzen, diese zusammenhängenden Argumente immer zusammen zu halten. Wenn die Beziehungen semantisch beschrieben wurden (z. B. mittels der Argumentationontologie), dann kann das Programm diese Information benutzen:

- Wenn A und B Synonyme sind, dann können beide durch den selben Knoten dargestellt werden, und ihre zugehörigen Informationen können gemischt werden.
- Wenn A die Verneinung Bs ist, dann können sie ebenfalls zusammen dargestellt werden, indem auf B „für“ bzw. „gegen“ überall getauscht wird.
- Wenn mehrere Argumente A unterstützen, aber mit verschiedenen Sicherheitsgraden, dann muss Modallogik verwandt werden. Aus dem Satz „Die meisten Hühner können fliegen“ folgt z. B. „Einige Hühner können fliegen“; jedoch folgt *vielleicht* nicht, „Alle Hühner können fliegen“.
- Wenn A eine Spezialisierung Bs ist, dann gilt alles, was für B gültig ist, auch für A. Wenn A eine Ausnahme ist, dann stimmt B nicht mehr.

4.5.3 Neue Formulierungen

In einer Diskussion ist es sehr üblich, dass sich die Ideen im Laufe der Zeit verändern, oder dass sie allmählich besser definiert werden.[44] Deshalb muss man damit rechnen, dass sich die eingetragene Information höchstwahrscheinlich verändern wird:

- Manchmal kann ein Argument verbessert werden. Z. B. ändert sich die Aussage A_1 „Alle Primzahlen sind ungerade Zahlen“ zu A_2 „Alle Primzahlen außer 2 sind ungerade Zahlen“. Dann sind die Kritiken an A_1 nicht mehr auf A_2 anwendbar.
- Eventuell kann ein Argument verschlechtert werden, z. B. von A_2 zu A_1.

[44] Beispielsweise haben die sog. *bösartigen Probleme* (siehe Punkt 2.5.1) die Eigenschaft, dass sich die Definition des Problems verbessert und entwickelt, indem man an der Lösung arbeitet.

Wenn viele Informationen auf ein Argument A_1 bezogen wurde, und dann dieses sich zu A_2 ändert, dann sind diese Informationen nicht mehr gültig, weil sie sich noch auf den alten Zustand beziehen. Sie zu löschen ist zu drastisch. Da A_1 und A_2 eigentlich zwei verschiedene Aussagen sind, schlagen wir folgenden Ansatz:

- Die Informationen nicht zu löschen, sondern A_1 und A_2 gleichzeitig behalten, indem A_2 als Aktualisierung von A_1 markiert wird..
- Dem Benutzer die –manuelle– Aufgabe bieten, die auf A_1 bezogenen Informationen so zu aktualisieren, damit sie für A_2 nochmals anwendbar sind.

4.5.4 Beimischung von Themen

Stets kann es den Fall geben, dass dasselbe Thema zweimal getrennt diskutiert wurde, und dass es deshalb gewünscht werde, die Informationen beizumischen. Das ist z. B. in Wikipedia typisch[45] und erfordert viel Arbeit, denn man muss wiederholte und eventuell widersprechende Informationen finden, und sorglich eine neue Version erfassen, die die Informationen beider Seiten beinhaltet.

In Wikipedia ist diese Aufgabe schwierig, weil Text in Prosa benutzt wird; aber für ein Diskussionssystem ist sie einfacher: um Themen A und B zu mischen, können die Aussagen jedes Themas einfach zusammengestellt werden, da sie allgemeingültig sind (d. h. sie hängen nicht von Annahmen an). Wenn A und B widersprüchliche Informationen hatten, dann ist es kein großes Problem, das die Beimischung verhindert. Widersprüche sind nämlich ein wesentlicher Teil von Diskussionen, und werden deshalb beim Diskussionssystem unterstützt und verwaltet. Die Widersprüche, die nach einer Mischung entstehen, können deshalb als normale Widersprüche betrachtet werden.

Eine Möglichkeit, zwei Aussagen A und B beizumischen, ist sie nicht zu verändern, sondern eine Beziehung hinzufügen, die besagt, dass sie gleich sind; darüber hinaus muss das Programm die Informationen beider immer zusammen zeigen. Auf diese Weise wird keine Information verloren.

4.5.5 Suche nach Fehlern und Fehlschlüsseln

Es gibt viele Zustände in einer Diskussion, die in der Praxis nicht möglich sind, z. B. weil sie der Wahrheit nicht entsprechen. Neben den schon erwähnten Einschränkungen bezüglich des Wahrheitswertes (z. B. A und nicht-A können nicht zugleich wahr sind) gibt es auch Fehler an den Informationen, Inkonsistenzen, und allerlei Fehlschlüsse; beispielsweise betrachte man den Fall mit den Aussagen „A stimmt, weil B“ und „B stimmt, weil A“: beide Aussagen sind zwar begründet, aber in einer Schleife.

[45] Unter anderen wegen ortographischer Varianten wie z. B. „Maria Luisa von Spanien“ und „Maria Luise von Spanien“. Siehe `http://de.wikipedia.org/wiki/Wikipedia:Artikel_zum_gleichen_Thema`

Ein Diskussionssystem kann derartige Fehler aufspüren, denn viele können nur aus den Beziehungen erkannt werden. Das Programm sollte dem Benutzer Hilfe anbieten, um diese Fehler irgendwie zu beheben.

Dafür ist es nötig, über eine *Argumentationstheorie* zu verfügen[46], um Beziehungen zwischen Aussagen auf einer höheren Abstraktionsebene zu analysieren; mit diesem Thema beschäftigt sich die Rhetorik, Philosophie und künstliche Intelligenz. Wenn die Struktur einer Diskussion von dem Programm „verstanden" wird, dann können sogar z. B. logische Fehlschlüsse aufgespürt werden.[47]

4.5.6 Archivierung

In jedem Diskussionssystem (einschließlich den Üblichen – E-Mail, Forum, Wiki, Chat –) gibt es oft wertlose Informationen. Beispielsweise alte Versionen der neu formulierten Informationen, veraltete Daten, nicht mehr berücksichtigte Alternativen, oder Abschweifungen.

Ein Diskussionssystem muss die Möglichkeit anbieten, Informationen abzulegen. Sonst besteht die oft unbeachtete Gefahr, *zu viele* Informationen zu haben. (und erinnere sich man an den Unterschied zwischen „Information" und „Wissen"[48]).

Eine Entfernung ist aber nicht immer günstig, denn die Informationen könnten in der Zukunft noch nützlich für andere Diskussionen sein: da es möglich ist, beliebige Argumente mit beliebigen anderen zu verknüpfen, könnte eine Abschweifung in einer Diskussion später in einer anderen Diskussion benutzt werden (eventuell nach einer Anpassung).

Die nicht mehr gewünschten Informationen können daher als „archiviert" markiert werden. Wir finden diese Unterscheidung sinnvoll, weil es viele nutzlose Informationen in Informationen gibt, und weil die Informationsüberflutung an Wissensmanagementsystemen ein Problem für die Nutzer darstellt.

4.5.7 Begründungen

Sowohl am Ende einer Diskussion als auch in der Mitte, muss ein Diskussionssystem fähig sein, alle Kenntnisse über die Diskussion zu sammeln um eine Zusammenfassung des jetzigen Zustandes zu zeigen. Für ein beliebiges Argument sollte der Benutzer sowohl den Wahrheitszustand des Argumentes *als auch die Gründe, aus denen es sich in diesem Zustand befindet*, klar sehen können.

Diese ist eine äußert wichtige Anforderung, denn der Benutzer braucht eine Methode, um die wichtigsten Punkte einer Debatte schnell zu erfahren. In z. B. einer Mailing-Liste muss der Benutzer alle Beiträge lesen um eine „Übersicht" des diskutierten Themas und dessen Verlaufes zu bekommen.

Darüber hinaus wird dies benötigt, wenn unser Ziel darin besteht, die Wahrhaftigkeit einer Aussage zu *beweisen* (dies war Anwendungsbeispiel 4); um einen

[46] Zum Beispiel: `http://www.ai.rug.nl/~verheij/publications/cd96/cd96.html`

[47] Die Arbeit (AWC07) erweitert das Programm Araucaria um Fehlschlüsse beschreiben und erkennen zu können.

[48] Dazu das Zitat: „Wir ertrinken in Informationen, aber uns dürstet nach Wissen" (John Naisbitt)

Beweis zu gestalten müssen alle Begründungen angegeben werden. Wenn man eine Entscheidung treffen muss (Anwendungsbeispiel 8) ist es ebenfalls sehr wichtig, die Begründung jeder Alternative zu wissen.

Ein Diskussionssystem sollte deshalb fähig sein, die Gründe jeder der vorgenommenen Operationen anzugeben. Da das Programm Regeln benutzt, um Wahrheitswerte zu berechnen, können dieselben Regeln „rückwärts“ angewandt werden, um zu erklären, warum ein Argument einen bestimmten Wahrheitswert hat.

4.5.8 Mögliche Erweiterungen

Es gibt noch weitere Anwendungen für die Informationen aus einer Diskussion. Da ein Programm nicht alles Mögliche unterstützen kann, sollte es erweiterbar sein, damit die spezialisierten Benutzer ihre benötigten Operationen programmieren und benutzen können.

Ein Beispiel davon sind **Abstimmungen** (über Argumente, über Alternativen, ...). Für viele Methodologien und Anwendungen ist es nötig, eine Abstimmung am Ende einer Diskussion stattfinden lassen, vor allem bei den Prozessen, bei denen eine Entscheidung getroffen werden muss. Genauere Beispiele gibt es in (EVS06).

Das Programm muss auch anderen –eventuell künftigen– Programmen die Nutzung der Informationen ermöglichen, z. B. einer **Inferenzmaschine**, die neues Wissen und neue Aussagen automatisch und ähnlich einem Menschen erzeugen kann.

4.6 Argumentensuche

Es ist notwendig, über eine gute Suchmaschine zu verfügen. Nämlich:

- Wenn ein Benutzer ein neues Argument eingeben will, sollte eine Suche ähnlicher Argumente bevor der Benutzer noch mehr Details angibt, denn es soll nicht nochmals eingetragen werden, wenn es schon existierte.
- Nachdem ein Benutzer ein Argument gerade eingetragen hat und er es begründen oder mit anderen verknüpfen will, sollte auch eine Suche potenziell zusammenhängender Argumente dem Benutzer bei der Ergänzung des Arguments helfen.
- Um den Wahrheitswert eines Argumentes automatisch zu berechnen, müssen alle Argumente aus der Datenbank, die einen bestimmten Bezug mit dem jeweiligen Argument haben, gefunden werden.
- Um noch unbestätigte oder mit den anderen Daten unverknüpfte Informationen regelmäßig zu finden.

Diese Suche muss textuell effektiv sein, weil es keine Standardmethode für das Schreiben von Aussagen gibt. Nicht nur müssen lexikalische Variationen auch gesucht werden (z. B. die Aussagen „*Vegetarisch* zu sein ist gut“ und „Der

Vegetarismus ist gut“), sondern ideal wäre es, auch neue Formulierungen einer Idee durch andere Wörter zu finden (z. B. die Aussage „Eine vorwiegend pflanzliche Ernährung ist vorteilhaft“ finden nach der Suche einer der zwei Früheren).

Die Argumentensuche muss natürlich auch die Nutzung der Begriffe und Beziehungen aus der Ontologie ermöglichen. Man sollte beispielsweise folgendes suchen können: „alle Argumente, die gegen A sind“, oder „alle Elemente, die gleichbedeutend wie A sind“. Dies ist nicht so sehr, um dem Benutzer zu helfen, sondern auch damit das Programm Argumente besser suchen bzw. finden kann. Der Benutzer zieht wohl vor, ein einfaches Argument zu finden und danach sich umschauen, eher als komplizierte und genaue Suchtextketten zu denken und schreiben.

Der schon beschriebene Ansatz ClaimFinder beschäftigt sich ausschließlich mit der Suche von Argumenten und implementiert ausreichende Funktionalitäten.

4.7 Kollaboration zwischen Benutzern

4.7.1 Kollaborationsarten

Da in den meisten der im Kapitel 2 beschriebenen Anwendungen mehrere Personen beteiligt sein können, ist es notwendig, ein Diskussionsystem zu gestalten, das die Kollaboration erlaubt. Ein solches System kann natürlich auch von einer einzelnen Person benutzt werden.

Aus mehreren Anwendungsfällen, die in Studien wie (KSE03) beschrieben sind, folgt, dass es zwei Methoden gibt, ein Diskussionssystem zu benutzen:

- Viele Personen treffen sich in einem Zimmer, und jeder bringt Argumente für bzw. gegen ein bestimmtes Thema vor, wobei ein **Moderator** gleichzeitig die Argumente für all Beteiligte auf einem Bildschirm sichtbar macht. Eine solche Diskussion nennt man **synchrone** Diskussion.

- Die Benutzer werden aufgefordert, ausschließlich durch das Internet (d. h. ohne gleichräumliche Dabeisein) ein Programm zu benutzen, mit dem sie zu einer Diskussion beitragen können. Normalerweise gibt es keinen Moderator. Die Diskussion in diesem Fall ist **asynchron**, da die Benutzer nicht gleichzeitig teilnehmen.

Synchrone Diskussion (z. B. per Chat) benachteiligt die Formalisierung, denn die Zeit für ausführlichere Beschreibungen fehlt, sie hat hingegen den Vorteil, dass die Themen prompt betrachtet werden können. (KSE03, S. 56) Allerdings werden die Benutzer, die synchron diskutieren (ganz gleich ob per Chat oder mündlich im selben Raum), nur einen Vorteil vom Programm beziehen können, wenn sie das Gespräch auf das gerade aufzubauende Diagramm orientieren. (KSE03, S. 70)

Die Möglichkeit, einen Moderator zu haben, ist sehr wichtig; die Effektivität mancher Ansätze basiert sogar auf diese Rolle (z. B.: *Dialog Mapping*, siehe (KSE03), S. 122). In (KSE03, S. 129) wird diese Rolle ausführlich beschrieben. Die Nutzung eines Moderators ist trotzdem ein noch nicht entschiedenes Thema.

Moderation ist einerseits sinnvoll:

- Sie trägt dazu bei, die geplanten Ziele zu erreichen, und die Ergebnisse sind besser als ohne Moderation. (DSW06, S. 184)
- In dem Szenario der Diskussion im selben Raum muss nur der Moderator das benutzte Programm erlernen; für die Benutzer ist es deshalb bequemer. (KSE03, S. 145)

Nichtsdestoweniger existieren ebenfalls soziale Probleme, z. B.:

- Nur wenige Personen können tüchtige Moderatoren werden: Entscheidend sind z. B. die Fähigkeit, neutral ein Gespräch zu lenken ohne sich zu beteiligen, oder die Gabe, gleichzeitig tippen und zuhören zu können. (KSE03)
- Wenn es einen Tutor gibt, der die Benutzer korrigiert (z. B. per Chat oder E-Mail), dann wird die Diskussion beeinträchtigt, denn viele Benutzer werden es nicht wagen, die vom Tutor vorgenommenen Korrekturen zu bezweifeln. Laut Experimenten aus (VAK99).
- Manchmal wird kein Moderator benötigt, um eine produktive Diskussion zu führen. Berichte wie (KSE03), S. 150 zeigen, dass die Diskussion „von allein“ laufen kann, wenn für das Publikum die Ziele der Diskussion sehr klar sind, oder wenn die von den Teilnehmern benutzte Fachsprache zu schwierig für den Moderator ist.

Da unser Interesse darin besteht, ein Programm zu erstellen (und keine Problemlösenmethodologie, die den Teilnehmern einer Sitzung Rollen zuordnet), ziehen wir vor, nach einer mehrbenutzerfähigen Software zu streben, mit der die Benutzer asynchron beitragen. Dies ermöglicht auch die anderen Anwendungen: mit oder ohne Moderator, und im selben Raum oder nicht. Aber damit die Benutzer wirklich selbst zum Programm beitragen können, muss das Diskussionssystem eine einfache Syntax haben (jeder wird sie nämlich lernen müssen, nicht nur der Moderator) und die Schnittstelle muss eine sehr schnelle Eingabe sowie Umwandlung der Informationen ermöglichen, denn eventuell müsste sie von einem Moderator benutzt werden, um ein Gespräch in Echtzeit abzuschreiben.

4.7.2 Prozesse, Probleme und Lösungen

In einem Diskussionssystem ist Kollaboration vielseitig wichtig. Beispielsweise um:

- Inhalte beizutragen.
- Änderungen in Diskussionen zu folgen.
- Aussagen in Teile zu zerlegen.
- informelle Informationen neu zu formalisieren.

- Argumente miteinander zu verbinden.
- wiederholte Informationen aufzuspüren und beizumischen.
- Fehler oder Lügen aufzuspüren und zu korrigieren.
- veraltete Informationen zu suchen und zu archivieren.
- den eigenen Zielen zu folgen und sicherzustellen, dass die Diskussion in die richtige Richtung läuft.
- zu ermöglichen, dass sich die Benutzer über allerlei Themen unterhalten können.[49]

Weiterhin können viele Kollaborationsprobleme auftauchen. Wir beschreiben manche kurz:

- schlecht formalisierte Informationen (sie stellen kein Argument dar).[50]
- Informationen, die erneut geschrieben werden müssen (zu mehrdeutig).
- wiederholte Informationen.
- nützliche Informationen, die aber an der falschen Stelle geschrieben oder verlinkt wurden.
- Beitragen ungeeigneter Texte (z. B. Inhalte mit Copyright oder mit restriktiver Lizenz).
- Vandalismus: Zerstören von Informationen, und Interesse nur an Belästigung.
- Beiträge ohne Quellen oder mit ungültigen Quellen.
- Themaabkommen in einer Diskussion, die gezielt sein sollte.
- zu viel Zeit an der Diskussion verloren ohne Fortschritte an der Lösungfindung zu erreichen.

Zunächst stellen wir einige Ideen vor, um die Folgen dieser Probleme zu vermindern:[51]

[49] Selbstverständlich wäre es interessant, dass auch die persönlichen Mitteilungen zwischen Benutzern auch mit demselben Diskussionssystem betrachtet werden, sodass die damit erzeugten Informationen eventuell für andere Zwecke benutztbar sind.

[50] Die meisten Leute sind nicht in der Lage, ihre Meinungen sogar über wichtige Themen zu formalisieren (wenn sie überhaupt eine Meinung haben). (KSE03, S. 104). Ein Diskussionssystem kann dabei helfen, diese Fähigkeit zu erlernen.

[51] Diese Schwierigkeiten zu erleichtern ist alles worauf wir streben können. Es ist für ein Programm nämlich nicht möglich, all diese Probleme zu beheben, denn bei diesen Kollaborationsthemen liegt das Problem oft nicht an der Software, sondern am Benutzer.

- Das Diskussionssystem so konzipieren, dass die nutzlosen bzw. unwahren Informationen unbeachtet werden. Das Entfernen der Informationen ist deshalb nicht notwendig. Jedes Argument, das einen „wahren“ Wahrheitswert nicht erreicht hat, ist nicht Teil der bekannten Wahrheit.

- Ebenfalls anstreben, dass Vandalismus nur dadurch möglich ist, indem neue Argumente angelegt werden, die die aktuellen Informationen widerspricht. Wenn die neuen Argumente unsinnig sind, werden sie den Stand der gut begründeten Wahrheit nicht ändern können.

- Da das Programm teilweise kennt, was falsch bzw. wahr ist, kann es dieses Wissen benutzen, um Vandalismus aufzuspüren. Beispielsweise können Benutzer gefunden werden, deren Beiträge schnell verfälscht werden, oder die die seit langem gut begründeten Argumente direkt als falsch markieren.

- Bei Themen, bei denen es verschiedene Möglichkeiten gibt, ein bestimmtes Problem zu verstehen (z. B. wegen unterschiedlichen Wortschatzes), kann es erlaubt werden, dass jeder Benutzer die Daten auf seine eigene bevorzugte Weise sieht. Das Programm kann dann die Ansichten aller Nutzer kombinieren.

- Jeder Beitrag sollte mit dessen Autor annotiert werden, damit es immer klar ist, wer verantwortlich für die Inhalte ist.[52]

- Eine Konvention darüber ausdenken, was als „absolut wahr“ angesehen werden kann (d. h. so wahr, dass keine Begründungen mehr benötigt werden). Dies kann z. B. mit einer Liste anerkannter Quellen erfolgen, oder mit der Bestätigung bestimmter Benutzer (Prüfer), oder „alles ist wahr bis es widersprochen wird“, etc.

- Das Themawechseln erlauben, wenn das Ergebnis interessant ist. Wenn das Ziel ist, allerlei Informationen zu sammeln, dann ist jedes Thema nützlich. Wenn das Thema festgelegt bleiben sollte, dann können jedem Argumente Etikette angehängt werden, die das Thema beschreiben, damit die Benutzer sich zu einem bestimmten Thema einschränken können.

- Übungen anbieten, um die Syntax des Programms zu erlernen.

- Viele Anleitungen anbieten, nämlich über Formalisierung, Gedankengang, und andere diskussionstypische Themen.[53]

Darüber hinaus muss die Kommunikation zwischen Benutzern als Teil der Arbeitsverfolgung vorhanden sein. Zum Beispiel:

[52] Laut (KSE03), S. 187, Punkt 3 sei die Urheberschaft der Inhalte als ein Hauptpunkt zu betrachten. Das dort beschriebene Programm ClaiMaker erlaubt, jedem Beitrag eine Literaturangabe zuzuordnen; außerdem werden bei jedem Beitrag die Daten des Verfassers klar angezeigt.

[53] Ein gutes Beispiel von Dokumentation ist diejenige des Programms „Convince me“, in `http://www.soe.berkeley.edu/~schank/convinceme/materials.html`

- Eine benutzereigene „Beobachtungsliste“ (wie auf Mediawiki, die Software Wikipedias), wo jeder sehen kann, was es Neues bei den bevorzugten Diskussionen gibt.
- Beiträge-Bewertung durch fortgeschrittene Benutzer, Kritiken an die Beiträge (natürlich mittels neuer Argumente), Metabewertungen (Kritiken an die Kritiken) usw.
- Arbeitsgruppen einrichten damit die Benutzer auf bestimmte Aufgaben spezialisiert sein können: Einige fügen nur Informationen hinzu, andere verbinden sie nur, andere überwachen die neuen Beiträge, etc.
- Die Benutzergruppen können auch nach Kenntnis eingeteilt werden, um nutzlose Diskussionen zwischen Experten und Anfängern zu vermeiden (falls gewünscht).

Einige dieser Vorschläge sind nicht technisch sondern sozial, und daher sollten sie zu Methodologien und Konventionen gehören. Andere können im Programm implementiert werden. Das Gebiet der Kollaboration ist sowieso ein komplexes Thema, und viele andere Arbeiten sind notwendig, um effizientere Arbeitsmethoden zu schaffen.

4.8 Graphische Schnittstelle und Visualisierung

4.8.1 Benutzung der Schnittstelle

Einer der wichtigsten Punkte eines Diskussionsystems ist die graphische Schnittstelle, denn es ist dort, wo alle vorher beschriebenen Verfahren stattfinden: Ein- und Ausgabe der Informationen, Wissenverarbeitung und Kollaboration der Benutzer.

Eine Anforderung einer Schnitteschelle ist, dass sie die schnelle Verarbeitung der Daten erlaubt (idealerweise so schnell und leicht wie das menschliche Gehirn). Daten müssen nämlich schnell eingegeben werden können, man muss durch die schon Existierenden problemlos stöbern können, sie zu verändern ohne auf Nebenwirkungen achten zu müssen, und beliebige Operationen auf irgendeiner Genauigkeitsebene durchführen können (z. B. sehr abstrakt, sehr genau, gemäßigt, ...).

An der Schnittstelle sollten mehrere Dimensionen dargestellt werden, einschließlich eine *zeitliche* (Evolution der Informationen) und *kontextuelle* (mehrere Ansichten oder Zusammenhänge) Dimension. (KSE03, S. 147). Zudem gibt es mehrere *Formulierungen* (gleichbedeutende Darstellungen).

4.8.2 Mehrere Schnittstellen

Wir schlagen vor, dass ein Diskussionsystem *modular* sein muss, und von daher sollte die Schnittstelle nicht zum Programm gebunden sein. Dies ermöglicht die Nutzung mehrerer Schnittstellen, z. B. eine im Textmodus, eine in Form einer

Gedankenkarte, eine innerhalb einer Webseite, eine nur als API (Programmierschnittstelle) zwischen verschiedenen Rechnern, ...[54]

Jede Darstellungsart stellt eigene Vorteile vor, die ein Programm profitieren sollte.

Beispielsweise kann eine Graphendarstellung, die ein großes Netz der Verbindungen zwischen Argumenten zeigt, bestimmte Teilmengen an Knoten sichtbar machen, die miteinander stark verknüpft sind, was höchstwahrscheinlich bedeutet, dass sie sich über das selbe Thema handeln (KSE03, S. 196), denn ein „Thema“ ist eine Menge an zusammenhängenden Ideen. Dieses durch die Graphenanalyse erzeugte Wissen kann profitiert werden, um die „wichtigen“ Themen zu finden und die Suchergebnisse so zu verbessern.

Eine Web-Schnittstelle verfügt dagegen über andere Möglichkeiten, vor allem die Kollaboration betreffend, und ermöglicht die einfache Nutzung von semantischen Webtechnologien (URIs, Metadaten, XML, ...), sowie auch das Nutzen und das reibungslose Verknüpfen auf viele externe Quellen.

Eine Programmierschnittstelle unterstützt die Modularität des Programms und erlaubt, dass sich die Komponenten miteinander kommunizieren können.

4.8.3 Genauigkeit

Ein Haupterfordnis eines Diskussionssystems, sowie auch vieler Gedankenkartenprogramme, ist die Fähigkeit, die Genauigkeitsebene der Sicht zu wechseln, d. h., von einer abstrakten Sicht des Themas zu einer genauen Sicht zu kommen, und umgekehrt.

Anstatt über etliche mögliche „Sichten“ zu verfügen (z. B. „abstrakt“ und „genau“) ist es besser, das allmähliche Zoomen zu erlauben: sich einem Begriff nähern bzw. entfernen zu können im figürlichen aber im graphischen Sinne. Eine Gedankenkarte-ähnliche Schnittstelle kann Zoom benutzen, um die Genauigkeitsebene zu ändern.[55]

Eine Textmodus-Schnittstelle kann ähnlicherweise das Genauigkeitsniveau jedes Begriffs berechnen, und die mit dem aktuellen Thema bezogenen Elemente im ersten Platz zeigen, vorausgesetzt, dass sie sich in der aktuell sichtbaren Genauigkeitsebene befinden.

4.8.4 Integrierte Suche

Die Argumentensuche ist eine unentbehrliche Operation, wie schon im Punkt 4.6 beschrieben wurde. Die graphische Schnittstelle sollte den Suchprozess stark mit dem Informationseingabeprozess integrieren; beispielsweise:

- durch mehrere Phasen: Man gibt ein noch ungenaues Argument ein, dann sucht das System ähnliche Argumente, und wenn keine existieren dann darf man weitere Details eingeben und anschließend das Argument hinzufügen.

[54] ClaiMaker kann die gespeicherten Daten sowohl als Webseite als auch als Graphen darstellen.

[55] Diese ist eine in vielen Systemen vorhandene oder geplante Eigenschaft, z. B.: ClaiMaker, iMapping (NEPOMUK), etc.

- in Form von Autovervollständingung: Während man eintippt werden Argumente angezeigt, die ähnlich sind.

Die Autovervollständing ist in vielen modernen Anwendungen nützlich, jedoch wäre sie in einem Diskussionssystem schwierig zu implementieren, weil die Daten, die vervollständigt werden müssen, *Aussagen* sind, und Aussagen sind in der Regel nicht mit einem kurzen Namen erkennbar, sondern sie sind zahlreich und gleichartig. Ein interaktives Menü kann aber die verfügbaren Möglichkeiten graphisch darstellen, z. B. durch ein kleines Diagramm für jedes Argument, damit der Zusammenhang klarer erkannt werden kann.

4.8.5 Ausgabe

Neben der Informationdarstellung mittels einer graphischen Schnittstelle ist es äußert wichtig, die Daten exportieren zu können. Besonders interessant ist die Fähigkeit, Prosatext aus den Begründungen eines beliebigen Argumentes zu erzeugen; dabei können die Ideen durch Partikel aus der Argumentationsontologie (wie z. B. „weil“, „nicht“, „hat folgende Teile“ etc.) miteinander verbunden werden.

Bei der Texterzeugung muss die Kohärenz in Betracht gezogen werden. Um kohärente Texte zu gestalten ist es ideal, /Rhetorical Structure Theory/[56] zu benutzen, weil es genau darauf abzielt.

In späteren Ansätzen könnte man einen ganzen wissenchaftlichen Artikel in Form von Argumenten schreiben, und einem Programm die Aufgabe lassen, den Prosatext zu erzeugen. Dazu kann semantisch annotierter LaTeX-Code erzeugt werden.[57]

5 Ergebnisse

Diese Arbeit hat sowohl einen Analysen- (Kapitel 2 und 3) als auch einen Synthesisteil (Kapitel 4).

5.1 Analyse

Im Kapitel 2 wurden verschiedene Szenarien studiert, an denen argumentationbezogene Programme benutzt werden. Damit wurde herausgefunden, dass viele wichtige, übliche und interessante Anwendungen als Diskussion verstanden und betrachtet werden können; unter diesen Fällen befinden sich so erhebliche Themen wie das Problemlösen, die Entscheidungsfindung, das kollaborative Lernen, die Ideenfindung und Sinnsuche, als auch ehrgeizige Aufgaben wie die Bestätigung der Wahrhaftigkeit beliebiger Internet-Webseite, darunter Wikipedia. Im Verlauf dieses Kapitels wurde der Begriff „Diskussionssystem“ sowie dessen Anforderungen und die allen gemeine, benötigte Infrastruktur entwickelt.

[56] Empfehlenswert ist die Einführung in RST, in `http://www.sfu.ca/rst/`

[57] Es gibt SALT (*Semantically Annotated LaTeX*, `http://salt.semanticauthoring.org/`), das auf RST basiert, und sogar eine RDFS-Ontologie für Rhetorik bietet.

Im Kapitel 3 wurden sowohl häufige als auch spezialisierte Programme verglichen und kritisiert, mit folgenden Ergebnissen:

- Es gibt relativ wenige Systeme (Programme, Methodologien, ...) und im Allgemeinen wenig Recherche über Diskussionswerkzeuge.
- Die einfachen Systeme werden noch weitgehend benutzt, weil sie sehr verbreitet sind und das Ziel teilweise erledigen können.
- Manche der spezialisierten Ansätze sind auf einem bestimmten Gebiet tauglich (z. B. Schnittstelle, Kollaboration, Darstellung, ...) aber häufig nicht in anderen, die auch wichtig sind.
- Nur wenige Programme haben als Ziel die logische Argumentation (Notation, Aussagen mit Wahrheitswert usw.).

Die Syntax, die dazu dient, Argumente zu beschreiben, wurde ebenfalls besprochen und etliche Ansätze wurden erwähnt. Es ist immer ein subjektiver Punkt, der per se schwierig ist, sogar wenn die Absicht darin besteht, eine einfache Notation zu gestalten. Es gibt immer einen Kompromiss zwischen Genauigkeit und Benutzerfreundlichkeit, wobei für die Nutzer einfache Notationen am meisten empfohlen werden.

5.2 Synthese

In Kapitel 4 wurde einem Prozess gefolgt, um Ideen für die Implementation eines besseren Diskussionssystems vorzustellen. Zuerst wurden die Anforderungen grob gelistet: unabhängig aller Problemlösen-Methodologien, mehrbenutzerfähig, im Internet benutzbar, mit graphischer Schnittstelle, modular, und einfach zu erweitern („freie Software“-Lizenz, plattformunabhängig, den Standards folgend).

Danach wurden vier anfängliche Ideen vorgeschlagen, um ein Diskussionssystem mittels Erweiterungen schon benutzter Systeme zu implementieren:

- Text mit einer Begründung pro Satz.
- Aussagenwiki.
- Gedankenkarte mit einem Argument pro Knoten.
- Forum mit semantischer Beschreibung der Nachrichten.

Aus diesen Ideen wurden die Hauptelemente eines universelen Diskussionssystemes herausgefunden: Notation (mit Referenzssystem und Eingabemethode), semantische Informationen auf jedem Argument, Regeln für die Nutzung der Information, Suchmaschine, System für die Kollaboration der Benutzer, und graphische Schnittstelle (und Ausgabemethoden).

Für jeden der Punkte wurden neue Gedanken angegeben und begründet, die die aktuellen Ansätze verbessern sollen, und entsprechend wurden Programme erwähnt, die diese Ideen schon implementieren. Es wurde erörtert:

- Notation: Sie muss einfach und erweiterbar sein, und die Prosa ergänzen. Einen Formalisierungsprozess (von Prosa in eigene Syntax) sollte es geben. Sie muss den Begriff „in einer Aussage benutzter Begriff“ kennen, um Aussagen vereindeutlichen zu können. Jedes Argument muss ein Satz mit Verb sein, nicht Fragen oder Begriffe. Die Beziehungen können am Anfang mehrdeutig sein und danach spezialisiert werden. Die Eingabe kann sowohl im Textmodus als auch mittels von Graphen erfolgen.

- Semantische Informationen auf jedem Argument: Jedes Argument besitzt einen „Wahrheitswert“; die Semantik dieses Begriffs muss entschieden und definiert werden, aber die möglichen Werte und Interpretationen können auch in einer erweiterbaren Hierarchie dargestellt werden. Der Wahrheitswert wird gelegentlich aus anderen Argumenten gerechnet. Jedes Argument ist allgemeingültig und daher gibt es keine Einschränkung auf Themen.

- Nutzung des Wissens: Das Programm muss die Wahrheitswerte berechnen und verwalten, Begründungen einer beliebigen Aussage erzeugen können, ähnliche Ideen zusammenbringen (wobei deren Beziehungen auch semantisch beschrieben werden müssen), Konflikte und Fehler aufspüren, und Informationen archivieren. Unbedingt muss es Ideen neuformulieren und Themen beimischen können. Das Programm muss offen für weitere Funktionen sein, z. B. Abstimmungen oder eine automatische Inferenzmaschine um Deduktion zu ermöglichen.

- Suchmaschine: Sie muss sowohl wortwörtlichen Text als auch Beziehungen und Elemente aus der Argumentationsontologie suchen können.

- Kollaboration: Asynchron und ohne Moderator ist es besser. Viele Ideen wurden vorgeschlagen um die Kommunikationsprobleme zu erleichtern und das System bequem und benutzerorientiert zu machen.

- Graphische Schnittstelle: Ist frei wählbar, denn es kann viele geben (z. B. Graphen, Text, Web, API) oder keine. Alle müssen die Arbeit auf verschiedenen Abstraktionsebenen ermöglichen, und zwar mittels Zoom. Die Suche muss stark integriert werden, beispielsweise mit Argumenten-Autovervollständigung. Außerdem kann der Inhalt zu Prosa exportiert werden.

5.3 Schlussfolgerung

Mit dieser Arbeit wurde der Begriff „Diskussionssystem“ eingeführt und beschrieben, und zwar nach der Art und Weise der logischen Argumentation (im Gegensatz dazu z. B. nur die Visualisierung zu berücksichtigen). Es wurde ein System vorgeschlagen, das alle Anwendungsbeispiele erfüllt, das aber grundsätzlich bleiben sollte (wie ein Wiki, Forum oder E-Mail-Programm) ohne sich mit schwierigen Problemlösenmethodologien zu befassen.

Diese Studienarbeit ist außerdem ein erster Beitrag zur deutschen Literatur über dieses Thema, denn es gibt nicht viel Recherche über Programme für die logische Argumentation, und das Meiste ist nur auf Englisch.

Die Arbeit wurde für den Autor eine sehr gute Übung, um Deutsch zu lernen, allerdings eine mühselige Aufgabe, die viel mehr Aufwand dargestellt hat als der eigentliche Inhalt. Als Ergebnisse davon ist inzwischen das deutsche Projekt Wikiwörterbuch mit neuen Wörtern, Definitionen und vielen Grammatik- und Ausdrucks-Zweifeln aufgewachsen.[58]

6 Nächste Schritte

Da das betrachtete Thema relativ neu und sehr aktuell ist, bleibt noch vieles zu erforschen. Hier werden manche Ideen für andere Studien- bzw. Diplomarbeiten oder ähnliche aufgelistet.[59]

Wir schlagen einige theoretische Arbeiten vor:

- Anstatt nur einen groben Durchblick des Themas hervorzubringen (es gibt schon viele solche Arbeiten, z. B. diese), nur *einen* der erwähnten Punkte auswählen (Notation, semantische Informationen, Wissensnutzungregeln, ...) und ihn ausführlich studieren bis zu einem nützlichen Ergebnisse. Manche Themen (z. B. effektives Kollaborationsmodell) sind immerhin so schwierig, dass sie vieler wissenschaftlichen Studien benötigen werden.

- Die Analyse der Diskussionssysteme vom Kapitel 3 fortsetzen, aber diesmal unbedingt im Hypertext-Format, denn es ist viel besser und bequemer als eine statische Arbeit auf Papier. Dadurch kann eine große Informationsquelle über Diskussionssysteme erschafft werden. Die Kollaboration der Benutzer sollte möglich sein, damit diese Datenbank wächst und nicht veraltet wird.[60]

- *Ausschließlich* die „wichtigen“ Diskussionssysteme aus Kapitel 3 auswählen[61], sie vergleichen, und entscheiden, welches das Aussichtreichste ist, d. h.: Welches soll in nachfolgenden Arbeiten ausführlich studiert werden; wie kann es korrigiert und mit den Eigenschaften der anderen ergänzt werden.

Wir schlagen ebenso praktische Lösungen vor:

- Leicht: Eines der üblichen Diskussionsysteme (Punkt 3.1) auswählen und versuchen, all seine (schon erwähnten) Probleme zu beheben damit es besser für die Diskussion taugt.

[58] `http://de.wiktionary.org/wiki/Benutzer:N142857`

[59] Siehe auch die Meinung von Douglas C. Engelbart auf (KSE03), S. 205.

[60] Natürlich wäre es interessant, Diskussionen über diese Diskussionssysteme zu ermöglichen... Dann gälte das Projekt nicht nur als Auflistung, sondern auch als Spielwiese.

[61] Diese sind zumindest: Compendium, ClaiMaker+ClaimFinder+ClaimSpotter, Araucaria, DebateMapper.

- Mittel: Ein neues System durch Kombination existierender Technologien gestalten (wie am Punkt 4.2 schon erklärt), und die Erfahrung dokumentieren und mit den Ideen dieser Arbeit beimischen und vergleichen.

- Schwierig: Ein neues Diskussionssystem programmieren, das alle auf Kapitel 4 erwähnten Ideen aufweist. Wir empfehlen, als Infrastruktur ein Wissensmanagementsystem zu benutzen; wir schlagen NEPOMUK vor, denn es hat alle benötigten Eigenschaften: modular, hat Gedankenkartenprogramm iMapping (mit Zoom), Wiki (mit Syntax, die mit anderen Wikis vereinbar ist), Autovervollständigen, semantisches Inhaltsrepositorium (SWECR), kompatibel mit anderen Dokumenten, formale Sprache um Beziehungen zu beschreiben (CDS), spezialisierbare Beziehungen, an dem normalen Computerbenutzer orientiert, BSD-Lizenz, ...[62]

7 Literatur

Die in dieser Arbeit am meisten benutzte Referenz ist „Visualizing Argumentation“ (KSE03), eines der wenige Bücher über Programme für die Darstellung und Nutzung von Argumenten. Das Buch ist aber nur eine Sammlung einzelner Artikel von verschiedenen Autoren, nicht eine Zusammenfassung des Themas oder *aller* wichtigen Punkte, die beim Design eines Diskussionsprogramms betrachten werden müssen. Das Buch weist immerhin auf die Ergebnisse vieler realer Experimente (einige erfolgreich, andere nicht) und auf die dabei gelernten Erfahrungen.

Auch hilfreich wurden persönliche Webseiten von Personen, die auch Interesse daran haben, bessere Diskussionsprogramme oder -methodologien zu haben, und die ihre Ideen und Meinungen über reichliche Themen der Argumentation und des Wissensmanagements dargelegt haben (z. B. Wikis, Philosophie, die Welt, die Wahrheit, ...).[63]

Zunächst folgt die Liste der in der Arbeit zitierten Werke. Alle außer (EW03) wurden komplett gelesen.

Literatur

[AWC07] A., Gibson, Rowe G. W. und Reed C.: *A computational approach to identifying formal fallacy.* Technischer Bericht, University of Dundee, 2007. `http://babbage.computing.dundee.ac.uk/chris/publications/2007/cmna2007-gibson.pdf`.

[BL99] Berners-Lee, Tim: *Tejiendo la red.* Siglo veintiuno, 1999.

[62] Ein anderer Grund ist, dass der Autor dieser Arbeit als HiWi am Projekt NEPOMUK gearbeitet hat...

[63] Beispiele sind `http://www.nooranch.com/synaesmedia/wiki/wiki.cgi` , `http://www.usemod.com/cgi-bin/mb.pl?DebateTool`

[DSW06] Davies, John, Rudi Studer und Paul Warren: *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. Wiley, 4 2006.

[EVS06] Engler, Michael, Denny Vrandecic und York Sure: *A Tool for Diligent Argumentation: Experiences, Requirements and Design.* Technischer Bericht, Universität Karlsruhe and BT Research, 2006. http://www.aifb.uni-karlsruhe.de/WBS/dvr/publications/stica2006engler.pdf.

[EW03] Eisenführ, Franz und Martin Weber: *Rationales Entscheiden*. Springer, 2003.

[Gö06] Göbel, Markus: *Semalan - Semantic Mailing List Analyzer*. Technischer Bericht, Universitaet Karlsruhe, 1 2006. http://xam.de/2006/2006-01-10-DA-Markus_Goebel.pdf.

[Har05] Harrell, Maralee: *Using Argument Diagramming Software in the Classroom*. Technischer Bericht, Carnegie Mellon University, 2005. http://www.hss.cmu.edu/philosophy/harrell/ArgumentDiagramsInClassroom.pdf.

[KSE03] Kirschner, Paul A., Simon J. Buckingham Shum und Chad S. Carr (Eds): *Visualizing argumentation*. Springer, 2003.

[RMRW06] Rowe, Glenn, Fabrizio Macagno, Chris Reed und Douglas Walton: *Araucaria as a Tool for Diagramming Arguments in Teaching and Studying Philosophy*. Technischer Bericht, University of Dundee, Catholic University of Milan, University of Winnipeg, 2006. http://babbage.computing.dundee.ac.uk/chris/publications/2006/tp2006.pdf.

[Sel99] Selvin, Albert M.: *Supporting Collaborative Analysis and Design with Hypertext Functionality*. Journal of Digital Information, 1(4), Januar 1999. http://jodi.tamu.edu/Articles/v01/i04/Selvin/.

[SSM04] Sereno, Bertrand, Simon Buckingham Shum und Enrico Motta: *ClaimSpotter: an Environment to Support Sensemaking with Knowledge Triples*. Technischer Bericht, Knowledge Media institute, 2004. http://kmi.open.ac.uk/publications/pdf/kmi-04-29.pdf.

[TPSS05] Tempich, Christoph, H. Sofía Pinto, York Sure und Steffen Staab: *An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT)*. Technischer Bericht, Instituto Sperior Técnico (Lisboa), Universität Karlsruhe, University of Koblenz Landau, 5 2005. http://www.aifb.uni-karlsruhe.de/WBS/cte/html/

publications/pdf/ESWC2005_cte_spi_Argu_Onto4Diligent_final.pdf.

[VAK99] Veerman, Arja L., Jerry E.B. Andriessen und Gellof Kanselaar: *Collaborative Learning through Computer-Mediated Argumentation*. Computer-Supported Collaborative Argumentation for Learning Communities, CSCL'99 Workshop, 11th-12th Dec., 1999, Stanford University, 1999. http://d3e.open.ac.uk/cscl99/Veerman/.

[VH06] Völkel, Max und Heiko Haller: *Conceptual Data Structures (CDS) – Towards an Ontology for Semi-Formal Articulation of Personal Knowledge*. Proc. of the 14th International Conference on Conceptual Structures 2006. Aalborg University - Denmark, July 2006., 2006. http://www.xam.de/2006/04-cds.pdf.

[VPST05] Vrandecic, Denny, H. Sofia Pinto, York Sure und Christoph Tempich: *The DILIGENT Knowledge Processes*. Journal of Knowledge Management, 9(5):85–96, OCT 2005. http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/2005_kmjournal_diligent.pdf.

„Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder Abänderungen entnommen wurde.“

Ich bin der einzige Autor der Arbeit und ich habe ausschließlich sprachliche Korrekturen erhalten. Kein Inhalt wurde aus anderen Quellen abgeschrieben, und die Urheberrechte der Quellen wurden respektiert.

Datum, Ortsangabe und Unterschrift: