

Ein wissenschaftlicher Ziel-Empfehlungsdienst

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Wirtschaftsingenieurwesen Informatik

eingereicht von

Maida Osmic-Ahmic

Matrikelnummer 0225045

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung

Betreuer/in: Univ.-Prof. Dr.rer.nat. Klaus Tochtermann, TU Graz

Mitwirkung: Dipl.-Ing. Dr.techn. Markus Strohmaier, TU Graz

Wien, 13.10.2010

(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

Bakk. techn. Maida Osmic-Ahmic
Hauslabgasse 9/8
A-8010 Graz

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit, einschließlich Tabellen, Karten und Abbildungen, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ort, Datum, Unterschrift

Kurzfassung

In Zusammenarbeit mit dem Partnerunternehmen *Xohana* ist das Ziel dieser Arbeit ein neuartiges Service zu entwickeln, das Benutzern einer sozialen Software mögliche Ziele vorschlägt. Diese Masterarbeit wendet sich damit einer praktischen Aufgabenstellung zu und untersucht das Potential der Agenten- und Zielorientierten Modellierungssprache i^* [46] für diesen Zweck. Die Aufgaben der Arbeit umfassen den Entwurf, die Implementierung basierend auf i^* und die Evaluierung des Services.

Nach der Einführung in die für die Arbeit wichtigen Themen wie *Xohana Software System*, *Modellierung von Zielen*, *Suchalgorithmen* sowie *Empfehlungsdienste*, wurde die Architektur des selbst entwickelten Ansatzes vorgestellt, gemeinsam mit dem Kern der entwickelten Applikation - 3 einfache Suchalgorithmen (*Tiefensuche*, *Breitensuche* und *heuristischer Ansatz*), die speziell für *Xohana* angepasst wurden. Durch die quantitative und qualitative Evaluierung wurden die Vorteile und Nachteile der oben erwähnten Ansätze untersucht sowie sowohl miteinander als auch mit dem Baseline-Algorithmus verglichen.

Sowohl quantitativ als auch qualitativ hat sich die Tiefensuche für diesen Fall im Durchschnitt als am besten geeignet gezeigt.

Abstract

In cooperation with the partner company *Xohana* is the objective of this thesis development of a new type of service that proposes goals to the users of social touristic application. This thesis is intended to be a solution for a practical problem and the potential of the agent and goal-oriented modeling language i^* [46] is investigated for this purpose. The tasks of the work include the design, implementation based on i^* and the evaluation of the service.

After the introduction to the major thesis issues such as *Xohana Software System*, *modeling of goals*, *search algorithms* and *recommender systems*, the architecture of the developed application is presented, together with the core of the developed application - 3 simple search algorithms - (adapted *depth*, *breadth* and *heuristic search*), that have been adjusted especially for *Xohana* needs. Quantitative and qualitative evaluation weighted the advantages and disadvantages of the above-mentioned approaches, and compared the algorithms with each other and with the baseline algorithm.

Both quantitatively and qualitatively has the depth search for this case, on average, appeared as the most appropriate for this purpose.

Inhaltsverzeichnis

Inhaltsverzeichnis	7
Abbildungsverzeichnis	9
Tabellenverzeichnis	13
1 Einleitung	15
1.1 Motivation	15
1.2 Kontext und Ziel der Arbeit	16
1.3 Aufbau der Arbeit	18
2 Literaturüberblick	3
2.1 Das Xohana Software System	3
2.2 Modellierung von Zielen	6
2.3 Einfache Suchalgorithmen	16
2.4 Empfehlungsdienste	21
3 Architektur des Ansatzes	65
3.1 Struktur des Projekts	65
3.2 Modell für Xohana	65
3.3 Die Input- und Outputmenge	67
3.4 Xohana-Algorithmen	68
3.5 Architekturskizze	71
4 Evaluierung	79
4.1 Vorbereitung der Evaluierung	79
4.2 Durchführung der Evaluierung	82
4.3 Evaluierungsergebnisse	84
5 Zusammenfassung und Ausblick	97
5.1 Zusammenfassung	97
5.2 Ausblick	98

A	Anwendungsfälle	99
A.1	Anwendungsfall 1	99
A.2	Anwendungsfall 2	101
A.3	Anwendungsfall 3	102
A.4	Anwendungsfall 4	103
A.5	Anwendungsfall 5	106
A.6	Anwendungsfall 6	107
A.7	Anwendungsfall 7	109
A.8	Anwendungsfall 8	110
A.9	Anwendungsfall 9	111
A.10	Anwendungsfall 10	113
A.11	Anwendungsfall 11	115
A.12	Anwendungsfall 12	115
A.13	Anwendungsfall 13	117
A.14	Anwendungsfall 14	117
A.15	Anwendungsfall 15	119
A.16	Anwendungsfall 16	120
A.17	Anwendungsfall 17	122
A.18	Anwendungsfall 18	123
A.19	Anwendungsfall 19	124
A.20	Anwendungsfall 20	126
B	i*-Modell für Xohana	129
C	Zusammensetzung aller Klassen	131
	Literaturverzeichnis	133
	Index	139

Abbildungsverzeichnis

1.1	Wissensquellen in Empfehlungsdiensten [11]	17
2.1	Eingabe-Interface	4
2.2	Akteure [47]	8
2.3	Akteure-Verbindungen-Links [47]	9
2.4	Strategische Abhängigkeiten [47]	10
2.5	Verwundbarkeitsarten [47]	11
2.6	Akteurabgrenzung [47]	12
2.7	Elemente [47]	12
2.8	Mittel-Zweck-Link [47]	13
2.9	Dekomposition-Links [47]	14
2.10	Beitragslinks [47]	14
2.11	SR-Modell für ein Kunden-betriebenes E-Commerce System mit Mittelsmann [48]	15
2.12	Ein Knoten im Suchbaum [36]	16
2.13	Allgemeiner Baum-Suchalgorithmus [36]	17
2.14	Breitensuche auf einem einfachen binären Baum. In jeder Phase ist der Knoten, der als nächstes erweitert wird, durch einen Marker gekennzeichnet [36].	18
2.15	Tiefensuche auf einem binären Baum [36].	19
2.16	Taxonomie der Empfehlungsdienste ([1], [3], [30])	26
2.17	Der Framework für die Analyse und Klassifikation der Empfehlungsdienste [22]	29
2.18	Ablauf eines fallbasierten Empfehlungsdienstes [4]	30
2.19	Kritik von Beispielen (example critiquing) [28]	31
2.20	Dialogstruktur in constraint-based Empfehlungsdiensten [11]	33
2.21	Empfehlungsvorgang [9]	39
3.1	Blockdiagramm des Projekts	66
3.2	Auszug aus dem i^* -Modell für Xohana, Das komplette i^* -Modell für Xohana, kann man im Anhang B.1 auf der Seite 130 finden.	67

3.3	UML Package-Diagramm	71
3.4	Klassendiagramm für Algorithms-Package	75
3.5	Klassendiagramm für Processors-Package	76
3.6	Klassendiagramm für Server-Package	76
3.7	Klassendiagramm für Test-Package	76
3.8	Zusammensetzung aller Klassen	77
4.1	Qualitativer Vergleich der vier Algorithmen (nach Durchführung der Evaluierung, wurden die Ergebnisse aus Fragenbogen gesammelt und verglichen). Den größten Entscheidungsaufwand stellte die Suche mit heuristischem Algorithmus. Das allerhöchste Zutrauen an die getroffene Entscheidung hatten die Testpersonen bei Tiefensuche. Bei der Absicht der Wiederbenutzung haben die Tiefensuche und die Breitensuche die gleiche durchschnittliche Bewertung erreicht.	85
4.2	Rankingergebnisse, die Testpersonen am Schluss der Evaluierung gegeben haben. Die Tiefensuche wurde am besten gerankt, nah gefolgt von der Breitensuche. Die schlechteste durchschnittliche Ranking hatte der Baseline-Algorithmus.	86
4.3	Länge der Benutzung (die Zeit, die die Testperson gebraucht hat, um ihre Ziele auszudrücken). Für das Ausdrücken von Zielen mittels dem Baseline-Algorithmus brauchten die Testpersonen durchschnittlich am wenigsten Zeit. Die Benutzung dauerte bei der Breitensuche am längsten.	88
4.4	Länge der Benutzung pro Ziel (durchschnittlichen Längen der Benutzung wurde mit der Anzahl der vorgeschlagenen Ziele dividiert, um den Einfluss der Anzahl der ausgedrückten Ziele zu verhindern). Dabei sieht man dass die Testpersonen durchschnittlich am wenigsten Zeit gebraucht haben, ein Ziel mittels der Tiefensuche auszudrücken. Weit langsamer waren die Testpersonen beim Ausdrücken von Zielen mittels heuristischem Ansatz.	89
4.5	Katalogabdeckung. Der Baseline-Algorithmus lieferte eine 100% Katalogabdeckung, Die Tiefensuche und die Breitensuche erreichten jedoch auch gute Ergebnisse. Die schlechteste Katalogabdeckung zeigte die heuristische Suche.	90
4.6	Vergleich der durchschnittlichen Anzahl der Zielen, die abgegeben wurden, die aus i^* -Modell stammten oder die vorgeschlagen wurden. Am meisten wurden die Ziele bei der Baseline-Suche abgegeben, gefolgt von Tiefensuche, bei der Tiefensuche wurde aber die höchste Anzahl an Ziele vorgeschlagen. Bei der Baseline-Suche stammten alle Ziele die ausgewählt wurden aus i^* -Modell.	91

4.7 Genauigkeit. Die weit größte Genauigkeit hatte der Baseline-Algorithmus. Hinter ihm waren die heuristische Suche, die Breitensuche und die Tiefensuche.	92
4.8 Trefferquote. Obwohl die Baseline-Suche wieder das beste Resultat gezeigt hat, war da die Tiefensuche im Gegensatz zu den Genauigkeitsergebnissen ganz gut. Die Breitensuche war nicht viel schlechter, die heuristische Suche hatte die weit niedrigste Trefferquote. . . .	93
4.9 Durchschnittliche Anzahl der Interaktionen (d.h., die durchschnittliche Anzahl der Sessions, die eine Testperson während der Benutzung der Applikation gehabt hat). Am wenigsten brauchten die Testpersonen mit der Applikation, die der heuristische Ansatz für das Vorschlagen von Zielen benutzt hat, zu interagieren. Mit der Applikation der Breitensuche brauchten die Testpersonen am meisten Hilfe. Die Interaktion mit der Baseline-Suche wurde hier nicht bewertet, da hierbei alles in einer Session entwickelt.	94
4.10 Normierte durchschnittliche Anzahl der Interaktionen (um den Einfluss der Anzahl der vorgeschlagen Zielen auch hier zu eliminieren, hat man wieder durch die Anzahl der vorgeschlagen Zielen, sowie zusätzlich durch zwei dividiert). Wieder haben die Testpersonen die Tiefensuche als am intuitivsten befunden.	95
A.1 Gegeben: Leere Anforderung	99
A.2 Anwendungsfall 2a: Teil- oder Gesamtsuchstring	101
A.3 Anwendungsfall 2b: Teil- oder Gesamtsuchstring	102
A.4 Anwendungsfälle 3a und 3b	103
A.5 Anwendungsfall 4	104
A.6 Anwendungsfall 5	106
A.7 Anwendungsfall 6	107
A.8 Anwendungsfall 7	109
A.9 Anwendungsfall 8	110
A.10 Anwendungsfall 9	112
A.11 Anwendungsfall 10	113
A.12 Anwendungsfall 11	115
A.13 Anwendungsfall 12	116
A.14 Anwendungsfall 13	117
A.15 Anwendungsfall 14	118
A.16 Anwendungsfall 15	119
A.17 Anwendungsfall 16	120
A.18 Anwendungsfall 17	122
A.19 Anwendungsfall 18	123
A.20 Anwendungsfall 19	125

A.21 Anwendungsfall 20	126
B.1 Das i*-Modell für Xohana	130
C.1 Zusammensetzung aller Klassen	131

Tabellenverzeichnis

1.1 Dimensionen des Empfehlungsproblems [30]	16
2.1 Überblick über ausgewählte Definitionen von Empfehlungsdiensten [22]	22
2.2 Dimensionen der Empfehlungstechnologie [30]	27
3.1 Die Klassenunterteilung	72
A.1 Ergebnisse der BFS für den Anwendungsfall 1	100
A.2 Ergebnisse der DFS für den Anwendungsfall 1	100
A.3 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 1 . .	101
A.4 Ergebnisse (gleich für alle 3 Algorithmen)	102
A.5 Ergebnisse für den Anwendungsfall 2b (gleich für alle 3 Algorithmen)	102
A.6 Ergebnisse der BFS für den Anwendungsfall 3	103
A.7 Ergebnisse der DFS für den Anwendungsfall 3	104
A.8 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 3 . .	104
A.9 Ergebnisse der BFS für den Anwendungsfall 4	105
A.10 Ergebnisse der DFS für den Anwendungsfall 4	105
A.11 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 4 . .	105
A.12 Ergebnisse der BFS für den Anwendungsfall 5	106
A.13 Ergebnisse der DFS für den Anwendungsfall 5	107
A.14 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 5 . .	107
A.15 Ergebnisse der BFS für den Anwendungsfall 6	108
A.16 Ergebnisse der DFS für den Anwendungsfall 6	108
A.17 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 6 . .	108
A.18 Ergebnisse der BFS für den Anwendungsfall 7	109
A.19 Ergebnisse der DFS für den Anwendungsfall 7	110
A.20 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 7 . .	110
A.21 Ergebnisse der BFS für den Anwendungsfall 8	111
A.22 Ergebnisse der DFS für den Anwendungsfall 8	111
A.23 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 8 . .	111
A.24 Ergebnisse der BFS für den Anwendungsfall 9	112

A.25 Ergebnisse der DFS für den Anwendungsfall 9	112
A.26 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 9 . .	113
A.27 Ergebnisse der BFS für den Anwendungsfall 10	114
A.28 Ergebnisse der DFS für den Anwendungsfall 10	114
A.29 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 10 .	114
A.30 Ergebnisse für alle 3 Algorithmensarten für den Anwendungsfall 11	115
A.31 Ergebnisse der DFS für den Anwendungsfall 12	116
A.32 Ergebnisse des heuristischen Ansatzes und BFS für den Anwen- dungsfall 12	117
A.33 Ergebnisse für alle 3 Algorithmensarten für den Anwendungsfall 13	118
A.34 Ergebnisse der BFS für den Anwendungsfall 14	118
A.35 Ergebnisse der DFS für den Anwendungsfall 14	119
A.36 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 14 .	119
A.37 Ergebnisse der BFS für den Anwendungsfall 15	120
A.38 Ergebnisse der DFS für den Anwendungsfall 15	121
A.39 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 15 .	121
A.40 Ergebnisse für alle 3 Algorithmensarten für den Anwendungsfall 16	122
A.41 Ergebnisse der BFS für den Anwendungsfall 17	123
A.42 Ergebnisse der DFS und des heuristischen Ansatzes für den Anwen- dungsfall 17	123
A.43 Ergebnisse der BFS für den Anwendungsfall 18	124
A.44 Ergebnisse der DFS für den Anwendungsfall 18	124
A.45 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 18 .	125
A.46 Ergebnisse der DFS für den Anwendungsfall 19	125
A.47 Ergebnisse des heuristischen Ansatzes und BFS für den Anwen- dungsfall 19	126
A.48 Ergebnisse der BFS für den Anwendungsfall 20	127
A.49 Ergebnisse der DFS für den Anwendungsfall 20	127
A.50 Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 20 .	128

Einleitung

1.1 Motivation

Wer kein Ziel vor Augen hat, kann auch keinen Weg hinter sich bringen. (Ernst Ferstl)

Empfehlungsdienste sind seit der Erscheinung der ersten wissenschaftlichen Arbeiten in ihrem Bereich Mitte der Neunziger zu einem sehr wichtigen Forschungsbereich geworden. Wie weit sich die Bedeutung dieses Begriffes über die Jahre hinweg entwickelt hat, kann man am Unterschied zwischen den früheren und den aktuellen Definitionen bemerken. 1997 definieren *Resnick und Varian* den Empfehlungsprozess wie folgt: «*people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients*» [34], 2002 bezeichnet *Robin Burke* den Empfehlungsdienst als «*any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options*» [3]. Die Definition der Empfehlungsdienste, und somit der gesamte Forschungsbereich setzen ihren Fokus heute neben der ursprünglichen Ausnutzung des Peer-Wissens auch auf die Personalisierung, die Interaktion, die Nützlichkeit und die Wichtigkeit der Ergebnisse für den Benutzer.

Begleitende industrielle Lösungen auf diesem Feld bestätigen das zusätzlich: zuzüglich zu den ursprünglichen *kollaborativen* Empfehlungsdiensten implementieren die Entwickler heutzutage eine Reihe von verschiedenen Arten von *inhaltsbasierten* und *wissensbasierten* Empfehlungsdiensten oder ihre Kombination, entscheiden ob die Interaktion mit dem Benutzer besser *interaktiv* (in einem Dialog, wobei der Benutzer ein Feedback abgibt oder die Fragen beantwortet [30]) oder *Single-Shot* (wo jede Interaktion mit dem Nutzer verwendet wird,

um eine unabhängige Empfehlung zu produzieren, ohne zusätzlichen Benutzer-Feedback [30]) erfolgen sollte und untersuchen ob der entwickelte Dienst mehr *statische* und *dynamische* Benutzerprofile benötigt.

Dimensionen des Empfehlungsproblems					
Fundamentale Problemeigenschaften			Sekundäre Problemeigenschaften		
Struktur des Problems	Domäne	Beziehung mit dem Benutzer	Benutzereingabe	Hintergrundwissen	Empfehlungsausgabe
<i>Auswahlproblem</i>	<i>Größe und Verschiedenartigkeit des Ergebnisraums</i>	<i>Beziehungslänge</i>	<i>Grad des Benutzeraufwandes</i>	<i>Soziales Wissen</i>	<i>Ausgabotyp</i>
<i>Konfigurationsproblem</i>	<i>Lebenserwartung</i>	<i>Beziehungstiefe</i>		<i>Inhaltswissen</i>	<i>Erklärungsgrad</i>
<i>Planungsproblem</i>	<i>Kritikalität</i>			<i>Wissen über Domäne</i>	
<i>Erforschungsproblem</i>	<i>Wichtigkeit und Dauer der Benutzerpräferenzen</i>				
	<i>Endbenutzertyp</i>				

Tabelle 1.1: Dimensionen des Empfehlungsproblems [30]

Zu den erwähnten Dimensionen der Empfehlungstechnologien müssen gleichzeitig verschiedene Dimensionen des Empfehlungsproblems [30] in Betracht genommen werden - neben den fundamentalen Problemeigenschaften wie der Struktur des Problems, der Domäne und der Beziehung mit dem Benutzer sollte man außerdem auf die sekundären Problemeigenschaften (die Benutzereingabe, das Hintergrundwissen und die Empfehlungsausgabe) achten. Einen Überblick über die Einzelheiten dieser Eigenschaften kann man in der nachfolgenden Tabelle (1.1, aus [30], auf der Seite 16) gewinnen.

Auf der anderen Seite muss der Entwickler auch untersuchen, welche Wissensquelle für den geplanten Empfehlungsdienst am besten geeignet sind (siehe Abbildung 1.1 aus [11], Seite 17) und welche Probleme (zwei Versionen vom *cold-start*-Problem, *first-user* und/oder *first item*, oder *Portfolio-Effekt*) auftreten können.

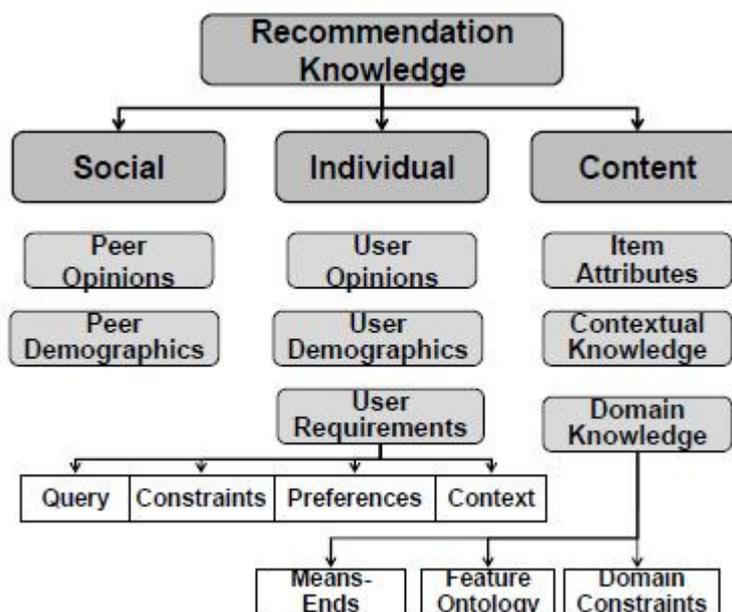


Abbildung 1.1: Wissensquellen in Empfehlungsdiensten [11]

1.2 Kontext und Ziel der Arbeit

In Zusammenarbeit mit dem Partnerunternehmen Xohana ist das Ziel dieser Arbeit ein neuartiges Service zu entwickeln, das Benutzern einer sozialen Software mögliche Ziele vorschlägt. Diese Masterarbeit wendet sich damit einer praktischen Aufgabenstellung zu und untersucht das Potential der Agenten- und Ziel-orientierten Modellierungssprache i^* [46] für diesen Zweck. Die Aufgaben der Arbeit umfassen den Entwurf, die Implementierung basierend auf i^* und die Evaluierung des Services.

Diese Masterarbeit präsentiert einen selbst entwickelten Ansatz im Bereich von wissensbasierten Empfehlungsdiensten und untersucht die Anwendungen der i^* -Modellierung, um eine Ontologie der Benutzerziele zu bilden, die als Hintergrundwissen für den Dienst benutzt werden sollte. Dabei werden drei einfache Empfehlungsalgorithmen entwickelt, die entsprechende Teile des Ontologie-Wissens in Form von intelligenten Vorschlägen zurückgeben, um dem Benutzer zu helfen, seine Bedürfnisse auszudrücken und interaktiv auf ihre Veränderung zu reagieren. Die Arbeit wird sich mit der Frage beschäftigen, ob dieser Ansatz eine gute Lösung für die Anforderungen von Xohana ist und er Probleme behebt, sowie wie gut die Leistung von Empfehlungsdiensten bei der Erstellung von Ziel-Empfehlungen, insbesondere im Vergleich mit anderen Empfehlungsdiensten, ist.

1.3 Aufbau der Arbeit

Um die Behandlung der oben genannten Fragen möglichst verständlich und nachvollziehbar zu gestalten, ist die Arbeit wie folgt aufgebaut: die Arbeit beginnt mit der oben beschriebenen Zielsetzung, mit welcher der Rahmen der Arbeit definiert ist (Kapitel 1). Der erste Teil des Kapitels 2 (*Literaturüberblick*) führt in die Themen NFR-Framework, i^* -Modellierung und Algorithmen ein, um sie anschließend in einen kontextuellen Zusammenhang zu bringen.

Welche Empfehlungsdienste zum heutigen Zeitpunkt vorhanden sind und welchen Zweck sie erfüllen, wird im zweiten Teil dieses Kapitels, ab 2.4 (*Empfehlungsdienste*) erläutert. Außerdem wird das Thema Evaluierung der Empfehlungsdienste beleuchtet.

Im Kapitel 3 (*Architektur des Ansatzes*) wird der praktische Teil der Arbeit präsentiert. Dabei wird einerseits die Architektur vorgestellt, andererseits ihr Ablauf und die verwendeten Algorithmen skizziert. Außerdem werden spezifische technische Aspekte und Abläufe mit Hilfe von Klassendiagrammen erläutert.

Mit der Evaluierung des entwickelten Empfehlungsdienstes beschäftigt sich Kapitel 4 (*Evaluierung*). Dort wird zuerst auf die Evaluierungsart, die Evaluierungsaspekte und die Evaluierungsumgebung eingegangen, sowie der Testablauf vorgestellt, um anschließend die Testergebnisse zu präsentieren.

Kapitel 5 (*Zusammenfassung und Ausblick*) fasst abschließend die wichtigsten Themen der Arbeit noch einmal kurz zusammen. Es werden die gesteckten Ziele mit den erarbeiteten Lösungen verglichen, um deren Erreichung darzustellen.

Literaturüberblick

In diesem Kapitel werden die notwendigen Grundlagen für die auszuführende Aufgaben vorgestellt. Als Einführung wird das Xohana Software System dargestellt. Der zweite Teil beschäftigt sich mit der Modellierung von Zielen. Im Rahmen dessen werden die Grundbegriffe der NFR- und i^* -Framework eingegeben sind. Der dritte Teil geht auf das Grundwissen der benötigten Algorithmen genauer ein, und der vierte Teil stellt die Empfehlungsdienste und ihre Evaluation vor.

2.1 Das Xohana Software System

Xohana ist ein Online Empfehlungsdienst, dessen Domäne der Gesundheitstourismus ist. Es handelt sich hier sowohl um ein Auswahl- als auch um ein Erforschungsproblem, da der Nutzer von einer endlichen Menge möglicher Urlaubsziele nur bestimmte davon besuchen möchte, er aber gleichzeitig nicht sicher ist, welche für ihn am interessantesten sind. Die Domäne ist von der mittleren Größe und die Gegenstände, die vorgeschlagen werden sollten sind homogen, da beim Vorschlagen die gleiche Art von Elementen empfohlen wird. Die Lebenserwartung der Empfehlungen ist auch mittelmäßig, begründet dadurch, dass die Ziele der Benutzer bei dieser Urlaubsart sich mit der Zeit nicht viel verändern.

Da ein Szenario beinhaltet an Krebs erkrankte Personen beinhaltet, die ihrem Gesundheitszustand entsprechend besondere Bedürfnisse bei ihren Freizeitaktivitäten, der Unterkunft und der Reise besitzen und ohne eine genauere Beachtung dieser kaum eine Möglichkeit haben, einen angenehmen Urlaub zu verbringen, sind die Benutzerpräferenzen von äußerster Bedeutung. Diese Touristen sind aber keine professionelle Urlaubssucher und wissen mehr darüber, was

sie möchten als wie das erreicht werden kann. Deshalb ist auch ein umfassendes Wissen über die Domäne und die Benutzer selbst eine Notwendigkeit. Die Benutzer haben bisher größtenteils schlechte Erfahrungen mit den Tourismus-Empfehlungsdiensten gemacht und müssen sorgfältig behandelt werden. Die Eingabe sollte so explizit wie möglich erfolgen, dem Nutzer aber nicht viel Mühe bereiten, weshalb eine angenehme Interaktion sehr vorteilhaft ist.

Wenn dem Nutzer aber *Xohana* gefällt, wird es mit der Zeit möglich, ausführliche Profile über die Benutzer zu bilden, sowie implizite Informationen und das Wissen der anderen Benutzer in der Empfehlungen miteinzubeziehen. Am Anfang kann aber nur das Wissen über die Domäne durch die Experten und Wissenserforschung gesammelt werden, da durch die geringe anfängliche Anzahl der Benutzer soziales Wissen nicht ohne den *cold-start* [17] Problem benutzt werden kann, was für eine kritische Domäne riskant ist. Der *Portfolio-Effekt* [3] ist in diesem Fall nicht von größerer Bedeutung, da beliebige Urlaubspakete wieder gern genommen werden könnten. Die Ausgaben sollen in Form von Empfehlungen, welche auf den Benutzeranforderungen basieren, dargestellt sein und klar und intuitiv wirken.

Eine der Prioritäten dieses Dienstes ist, eine tiefe Beziehung mit dem Benutzer herzustellen und diesem ermöglichen, das Vertrauen an die Empfehlungsdienste durch nützliche und intuitive Vorschläge zu gewinnen.

Allgemeine Vorgehensweise

New inquiry

Title:

<input type="text"/>	should be	<input type="text"/>	Remove
<input type="text"/>	must be	<input type="text"/>	Remove
<input type="text"/>	should be	<input type="text"/>	Remove

Add requirement

Abbildung 2.1: Eingabe-Interface

Zuerst kommt der Benutzer in Kontakt mit Eingabe Interface (Abbildung 2.1 auf der Seite 4), wo er seine Anforderungen (bzw. Ziele) ausdrücken sollte. Bevor der User mit der Eingabe beginnt (d.h. bevor der leere String an den Server

übergeben wird) bekommt er die Standardempfehlungen, die mit der jeweiligen Algorithmenart verbunden sind. Bei der Erzeugung von Empfehlungen, welche mit mehreren Anforderungen verbunden sind, und bei der Eingabe dieser, ist auf folgende Dinge zu achten:

- Die Empfehlungen dürfen sich nicht wiederholen, d.h. die Empfehlungen die schon vorgeschlagen sind, müssen aus der Liste der Ergebnisse entfernt werden.
- Falls man eine Anforderung eingeben möchte, und dabei beide Felder Kriterium (*Englisch: criterium*) und Nachfrage (*Englisch: demand*) leer sind, bezieht sich die Empfehlung für diese Zeile auf die Empfehlung für die vorherige Anforderungen, d.h. einer der oben genannten Suchalgorithmen wird auf vorigen Anforderung angewandt (natürlich müssen wieder bereits genützte Empfehlungen entfernt werden.)
- Falls die Suche nach der vorigen Anforderung keine Ergebnisse liefert, sieht sich der parametrisierte Algorithmus (deren Aufgabe es auch ist, den entsprechenden Algorithmus aufzurufen) den ganzen Baum an: zuerst überprüft er ob alle Ziele, die sich auf der höchsten Ebene im Baum befinden, vorgeschlagen sind, falls nicht, sendet er diese als Empfehlung, falls ja, geht er zur zweiten Ebene und überprüft ob alle Ziele von dieser Ebene vorgeschlagen sind usw..

Während der Benutzer seine Anforderungen ausdrückt, wird alles was er einträgt von einem Server, der diese Umgebung kontrolliert, empfangen und via HTTP an den Empfehlungsserver übermittelt, der mittels des Anforderungsprozessors diese Information richtig versteht und sich an den Algorithmus richtet, um das i^* -Modell entsprechend durchsuchen zu können, welche Ziele mit den Zielen der Benutzer verknüpft sind. Diese Auswahl wird dabei dem jeweiligen Algorithmus überlassen aber auch vom Baumprozessor aktiv unterstützt. Der Empfehlungsserver übermittelt dann dem Eingabeserver eine XML-Datei mit den ausgewählten Zielen, welche vom Ausgabeprozessor (der vom Algorithmus berufen wurde, wenn die Ergebnisse vorhanden waren) früher in dieser Datei rechtsgemäß eingetragen wurden, zurück. Das ganze Prozess findet interaktiv statt. Dem Benutzer ist die hintergründige Arbeit nicht bewusst, er wird dabei aber aktiv unterstützt, da er nur die hilfreichen Empfehlungen im Interface-Fenster sieht und sich mit dem Background nicht beschäftigen muss.

2.2 Modellierung von Zielen

Damit den Xohana-Benutzern die Ziele bei ihrer Urlaubssuche empfohlen werden könnten, muss ein ausführliches Modell dieser Ziele gebaut werden. Zu diesen Zwecken wurden die Prinzipien der i^* -Modellierung und des NFR-Frameworks ausgenutzt. In den folgenden Abschnitten werden diese kurz eingeführt. Kapitel 3 (*Architektur des Ansehens*) wird an ihrer Umsetzung im *Xohana Software System* angehen.

NFR Framework

Um die Ziele der Benutzer modellieren zu können, muss zuerst das Prinzip des Softgoals, welches im NFR Framework [7] eingeführt wurde, erklärt werden.

Der NFR Framework ist für die Repräsentation und die Erläuterung der nicht-funktionellen Anforderungen (Engl. «*non-functional requirements*»), die ein wichtiger Teil der Systementwicklung sind, gedacht. Im Gegensatz zu den *funktionellen Anforderungen*, die Funktionen des Systems/Software-Systems oder einer Software-Komponente sind, bei welchen das System oder die Komponente in der Lage sein muss, ausführen zu können, beschreiben die nicht-funktionelle Anforderungen nicht was die Software tun sollte, sondern wie sie das machen kann [42]. Da sie sich mit der Qualität des Software-Systems beschäftigen, ist ihre sorgfältige Untersuchung von großer Bedeutung, bringt aber zusätzlichen Aufwand, da sie miteinander interagieren (sich helfen, verletzen u.s.w.) und ihre Interpretation, Wichtigkeit und Ausführung relativ zu dem untersuchten System sind und infolgedessen variieren.

Wegen dieser Eigenschaften kann man die Begründung der nicht-funktionellen Anforderungen nicht dualistisch (e.g. *wahr* oder *falsch* - *befriedigt* oder *nicht*) behandeln, da eine klare Aussage darüber nicht möglich ist. Deswegen wird im NFR-Framework in solchem Fall von Herbert Simon [23] der in 1950 entstandene Begriff *Satisfizierung* (Engl. «*satisficing*») eingeführt, um eine *ausreichend befriedigte* nicht - funktionelle Anforderung zu beschreiben, und daher die Begründung dialektisch auszuführen.

Die nicht-funktionellen Anforderungen werden im NFR-Framework als *Softgoals* behandelt. Ob diese «*satisficed*» oder nicht «*satisficed*» sind wird dann durch die Analyse der *gegenseitigen Abhängigkeiten der Softgoals* (Engl. «*softgoal interdependencies*») weiter überprüft. Die Visualisierung der beiden Komponentenarten erfolgt in den Graphen der gegenseitigen Abhängigkeit (Engl. «*Softgoal Interdependency Graphs*», «*SIGs*»). Weitere Komponenten des NFR-Frameworks sind das *Evaluationsverfahren*, das festsetzt, in welchem Ausmaß die nicht-funktionelle Anforderung «*satisficed*» ist, die *Methoden*, die durch Verfeinerung der Softgoals realisiert sind und eine detaillierte Liste der Entwick-

lungstechniken anbieten und *Korrelationen*, die Kataloge der möglichen Zusammenhänge zwischen den Softgoals vorsehen.

Softgoals

«Der Meilenstein des (NFR-)Frameworks ist das Konzept des Softgoals, das ein Ziel repräsentiert, welches keine klare Definition und/oder Kriterium ob er zufrieden gestellt ist oder nicht, hat» [7].

NFR-Softgoals

Wenn die Softgoals identifiziert sind, fängt der Entwickler mit dem Zerlegen jedes NFR-Softgoals in mehrere spezifische Untersoftgoals (Engl. «*subsoftgoals*») an. Da diese einen *NFR-Typ* der auf die bestimmte nicht-funktionelle Anforderung die durch das Softgoal beschreibt ist hinweist, und ein *Thema*, das die Essenz des Softgoals beschreibt, hat, kann der Entwickler zwischen einer *Typ-* oder einer *Thema-*Dekomposition auswählen, in Abhängigkeit davon, ob sich die Untersoftgoals auf die gleiche nicht-funktionelle Anforderung beziehen oder ein gemeinsames Thema haben. Wenn alle Softgoals verbraucht sind, damit das obere Softgoal erfüllt wird, benutzt man eine *AND-Dekomposition*, wenn nur ein Softgoal genügt, steht die *OR-Dekomposition* zur Verfügung.

Operationalisierte Softgoals

Ein anderer Softgoaltyp der oben beschriebenen *NFR-Softgoals* sind die *operationalisierten Softgoals*, die die Kluft zwischen den NFR-Softgoals und den gewünschten Lösungen für das System überbrücken. Wenn die NFR-Softgoals ausreichend verfeinert sind, werden diese in den Entwicklungstechniken operationalisiert, die man im NFR-Framework als *operationalisierte Softgoals* bezeichnet. Diese können ihre oberstehende Softgoals wieder *positiv* oder *negativ* beeinflussen.

Claims

Der dritte Softgoaltyp sind *Behauptungen* (Engl. «*claims*»), die ermöglichen, die Entwicklungsentscheidungen gut argumentieren zu können. Die *Behauptungs-Softgoals* können benutzt werden, um die Gründe für die Verfeinerungen, die Wahl einer bestimmten Alternative oder *tradeoff* erklären zu können. Im Gegensatz zu anderen Softgoaltypen beziehen sich die Behauptungen auf die Darlegung selbst, d.h. auf gegenseitige Beziehungen der Softgoals anstatt direkt auf die Softgoals.

Die Softgoals werden dann im *i** Framework [49] weiterverwendet.

In dem folgenden Abschnitt wird unter Benutzung von [46] eine Übersicht über i^* -Modellierungsframework zusammengefasst.

i^* Framework

Der i^* -Framework ist für die Analyse, das Design und die Modellierung von geschäftlichen Prozessen gedacht [46]. Der Framework nimmt Bezug darauf, dass die heutigen Beziehungen und Prozesse in einer Multi-Agenten Welt, in welcher die Agenten voneinander abhängig sind, passieren. Deshalb besteht ein i^* -Modell üblicherweise aus Akteuren, welche für das Erreichen von Softgoals voneinander abhängig sind, aus Zielen die durchzusetzen sind, aus Aufgaben, die zu erledigen sind sowie aus Ressourcen, die die Bedürfnisse decken sollten [49]. i^* präsentiert daher eine Agenten- und Ziel-orientierte Modellierungsparadigma, die sich mit den Verhältnissen der Agenten auf der intentionellen Ebene beschäftigt. Die i^* -Modellierungssprache wird gewöhnlich für die Modellierung in der frühen Phase der System-Modellierung benutzt, um das Problem domain besser verstehen zu können.

Der i^* -Framework besteht aus zwei Modellarten: dem Modell der strategischen Abhängigkeiten (Engl. «*Strategic Dependency Model, SD*») und dem Modell der strategischen Begründungen (Engl. «*Strategic Rationale Model, SR*»).

Modell der strategischen Abhängigkeiten (SD)

Das Modell der strategischen Abhängigkeiten (Engl. «*Strategic Dependency Model*») fokussiert sich auf Abhängigkeitsbeziehungen. Dieses Modell beschreibt das Netzwerk der intentionellen Beziehungen zwischen Akteuren [48]. Er besteht aus Knoten und Verknüpfungen, wobei die Knoten, die Akteure und die Verknüpfungen in verschiedenen Abhängigkeiten (Ziel-Abhängigkeiten, Aufgabe-Abhängigkeiten, Ressourcen-Abhängigkeiten und Softgoal-Abhängigkeiten) zueinander stehen.

Akteure

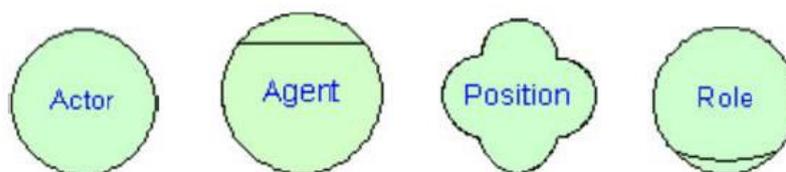


Abbildung 2.2: Akteure [47]

Akteure (Engl. «*Agents*») sind aktive Einrichtungen, die Maßnahmen zur Erreichung von Zielen durch die Ausübung ihres *Know-how* enthalten [50].

Die Aktoren (Engl. «*Actors*»), siehe Abbildung 2.2, Seite 8, können in Agenten (Engl. «*agents*»), Rollen (Engl. «*roles*») und Positionen (Engl. «*positions*») unterteilt werden [46].

Agenten sind Akteure mit konkreter Manifestation, wie z.B. eine menschliche Person oder ein Software/Hardware Agent.

Rollen präsentieren eine Zusammenfassung der Charakterisierung des Verhaltens eines sozialen Akteurs in einem speziellen Kontext oder Bereich der Bemühung.

Positionen bezeichnen eine vermittelnde Abstraktion, die zwischen einer Rolle und einem Agent benutzt werden kann.

Akteure-Verbindungen-Links

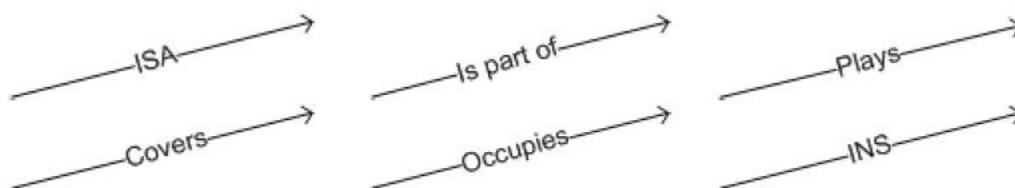


Abbildung 2.3: Akteure-Verbindungen-Links [47]

Akteure-Verbindungen-Links (Engl. «*Actor Association Links*») in Abbildung 2.3, Seite 9, beschreiben graphisch die Beziehungen zwischen unterschiedlichen Akteuren [46].

Ist-Teil-Von Verbindung(Engl. «*Is-part-of, Part Association*») ist eine intentionelle Abhängigkeit zwischen einem Akteurstyp (Agent, Rolle, Positione) als eine Gesamtheit und seinen Bestandteilen, die zum Aufrechterhalten der Einheit gedacht sind.

Ist-Verbindung (Engl. «*IsA Association*») entsteht, wenn man einen Akteur als einen Spezialfall eines anderen Akteurs bezeichnen möchte.

Spielt-Verbindung (Engl. «*Plays Association*») bezeichnet eine Verbindung zwischen Agenten und einer Rolle, die dieser Agent spielt.

Deckt-Beziehung (Engl. «*Covers Relationship*») ist eine Beziehung zwischen einer Position und einer Rolle, die diese Position deckt.

Besetzt-Beziehung (Engl. «Occupies Relationship») zeigt vor, dass ein Agent alle Rollen die von einer Position gespielt werden, besetzt.

Instanziierungsbeziehung (Engl. «Instantiation, INS Relationship») präsentiert die Verknüpfung zwischen einer generalisierten Einheit und ihrer Instanz.

Strategische Abhängigkeiten

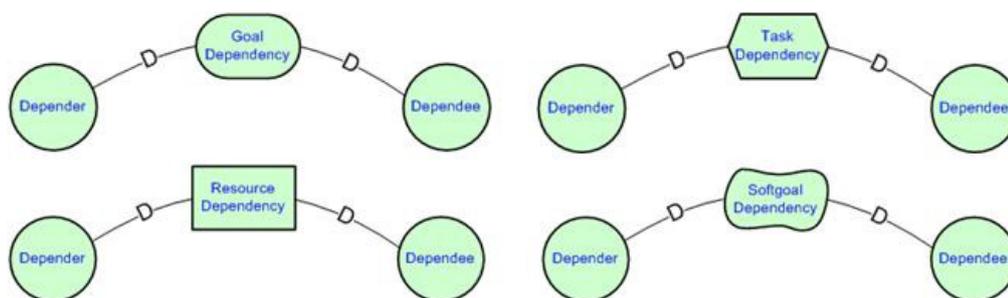


Abbildung 2.4: Strategische Abhängigkeiten [47]

Eine strategische Abhängigkeit (Engl. «*Strategic Dependency*») - Abbildung 2.4, Seite 10 - besteht aus einem *Verlassenden* (Engl. «*Dependee*»), einem *Abhängigen* (Engl. «*Depender*») und einem *Dependum*, um das sich die Beziehung fokussiert. Der Verlassende ist der Akteur, auf den man sich in einer Abhängigkeitsbeziehung verlässt, der zweite Akteur, der in dieser Beziehung vom Verlassenden abhängig ist, bezeichnet man als den Abhängigen, und das Objekt dieser Beziehung ist das *Dependum* [46]. Man unterscheidet zwischen folgenden Abhängigkeiten:

Ziel-Abhängigkeit (Engl. «Goal Dependency»), wo der Abhängige sich auf den Verlassenden verlässt, um ein bestimmtes Ziel zu erreichen,

Aufgabe-Abhängigkeit (Engl. «Task Dependency»), wo der Abhängige vom Verlassenden erwartet, eine bestimmte Aktivität in die Tat umzusetzen,

Ressource-Abhängigkeit (Engl. «Resource Dependency»), wo der Abhängige von dem Verlassende abhängig ist, eine bestimmte Einheit zur Verfügung zu stellen und

Softgoal-Abhängigkeit (Engl. «Softgoal Dependency»), wo der Abhängige vom Verlassende erwartet, eine Aufgabe zu erledigen, welche zum Erreichen eines bestimmten Softgoals führen sollte.

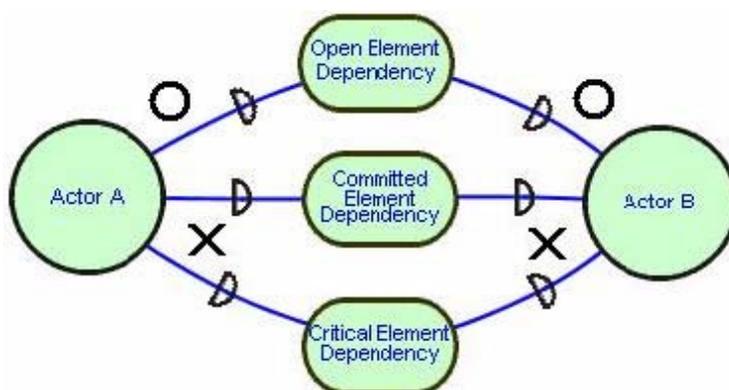


Abbildung 2.5: Verwundbarkeitsarten [47]

Verwundbarkeit (Engl. «*Vulnerability*»)

Da der Abhängige vom Verlassenden abhängig ist und auch die Möglichkeit besteht, dass die Erwartungen an den Verlassenden von ihm nicht erfüllt werden, wird der Abhängige durch den Verlassenden *verwundbar* (Engl. «*vulnerable*»). Abgesehen von der Stärke kann diese Verwundbarkeit drei verschiedene Formen annehmen [46]:

Offene Abhängigkeit (Engl. «*Open dependency*»): bezeichnet man als «O» auf der passenden Seite der Verknüpfung. Der Effekt des Nicht-Erlangens des Dependens ist vorhanden, aber nicht schwerwiegend.

Verpflichtete Abhängigkeit (Engl. «*Committed dependency*»): Eine Aktivität für die Erreichung eines Zieles des Dependens wird durch Nicht-Erlangen des Dependens versagen.

Kritische Abhängigkeit (Engl. «*Critical dependency*»): ist durch ein «X» auf der betroffenen Seite der Abhängigkeit repräsentiert. Das Nicht-Erlangen des Dependens wird verursacht, dass alle Aktivitäten, die der Depender um ein Ziel zu erreichen zu unternehmen geplant hat, nicht erreicht werden.

Modell der strategischen Begründungen (SR)

Modell der strategischen Begründungen (Engl. «*Strategic Rationale Model*», «*SR-Modell*») ist mehr detailliert bei der Modellierung - als bei der Beschreibung der «äußeren» Verhältnisse zwischen Akteuren. Hier werden «interne» Elemente (Aufgaben (Engl. «*Tasks*»), Ziele (Engl. «*Goals*»), Ressourcen (Engl. «*Resources*») und sanfte Ziele (Engl. «*Softgoals*»)) benutzt um die internen intention-

nelle Beziehungen (die innerhalb des Akteurs passieren) darzustellen. Diese Beziehungen werden mittels Strukturen der Mittel-Zweck- (Engl. «*Means-Ends*»), Aufgaben-Dekomposition- (Engl. «*Task Decomposition*») und Beitrags- (Engl. «*Contribution*») -Beziehungen arrangiert [46].

Abgrenzung/Akteurabgrenzung

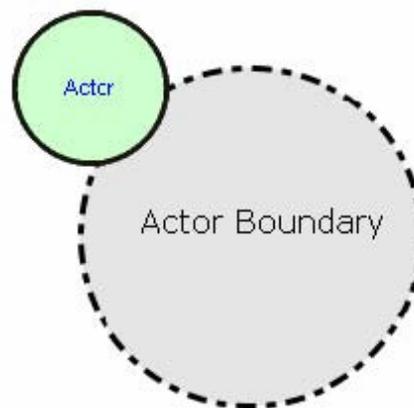


Abbildung 2.6: Akteurabgrenzung [47]

Die Akteurabgrenzung (Engl. «*Actor Boundary*») in Abbildung 2.6, Seite 12) gibt die Grenzen der Absichtlichkeit eines bestimmten Akteurs an [46]. Alle Elemente, die sich innerhalb dieser Abgrenzung befinden, sind ein Teil des intentionellen Verhaltens dieses Akteurs. Wenn diese von den intentionellen Elementen eines anderen Akteurs abhängig sind, werden sie mit den Abhängigkeitslinks, die sich über verschiedene Akteurabgrenzungen erstrecken, verbunden.

Elemente

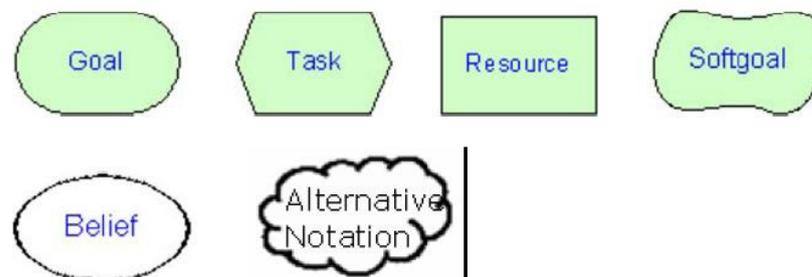


Abbildung 2.7: Elemente [47]

Es gibt 4 Arten von intentionellen Elementen (Engl. «*Elements*»), in Abbildung 2.7, Seite 12: die Ziele (Engl. «*Goals*»), die sanften Ziele (Engl. «*Softgoals*»), die Aufgaben (Engl. «*Tasks*»), die Ressourcen (Engl. «*Resources*») und die Vorstellungen (Engl. «*Claims*») [46].

Die Ziele beschreiben intentionelle Bedürfnisse eines Akteurs. Außerdem ist ihre Erfüllung durch ein klares Kriterium definiert.

Die Softgoals sind den (harten) Ziele sehr ähnlich, mit der Ausnahme, dass die Kriterien für die Zufriedenstellung des Softgoals nicht eindeutig sind.

Die Aufgaben wünscht ein Akteur in bestimmter Art und Weise zu erfüllen.

Ressourcen sind informationelle oder physische Entitäten die dem Akteur zu Verfügung gestellt werden sollten.

Vorstellungen sind Bedingungen über die Welt, über die der Akteur glaubt dass sie wahr sind, obwohl sie nicht wahr sein müssen.

Mittel-Zweck-Links

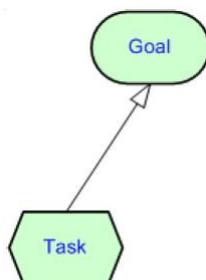


Abbildung 2.8: Mittel-Zweck-Link [47]

Mittel-Zweck Link (Engl. «*Means-Ends-Link*») - siehe Abbildung 2.8, Seite 13 - ist eine Beziehung zwischen einem Ende, das durch ein Ziel-Element repräsentiert wird, und einem Mittel zu seiner Erreichung, die eine Aufgabe ist [46].

Dekomposition-Links

Ein Aufgabe-Element kann in verschiedene Unterelemente zerlegt («dekompositioniert») sein (siehe Abbildung 2.9 auf der Seite 14). Abhängig davon, um welches Element es sich auf der anderen Seite des Dekomposition-Links (Engl. «*Decomposition Link*») handelt, wird zwischen *Aufgabe/Ziel - Dekomposition*, *Aufgabe/Aufgabe - Dekomposition*, *Aufgabe/Ressource - Dekomposition* und *Aufgabe/Softgoal - Dekomposition* unterschieden [46].

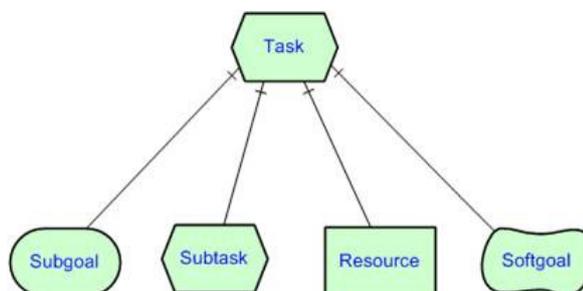


Abbildung 2.9: Dekomposition-Links [47]

Beitragslinks

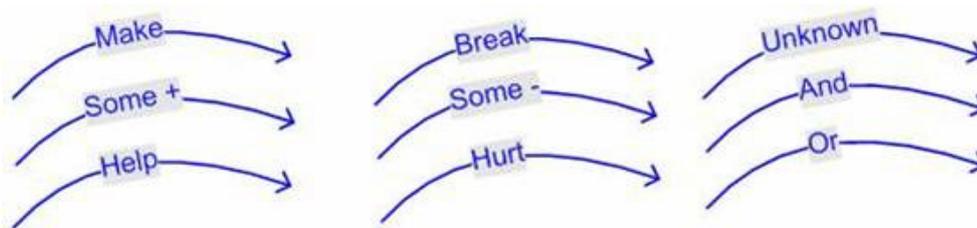


Abbildung 2.10: Beitragslinks [47]

Die Beitragslinks kann man in *Make*, *Some+*, *Help*, *Unknown*, *Break*, *Some-*, *Hurt*, *Or* und *And* Links unterteilen.

Make: Positiver Beitrag der stark genug ist, ein Softgoal zu *satisfizieren*

Some+: Positiver Beitrag über dessen Stärke keine Informationen verfügbar sind, kann *Make* oder *Help* Beitrag sein

Help: Teilweise positiver Beitrag, der von sich aus nicht in der Lage ist, ein Softgoal zu *satisfizieren*

Unknown: Beitrag, dessen Polarität nicht bekannt ist

Break: Negativer Beitrag, der stark genug ist, ein Softgoal abzuweisen

Some-: Negativer Beitrag über dessen Stärke keine Informationen verfügbar sind, kann *Break* oder *Hurt* Beitrag sein

Hurt: Teilweise negativer Beitrag, der von sich aus nicht in der Lage ist, ein Softgoal abzuweisen

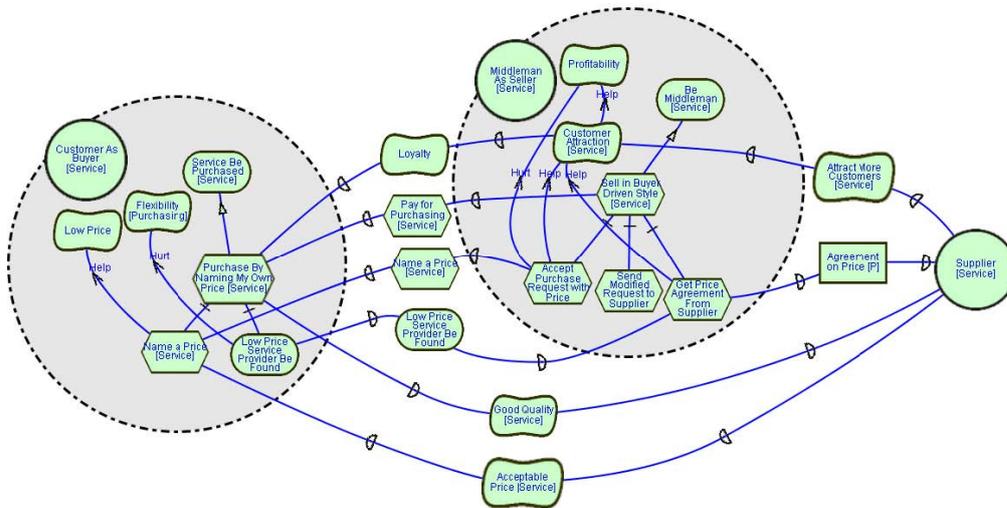


Abbildung 2.11: SR-Modell für ein kunden-betriebenes E-Commerce System mit Mittelsmann [48]

Or: Der Elternteil ist satisfiziert, wenn einer der Nachkommen satisfiziert (Engl. «satisficed») ist

And: Der Elternteil ist satisfiziert, wenn alle Nachkommen *satisfiziert* sind

Beispiel eines möglichen SR-Modells

Ein Beispiel eines möglichen SR-Modells für ein kunden-getriebenes E-Commerce System mit Mittelsmann ist in der Abbildung 2.11 ([48]) (Seite 15) graphisch dargestellt.

Das Modell stellt graphisch die interne logischen Begründungen (Engl. «*rationale*») die zwischen einem Kunden und einem Mittelsmann bei der Verhandlung stattfinden.

2.3 Einfache Suchalgorithmen

Wenn das Modell der Ziele mittels t^* gebaut wird, entsteht ein baumförmiger Graph, der als Basis für den Empfehlungsprozess dient. Für die Durchsuchung dieses Graphen braucht man eine Suchstrategie, mit der in Abhängigkeit von der Art der Suchstrategie, einen Teil der Ziele in diesem Graphen, welche der Benutzereingabe entsprechen, auswählen kann.

Der entwickelte Graph wird in dieser Arbeit als ein *Suchbaum* behandelt. Ein Suchbaum wird von einem *Anfangszustand* und einer *Nachfolgerfunktion* erzeugt, die gemeinsam einen Zustandsraum definieren. Die Wurzel des Suchbaumes ist ein Suchknoten, der dem Anfangszustand entspricht. In einem ersten Schritt ist zu prüfen, ob dies ein *Zielzustand* ist. Wenn dies nicht der Fall ist, muss man die anderen Zustände in Erwägung ziehen. Dies wird durch das Erweitern des aktuellen Zustandes gemacht; die Nachfolgerfunktion wird auf den aktuellen Zustand angewendet und dadurch ein neues Zuständeset erzeugt. Die Entscheidung, welcher Zustand als nächstes ausgedehnt werden soll, hängt von der *Suchstrategie* ab [36].

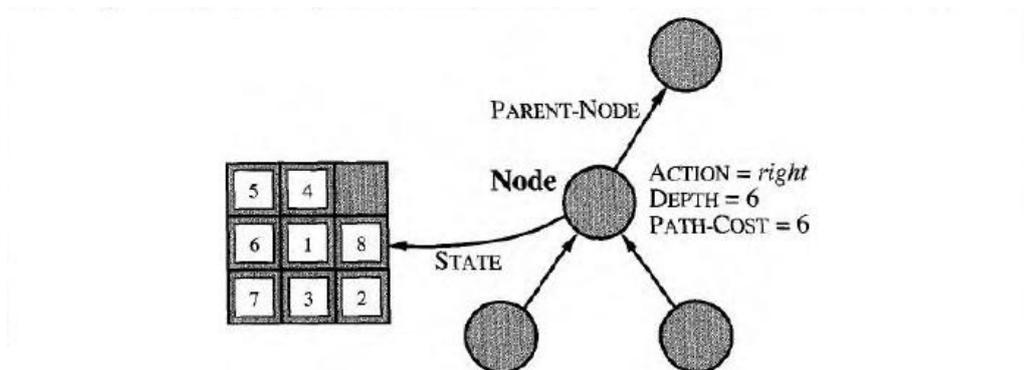


Abbildung 2.12: Ein Knoten im Suchbaum [36]

In [36] wird ein Knoten (Abbildung 2.12, Seite 16) als eine Datenstruktur mit 5 Bestandteilen definiert:

Zustand (Engl. «state»): der Zustand im Zustandsraum, dem der Knoten entspricht;

Elternknoten (Engl. «parent node»): der Knoten im Suchbaum, der diesen Netzknoten erzeugte;

Handlung (Engl. «action»): die Handlung, die auf den Elternteil angewandt wurde, um den Knoten zu erzeugen;

Pfadkosten (Engl. «*path cost*»): die angedeuteten Kosten des Pfades vom Anfangszustand bis zum Knoten und

Tiefe (Engl. «*depth*»): die Anzahl der Schritten am Pfad vom Anfangszustand.

Man muss aber auch die Sammlung von Knoten darstellen, die erzeugt, aber noch nicht ausgedehnt wurden - diese Sammlung wird in [36] als *Fringe* bezeichnet. Jedes Element der *Fringe* ist ein Blattknoten (Engl. «*leaf node*») - ein Knoten, der keine Nachfolger hat.

Der allgemeine Baum-Suchalgorithmus aus [36] ist in der Abbildung 2.13 auf der Seite 17 beschrieben.

```

function TREE-SEARCH(problem, fringe) returns a solution, or failure
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if EMPTY?(fringe) then return failure
    node ← REMOVE-FIRST(fringe)
    if GOAL-TEST[problem] applied to STATE[node] succeeds
      then return SOLUTION(node)
    fringe ← INSERT-ALL(EXPAND(node, problem), fringe)

function EXPAND(node, problem) returns a set of nodes
  successors ← the empty set
  for each (action, result) in SUCCESSOR-FN[problem](STATE[node]) do
    s ← a new NODE
    STATE[s] ← result
    PARENT-NODE[s] ← node
    ACTION[s] ← action
    PATH-COST[s] ← PATH-COST[node] + STEP-COST(STATE[node], action, result)
    DEPTH[s] ← DEPTH[node] + 1
    add s to successors
  return successors

```

Abbildung 2.13: Allgemeiner Baum-Suchalgorithmus [36]

Im Allgemeinen kann man bei der Suchstrategie im Graphen zwischen uninformierte (Engl. «*uniformed search*») und informierte Suche (Engl. «*informed search*») unterscheiden [36].

Uninformierte Suchstrategien

Bei der uniformierten Suche (auch *Blindesuche* genannt) hat man keine zusätzlichen Informationen über die Zustände außer derjenige die in der Problemde-

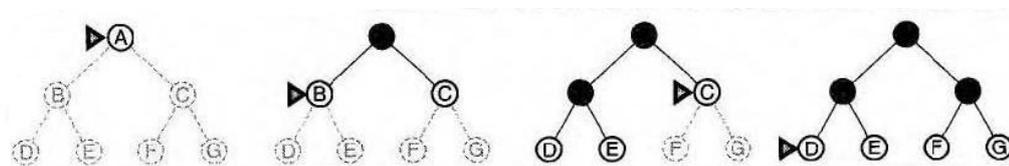


Abbildung 2.14: Breittensuche auf einem einfachen binären Baum. In jeder Phase ist der Knoten, der als nächstes erweitert wird, durch einen Marker gekennzeichnet [36].

definition vorausgesetzt sind. Alles was diese Sucharten machen können ist die Nachfolger zu generieren und zu unterscheiden, ob der Zielzustand erreicht wurde oder nicht [36]. Die bekanntesten uninformierten Sucharten sind die Breittensuche (Engl. «*breadth-first search*»), die uniformierte Kostensuche (Engl. «*uniform-cost search*»), die Tiefensuche (Engl. «*depth-first search*»), die Tiefebeschränkte Suche (Engl. «*depth-limited search*»), die iterative Vertiefungssuche (Engl. «*iterative deepening search*») und die bidirektionale Suche (Engl. «*bidirectional search*»). Diese Arbeit beschäftigt sich mit der Anwendung der Breittensuche und der Tiefensuche.

Breittensuche

Die allgemeine Breittensuche (Engl. «*Breadth-First Search, BFS*») ist ein Verfahren, in dem zuerst der Wurzelknoten untersucht und expandiert wird. Wenn in seinem direkten Nachfolger kein Zielzustand enthalten ist, geht die Suche weiter und expandiert deren Nachfolger und so weiter, d.h., dass alle Knoten, die sich auf einer Ebene e befinden vor alle Knoten, die sich auf der Ebene $e + 1$ befinden, expandiert werden müssen [36]. Die Reihenfolge der Ausweitung kann man in der Abbildung 2.14 (Seite 18) ansehen.

Tiefensuche

Die allgemeine Tiefensuche (Engl. «*Depth First Search - DFS*») geht bei ihrer Arbeit, wie die Name schon sagt, von der Tiefe aus: immer wird die Knoten erweitert, der sich auf der tiefsten Ebene im Baum befindet. Nur wenn es im Tiefstand keine Knoten mehr auszudehnen gibt, geht die Suche zurück und expandiert die Knoten auf einer höheren Ebene. Die Arbeitsweise des Algorithmus kann man in der Abbildung 2.15 (aus [36]) auf der Seite 19 genauer betrachten.

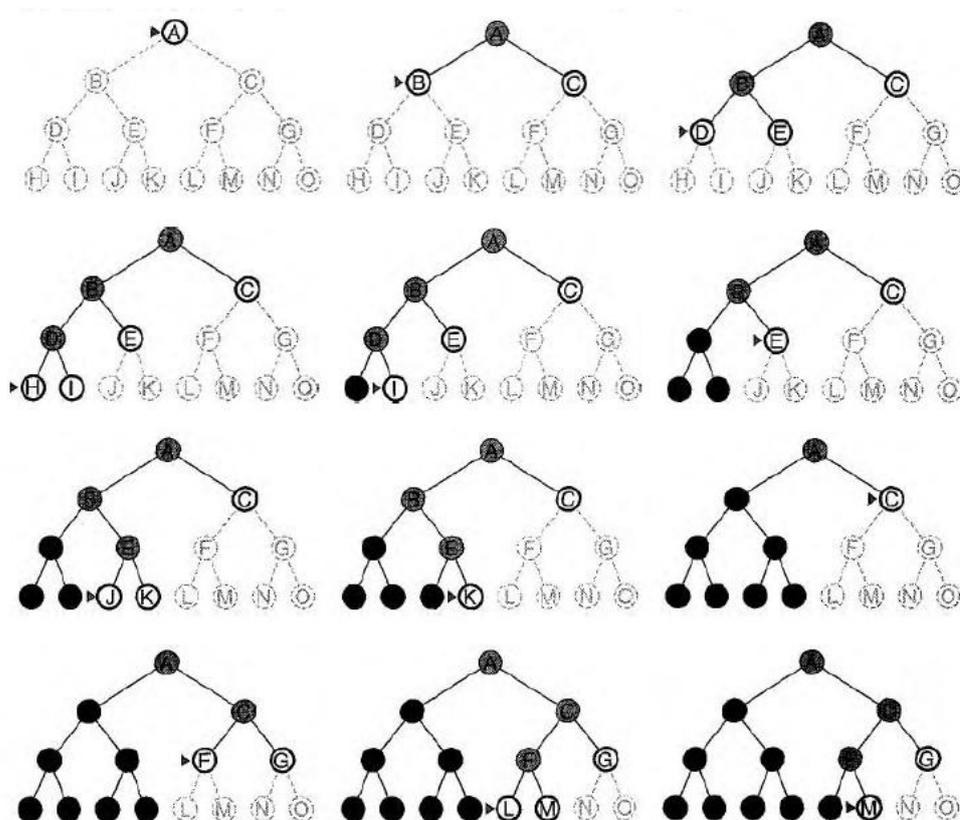


Abbildung 2.15: Tiefensuche auf einem binären Baum [36].

Informierte Suchstrategien

Im Gegensatz zur uninformierten Suchstrategien, sind bei den informierte Suchstrategien einige relevante Daten über den Graphen schon vorhanden (diese Art der Suche wird auch *heuristische Suche* genannt). Diese Suchstrategieart benutzt das Problem-spezifische Wissen, um die Baumsuche effektiver zu machen [36].

Der allgemeine Ansatz in diesem Bereich wird Beste-Erste Suche (Engl. «*best-first search*») genannt. Beste-Erste Suche ist ein Beispiel des allgemeinen Graph-Suchalgorithmus, in dem ein Knoten zum Ausdehnen anhand einer Evaluierungsfunktion $f(n)$ gewählt wird [36]. Der Knoten mit dem niedrigsten Evaluierungswert wird für Ausdehnung ausgewählt, weil die Evaluation die Entfernung bis zum Ziel misst. Allerdings muss man hier in Betracht nehmen, dass die Beste-Erste Suche den Knoten auswählt der dem Ziel am nächsten *erscheint* und deshalb auch ungenau sein kann - denn wenn dies immer der Fall wäre, würde man überhaupt keine Suchstrategie brauchen.

Es existiert eine ganze Familie von Beste-Erste Suchalgorithmen. Der Haupt-

teil all dieser Algorithmen ist eine *heuristische Funktion*, die man als $h(n)$ bezeichnen kann:

$$h(n) = \text{geschätzte Kosten für den billigsten Pfad von Knoten } n \text{ zu einem Zielknoten [36]}$$

Heuristische Funktionen können vereinfacht als beliebige Problem - spezifische Funktionen definiert werden, bei denen gilt: wenn n der Zielknoten ist, dann ist $h(n) = 0$ [36].

Einige bekannte Vertreter dieser Suchstrategiefamilie sind vor allem A^* - und *Greedy Beste-Erste Suche* (Engl. «*greedy best-first search*»).

In dieser Arbeit wird das Wissen über informierte Suchstrategien benutzt, um einen eigenen heuristischen Ansatz zu entwickeln.

Greedy Beste-Erste Suche

Greedy Beste-Erste Suche bemüht sich den Knoten auszudehnen, der dem Ziel am nächsten ist, mit der Begründung, dass diese Vorgangsweise mit großer Wahrscheinlichkeit schnell zu einer Lösung führt. Das bedeutet, sie evaluiert den Knoten in dem sie die heuristische Funktion $f(n) = h(n)$ verwendet [36].

A* Suche

Die Evaluierungsfunktion bei der A^* -Suche kombiniert $g(n)$, die Kosten um den Knoten zu erreichen und $h(n)$, die Kosten, um vom Knoten bis zum Zielknoten zu kommen:

$$f(n) = g(n) + h(n)[36]$$

Weil $g(n)$ die Pfadkosten für den Anfangsnetzknoden zum Knoten n und $h(n)$ die geschätzten Kosten vom billigsten Pfad von n zum Ziel darstellt, erhält man:

$$f(n) = \text{geschätzte Kosten für die billigste Lösung durch } n \text{ [36]}$$

Deshalb fängt man hier bei der Suche nach der billigsten Lösung mit dem niedrigsten Wert von $g(n) + h(n)$.

2.4 Empfehlungsdienste

Empfehlungen sind ein erfolgreiches Beispiel der Anwendung der künstlichen Intelligenz (AI). Ihre Geschichte ist aber durch schnelle Dynamik, häufige Aktualisierungen und einen großen Bedarf nach vollständiger Klassifizierung charakterisiert. Dieser Abschnitt beschäftigt sich mit der Diskussion ausgewählter bisheriger Erkenntnisse. Der Abschnitt befasst sich mit den Definitionen, der Unterteilung und den Klassifizierungsdimensionen der Empfehlungsdienste, fokussiert sich dann auf die wissensbasierten Empfehlungsdienste, beschreibt als nächstes die bestehenden Evaluationsansätze und schließt mit dem Evaluationsvorschlag für *Xohana* ab.

Überblick

Definitionen

Wie im Einleitungskapitel bereits erwähnt, entwickelt sich die Definition des Empfehlungsdienstes ziemlich rasch. Die Tabelle 2.1 (aus [22]) auf der Seite 22 zeigt einen Überblick über interessante verwandte Definitionen der letzten 17 Jahre. Es ist leicht zu beobachten, dass Empfehlungsdienste am Anfang nur mit kollaborativer Filtrierung verknüpft waren, die inhaltsbasierten und wissensbasierten Empfehlungsdienste mit der Zeit jedoch immer mehr an Bedeutung gewonnen haben.

In dieser Arbeit wird die Definition in [3] benutzt:

«Der Empfehlungsdienst ist jeder Dienst, der den Benutzer in einer persönlichen Art zu interessanten oder nützlichen Gegenstände durch einen großen Raum von möglichen Optionen führt oder solche Gegenstände als Ausgabe erzeugt.» [3]

Unterteilung

Die Klassifikation der Empfehlungsdienste hat parallel im Laufe der Zeit gleichzeitig immer mehr an Komplexität bekommen. Eine detaillierte Unterteilung ist in mehreren Forschungsarbeiten gegeben, wobei [1], [3] und [30] bemerkenswerter als die anderen erscheinen.

In [1] wird eine gute Einführung in den Forschungsbereich der Empfehlungsdienste sowie die klassische Einordnung der Empfehlungsdienste angegeben. *Adomavicius und Tuzhilin* unterscheiden zwischen:

Nr.	Quelle	Definition	Jahr
1	[15]	„Collaborative filtering simply means that people collaborate to help one another perform filtering by recording their reactions to documents they read.“	1992
2	[33]	„Collaborative filters help people make choices based on the opinions of other people.“	1994
3	[40]	„Social information filtering essentially automates the process of ‘word-of-mouth’ recommendations: items are recommended to a user based upon values assigned by other people with similar taste.“	1995
4	[34]	„In a typical recommender system people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients.“	1997
5	[26]	„The term ‘collaborative filtering’ describes techniques that use the known preferences of a group of users to predict the unknown preferences of a new user; recommendations for the new users are based on these predictions. Other terms that have been proposed are ‘social information filtering’ and ‘recommender system’.“	1999
6	[8]	„In collaborative filtering, entities are recommended to a new user based on the stated preferences of other, similar users.“	2000
7	[37]	„Recommender systems use product knowledge—either hand-coded knowledge provided by experts or ‘mined’ knowledge learned from the behavior of consumers—to guide consumers through the often-overwhelming task of locating products they will like.“	2001
8	[3]	„Recommender systems represent user preferences for the purpose of suggesting items to purchase or examine...(the term describes) any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options.“	2002
9	[20]	„Recommender systems use the opinions of members of a community to help individuals in that community identify the information or products most likely to be interesting to them or relevant to their needs.“	2004
10	[17]	„Recommender systems use the opinions of a community of users to help individuals in that community more effectively identify content of interest from a potentially overwhelming set of choices.“	2004
11	[10]	„Recommender systems—a personalized information filtering technology used to either predict whether a particular user will like a particular item (prediction problem) or to identify a set of N items that will be of interest to a certain user (top-N recommendation problem).“	2004
12	[16]	„Recommender system is a system that helps users to find their wanted items by making recommendations based on either the content of the recommended items (content-based filtering), or ratings of similar users on the recommended items (collaborative filtering).“	2004
13	[18]	„A personalized recommendation system can provide one-to-one service to customers based on customers’ past behavior and through inference from other users with similar preferences. The aim of personalization is to offer customers what they want without asking explicitly and to capture the social component of interpersonal interaction.“	2005
14	[19]	„A recommender system is a typical software solution used in e-commerce for personalized services. It helps customers find the products they would like to purchase by providing recommendations based on their preferences, and is partially useful in e-commerce sites that offer millions of products for sale.“	2005
15	[39]	„Recommender systems suggest items of interest to users based on their explicit and implicit preferences, the preferences of other users, and user and item attributes.“	2005
16	[24]	„A recommender system is the information filtering that applies data analysis techniques to the problem of helping customers find the products they would like to purchase by producing a predicted likeness score or a list of recommended products for a given customer.“	2005

Tabelle 2.1: Überblick über ausgewählte Definitionen von Empfehlungsdiensten [22]

Inhaltsbasierte Empfehlungen (Engl. «*content-based recommendations*»), sind als Empfehlungen von Gegenständen, welche den, vom Benutzer früher präferierten, ähnlich sind, definiert,

Kollaborative Empfehlungen (Engl. «*collaborative recommendations*»), sind Empfehlungen der Gegenstände an den Benutzer, die andere Benutzer mit ähnlichem Geschmack und ähnlichen Vorlieben vorgezogen haben, und

Hybride Empfehlungen (Engl. «*hybrid recommendations*»), stellen die Methoden, welche inhaltsbasierte und kollaborative Methoden kombinieren, dar.

[1] untersucht zusätzlich sowohl die Vorteile und Nachteile, als auch die Verbesserungsmöglichkeiten der jeweiligen Empfehlungstechnik und fokussiert sich auch auf die Möglichkeiten der Expansion der Kapazitäten der Empfehlungsdienste und die Behebung der derzeitigen Begrenzungen.

Burke klassifiziert in [3] die Empfehlungsdienste in ähnlicher Weise - neben der klassischen *inhaltsbasierten* und *kollaborativen* Empfehlungsdienste führt er auch drei zusätzliche Empfehlungsdienste ein:

Demographische Empfehlungsdienste (Engl. «*demographic recommender systems*»), die versuchen Benutzer anhand ihrer persönlichen, demographischen Attribute die entsprechenden Empfehlungen vorzuschlagen,

Nützlichkeitsbasierte Empfehlungsdienste (Engl. «*utility-based recommender systems*»), die die Empfehlungsenergie im Bezug auf den Nutzwert jedes Gegenstandes für den Benutzer durchführen und

Wissensbasierte Empfehlungsdienste (Engl. «*knowledge-based recommender systems*»), die die Gegenstände anhand des Wissen über die Bedürfnisse und die Präferenzen des Benutzers empfehlen.

Beim Vergleich dieser Dienste identifiziert man in [3] weiter die größten Probleme, die in Abhängigkeit von der Dienstart auftreten:

Die zwei Varianten von *Ramp-Up Problem* -

Neuer Benutzer (Engl. «*New User*»): Weil Empfehlungen auf einem Vergleich zwischen dem Zielbenutzer und den anderen Benutzern, welcher einzig und allein auf der Ansammlung von Bewertungen basiert, ist neuer ein Benutzer mit wenigen Bewertungen nur schwer zu kategorisieren.

Neuer Gegenstand (Engl. «*New Item*»): Ähnlich kann ein neues Gegenstand, der noch nicht vielen Bewertungen hatte, auch nicht leicht zu empfehlen sein.

Die Probleme mit der Spärlichkeit des Bewertungsraumes: Kollaborative Empfehlungsdienste hängen von der Überschneidung in den Benutzerbewertungen ab und haben eine Schwierigkeit, wenn der Raum der Bewertungen spärlich ist: d.h., wenn wenige Benutzer die gleichen Gegenstände bewertet haben.

'Grey Sheep' Problem: Kollaborative Empfehlungsdienste arbeiten am besten für einen Benutzer, der in eine Nische mit vielen Nachbarn ähnlichen Geschmacks zusammenpasst. Diese Technik arbeitet jedoch für ein sogenanntes '*Grey Sheep*' [25], deren Geschmack sich auf einer Grenze zwischen existierenden Cliquen von Benutzern befindet, jedoch nicht so gut. Dies stellt auch ein Problem für demographische Systeme, welche versuchen, die Benutzer anhand von persönlichen Merkmalen zu kategorisieren, dar.

Start-up Problem: Inhaltsbasierte Empfehlungsdienste haben üblicherweise das Problem, dass sie ausreichend Bewertungen ansammeln müssen, um einen zuverlässigen Klassifizierer aufbauen zu können.

Beschränkungen des Empfehlungsraums: Relativ kollaborative, inhaltsbasierte Empfehlungsdienste haben auch das Problem, dass ihre Empfehlungen auf Merkmalen, welche deutlich mit den Gegenständen, die sie empfehlen, assoziiert sind, beschränkt sind.

Portfolio-Effekt (Engl. «*portofolio effect*»): Ideale Empfehlungsdienste würden keine Aktien vorschlagen, die der Benutzer schon besitzt, auch keinen Film, den er schon gesehen hat. Sowohl die kollaborative als auch die inhaltsbasierte Empfehlungsdienste sind in diesem Bereich mangelhaft.

Mangelnde Flexibilität: Die nützlichkeitsbasierten Empfehlungsdienste versagen teilweise im Bereich der Flexibilität. Der Benutzer muss eine vollständige Präferenzfunktion konstruieren, und dadurch die Wichtigkeit jedes möglichen Merkmals abwägen. Oft belastet das die Interaktion mit dem Benutzer beträchtlich.

Bedarf nach Aneignung von Wissen: Wissensbasierte Empfehlungsdienste haben vor allem den Nachteil, dass ihr Bedürfnis nach Wissenserwerb zeit- und kostenaufwendig sein kann.

Aus diesen Gründen sieht der Autor eine hybride Kombination von Empfehlungsdiensten als sehr wünschenswert, um die Vorteile der Dienste auszunutzen und mangelnde Eigenschaften zu beheben. Weiters werden in [3] die 7 möglichen hybriden Methoden präsentiert:

- *Gewichteter Hybrid* (Engl. «*weighted hybrid*»), wo die Punktzahl des Empfehlungsgegenstandes von den Ergebnissen aller in einem Dienst vorhandenen Empfehlungsmethoden berechnet wird,
- *Schaltender Hybrid* (Engl. «*switching hybrid*»), wo der Empfehlungsdienst anhand der aktuellen Bedingungen zwischen den verschiedenen Empfehlungsmethoden umschaltet,
- *Gemischter Hybrid* (Engl. «*mixed hybrid*»), wo die Empfehlungen von mehreren Empfehlungstechniken gleichzeitig präsentiert werden,
- *Merkmalkombination - Hybrid* (Engl. «*feature combination hybrid*»), wo Merkmale aus verschiedenen Empfehlungsdatenressourcen in einen einzigen Algorithmus zusammengesetzt werden,
- *Merkmalaugmentation - Hybrid* (Engl. «*feature augmenation hybrid*»), wo die Ausgabe einer Empfehlungstechnik als die Eingabe der anderen Empfehlungstechnik benutzt wird,
- *Meta-Ebene Hybrid* (Engl. «*meta-level hybrid*»), wo das Modell, das in einem Empfehlungsdienst gelernt wird in einem anderen Empfehlungsdienst als Input verwendet wird und
- *Kaskadenartiger Hybrid* (Engl. «*cascade hybrid*»), wo ein Empfehlungsdienst die Ergebnisse des anderen Empfehlungsdienstes verfeinert.

In [30] folgen *Ramezani et al.* das klassische Unterteilungsprinzip aus [1], verfeinern drei der erwähnten Empfehlungstechniken aber weiter.

So kann man bei den **kollaborativen Empfehlungsdiensten - KE** (Engl. «*collaborative filtering recommenders, CFR*») zwischen

- *Erinnerungsbasierten* (Engl. «*memory-based*») KE, die die Nachbarpräferenzen noch in Runtime benutzen, um die Empfehlungen zu generieren und
- *Modellbasierten* (Engl. «*model-based*») KE, die ein Modell des Benutzerhaltens vor dem Empfehlungsvorgang erlernen und anhand dieses dann die Empfehlungen erstellen,

unterscheiden. Die **inhaltsbasierten Empfehlungsdienste - IE** (Engl. «*content-based recommenders, CBR*») unterteilt man in

- *lernende* (Engl. «*learning*») *IE*, die die Empfehlungen auf Basis von persistenten Benutzerprofilen bilden und
- *Suchmaschinen* (Engl. «*search engines*»), die die Empfehlungen auf Basis von kurzlebigen Benutzerprofilen erstellen.

In den **wissensbasierten Empfehlungsdiensten - WE** (Engl. «*knowledge-based recommenders, KBR*») können

- *Fallbasiertes Schließen* (Engl. «*Case-based Reasoning*») *WE*, das das Wissen von früher erlebten Situationen wiederverwendet,
- *constraint-based* (Engl. «*Constraint-based*») *WE*, die die Empfehlungen, basierend auf einem Zwangssatz, das von jedem Benutzer gesammelt wird, anbieten und
- *Regelbasierte* (Engl. «*Rule-based*») *WE*, die explizite Regel verwenden um zwischen den Benutzerbedürfnissen und vorhandenen Gegenständen abzubilden,

aufzutreten.

Die drei angeführten Unterteilungsmethoden werden in der Abbildung 2.16 auf der Seite 26 zusammengefasst.

Taxonomie der Empfehlungsdiensten				
Allgemeine Unterteilung	Weitere Unterteilung			
Kollaborative Empfehlungsdienste (KE)	Erinnerungsbasierte KE		Modell-basierte KE	
Inhaltsbasierte Empfehlungsdienste (IE)	Lernende IE		Suchmaschinen	
Wissensbasierte Empfehlungsdienste (WE)	Fallbasiertes Schließen WE	Zwangsbasierte WE	Regelbasierte WE	
Hybride Empfehlungsdienste	Gewichteter Hybrid	Schaltender Hybrid	Gemischter Hybrid	
	Merkmal-kombination	Merkmal-augmentation	Kaskadenartiger Hybrid	Meta-Ebene Hybrid
Demographische Empfehlungsdienste				
Nützlichkeitsbasierte Empfehlungsdienste				

Abbildung 2.16: Taxonomie der Empfehlungsdienste ([1], [3], [30])

Klassifikationsdimensionen

Einer der ersten Versuche, die Dimensionen des Empfehlungsproblems zu identifizieren, findet man in [37], wo eine ausführliche Untersuchung und Klassifizierung der Empfehlungsdienste in E-Commerce von *Schafer et al.* - die Forschungsgruppe hinter des *GroupLens* Projekts durchgeführt wurde. Obwohl sich die Autoren auf den Bereich der kommerziellen Anwendung der Empfehlungsdienste beschränken, können die Dimensionen, die dort eingeführt wurden - wie *Benutzereingabe, Ausgaben, Empfehlungsmethoden, Personalisierungsgrad* und *Lieferungsmodus* auch die andere Arten der Empfehlungsdiensten zugewiesen werden. Die im letzten Unterkapitel erwähnten Arbeiten in [1] und [3] bieten eine tiefere Analyse der Probleme, die für die jeweilige Empfehlungsmethode charakteristisch sind. Eine allgemeine Klassifikation aller bestehenden Dimensionen des Empfehlungsproblems kann man in [30] und [22] finden.

[30] analysiert auf der einen Seite die drei Dimensionen der Empfehlungstechnologie - nämlich die *Algorithmen*, die *Interaktion mit den Benutzern* und das *Benutzer-Profiling* (Tabelle 2.2, Seite 27), und auf der anderen Seite schlägt er sechs möglichen Dimensionen eines Empfehlungsproblems vor - die *Problemstruktur*, die *Beziehung mit dem Benutzer*, die *Domäne*, die *Benutzereingabe*, das *Hintergrundwissen* und die *Empfehlungsausgabe* (Tabelle 1.1, Seite 16).

Dimensionen der Empfehlungstechnologie		
Algorithmen	Interaktion mit den Benutzern	Benutzer-Profiling
<u>Kollaborative</u> Erinnerungsbasierte Modellbasierte	<u>Gesprächige</u> Frage/Antwort Candidate/Critique	<u>Persistente Benutzermodelle</u> <u>Kurzlebige Benutzermodelle</u>
<u>Inhaltsbasierte</u> Lernende Suchmaschinen	<u>Single-Shot</u>	<u>Statische Benutzerprofile</u> <u>Dynamische Benutzerprofile</u>
<u>Wissensbasierte</u> Case-based Reasoning Constraint-basierte Regelbasierte		
<u>Hybride</u>		

Tabelle 2.2: Dimensionen der Empfehlungstechnologie [30]

Wie die verschiedenen Algorithmenarten in [30] beschrieben wurden, wurde bereits im vorangegangenen Abschnitt behandelt. *Ramezani et al.* definieren des-

weiteren *gesprächige* und *Single-Shot* Interaktionen mit den Benutzer, wo die erste durch die Anfragen an die Benutzer um ein zusätzliches Feedback (*Candidate/Critique* Benutzerinteraktion) oder die Beantwortung der Fragen (*Frage/Antwort* Benutzerinteraktion) gekennzeichnet ist und die zweite für jede Interaktion mit dem Benutzer eine neue, unabhängige Empfehlung erzeugt. Was das Benutzer-Profilung angeht, verbinden die Autoren mit *persistenten Benutzermodellen* und *statischen Benutzerprofilen* eine Ansammlung der Benutzerinformationen über die Zeit hinweg. Im Gegensatz dazu beinhalten die *kurzlebige Benutzermodelle* und *dynamische Benutzerprofile*, nur aktuelle Benutzerinformationen (die aus der aktuellen Session mit dem Benutzer erzeugt wurden).

Bei den Dimensionen des Empfehlungsproblems kann man laut [30] eine grobe Klassifizierung in fundamentale Problemeigenschaften (*Struktur der Problems*, *Beziehung mit dem Benutzer* und *Domäne*) und sekundäre Problemeigenschaften (*Benutzereingabe*, *Hintergrundwissen* und *Empfehlungsabgabe*) machen. Die *Problemstrukturen* werden in 4 Kategorien klassifiziert: das *Auswahlproblem*, wobei das Problem darin liegt, aus einer gewaltigen Anzahl an Optionen eine gute Auswahl zu treffen, das *Konfigurationsproblem*, wo die besten Wege, um Komponentensets zu kombinieren, gesucht sind, das *Planungsproblem*, wo eine Aktivitätensequenz um ein bestimmtes Ziel zu erreichen gesucht wird und das *Erforschungsproblem*, wo eine Benutzerunterstützung bei der Suche nach etwas Interessantem gebraucht wird. Bei der Domäne müssen die *Größe* und die *Verschiedenartigkeit des Ergebnisraums*, die *Lebenserwartung* und die *Kritikalität* der Empfehlungen, die *Wichtigkeit* und die *Dauer der Benutzerpräferenzen* sowie der *Endbenutzertyp* in Betracht genommen werden. Die *Länge* und *Tiefe der Beziehung mit dem Benutzer* hängt auch von dem zu entwickelnden Dienst ab.

Was die sekundären Problemeigenschaften anbelangt, wird bei der *Benutzereingabe* der *Grad des Benutzeraufwandes* (d.h. ob explizite oder implizite Benutzereingabe verwendet werden sollte) und der *Eingabetyp* ausführlich beschrieben, wobei beim letzteren die Autoren wieder zwischen *impliziten* und *expliziten* Benutzereingabetypen unterscheiden. Einen Überblick über die Dimensionen des *Hintergrundwissens* in Empfehlungsdiensten kann man in Abbildung 1.1 auf der Seite 17 gewinnen. Die *Empfehlungsabgabe* sehen die Autoren als ein zwei-dimensionelles Problem, wobei einerseits der *Ausgabebetyp* in *Empfehlungen* und *Voraussagen* unterteilt wird und andererseits die Wichtigkeit der *Erklärung* erforscht sein sollte.

Manouselis und Costopoulou schlagen in [22] eine ähnliche Klassifikation der Dimensionen des Empfehlungsproblems. Der von ihnen vorgeschlagene Fra-

network für die Analyse und die Klassifikation der Empfehlungsdienste (siehe [22]) ist in der Abbildung 2.17 auf der Seite 29 ersichtlich. Die Empfehlungsdimensionen sind relativ ähnlich, die Unterteilung erfolgt aber auf eine andere Art und Weise. Neu ist hier die *Operationsdimension*, die mehr technisch bezogen ist und sich an die Architektur-, die Empfehlungsörtlichkeit- und die Modus-Aspekte konzentriert.

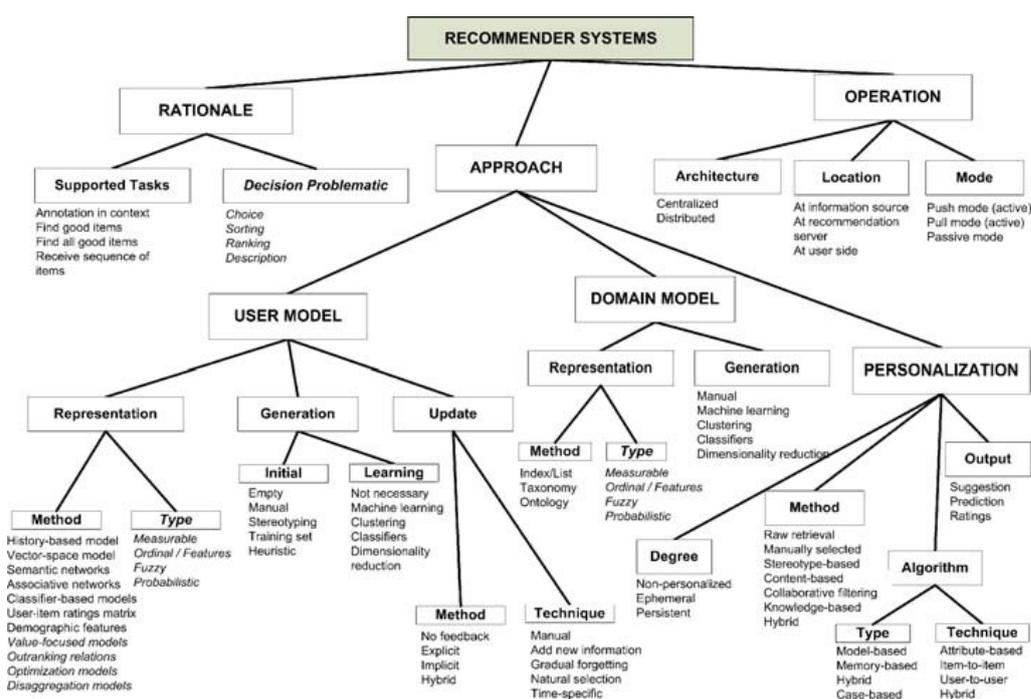


Abbildung 2.17: Der Framework für die Analyse und Klassifikation der Empfehlungsdienste [22]

Wissensbasierte Empfehlungsdienste

Obwohl die kollaborativen Empfehlungsdienste die bekanntesten und die meist verbreiteten Empfehlungsdienste sind, gewinnen die wissensbasierten Empfehlungsdienste immer mehr an die Bedeutung. Schon im Jahr 2000 definiert *Burke* in [2] den wissensbasierten Empfehlungsdienst als «ein Dienst, das das Wissen über die Benutzer und die Produkte benutzt, um einen wissensbasierten Ansatz für die Erstellung der Empfehlungen zu verfolgen, der die Behauptung darüber ermöglicht, welche Produkte die Anforderungen der Benutzer erfüllen». In den ersten Klassifikationen der Empfehlungsdienste treten sie aber nicht selbststän-

dig auf, sondern wurden üblicherweise in den Bereich der inhaltsbasierten Empfehlungsdienste zusammengefasst.

Am besten kann man die Begrenzung zwischen den wissensbasierten und den anderen Empfehlungsdiensten am Hintergrundwissen (siehe Tabelle 1.1, Seite 17), das für die Empfehlungen verwendet wird, bemerken: weil die kollaborativen Empfehlungsdienste das Peer-Wissen und die Benutzermeinungen für die Empfehlungen und die inhaltsbasierte Dienste die Benutzermeinungen und Gegenständeattributen und die demographische Wissensquellen ausnutzen, können in der wissensbasierten Empfehlungsdiensten alle andere Wissensquellen angewendet werden [4], d.h., als Hintergrundwissen können dann kontextuelles Wissen, alle Formen der Benutzeranforderungen (Abfrage, Einschränkungen, Präferenzen oder Kontext) sowie das bestehende Domänenwissen (Means-Ends, Eigenschaftenontologie oder Domäneinschränkungen) auftreten. Dieser umfangreiche Quellenbereich führt aber auch zu einer nicht klaren Unterscheidung zwischen den jeweiligen Arten der wissensbasierten Empfehlungsdienste, so dass die heutige interne Klassifikation dieses Empfehlungsbereichs nicht alle auftretenden Arten aufdeckt.

Die Vorteile der wissensbasierten Empfehlungsdienste umfassen das Vermeiden des *Cold-start-Problems* [38] und *Ramp-Up-Problems* [3], und sind für die Domäne, bei der die Kosten, die mit einem Dienstfehler auftreten können hoch sind, am besten geeignet, da durch ihre Geräusche-Abwesenheit die Empfehlungen sehr verlässlich sind. Auf der anderen Seiten fordern sie eine extensive und teure Wissenserwerbung und sind deshalb für die Dienste, für welche eine einfachere Lösung gefunden werden könnte, nicht empfehlenswert [30].

In diesem Teil unterscheidet man zwischen den *Fallbasiertes-Schließen* ([2], [45], [5], [28], [32], [6], [27]), den *constraint-based* ([13], [11]) und den *modellbasierten* ([44], [51], [29]) wissensbasierten Empfehlungsdiensten. Alle hier nicht aufgezählten Dienste werden als «Andere» ([41], [52]) bezeichnet.

Fallbasiertes-Schließen wissensbasierte Empfehlungsdienste

Fallbasiertes-Schließen-Empfehlungsdienste sind eine der ersten Vertreter dieses Bereiches. Sie basieren auf der Idee des fallbasierten Schließens. Ein fallbasiertes Schließen «löst neue Probleme durch die Anpassung der Lösungen, die für die Lösung der alten Probleme verwendet wurden» (Riesbeck und Schank, 1987). Ein Ablauf eines fallbasiertes Empfehlungsdienstes kann man in der Abbildung 2.18 (Quelle: [4]) auf der Seite 30 ansehen.

Eine Datenbank der vorherigen Fälle/Erfahrungen (Engl. «*Case-base*») ist vorhanden. Wenn der Benutzer den Dienst betreten werden die ähnlichsten Fälle für die der Empfehlungsdienst annimmt, dass der Benutzer sie haben möchte, aus dieser Datenbank zurückgeholt. Der Benutzer adaptiert die erhaltenen Fälle

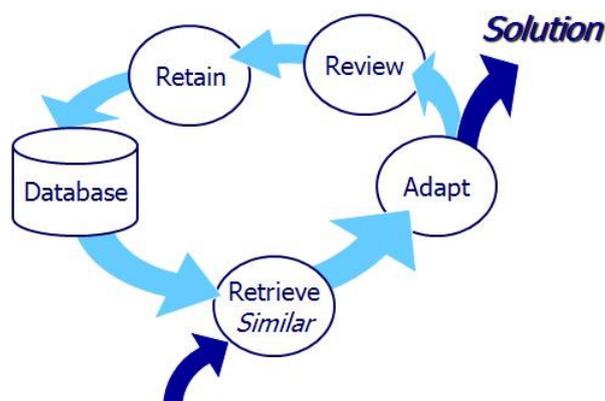


Abbildung 2.18: Ablauf eines fallbasierten Empfehlungsdienstes [4]

bis er die benötigten Lösungen gefunden hat. Die verwendete Lösung wird dann überprüft und in der Datenbank gespeichert. In [35] findet man ein klassisches Beispiel dieses Empfehlungsprozesses.

Bei den Fallbasiertes-Schließen-Empfehlungsdiensten haben sich weiter Unterkategorien entwickelt:

- Kritik von Beispielen (Engl. «*example critiquing*») - [2], [5], [6], [28], [45] (siehe Abbildung 2.19, Seite 31),
- dynamische Kritik (Engl. «*dynamic critiquing*») - [5], [6],
- zunehmende Kritik (Engl. «*incremental critiquing*») - [31] und
- zusammengesetzte Kritik (Engl. «*compound critiquing*»)- [32].

Alle diese Dienste benutzen das Konzept der *kritischen Abhandlung* (Engl. «*critiquing*»), bei der der Benutzer seine Reaktion auf den empfohlenen Gegenstand in Form von Rückmeldung aller Abweichungen dieses Gegenstandes von seinem Ideal abgibt [11]. Kritiken können als *Einheiten* (Engl. «*unit critiques*») oder *zusammengesetzt* (Engl. «*compound critiques*») abgegeben werden. Bei der ersten handelt der Benutzer bestimmte Gegenstandsattribute kritisch ab, die zweiten können aber mehrere Eigenschaftendimensionen auf einmal abhandeln. Die *zunehmenden* kritischen Abhandlungen nehmen sowohl die vorherige als auch die aktuellen kritischen Abhandlungen des Benutzers in Betracht. Bei den *dynamischen* kritischen Abhandlungen werden häufig auftretende Muster in früher durchgeführten kritischen Abhandlungen beobachtet und als nächste mögliche Kritik vorgeschlagen.

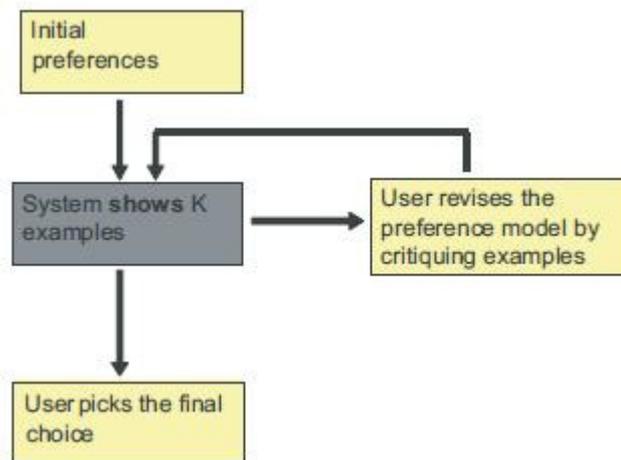


Abbildung 2.19: Kritik von Beispielen (example critiquing) [28]

Beispiele

Einige erfolgreiche Beispiele dieser Art von wissensbasierten Empfehlungsdiensten sind:

[35]: **ITR** ist ein Reiseberater, basierend auf fallbasiertem Schließen, der dem Benutzer bei der Auswahl eines geeigneten Urlaubs helfen sollte. Dabei unterstützt er nicht nur die Selektion der entsprechenden Urlaubsziel sondern auch die Auswahl der gewünschten Aktivitäten und Attraktionen und ermöglicht die Entwicklung benutzerspezifischen Reiseplans, der als *Reisetasche* bezeichnet wird. Das Wissen dahinter beruht auf der Basis der Angelegenheiten in den Reiseplänen eines Benutzernetzwerks und den Katalogen von *Destination Management Organisation (APT Trentino)*. Diese hierarchische Angelegenheitsstruktur wird als XML-Ansicht über die relationale Datenbank implementiert.

[2]: **Entree, Recommender.com, FindMe**. Die gezeigten Projekte *Entree*, ein Restaurantempfeher und *Recommender.com*, ein Filmempfehlungsdienst, sind wissensbasierte Empfehlungsdienste die auf dem *FindMe-Ansatz* basieren, und bei welchen die Beispiele benutzt werden um eine effiziente Suche durchzuführen, wobei gleichzeitig eine soziale Interaktion durch das Weichen oder die Änderung der Charakteristiken vom Beispiel stattfindet. Die Herausholung der Präferenzen erfolgt stufenweise, anstatt alles-auf-einmal wie z.B. bei einer Query. Das kann den Benutzer von den leeren Teilen des Produktraums weiterbringen.

[45]: **Flatfinder**, ein Hilfswerkzeug für die Suche nach einer entsprechenden Studentenunterkunft. [45] schlägt einen anderen Ansatz für die Empfehlungsgenerierung vor: durch die Ausschöpfung des existierenden Domänenwissens über die Verbreitung von Vorzügen der Benutzer. Diese Strategie sollte sich aber auch auf die Benutzerreaktion auf die vorgestellten Beispiele dynamisch adaptieren. Der Ansatz hat sein Fundament in der kritischen Abhandlung von Beispielen, wo dem Benutzer in jedem Zyklus ein Set von k Beispielen gezeigt wird. Durch die Interaktionen mit diesen Beispielloptionen drückt er dann seine Präferenzen aus, und gibt dem Dienst ein Feedback ab, das benutzt wird um das nächste Optionenset zu kreieren.

Constraint-based wissensbasierte Empfehlungsdienste

In Situationen wenn der Empfehlungsdienst nicht häufig benutzt wird und die angebotenen Gegenstände sehr komplex sind, ein Empfehlungsfehler große Auswirkung haben kann, und die Mehrheit der Benutzer nicht mit allen nötigen Merkmalen vertraut ist, kann ein constraint-based wissensbasierter Empfehlungsdienst von großer Hilfe sein.

Was die constraint-based wissensbasierten Empfehlungsdienste von anderen dieser Art unterscheidet wird in [11] ausführlich beschrieben. Außer dass die Autoren am Anfang dieser Arbeit eine aktuelle Taxonomie des Empfehlungswissen der existierenden Empfehlungsdienste die Empfehlungsdienste nach Wissensquelle unterteilt präsentieren (wobei das Wissen vom *Benutzer selbst*, von *anderen Anwendern* des Diensts, die *von den Gegenständen* und von der *Empfehlungsdomäne* stammen könne), und die vorhandenen Empfehlungstechniken kurz erklärt und ihre Vorteile und Nachteile hervorbringt, definieren sie auch die constraint-based wissensbasierten Empfehlungsdienste als «Anwendungen, bei welchen spezifische Produkteigenschaften als auch die Beziehung zwischen den Anforderungen der Kunden und der Produkte in Form von Beschränkungen (Engl. *<constraints>*) modelliert wird». Die Unterstützung der Kunden erfolgt durch die Erklärung der Gegenstände, das Anbieten von Ausbesserungsmöglichkeiten im Falle, dass keine Ergebnisse geliefert werden, und das Vorschlagen von Attributeneinstellungen, basierend auf den Präferenzen der Community.

Diese Dienste fordern aber auch eine explizite Eingabe der Benutzeranforderungen. Diese sind in einer Wissensbasis (Engl. *<knowledge base>*) in Form eines Zwänge-Netzwerks zusammengesetzt. Ein Beispiel eines Ablaufes in einem constraint-based Empfehlungsdienst für Web Space Hosting kann man in der Abbildung 2.20 (aus [11]) auf der Seite 33 betrachten. Die Interaktionssequenzen, in welchen die Kunden die Fragen beantworten können, werden explizit definiert und benutzen das Prinzip der *finite Zustandsautomaten*. In den Zustän-

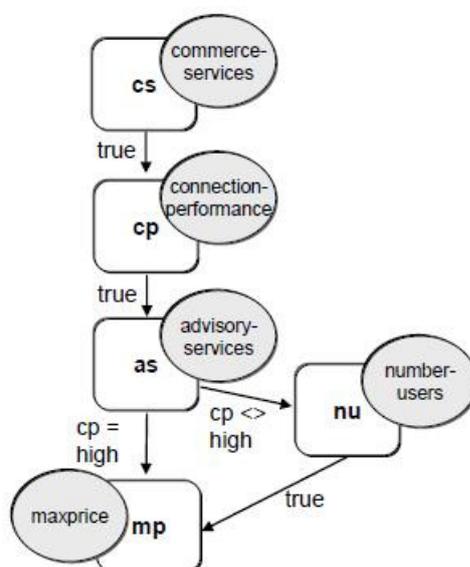


Abbildung 2.20: Dialogstruktur in constraint-basierten Empfehlungsdiensten [11]

den werden die Fragen an den Kunden gestellt und in Abhängigkeit von den von ihnen ausgedrückten Präferenzen wird eine der folgenden möglichen Zustände ausgewählt. So wird z.B. beim Web Space Hosting nur die Frage über die Anzahl der zukünftigen Benutzer gestellt, wenn sich der Kunde für eine niedrige oder mittlere Verbindungsleistung entschieden hat.

Beispiele

Eine aussagekräftige industrielle Anwendung dieser Art von wissensbasierten Empfehlungsdiensten findet man in:

[13]: FSAdvisor wird für das wissensbasierte Verkaufen von finanziellen Dienstleistungen benutzt. Er hilft den Handelsvertretern in verschiedenen Situationen - beim Überprüfen der Konsistenz der Kundenanforderungen sowie beim Support bei der Behebung der Errors, um für die Bedürfnisse des Kunden ein geeignetes Produkt zu finden, bei der Findung und beim Korrigieren der inkonsistenten Kundenanforderungen und bei der Dokumentation. Der Dienst hat durch die Benutzung von intelligenten kundenorientierten Dialogen, die auf modellbasiertem Argumentieren und der Personalisierung basieren, zur Verbesserung der Handelsdialoge beim Verkauf von finanziellen Diensten geführt.

Modellbasierte wissensbasierte Empfehlungsdienste

Modellbasierte Empfehlungsdienste treten sowohl als Unterteil der kollaborativen als auch der wissensbasierten Empfehlungsdienste. In beiden Fällen wird ein Modell des Benutzerverhaltens vor dem Empfehlungsvorgang gelernt, und anhand dessen werden dann die Empfehlungen erstellt. Der Unterschied besteht darin dass, für die Erstellung des Benutzermodells bei der ersten das Peer-Wissen ausgenutzt wird, während für die zweite ein umfassendes Wissensengineering benötigt wird.

Beispiele

Obwohl diese Art von wissensbasierten Diensten in der offiziellen Klassifikation noch nicht auftaucht, können zahlreiche Beispiele dieser Empfehlungsart in der Literatur trotzdem vorgefunden werden. Einige davon sind:

[44]: Wissensbasierte Empfehlungsdienste mit expliziten Benutzer-Modellen: Als die größten Herausforderungen für die Empfehlungsdienste von heute identifizieren die Autoren hier der *Genauigkeitsgrad*, das *konstante Auftauchen von neuen Gegenständen* und die *Heterogenität* der Produkteigenschaften. Als eine Lösung schlagen sie vor, die explizite Modellierung für jeden Benutzer und jedes Produkt am entsprechenden Marktplatz zu benutzen, mit einem zusätzlichen personalisierten Dienst, der die Abbildung zwischen den beiden verrichtet. Hierzu könnte eine datengetriebene Komponente addiert werden, die sich um jene Fälle bei welchen der Dienst keine korrekte Ergebnisse liefert, kümmern sollte. Als die Beispiele wie das in der Praxis funktionieren sollte wurden die Fälle von Empfehlungen für Bücher und elektronische Geräte genannt.

[51]: Reiseempfehlungen basierend auf abhängigen Benutzerpräferenzen: Die Forscher in [51] betrachten die Situationen wo die Benutzerpräferenzen voneinander abhängig sind und demonstrieren, wie dieses Problem mit Hilfe eines effektiven Empfehlungsdienstes, welches auf *abhängigen Präferenzen* (Engl. «*conditional preferences*») basiert, behoben werden kann. Dieser Dienst wird am Beispiel der Reiseempfehlungen erklärt. Das quantitative Modell der abhängigen Präferenzen wird anhand des existierenden Domänenwissens kreiert. Das Modell beinhaltet 4 Submodelle: das *Domänenwissensmodell*, das *Modell der abhängigen Präferenzenregeln*, das *Einstufungsmodell* und das *Modell der Gruppenpräferenzen*. Nachträgliche Präferenzenschlußfolgerungen übersetzen dabei die Regeln über abhängige Präferenzen in den Bemessungs-Größen der Gegenstände.

[29]: Modellbasierter Schlussfolgerungsmotor für die Autoindustrie: Um die Entscheidungen auch ohne umfassendes technisches Wissen genau treffen zu können, schlagen die Autoren in [29] bei der Entwicklung eines gesprächigen Empfehlungsdienstes die Integration eines intuitiven Kundenmetamodells, mit verbundenem Schlussfolgerungsmotor (Engl. «*Inference Engine*») vor, um die Argumentation über Kundenbedürfnisse anbieten zu können. Am Beispiel der Autoindustrie erklären die Autoren das vorgeschlagene Metamodell, das die domäne-abhängige Beschreibung der zukünftigen Kunden darstellt. Das Kundenmodell wird als ein Bayesianisches Netzwerk repräsentiert.

Andere Arten von wissensbasierten Empfehlungsdiensten

Da die wissensbasierten Empfehlungsdienste für komplexere, maßgeschneiderte Kundenprojekte entwickelt sind, unterscheidet sich üblicherweise jeder Dienst vom anderen in mehreren Aspekten. Diese Heterogenität der Empfehlungsdienste verursacht aber zusätzliche Probleme bei der Klassifikation (aber auch bei der Evaluierung, was in nächstem Abschnitt behandelt wird), so dass ständig neue Kategorien hinzugefügt werden (wie bei den modellbasierten wissensbasierten Empfehlungsdiensten) oder einige Vertreter dieses Anwendungsbereiches nicht eindeutig klassifiziert sind.

Beispiele

Ein interessantes Beispiel dafür ist:

[41]: Szenario-orientierte Empfehlung gibt einen besonderen Ansatz für das benutzerorientierte Design in Empfehlungsdiensten. Diese Empfehlungen werden benutzt um den Benutzern in ihren täglichen Lebensszenarien (z.B. «Besuch beim Arzt», «Job-Interview», «Hochzeit» usw.) bei der Produktauswahl zu helfen. Die Idee wird anhand des Mode-Empfehlungsdienstes «What am I gonna wear?» erklärt. Die Kleidungsauswahl wird in drei Unterprobleme geteilt: die Funktion der Kleidung (Aufgabe des *Funktions-sensors*), die Kleidung als Ausdruck des Charakters (Aufgabe des *Style-Sensors*) und die Bedeutung der Kleidung zu einem bestimmten Anlass und für die anwesenden Leute (Aufgabe des *Gelegenheit-Netzwerks*).

Evaluierung von Empfehlungsdiensten

Die Evaluierung beschreibt die systematische Sammlung und die Analyse von nicht routinemäßig verfügbaren Informationen über verschiedene Aspekte eines

gegebenen Studienobjekts, um seine kritische Beurteilung zu ermöglichen. Einfach gesagt, untersucht sie die Unterschiede zwischen dem Soll- und Ist-Zustand eines zu untersuchenden Dienstes.

Die Evaluierung der Empfehlungsdienste ist in der wissenschaftlichen Arbeiten ausführlich beschrieben. Die Fragen, die dabei gestellt werden kann man grob in drei Gruppen unterteilen:

- Funktioniert der Empfehlungsdienst ordnungsgemäß? ([17]), [21])
- Ist der Algorithmus *A* besser als der Algorithmus *B*? ([5], [28], [32], [6], [27])
- Was ist der beste Algorithmus für bestimmte Aufgaben und Benutzergruppen? ([52], [12], [27])

Das Problem besteht darin, dass durch die Vielfalt der Empfehlungsdienstarten auch zahlreiche Evaluierungsmethoden entstanden sind - der langjährige Mangel an umfassender Standardisierung in diesem Bereich hat dieses Problem noch vertieft.

Evaluierungsaspekte

Herlecker et al. geben in [17] einen systematischen Überblick über den Evaluierungsbereich der Empfehlungsdienste, mit besonderem Fokus auf kollaborative Empfehlungsdienste.

Die Autoren beschäftigten sich nicht nur mit dem Weg, die Qualität der Empfehlungen zu messen, sondern auch mit der Entscheidung welche *Benutzeraufgaben* zu evaluieren sind, welche *Datensets* und *Analysetypen* verwendet werden sollten, welche andere *Empfehlungsnattribute* außer der Qualität betrachtet werden sollten und mit der Benutzerevaluierung des gesamten Dienstes.

Ein anderer Bereich, den die Autoren dargestellt haben, sind die Ergebnisse der selbst durchgeführten Analysen der Genauigkeitsmetriken an einem bestimmten Domäneinhalt.

Vor Beginn der Evaluierung sollte die Hauptaufgabe des Dienstes definiert werden, damit gemessen werden kann, wie gut diese erfüllt wird. Als bei der Evaluierung am häufigsten auftretende Aufgaben, welche die Ziele, die bei den Benutzern der Empfehlungsdienste auftreten können, charakterisieren, identifizieren sie *Erläuterung im Kontext*, *Finde gute Gegenstände*. Die anderen Aufgaben, *Finde alle guten Gegenstände*, *Empfehlungssequenz*, *Nur Browsing*, *Finde glaubwürdigen Empfehlungsdienst*, *Verbessere mein Profil*, *Drücke mich aus*, *Helfe den anderen*, *Beeinflusse die anderen* sind in der Literatur weniger verbreitet, stehen aber auch als geeignete Hauptaufgabe des Empfehlungsdienstes zur Auswahl.

Wenn man die Aufgabe, die evaluiert werden sollte, festgelegt hat, ist das am besten geeignete Datenset auszuwählen. Ob man sich für die *Offline*-Evaluierung mit existierenden Datensets entscheiden sollte oder ein *Online*-Experiment bessere Ergebnisse zeigt, hängt von der Art des jeweiligen Dienstes ab. Falls der Akzent auf Leistung, Zufriedenheit oder Beteiligung der Benutzer gesetzt wird, ist die zweite Evaluierungsmethode laut [17] die bessere Option. Weiters sollte man die Unterschiede zwischen synthetischen und natürlichen Datensets für das ausgesuchte Problem, die anderen Eigenschaften des Datensets (*Domänenmerkmale, inhärente Merkmale, Mustermerkmale*) sowie die frühere und die heutige Entwicklung in Datensets angehen.

Die Genauigkeitsmetriken kommen bei der Evaluierung der kollaborativen Empfehlungsdienste am meisten vor. Deshalb machen die Autoren eine ausführliche Evaluierung von einer Menge früher benutzter Genauigkeitsmetriken, die in drei Klassen unterteilt werden können - *Vorhersagende Genauigkeitsmetriken* - *MAE* und *Korrelation*, *Klassifikationsgenauigkeitsmetriken* - *NPDM* und *durchschnittliche Genauigkeit* sowie *Rang* - *Genauigkeitsmetriken* - *ROC* und *Half-Life*, welche Metriken zu welchen Aufgaben besser passen, vergleichen dann die Ergebnisse, die diese an einem bestimmten Datenset (*MovieLens*) bei einer ausgewählten Algorithmusart zeigen und untersuchen ob die Ergebnisse die die jeweilige Gruppe zeigt hoch korreliert sind.

Andere Dimensionen der Empfehlungsdienste mussten laut den Autoren bei der Evaluierung ebenfalls betrachtet werden, vor allem *erfasstes Gebiet* der Empfehlungsdienste, *Rate der Gelehrsamkeit der Algorithmen*, *Neuheit* und *Serendipity* (*Neigung überraschende Sachen per Zufall zu entdecken*) des Dienstes und *Zuversicht* der Empfehlungen. Die Evaluierung der *Benutzerreaktion* auf den Dienst wird in diesem Rahmen auch sehr empfohlen.

Als Bereiche, die es in der Zukunft zu untersuchen gilt, werden in [17] die *Benutzersensivität auf die Algorithmusgenauigkeit*, die *algorithmische Konsistenz zwischen Domains*, die *ausführliche Qualitätsmaßeinheiten* und die *Entdeckung von inhärenten Veränderlichkeit in den Empfehlungsdatensets* beschrieben.

In [9] wird der Mangel an Einheitlichkeit der Gestaltung der aktuellen Metriken für die Evaluierung der Empfehlungsdienste erkannt und ein neues Framework für die Analyse aller Empfehlungsdiensten vorgeschlagen, sowie eine neue Leistungsmetrik für jeden Dienst, der in diesem Framework definiert wird, vorgestellt.

Die heutigen Evaluierungsmetriken unterteilen die Autoren in drei Kategorien:

- *Bewertungsvoraussagen* - (MAE und ähnliche Metriken), die sich auf Vor-

hersagen über das Rating, das der Benutzer geben wird, noch bevor er dies gemacht hat, konzentrieren,

- *Klassifizierungsvoraussagen* - (Ranking-Metriken und ähnliche Metriken), die sich auf Vorhersagen über die Art der Klassifizierung, die der Benutzer für die vorgeschlagenen Gegenstände durchführen wird, konzentrieren und
- *Kapazität der erfolgreichen Entscheidungsfindung* (Engl. «*successful decision making capacity*») - (Genauigkeit und ähnliche Metriken), die sich auf die Messung der Dienstkapazität für die erfolgreiche Entscheidungsfindung konzentrieren.

Nur die letzteren passen in ein allgemeines Framework für die Evaluierung der Empfehlungsdienste, da die anderen zwei für ganz bestimmte Arten der Empfehlungsdienste gedacht sind.

Basierend auf der Definition von Burke in [2], unterteilen sie dieses Framework in zwei Subsysteme: ein interaktives Subsystem, den *Leiter* (Engl. «*guide*»), der sich um Fragen *wie* und *wann* eine Empfehlung gezeigt wird, kümmert und ein nicht-interaktives Subsystem, den *Filter*, der entscheidet, *welche* Gegenstände für die Benutzer nützlich oder interessant sein könnten. Ein Empfehlungsvorgang in diesem Framework wird in der Abbildung 2.21 ([9]) auf der Seite 39 gezeigt. Die Teile dieses Frameworks stellen die grundlegenden Elemente dar, *Event e* und *Session s(u)*, Ziele der Subsysteme sowie den generalisierten Empfehlungsprozess.

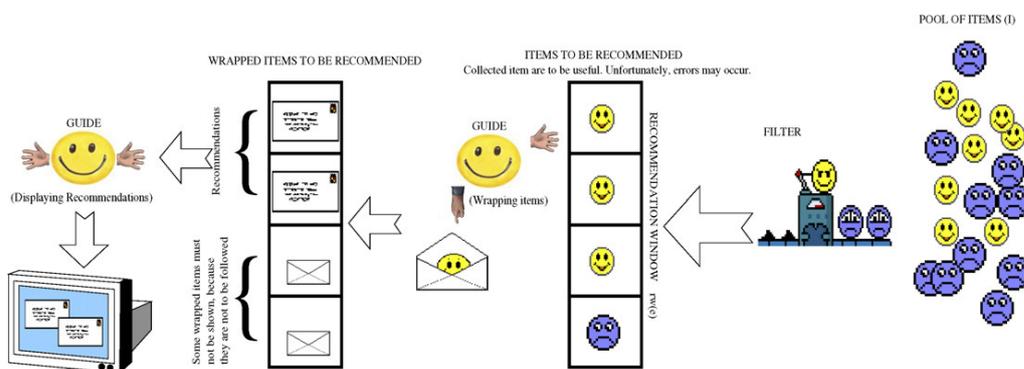


Abbildung 2.21: Empfehlungsprozess [9]

Die vorgeschlagene SDMC Evaluierungsmetrik, die für jeden Dienst aus diesem Framework gilt, wird definiert als

$$P(S) = \frac{\alpha}{|ev_r(S)|}$$

wobei α für die Anzahl der Empfehlungen die dem Benutzer gezeigt wurden und die er gefolgt hat oder als interessant gefunden und $|ev_r(S)|$ für die Anzahl der Empfehlungsevents steht.

Evaluierungsarten

Die heutigen Evaluierungsansätze kann man in zwei Kategorien unterteilen: Online-Evaluierung und Offline-Evaluierung.

Offline-Evaluierung

In der Offline-Evaluierung werden verschiedene Evaluierungsdimensionen bestehende Datensets angewendet, um Leistungseigenschaften zu prüfen. Im Unterschied zur Online-Evaluierung, werden hier retrospektive statt direkte Benutzerdaten verwendet, meistens Benutzer/Rating Paare, da diese Evaluierung sehr oft bei kollaborativen Empfehlungsdiensten auftritt. Bei diesen Evaluierungsmethoden ist es einfach viele Experimente laufen zu lassen und Algorithmen zu vergleichen. Auf der anderen Seite sind die Auswertungsmöglichkeiten begrenzt, und nicht alle Voraussagen können bewertet werden. Vor allem können die gemischten - Initiative - Dienste, interaktive Dienste oder Dienste mit Abfragen, welche typisch für die wissensbasierten Empfehlungsdienste sind, anhand dieser schwer evaluiert werden.

Offline-Evaluierungen treten aber auch selten bei der Evaluierung von wissensbasierten Empfehlungsdiensten auf. Ein gutes Beispiel dafür bringt [52]. Um zu sehen, welcher Ansatz für den kommerziellen Kontext am nützlichsten ist, schlagen die Autoren vor, eine komparative Analyse der acht verschiedenen (hybriden und klassischen) Empfehlungstechniken anhand eines konkreten Datensets (das sie in ihrem Fall von einem Web-Shop für feine kubanische Zigarren erhalten haben) offline zu machen und dadurch die implizite Informationen zu sammeln. Die 8 Empfehlungsarten, deren Leistung im Datenset laut [52] überprüft werden könnte, sind:

- «Top n» Empfehlung basierend auf Handelsberichten (*Topn*),
- wissensbasierte und nutzenbasierte kaskadierte Hybride ohne Entspannung (*KBUt*),
- wissensbasierte und «Top n» kaskadierte Hybride ohne Entspannung (*KB-Topn*),
- wissensbasierte und nützlichkeitsbasierte kaskadierte Hybride mit Entspannung (*KBUt-relax*),

- wissensbasierte und «Top n» kaskadierte Hybride mit Entspannung (*KB-Topn-relax*),
- kollaborative Filterung (*CF*),
- inhaltsbasierte Filterung (*CB*) und
- kollaborative und inhaltsbasierte Filterung Merkmal-Augmentation Hybride (*CBCF*).

Aus den binären Transaktionsdaten, die durch die Benutzung des Dienstes entstehen, können zwei Typen von (impliziten) Ratings hervorgehoben werden: frühere *Geschichte der Kunden-Kauftransaktionen* und *Click-Stream Daten*. Die Gegenständeähnlichkeit kann dabei aus der Logdatei mit gekennzeichnete Interaktion mit dem Berater und den benutzten Produktbeschreibungen heraus genommen werden.

Als Evaluierungsdimensionen werden

die Genauigkeit, die durch die *Recall*-Metrik berechnet werden kann,

die Benutzerabdeckung, die als Anzahl der Benutzer, den mindestens ein Gegenstand empfohlen wird, definiert werden kann und

die Katalogabdeckung, die die Neuheit und die Neigung, überraschende Sachen per Zufall zu entdecken, berechnet und deshalb durch als prozentuellen Anteil der Gegenstände im Katalog, die dem Benutzer irgendwann empfohlen wurde, gemessen werden kann,

verwendet. Pro Empfehlungserprobung werden dem Benutzer in der Regel 5 Gegenstände gezeigt. Die Kardinalität der Überschneidung zwischen der Empfehlungsliste und dem Testset errichtet das *Trefferset (hit-list)* für einen bestimmten Benutzer. *Recall* errechnet man dann durch $recall = \frac{|hit-set|}{|testing-set|}$ und wird für jeden Benutzer getrennt berechnet, wobei das *Overall-Recall* als den Durchschnitt aller Recall-Werte für alle Benutzer, die eine Empfehlung bekommen haben, darstellt.

Vier unterschiedliche Datensets können organisiert werden: *Buys-All*, das alle Kauftransaktionen von allen Benutzern enthält, *Buys-Advisor*, das nur den Anteil der Benutzer von *Buys-All* enthält, die Kontakt mit dem Kaufberater hatten, *Views-All*, das binäre Bewertungen von verschiedenen Benutzern, die ein Interesse an den Gegenstände gezeigt haben, enthält und *Views-Advisor*, der ein Subset von Usern, die mit dem Kaufagenten agiert haben, ist.

Die Antwort auf folgende zwei Forschungsfragen wird gesucht:

- Wie genau sind die verschiedenen Empfehlungsalgorithmen beim Voraussagen der Benutzertransaktionen in einer Online-Shopping Umgebung [52]?
- Welche Unterschiede gibt es zwischen den Algorithmen in Bezug auf das *User-Coverage* (Anteil der Benutzer im ganzen Datenset, denen mindestens einmal während der Probezeit ein Gegenstand empfohlen wurde [17]) und die *Produktentdeckung* (Anteil der Gegenstände im Katalog, die dem Benutzer mindestens ein Mal empfohlen wurden [17])[52]?

Online-Evaluierung

In der Online-Evaluierung wird untersucht, ob und wenn ja, in welchem Ausmaß dem Benutzer ein Vorteil durch den Empfehlungsdienst gebracht wurde. Üblicherweise werden hier standardisierte Techniken für die Überprüfung der Zufriedenheit der Nutzer, wie Benutzerstudien und Fokusgruppen benutzt. Diese Evaluierungsart wird häufig für die Evaluierung der wissensbasierten Dienste angewendet. Der Vorteil der Online-Evaluierung besteht darin, dass durch diese eine echte «*Ground Truth*» der Stellungnahmen der Benutzer entnommen werden kann. Diese Arten von Evaluierungsmethoden können aber nicht ausgeführt werden, bevor der Empfehlungsdienst komplett ist und haben den Nachteil, dass sie oft teuer und zeitaufwendig sind und sehr empfindlich auf experimentelle Bedingungen reagieren können.

Die Online-Evaluierung kann unterschiedlich durchgeführt werden:

- durch die Bereitstellung eines entsprechenden Fragebogens vor, während oder nach der Benutzung des Empfehlungsdienstes ([11]) oder
- die Bereitstellung verschiedener Online-Prozeduren und passender Szenarien, die ermöglichen, dass die Benutzer die Suche nach einem bestimmten Produkt simulieren, um zu sehen, ob und wie der Empfehlungsprozess funktioniert ([27]) und
- die Experteneinschätzung wo, nach dem Konstruieren eines Empfehlungsszenariums, die selbes für die Benutzung sowohl Experten als auch der Empfehlungsdienst angeboten wird, um zu überprüfen, welche Vorteile und Nachteile sie haben ([4]).

In Bezug auf die Art, in der man mit einer Benutzerstudie umgehen kann, unterscheidet man in der Literatur zwischen:

Between-Group Experiment, wo für jede Gruppe ein anderer Versuch durchgeführt wird und jede Gruppe eine bestimmte Empfehlungsmethode evaluiert, wobei diese unterschiedlich zu den Empfehlungsmethoden, die die anderen Gruppen evaluieren, ist ([28], [6], [32]).

Within-Subject Experiment, wo jeweilige Gruppe mehrere Empfehlungsmethoden zur Evaluierung zur Verfügung gestellt sind ([28], [32]).

Die Benutzung dieser beiden Experimentenarten ermöglicht, den Einfluss möglicherweise auftretender Voreingenommenheiten (wie z.B. die Reihenfolge in welcher die Empfehlungsdienste angezeigt werden) im Voraus begrenzt zu halten.

Die Empfehlungskriterien hängen vom jeweiligen Empfehlungsdienst ab und können Folgendes umfassen:

die Entscheidungsgenauigkeit, die die Anwender am Ende der Benutzung der Applikation erreicht haben. Sie wird in Abhängigkeit von der jeweiligen Evaluierung unterschiedlich berechnet. So wird sie in [28] als Anteil der Teilnehmer, die auf eine andere, bessere Option umgestiegen sind, wenn sie gebeten wurden, die vom Dienst vorgeschlagene Lösung mit allen Alternativen in der Datenbank zu vergleichen, definiert (Engl. «*switching rate*»);

den Entscheidungsaufwand, eine Wahl zu machen. Dieser Aufwand kann *objektiv* (durch die Überprüfung der Länge und der Anzahl der Sessions, der Berechnung der Zeit bis die entsprechende Lösung gefunden wurde u.s.w.), durch die Berechnung der gesamten Interaktionszeit aber auch *subjektiv* (durch die Stellung der entsprechenden Fragen) eingeschätzt werden. Der Entscheidungsaufwand kann bei critiquing-basierten Empfehlungsdiensten als Kritikaufwand, d.h., in Bezug auf die Anzahl der Produkten, die gesehen werden, bevor die Entscheidung getroffen wurde, gemessen werden [28];

das Vertrauen an die Entscheidung beschreibt das Sicherheitsausmaß, das der Benutzer hat, die richtige Entscheidung getroffen zu haben [27];

die Effekte der Bereitstellung von Erklärungen, die argumentieren, wieso ein bestimmter Gegenstand für einen Benutzer am passendsten ist [12];

die Effekte des Produktvergleiches: Der Einfluss des Produktvergleiches wird oft als Evaluierungsdimension benutzt [12];

die Effekte der Bereitstellung von Reparaturaktionen, wenn für die gegebene Anforderungen kein Ergebnis geliefert wurde und die Suchkriterien angepasst (d.h. *repariert*) werden sollten;

die Willigkeit, das Produkt zu kaufen Die Bereitschaft, das Produkt zu kaufen, vergrößert sich mit dem Steigen des Vertrauens und der Sicherstellung, dass der Empfehlungsdienst kompetente Empfehlungen ausgibt [12];

die Verbesserung des Wissens über die benutzte Domäne [12];

die Informativität des jeweiligen Empfehlungsdienstes [12];

die Absicht auf Wiederbenutzung des Empfehlungsdienstes [12];

die Frequenz der Benutzung des Empfehlungsdienstes [12];

demografische Informationen wie Alter, Herkunft u.s.w. [6].

Bei der Evaluierung sollte man aber auch auf gewisse Begrenzungsfaktoren achten: vor allem die Art des Produktdatensets, seine Größe und das Ausmaß seiner Verschiedenartigkeit, da bestimmte Produkte bestimmten Benutzer leichter zu empfehlen sind, sowie das Interfacedesign, das die Benutzer sowohl anziehen und ihnen helfen, als sie auch stören oder sie von der Produktsuche abbringen kann, als auch die frühere Erfahrung mit der Benutzung dieses oder eines ähnlichen Empfehlungsdienstes, da ein neuer Benutzer im Unterschied zum erfahrenen Benutzer zuerst eine Erlernungsphase durchmachen muss.

Als Lösungen für diese Art von Problemen werden in der Literatur Benutzungsanweisungen, Reparaturvorgänge oder mehrere Evaluierungsdatsensets vorgeschlagen.

Es kommen drei Arten der Online-Evaluierung in der Literatur häufig vor, wobei die Grenze zwischen dem verschiedenen Empfehlungsevaluierungen nicht klar definiert wird:

- *reine Online-Evaluierung*, wo die Evaluierungsdaten nur durch das explizite Feedback vom Benutzer (meistens anhand der Verwendung verschiedener Benutzerstudien) herausgezogen werden ([21]),
- *komparative Online-Evaluierung*, wo durch die Anfragen an den Benutzer verschiedene Empfehlungstechniken verglichen werden - [5], [28], [32], [12], [27] (ein wichtiger Unterteil dieser Empfehlungsart stellt der Vergleich mit dem *Baseline-Algorithmus*, üblicherweise eine Art von *Top_n/Ranked List/Form-Ausfüllen* Ansatz - [28], [12], [27]) dar und
- *gemischte Online-Evaluierung*, wo zusätzlich eine Offline - Evaluierungskomponente zugefügt wird, um die Evaluierungsergebnisse zu verfeinern ([32], [6],[12], [27]).

Reine Online-Evaluierung

In einer reinen Online-Evaluierung werden durch die Befragung der Benutzer neue Kenntnisse über den untersuchten Dienst gebracht. Dem Benutzer werden die Fragen über die Funktion des Empfehlungsdienstes gestellt oder es wird versucht, die vor dem Empfehlungsdienst anstehenden Aufgaben an den Benutzer zu übergeben, um zu sehen, welche Unterschiede zwischen dem Benutzer und dem Empfehlungsagent bestehen. Dabei benutzt man oft das *Mean Absolute Error - MAE*, um die *Genauigkeit* und die Einflüsse der verschiedenen Faktoren (Zeit, Vertrauen am Dienst u.s.w.) auf den jeweiligen Agenten zu vergleichen.

Beispiele

Ein typisches Beispiel dafür findet man in:

[21]: Die Wissenschaftler der *GroupLens Research Group* stellen sich in [21] die Frage, ob man sich bei der Suche nach dem richtigen Produkt lieber auf einen Empfehlungsdienst oder auf den menschlichen Berater verlassen sollte. Sie gehen bei ihrer Forschung von der Voraussagen ihres Projekts *MovieLens* aus, das ein auf kollaborativer Filterung basierter Empfehlungsdienst für Filmempfehlungen ist. Dabei berichten sie über die Studie, die auf eine Gruppe von *MovieLens*-Benutzern die schon mehr als 1000 Films bewertet haben, geführt wurde, wobei diese gebeten wurden, nachdem sie das Profil eines anderen Benutzers, das nur Informationen über die von ihm früher bewerteten Films enthalten hat, gesehen hatten, ihre Voraussagen über die Bewertungen für die anderen 10 Films abzugeben.

Komparative Online-Evaluierung

Der Vergleich der verschiedenen Empfehlungsarten wird bei den Evaluierungsvorgängen sehr oft vorgenommen, insbesondere bei der Evaluierung der wissensbasierten Empfehlungsdiensten. Dabei handelt es sich meistens um ähnliche Empfehlungsmethoden, die aber auch bedeutsame Unterschiede haben, und die Anzahl der in dieser Art und Weise bewerteten Empfehlungsmethoden pro Evaluierungsvorgang kann zwischen zwei ([28]) und acht ([12]) variieren.

Ein üblicher Ansatz bei dieser Methode ist, wie bereits erwähnt, ein Vergleich des Empfehlungsdienstes mit einem Baseline-Algorithmus, wo der Benutzer seiner Anforderungen in einer einfachen Form abgeben kann und/oder wo die Ergebnisse in einer rangierten Liste dargestellt werden ([28], [27]).

Da es hier um die Evaluierung mehrerer Empfehlungsdienste handelt, kann dem Benutzer nach der Benutzung des jeweiligen Dienstes ein Fragebogen angeboten werden, der die Performance des Dienstes abschätzt, wobei der Vergleich

der Dienste mittels eines abschließenden, zusammenfassenden Fragebogens erfolgen kann.

Beispiele

Konkrete Beispiele findet man in:

- [5]: präsentiert eine Benutzerstudie, die versucht, zwei verschiedene wissenschaftliche Empfehlungsdienste, wo einer auf der *dynamischen kritischen Abhandlung* (Engl. «*dynamic critiquing*») und der andere auf der *kritischen Abhandlung von Beispielen* basiert, zu evaluieren und zu vergleichen, um die Vorteile und Nachteile der beiden Dienste besser herausziehen zu können. Dabei werden auch zwei andere wichtige Konzepte gegenübergestellt, da die erste Technik vom Dienst vorgeschlagen wird, und die zweite Benutzer-motiviert wird. Die Studie beschäftigte sich mit drei Kriterien: der Entscheidungsgenauigkeit, die die Anwender am Ende der Benutzung der Applikation erreicht haben sollte, dem Entscheidungsaufwand und der erreichten Zuversicht und dem Vertrauen auf den Dienst.
- [28] In [28] wird die Evaluierung von vorzugsbasierten Empfehlungsdiensten präsentiert, die eine vielversprechende Art der wissenschaftlichen Empfehlungsdienste darstellen. Bei diesem Ansatz wird der Benutzer bei seiner Suche durch die Benutzung seiner Präferenzen über den gewollten Gegenstand angeleitet. Die Autoren erklären weiter wie die Kritik von Beispielen sowie die Empfehlungsgenerierung im von ihnen entwickelten Tool *FlatFinder* modelliert wurde, und machen eine ausführliche Evaluierung von drei Ansätzen für die Unterstützung der Entscheidungen: das einfache Formausfüllen, die Kritik von Beispielen ohne zusätzliche Empfehlungsmodellierung und mit zusätzlicher Empfehlungsgenerierung.
- In [6] führen die Forscher einen hybriden Empfehlungsdienst vor, der durch Vermischung der Konzepte der Präferenz-basierten Empfehlung (*PrefORG*) und der selbst motivierten kritischen Abhandlung von Beispielen (*EC*) durch den Benutzer entstanden ist. Die Evaluierung des hybriden Dienstes (*Pref-ORG+EC*) war durch den Vergleich mit einem anderen hybriden Empfehlungsdienst (*DC+EC*), der neben den Kritiken von Beispielen (*EC*) auch die dynamischen kritischen Abhandlungen (*DC*) zufügt, durchgeführt. Sie wird als ein *Between-Group Experiment* ausgeführt, das sowohl offline (durch Bemessung der Interaktionszeit und -aufwand) als auch online (durch die Befragung der Benutzer nach der wahrgenommenen kognitiven Anstrengung, der Entscheidungssicherheit, der Kaufabsicht und der Rückkehrabsicht) durchgeführt werden kann.

Gemischte Online-Evaluierung

Bei der gemischten Online-Evaluierung ist man neben dem expliziten auch am impliziten Feedback interessiert. Die Offline Aspekte, die bei der gemischten Online-Evaluierung in der verwendeten Literatur betrachtet werden, sind:

- durchschnittliche Länge der Sessions [32],
- durchschnittliche Anzahl der Sessions [6],
- durchschnitt der Critiquing-Zyklen, bevor das gesuchte Produkt gefunden wurde [32],
- die Gesamtanzahl der betrachteten Produkte, bis der Benutzer mit einem zufrieden war [12],
- die Anzahl der besuchten Webseiten, der benutzten Empfehlungen, der aktivierten Reparaturaktionen und Produktvergleichen, sowie der Clicks an Produktdetails [12] und
- der Fehlerrate, die die Anzahl der gemachten Fehler pro bestimmter Aufgabe misst [27].

Beispiele

[12] betrachtet die wissensbasierten Empfehlungsdienste in Hinsicht auf das Benutzerverhalten. Dabei werden die Effekte von 8 verschiedenen Empfehlungsmethoden in einer Benutzerstudie durch implizites und explizites Feedback analysiert. Es handelt sich dabei um eine Empfehlungssapplikation für die Internet Provider. Es wurden die Fragen gestellt, in welchem Ausmaß die Empfehlungsdienste bessere Ergebnisse liefern, dies wenn der Benutzer bei der Auswahl keine Hilfe sondern nur eine einfache Produktliste bekommt, weiters ob die Erklärungen der Produktempfehlungen dem Benutzer helfen mehr über die benutzte Domäne zu lernen und auch sein Vertrauen an den Empfehlungsprozess zu erhöhen und ob der Vergleich der Produkten diesen Prozess unterstützt oder ihm schadet.

[32] In [32] beschreiben die Autoren von 2 Forschungsgruppen die Günstigkeiten von zwei verschiedenen interaktiven Empfehlungsdiensten: das *A-priori-basierte Verfahren* und das Verfahren, das auf der Multi-Attributen Nutzentheorie (MAUT) basiert und führen eine ausführliche Evaluierung der Dienste durch zwei Versuchsverfahren, für die speziell eine online Evaluationsplattform entwickelt wurde, vor. Nach der Evaluierung, bei der beide Empfehlungsstrategien durch die Befragung der Anwender mittels

online Evaluierungsplattform über Themen wie Empfehlungseffizienz, -genauigkeit, -qualität und das Benutzerinterface geprüft wurden, aber auch das implizite Feedback, sowie die Interaktionszeit und der Aufwand untersucht wurden, haben die beiden Dienste sehr gute Ergebnisse erreicht.

- [27] Anhand von zwei unterschiedlichen Benutzerstudien in Rahmen der Evaluierung von Empfehlungsdiensten die auf Basis der *kritischen Abhandlung von Beispielen* funktionieren, schlagen *Pu et al.* in [27] ein allgemeines Evaluierungsframework für die Multi-Attributen-Produktsuche und -Empfehlungsdienste (*MAPST*) vor, die dieses anhand von drei Kriterien überprüfen sollte: der Entscheidungsgenauigkeit, dem Entscheidungsaufwand und dem Benutzervertrauen in den Dienst. Aus diese Studien ziehen sie auch die anderen Faktoren die bei der Evaluierung betrachtet werden sollten, heraus: die Hilfsmittelvorbereitung, den Benutzer, das Design der Benutzeraufgaben, den Evaluierungskontext, sowie die Kriterien, die Maßnahmen und die Methodologie der Ergebnisanalyse.

Architektur des Ansatzes

3.1 Struktur des Projekts

Das Projekt besteht im Allgemeinen aus 4 Teilen: dem i^* Modell, das mit Hilfe von OpenOME-Plugin [43] für Eclipse [14] in Betracht auf mögliche Zielen der Benutzer entwickelt und in XML exportiert wurde; den Algorithmen, die diesen XML-Baum anhand des beim Empfehlungsserver angelangten Benutzer-Inputs untersuchen und bestimmte Ergebnisse an den Server übermitteln; dem Empfehlungsserver, der die Argumente, die im User-Interface der Applikation von den Benutzern eingetragen wurden, in Form einer XML-Datei vom Server empfängt und diesem die entsprechenden Empfehlungen wieder in XML zurück liefert, sowie den Prozessoren, die die Kommunikation zwischen dem Modell, den Algorithmen, dem Server und dem User-Interface erleichtern.

Die Struktur des Projekts kann man im Blockdiagramm des Projektes auf Abbildung 3.1 in der Seite 66 betrachten.

3.2 Modell für Xohana

In der Abbildung 3.2 auf Seite 130 absehbare Teil des i^* -Modells ist relativ vereinfacht - das volle i^* Modell würde neben den Softgoals und den Tasks auch andere Elemente beinhalten (wie z.B. Ressourcen, Hardgoals, Akteure usw.) Dies ist aber jedoch nur ein Auszug aus dem kompletten i^* -Modell für Xohana, das im Anhang B.1 auf der Seite 130 betrachtet werden kann.

Sowohl in dem Auszug als auch in komplettem Modell wird die Softgoals anhand des [7] systematisch dargestellt, d.h. der erste Teil des Softgoals reprä-

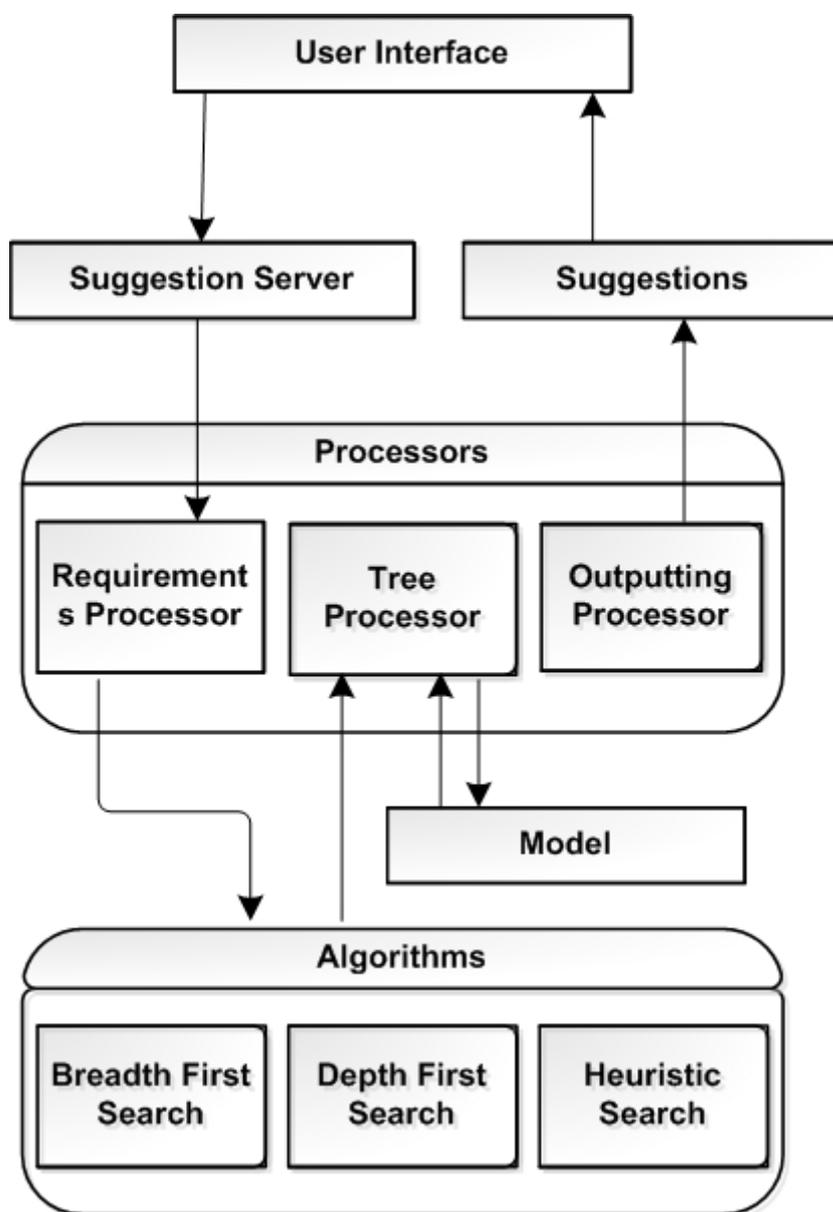


Abbildung 3.1: Blockdiagramm des Projekts

sentiert den Typ (Engl. «type») und in der eckigen Klammer wird das Thema des Softgoals (Engl. «topic») gezeigt (s. [7] für weitere Details).

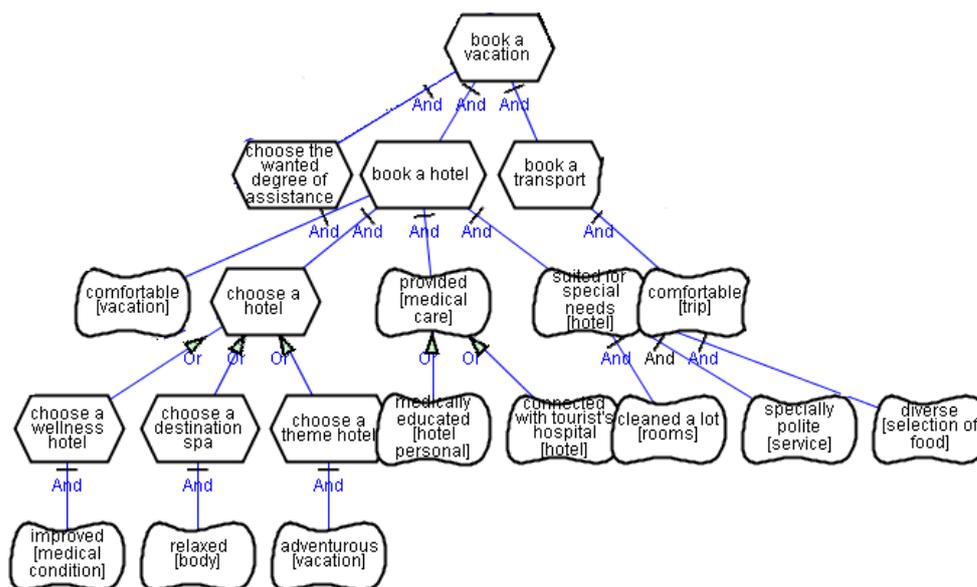


Abbildung 3.2: Auszug aus dem i^* -Modell für Xohana, Das komplette i^* -Modell für Xohana, kann man im Anhang B.1 auf der Seite 130 finden.

3.3 Die Input- und Outputmenge

Die Inputmenge beinhaltet Ziele, die der Benutzer mittels User-Interface (Abbildung 2.1, Seite 4) eingeben kann. In Abhängigkeit von der Wichtigkeit für den Benutzer, wählt er ob er sie als ein «Muss» oder ein «Soll» ausdrückt. Der Benutzer wird aber gleichzeitig auch durch kontinuierliches Vorschlagen von Zielen sowohl unterstützt aber auch auf die wichtigsten Ziele hingewiesen. Der User hat aber auch die Möglichkeit, seine Ziele unabhängig von den gebrachten Vorschlägen individuell zu formulieren.

Die Outputmenge besteht wieder aus möglichen Ziele für den Urlaub, die vom Benutzer auszuwählen oder gleich zu gestalten sind.

Empfehlungen

Die vom Empfehlungsdienst vorgeschlagenen Empfehlungen kann man in 2 Arten unterteilen:

1. Empfehlungen, die dem Benutzer vor der Eingabe der Anforderungen vorgeschlagen werden und
2. Empfehlungen, die der Benutzer während der Eingabe der Anforderungen bekommt.

Der Benutzer kann auch während des Ausdrückens der Ziele zu einer der Anforderungen zurückkehren. Dabei werden entsprechende Empfehlungen für diese frühere Anforderung in Erinnerung gebracht. Die Empfehlungen sind auch davon abhängig in welchem Feld sich der Benutzer gerade befindet. Zwei mögliche Felder stehen zur Auswahl: das linke Feld *Kriterium* («*Criterion*»), und das rechte Feld *Forderung* («*Demand*»). Wenn man eine der Zeilen im Eingabe-Interface mit den *i**-Softgoals vergleicht, kommt man zum folgenden Ergebnis: Jede Zeile sollte ein *Softgoal*, das rechte Feld das *Thema des Softgoals* und das linke Feld den *Typ des Softgoals* repräsentieren.

Die folgenden Paragraphen zeigen verschiedene Anwendungsfälle, wo zuerst das Benutzerinterface mit den Anforderungen und dann die Empfehlungen für beide Anforderungsfelder (die in der Applikation in der Datei «*suggestions.xml*» getrennt gesendet werden, abhängig davon in welchem Feld der Benutzer ist) erklärt sind.

3.4 Xohana-Algorithmen

Damit der Algorithmus weiß, wo er seine Suche beginnen sollte (falls das nicht der Fall ist, wo der Benutzer noch nichts eingetragen hat und ihm die Standardwerte für die jeweiligen Algorithmen geschickt wurden und vom Startelement im XML-Baum angefangen wurde) kann eine von der zwei Arten der Stringsuche benutzt werden: die **Teil-Stringsuche** (Engl. «*Part String Search*») oder die **strikte Stringsuche** (Engl. «*Exact String Search*»).

Teil-Stringsuche implementiert einige Methoden, um nach den Strings im XML-Baum suchen zu können. Grundsätzlich geht es darum, die Strings, die in einem Element des Baums enthalten sind, zu untersuchen, ausgehend von den Strings, die vom Benutzer der Applikation eingetragen wurden. Zuerst prüft eine der Methoden, ob der Elementstyp als Ziel beschrieben ist, indem sie seinen lokalen Namen genauer analysiert und konzentriert sich dann auf die Attribute des Elements die mit «*May*» anfangen, da diese die Softgoals darstellen. Wenn diese Attribute verfügbar sind, werden sie extrahiert, falls nicht, wird Null zurückgegeben. Der Vorgang wird weiter auf alle Elemente im Baum angewendet.

Präzise Stringsuche ist eine erweiterte Version der Teil-Stringsuche. Sie funktioniert in einer ähnlichen Weise und nutzt die annähernd gleiche Methode, aber nimmt nur exakte Treffer-Elemente die anhand der Suchbegriffe gefunden wurden, heraus. Die rekursive Arbeitsweise, um alle Elemente im Baum erreichen zu können, wird auch hier angewandt.

Gewichtungsfunktionen

Ein wichtiger Teil jedes für die *Xohana* entwickelten Algorithmus ist die dazugehörige Gewichtungsfunktion, die den Suchergebnissen gewisse Werte (im Bereich zwischen 0 und 1), entsprechend ihrer Wichtigkeit für die Benutzer hinzufügen, wobei 1 das wichtigste, und 0 am wenigsten wichtige Ergebnis darstellt.

Für die Auswahl der Vorschläge anhand des entwickelten Modells stehen 3 Algorithmen zur Verfügung. Für zwei werden klassische uninformierte Suchstrategien auf dem XML-Baum aus dem i^* formulierten Modell adaptiert, einer ist, reine informierte Suchstrategie. Der erste Algorithmus basiert auf der Breitensuche, der zweite auf der Tiefensuche und der dritte funktioniert auf einem selbst-entwickelten heuristischen Ansatz. Weiter ist für jeden der drei Algorithmen eine Gewichtungsfunktion vorhanden, die die Ergebnisse der Algorithmen entsprechend ihrer Relevanz für die Benutzer auswertet.

Adaptierte Breitensuche

Was die für *Xohana* entwickelte adaptierte Version von Breitensuche-Algorithmus angeht, gilt folgendes: Zunächst sieht die Breitensuche, ob das Startelement direkte Nachfolger hat und wenn ja, gibt sie diese als eine Liste von Elementen zurück. Wenn diese Liste aber über keine Ziel-Elemente verfügt, geht die Suche weiter und sucht von einer höheren Ebene höher, wer die Nachfolger zwei Ebenen unterhalb sind, d.h. immer wird das Startelement in der n -ten Ebene von einer höheren Startebene ausgewählt und die Nachfolger in noch einer n -Ebene tiefer (wieder ausgehend von der Ebene wo das ursprüngliche Startelement gefunden wurde) gesucht bis die Ziele gefunden sind oder der ganze Baum untersucht wird.

Gewichtungsfunktion für Breitensuche

Die direkten Nachfolger bekommen eine Gewichtung von 0.80, je weiter man ist, wird die Gewichtung mit 0.20 subtrahiert.

Adaptierte Tiefensuche

Da unser Modelbaum aus vielen Blättern (Endknoten) besteht, ist die für *Xohana* adaptierte Version von Tiefensuche so modifiziert, dass, wenn ein Blatt mit einem Task-Element verbunden ist, die Suche von diesem Element ausstartet. Falls es dabei um kein Blatt handelt, funktioniert die Suche standardmäßig. Die Arbeitsweise wird (wie bei der Breitensuche) iterativ auf alle Elemente des Baums angewendet.

Gewichtungsfunktion für Tiefensuche

Die Gewichtungen werden folgender Weise berechnet: Man berechnet die Länge des Pfades zwischen dem Startelement und jedem anderen Element, sowie auch den maximalen Pfad. Die Pfade der Elemente werden dann an (*maximalenPfad*+1) proportioniert, damit das am weitesten entfernte Element auch eine Gewichtung bekommen kann.

Heuristischer Ansatz

Der heuristische Ansatz sieht als Erstes nach, ob das Startelement direkte Nachfolger hat (s.g. Kinder, die sich eine Ebene tiefer befinden) und liefert sie in einer Liste von Elementen. Im Fall, dass diese Liste keine Zielelemente besitzt, prüft die Suche weiterhin ob das Element Geschwister hat (Elemente, die zu dem gleichen übergeordneten Element gehören). Wenn sich hier auch keine Ziele ergeben, sieht die Suche nach Tanten- und Onkeln - Elementen (Elemente, die Kinder der gleichen Großeltern - Elementen sind, und die eine Ebene höher liegen) nach und gibt sie zurück.

Gewichtungsfunktion für den heuristische Ansatz

Die Kinderelemente werden mit 1.0 bewertet, die Geschwisterelemente mit 0.75 und Tanten- und Onkelemente mit 0.5. Diese Bewertung der Elemente ist durch die Länge der Verbindung zwischen dem untersuchten Element und die Elemente, die sich in seiner Umgebung befinden, bestimmt - je näher sich ein Element zum untersuchten Element befindet, desto größer ist Bewertung - die ist dann natürlich für die Kinderelemente, die sich gleich unter dem Element befinden, am größten - 1.0.

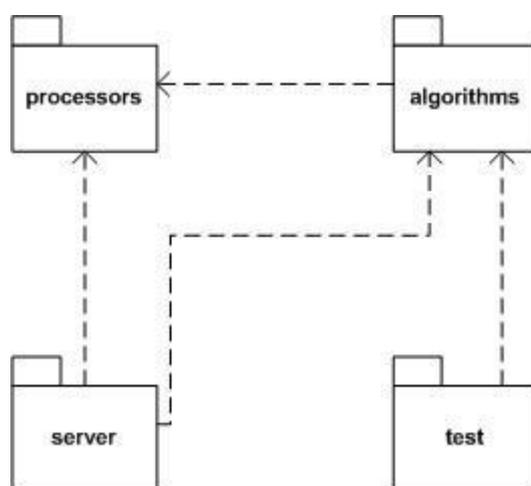


Abbildung 3.3: UML Package-Diagramm

3.5 Architekturskizze

UML Package-Diagramm

In der Abbildung 3.3 auf der Seite 71 kann man das Package-Diagramm des Projekts ansehen. Das Projekt besteht aus 4 Packages: *Processors*, *Algorithms*, *Server* und *Test*. Das *Server*-Package ist vom *Algorithms*-Package abhängig, da diese die parametrisierte Version der Algorithmen besitzt, die das *Server*-Package braucht, um Empfehlungen zu kreieren. Das *Processors*-Package enthält die *RequirementsProcessor*-Klasse, die der Server braucht, um die Requirements verstehen zu können. Das *Algorithms*-Package hängt auch vom *Processors*-Package ab, da die Klassen *OutputtingProcessor* so wie *TreeProcessor* ein Teil von ihm sind und vom *Algorithms*-Package gleichzeitig verwendet werden. Das *Test*-Package hängt direkt vom *Algorithms*-Package ab, in der gleichen Art und Weise wie das *Server*-Package: um Algorithmen testen zu können, muss man schon *ParametrizedAlgorithm* ausführen.

Die Klassenunterteilung

Wie die Klassen in den Packages untergeordnet werden kann man aus der Tabelle 3.1 auf Seite 72 ersehen.

Klassenunterteilung			
algorithms	processors	server	test
<u><i>BreadthFirst</i></u>	<u><i>TreeProcessor</i></u>	<u><i>HttpServer</i></u>	<u><i>TryAlgorithm</i></u>
<u><i>DepthFirst</i></u>	<u><i>OutputtingProcessor</i></u>		
<u><i>Heuristic</i></u>	<u><i>RequirementsProcessor</i></u>		
<u><i>ParametrizedAlgorithm</i></u>	<u><i>Constants</i></u>		
<u><i>ExactStringMatch</i></u>			
<u><i>PartStringMatch</i></u>			

Tabelle 3.1: Die Klassenunterteilung

Kurze Klassenbeschreibung

Klasse **BreadthFirst**

Diese Klasse implementiert die nicht-informierte Suche - Breitensuche (Engl. «*Breadth First Search*»).

Klasse **DepthFirst**

Diese Klasse implementiert die Tiefensuche (Engl. «*Depth-First Suche*»).

Klasse **Heuristic**

Diese Klasse implementiert einen möglichen Ansatz für informierte Suche (heuristischer Ansatz).

Klasse **ExactStringMatch**

Diese Klasse implementiert die Methoden, die die Ergebnisse von exakten String-Matching ergeben. Es erweitert die Klasse *PartStringMatch* so, dass sie die *exactMatch()*-Methode, hinzufügt, um nur exakte String-Matching Ergebnisse zu bekommen.

Klasse **PartStringMatch**

Diese Klasse implementiert die Methoden, die für Teil-String Matching genutzt werden.

Klasse ParametrizedAlgorithm

Die Klasse *ParametrizedAlgorithm* kann verwendet werden, um auswählen zu können, welche Art von Algorithmus benutzt wird. Abhängig vom Such-String (der von der Benutzer-Schnittstelle bezogen werden kann), der Art des Feldes, in dem sich der Benutzer derzeit befindet, der Art der benötigten Suche und der Art der String-Matching, werden die entsprechenden Parameter an die Methode `start()` geleitet, die dann wiederum die benötigten Methoden abrufen.

Interface Constants

Dieses Interface implementiert eine Menge von benötigten Konstanten, die im gesamten Programm benutzt wurden.

Klasse OutputtingProcessor

Die Klasse *OutputtingProcessor* sammelt die Methoden, die man für die Ausgabe der verarbeiteten Anforderungen in einer XML-Datei braucht.

Klasse RequirementsProcessor

Die Klasse *RequirementsProcessor* wird für die Verarbeitung der Anforderungszeichenfolge, die man vom Server bekommt, sowie und auch für das Extrahieren der Parameter, die für die zukünftige Verarbeitung von Bedeutung sind, verwendet.

Die Klasse TreeProcessor

Der *TreeProcessor* wird für die Verarbeitung des XML-Baums verwendet, der vom *i**Model erzeugt wurde.

Die Klasse HttpServer

Einfacher, aber voll-funktionierender HTTP/1.1 File Server. Da durch die Kooperation mit dem Server, der die Suche-String an diesen Server sendet, ein Fehler entsteht, kann dieser nicht gerade für das Testen der Algorithmen verwendet werden. Allerdings sollte das mit einer kleinen Anpassung dieser zwei Komponenten schnell möglich sein.

Die Klasse TryAlgorithm

Diese Klasse kann für das Testen der möglichen Kombination von Parametern verwendet werden.

Die Klassendiagramme

Die Klassendiagramme werden angesichts ihrer Zuordnung zu dem entsprechenden Package in 4 Gruppen unterteilt.

Algorithms

Das *Algorithms*-Package besteht aus 6 Klassen: *BreadthFirst*, *DepthFirst*, *Heuristic*, *ExactStringMatch*, *PartStringMatch* und *ParametrizedAlgorithm*. In welchen Beziehungen diese Klassen stehen, kann man aus dem Klassendiagramm für dieses Package leicht erkennen (Abbildung 3.4 auf der Seite 75): *PartStringMatch* ist eine Superklasse von der Klasse *ExactStringMatch*, die deshalb alle Klassen von *PartStringMatch* erbt und mit dem Benutzen der *exactMatch()*-Methode modifiziert. Der Zentralpunkt dieses Packages ist die *ParametrizedAlgorithm* Klasse - sie ist mit allen anderen Klassentypen (Algorithmen-Typen) verknüpft und ruft die Kombination dieser mittels der Methode *start()* und der gegebenen Parameterwerte auf.

Processors

Das *Processors*-Package besteht aus 3 Klassen (*TreeProcessor*, *OutputtingProcessor* und *RequirementsProcessor*) und einem Interface (*Constants*). Die Interaktion dieser Elemente kann man in der Abbildung 3.5 (Seite 76) verfolgen: *OutputtingProcessor* ist die Superklasse vom *TreeProcessor*, da dieser die *PrintingUtilities* hat, der das *TreeProcessor* immer wieder benutzt. Allerdings ist der *TreeProcessor* auch mit dem Interface *Constants* verbunden, da dieser das Interface implementiert. Der *RequirementsProcessor* steht hier ohne Verbindung - das aber deshalb, weil die Beziehung zwischen allen Klassen erst später untersucht wird - er wird dann eine Abhängigkeitsverbindung mit dem *HttpServer* haben.

Server

In diesem Package befinden sich 4 Klassen: *HttpServer*, *HttpFileHandler*, *WorkerThread* und *RequestListenerThread*. Die letzten 3 sind die inneren Klassen des *HttpServers*. Da es sich hier um ein Multithread Server handelt, steht die Beziehung zwischen ihnen im Verhältnis 1:n. Für das Projekt ist aber der *HttpFileHandler* am wichtigsten, da sich in ihm die Methode *handle()* befindet, die die Ausrufung der Bearbeitung der empfangenen Parameter und der Erzeugung der Empfehlungen erledigt (siehe 3.6 auf der Seite 76).

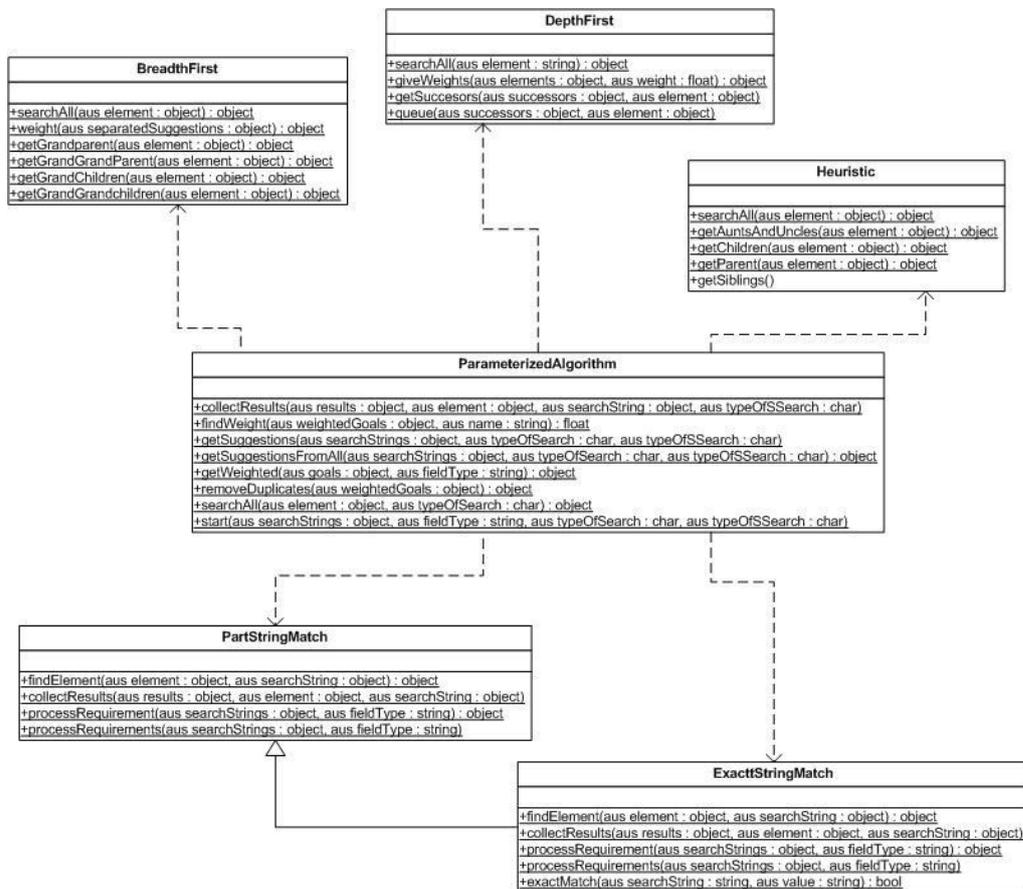


Abbildung 3.4: Klassendiagramm für Algorithms-Package

Test

Das *Test*-Package ist nur für das Testen gedacht und beinhaltet eine einzige Klasse: *TryAlgorithm*. Die *main()* Methode ruft den *ParameterizedAlgorithm* auf, was man auf der Seite 76 in der Abbildung 3.7 ansehen kann.

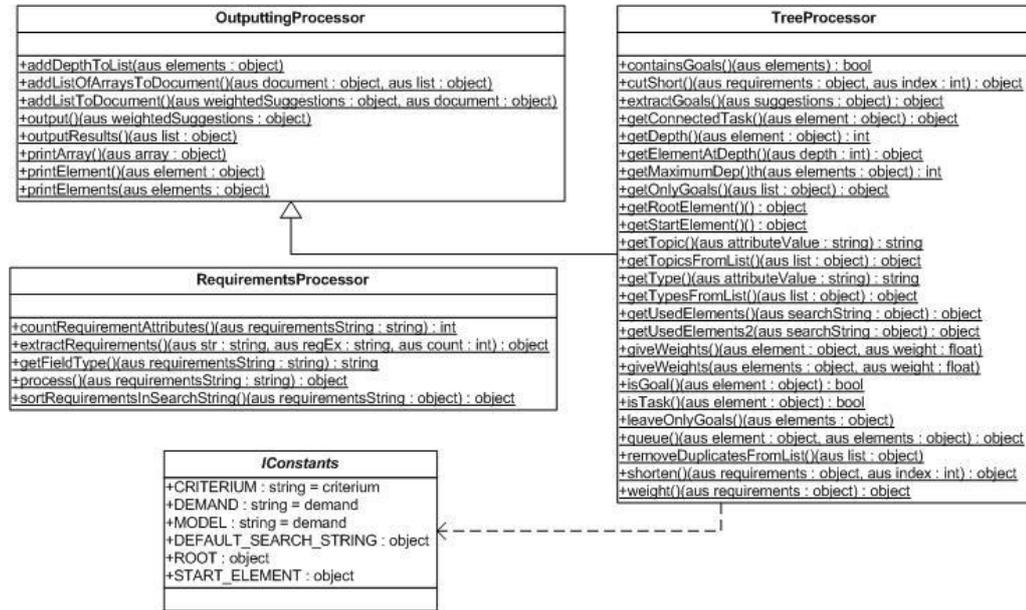


Abbildung 3.5: Klassendiagramm für Processors-Package

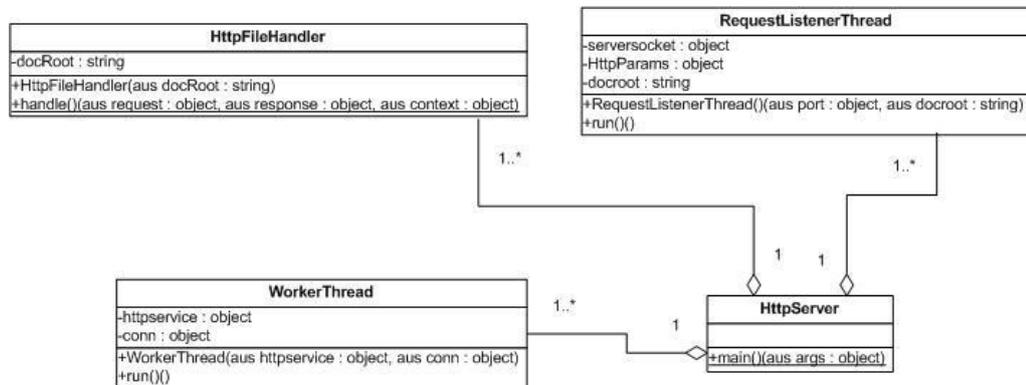


Abbildung 3.6: Klassendiagramm für Server-Package

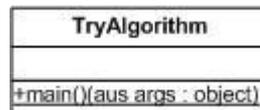


Abbildung 3.7: Klassendiagramm für Test-Package

Zusammensetzung aller Klassen

Die Abbildung 3.8 auf der Seite 77 zeigt einen Überblick über alle bestehenden Beziehungen zwischen allen Klassen. Das Hauptklassendiagramm kann man im Anhang C.1 auf der Seite 131 ansehen.

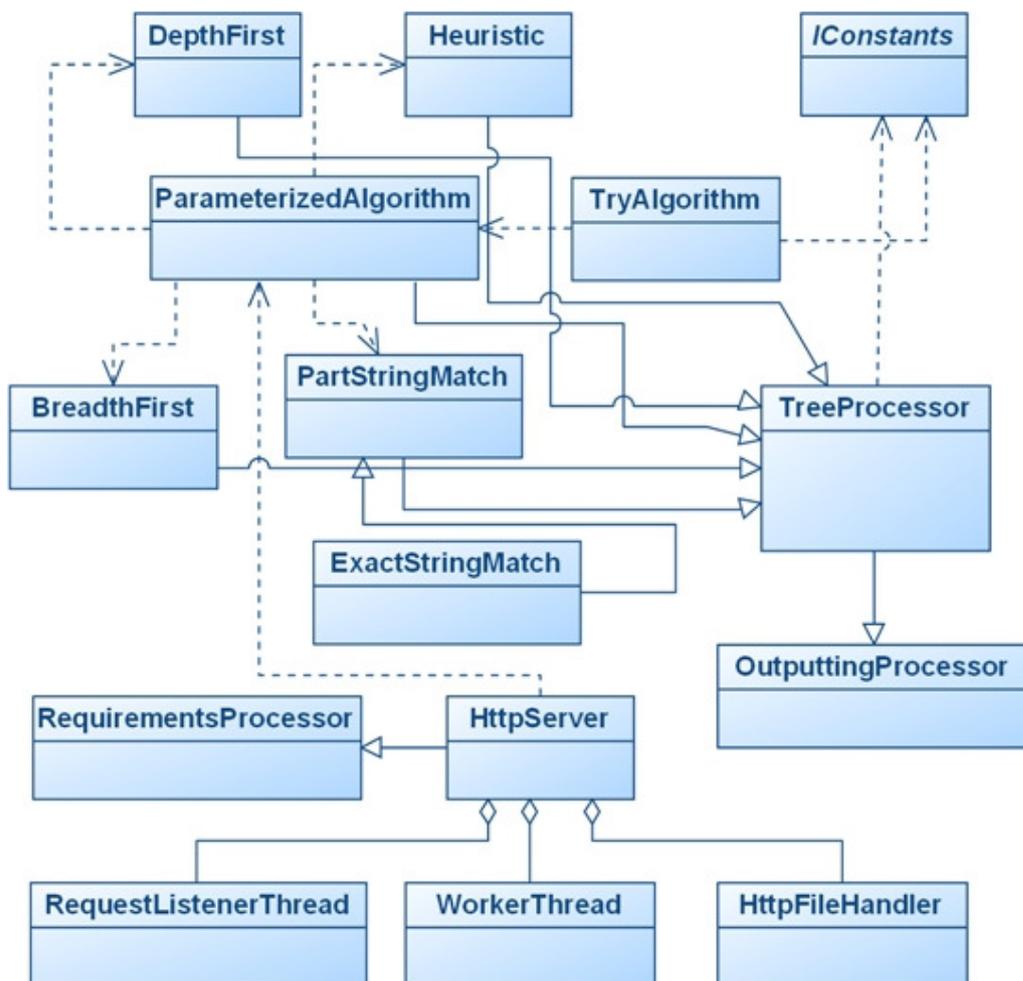


Abbildung 3.8: Zusammensetzung aller Klassen

Evaluierung

In diesem Kapitel wird die für den *Xohana*-Empfehlungsdienst durchgeführte Evaluierung vorgestellt. Die Evaluierung wird sowohl qualitativ (durch die Befragung der Testpersonen) als auch quantitativ (mittels numerischer Evaluierung der entstandenen Ergebnisse) durchgezogen. Als Ziele setzt man sich dabei, die Richtigkeit des Ablaufs des Empfehlungsdienst zu überprüfen und auszuwählen, welche Art der Empfehlungsgenerierung für *Xohana* am besten passt.

4.1 Vorbereitung der Evaluierung

Evaluierungsfragen

Während der Evaluierung wird man versuchen, die folgenden Fragen zu beantworten:

- Läuft der Empfehlungsdienst wie geplant ab? (vergleiche [17]) und [21])
- Welcher der drei Algorithmen (adaptierte Breitensuche, adaptierte Tiefensuche, heuristischer Ansatz) ist für die Empfehlungsgenerierung in *Xohana* am besten geeignet? ([52], [12], [27])

Evaluierungsaspekte

Evaluierungsaufgabe

Als Hauptaufgabe der *Xohana*, die evaluiert werden soll, gilt es *Finde alle gute Gegenstände zu identifizieren* [17].

Experimental Setup

Die benötigten Daten werden durch die Online-Evaluierung mittels Befragung der Benutzer der Applikation nach der Benutzung des Xohana gesammelt, was den *qualitativen* Aspekt der Evaluierung darstellt. Die erhaltenen Daten werden danach *quantitativ* bearbeitet.

Da Krebskranke eine sensitive und in der Regel schwer erreichbare Befragungsgruppe sind, wird man eine Gruppe von 10 Personen, der vorerst das Szenario als was für eine Person sie agieren sollten, erklärt wird, für die Befragung benutzen. Daher handelt es hier um eine szenariobasierte Evaluierung.

Präzision

Es wird die Exaktheit (Engl. «*Precision*») P und die Trefferquote (Engl. «*Recall*») R als Präzision benutzt, wobei die erste als «*the ratio of relevant items selected to number of items selected*» [17], und die zweite als «*the ratio of relevant items selected to total number of relevant items available*» [17] definiert wird, d.h.

$$P(S) = \frac{N_{rs}}{N_r}$$

wobei N_{rs} für die Anzahl der Empfehlungen die dem Benutzer gezeigt wurden und die er gefolgt oder als interessant gefunden hat, steht und N_r für die Anzahl der Empfehlungsevents ist, beziehungsweise gilt:

$$R(S) = \frac{N_{rs}}{N_s}$$

wobei N_{rs} die Anzahl der Empfehlungen die zu dem Benutzer gezeigt wurden und die er gefolgt hat oder als interessant gefunden hat ist und N_s die Anzahl der existierenden Empfehlungsmöglichkeiten darstellen.

Evaluierungsart

Es wird sowohl eine komparative als auch eine gemischte Online-Evaluierung durchgeführt, und die Evaluierung erfolgt dabei sowohl quantitativ als auch qualitativ.

Qualitativer Aspekt der Evaluierung

Dem Benutzer der Xohana wird nach der getrennten Benutzung aller drei Algorithmen des Xohana-Systems der entsprechende Fragebogen bereitgestellt. Es wird sich hier um ein Innerhalb-Betreff-Experiment handeln, da jeder Benutzer

alle drei Empfehlungsmethoden evaluieren wird. Mittels des Fragebogens werden folgende Evaluierungskriterien überprüft:

Entscheidungsaufwand - inwiefern den Benutzern das Ausdrücken ihrer Urlaubsanforderungen mit dem jeweiligen Algorithmus erleichtert wird (Entscheidungsaufwand wird zusätzlich auch quantitativ mittels der durchschnittlichen Länge der Sessions geprüft) [28],

Zutrauen an Entscheidung - wie sicher die Benutzer sind, dass sie ihre Urlaubswünsche gut ausgedrückt haben [27],

Absicht der Wiederbenutzung - um zu prüfen, welchen Eindruck das System hinterlassen hat [12].

Komparative Online-Evaluierung Dem Benutzer wird nach der Benutzung der jeweiligen Xohana-Empfehlungsmethode, sowie nachdem ihm die Benutzung eines Baseline-Algorithmus angeboten wurde, der gleiche Fragebogen angeboten, der in diesem Fall aus einer Liste aller Urlaubsziele besteht, aus der der Benutzer seine Ziele wählen kann. Abschließend wird ein Fragebogen bereitgestellt, in welchem der Benutzer evaluieren kann, welche der 4 Empfehlungsmethoden bei unterschiedlichen Kriterien besser abgeschnitten hat.

Quantitativer Aspekt der Evaluierung

Innerhalb der quantitativen Evaluierung werden folgende Aspekte überprüft:

Genauigkeit (Engl. «*precision*») und **Trefferquote** (Engl. «*recall*»), mittels der oben genannten Präzisionsdefinitionen [12],

Benutzerabdeckung (Engl. «*user coverage*»), die als Anteil der Anzahl an Benutzer deren mindestens ein Ziel erfolgreich empfohlen wird (N_{US}) an der Gesamtanzahl der Benutzern (N_U), definiert wird [52],

$$C_U(S) = \frac{N_{US}}{N_U}$$

Katalogabdeckung (Engl. «*catalog coverage*»), die durch die Anteil der Anzahl der Ziele aus dem Modell, die dem Benutzer je empfohlen wurden N_{RS} an der Gesamtanzahl an vorhandenen Empfehlungen N_C , gemessen werden kann [52],

$$C_C(S) = \frac{N_{RS}}{N_C}$$

durchschnittliche Länge der Sessions [32] und

durchschnittliche Anzahl der Sessions [6].

4.2 Durchführung der Evaluierung

Die Evaluierung wurde in einem Grazer Studentenheim ausgeführt, wobei die Testpersonen von verschiedenem Alter, Nationalität und Geschlecht gewählt wurden. Bei der Evaluierung haben 10 Testpersonen teilgenommen, im Alter zwischen 22 und 30 Jahre. Davon waren zwei Personen weiblich und acht männlich.

Allgemeine Vorgehensweise

Vom Standpunkt der Testperson

- Die Testperson wählt zuerst einen der vorhandenen Algorithmen auf dem Startfenster aus (Breitensuche, Tiefensuche oder Heuristische Suche stehen zur Auswahl bereit).
- Wenn die Testperson vorher schon einen Algorithmus evaluiert hat, wählt sie dann einen, den sie noch nicht benutzt hat.
- Es erscheint ein Formular mit Inputfeldern, wo sich das erste auf die eindeutige Form-ID und das zweite auf die auszudrückenden Softgoals bezieht.
- Bei der Eingabe hat die Testperson die Möglichkeit, eine der Zielempfehlungen auszuwählen oder geben ihre eigenen Empfehlungen einzugeben.
- Wenn die Testperson mit der Eingabe ihrer Urlaubsziele fertig ist, d.h., sicher ist, dass sie der Auswahl ihrer Ziele den Urlaub, den sie sich wünscht bekommen wird, klickt sie auf den «Submit»-Button. Somit werden ihre Ziele mit der eindeutiger Form-ID in der Datenbank gespeichert.
- Nach Abgabe ihrer Bedürfnisse meldet sich die Testperson beim Bewerter an, der ihr die entsprechenden Evaluierungsformulare aushändigt und die Applikation für die neue Benutzung vorbereitet.
- Nach der Benutzung und der Evaluierung von Breitensuche, Tiefensuche und Heuristische Suche wird der Testperson die Liste mit der Baseline-Tabelle vorgestellt sowie die entsprechenden Evaluierungsdokumente für diese Art von Empfehlung ausgehändigt.
- Wenn sie alle Suche-Algorithmen evaluiert hat, wird die Testperson angeboten, alle vier Arten durch die Bewertung zu vergleichen.

Vom Standpunkt des Bewerter

- Der Bewerter startet die Applikation für die Testperson und trägt die Startzeit auf das entsprechende Evaluierungsformular (die aus Server-Log auch eingelesen werden kann) ein.
- Nachdem die Testperson ihre Bedürfnisse ausgedrückt hat, trägt der Bewerter die Endzeit ein und speichert die Konsoleausgabe.
- Wenn die Testperson alle drei Xohana-Algorithmen innerhalb der Webapplikation bewertet hat, bietet der Bewerter der Testperson den Baseline - Algorithmus an, misst die Benutzungszeit und händigt ihr das Baseline-Evaluierungsformular aus.
- Nach dem alle Algorithmen benutzt und bewertet wurden, reicht der Bewerter den Vergleich- Bewertungsfragebogen der Testperson zu.

Bewertungsbogen

Nach der jeweiligen Benutzung des Algorithmus, wurde jeder Testperson angeboten, den benutzten Algorithmus mittels schriftlichem Bewertungsbogen zu bewerten. Die Bewertungsbogen waren für alle vier Algorithmenarten gleich und versuchten, mit Hilfe von folgenden Fragen, den Entscheidungsaufwand, das Zutrauen an die Entscheidung sowie die Absicht der Wiederbenutzung zu evaluieren:

Entscheidungsaufwand: Inwiefern wird Ihnen das Ausdrücken ihrer Urlaubsanforderungen mit Hilfe dieses Algorithmus erleichtert?

Zutrauen an die Entscheidung: Wie sicher sind Sie, dass Sie ihre Urlaubswünsche mit Hilfe dieses Algorithmus gut ausgedrückt haben?

Absicht der Wiederbenutzung: Wie gern würden Sie die Suche mit diesem Algorithmus wieder verwenden?

Die Testperson könnte auf die oben genannten Fragen mit Noten zwischen 1 und 5 antworten wobei 1 wenig und 5 viel bedeutete.

Der Fragenbogen zum Vergleich der vier Algorithmenarten, der nach der Benutzung aller Algorithmenarten der Testperson angeboten wurde, bewertete wieder die gleichen Kriterien, wobei die Testpersonen die benutzten Algorithmen in diesen drei Kategorien rangieren konnten und auf Platz eins den Algorithmus setzten, den sie in dieser Kategorie am besten gefunden haben, auf Platz vier den schlechtesten Algorithmus dieser Kategorie und die zwei anderen entsprechend dazwischen.

Die Testpersonen konnten ihre vorherige Eindrücke über die benutzten Algorithmen mittels der folgenden drei Fragen rangieren:

Entscheidungsaufwand: Welcher der vier Sucharten war am angenehmsten zu benützen?

Zutrauen an die Entscheidung: Bei welchem der vier Empfehlungsarten sind Sie am sichersten dass Sie damit den richtigen Urlaub finden werden?

Absicht der Wiederbenutzung: Welcher der vier Empfehlungsarten würden Sie den anderen Benutzern weiterempfehlen?

4.3 Evaluierungsergebnisse

Ergebnisse der qualitativen Evaluierung

Nach Durchführung der Evaluierung, wurden die Ergebnisse gesammelt und verglichen, beginnend mit der qualitativen Evaluierung. Einen qualitativen Vergleich der vier Algorithmen mittels der, in der Vorbereitung definierten, Fragen kann man in der Abbildung 4.1 auf der Seite 85 ansehen.

Qualitativer Vergleich der vier Algorithmen

Die Testpersonen haben das Benutzen am leichtesten empfunden, wenn die Ziele mittels der Tiefensuche oder der Breitensuche vorgeschlagen wurden - diese beiden Algorithmen wurden beide mit der durchschnittlichen Note von 4,2 bewertet. Den größten Entscheidungsaufwand stellte die Suche mit heuristischem Algorithmus (dieser wurde mit einer durchschnittliche Note von 3,7 bewertet). Der Baseline-Algorithmus erreichte dabei eine durchschnittliche Bewertung von 4.

Das allerhöchste Zutrauen an die getroffene Entscheidung bei der Auswahl von Zielen, die ihren Urlaub bestimmen sollten, hatten die Testpersonen bei Tiefensuche (durchschnittliche Note lag bei 4,4. Den zweite Platz holte sich die Baseline-Suche (durchschnittliche Note von 4,1), gefolgt von Breitensuche mit 3,9. Der letzte bei der Benotung war wieder der heuristische Ansatz mit der Bewertung von 3,8.

Bei der Absicht der Wiederbenutzung haben die Tiefensuche und die Breitensuche wieder die gleiche durchschnittliche Bewertung erreicht (4,3). Der heuristische Ansatz sowie der Baseline-Algorithmus waren dabei schlechter, mit einer durchschnittlichen Note von 4.

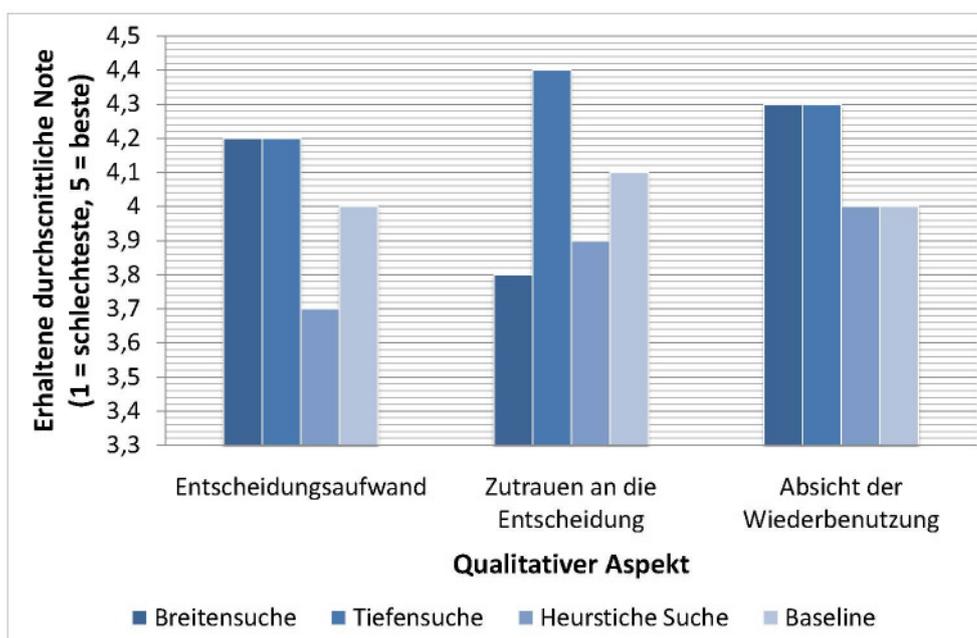


Abbildung 4.1: Qualitativer Vergleich der vier Algorithmen (nach Durchführung der Evaluierung, wurden die Ergebnisse aus Fragebogen gesammelt und verglichen). Den größten Entscheidungsaufwand stellte die Suche mit heuristischem Algorithmus. Das allerhöchste Zutrauen an die getroffene Entscheidung hatten die Testpersonen bei Tiefensuche. Bei der Absicht der Wiederbenutzung haben die Tiefensuche und die Breitensuche die gleiche durchschnittliche Bewertung erreicht.

Ranking

Die Ergebnisse der Ranking, die die Testpersonen am Schluss der Evaluierung gegeben haben, kann man in der Abbildung 4.2 auf der Seite 86 betrachten.

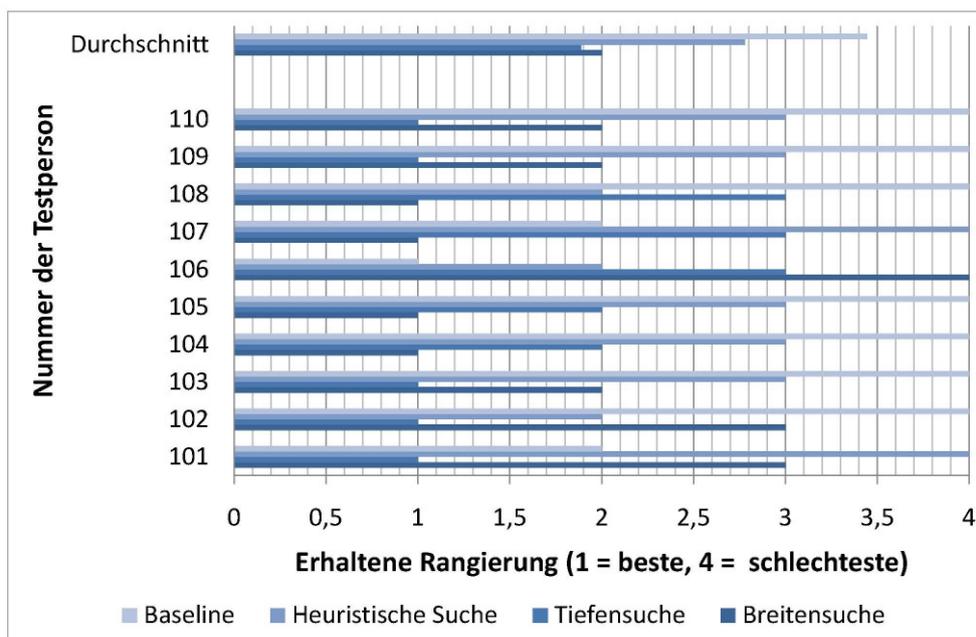


Abbildung 4.2: Rankingergebnisse, die Testpersonen am Schluss der Evaluierung gegeben haben. Die Tiefensuche wurde am besten gerankt, nah gefolgt von der Breitensuche. Die schlechteste durchschnittliche Ranking hatte der Baseline-Algorithmus.

Wenn man die Resultate der Ranking betrachtet, war die Tiefensuche mit durchschnittlicher Ranking 1,89 am besten gerankt, nah gefolgt von der Breitensuche mit durchschnittlicher Ranking 2. Die schlechteste durchschnittliche Ranking hatte der Baseline-Algorithmus (3,4). Nicht viel besser im Durchschnitt war die heuristische Suche (2,78).

Interpretation der Ergebnisse der qualitativen Evaluierung

Wenn man die Ergebnisse der Umfrage, die mit den Testpersonen durchgeführt wurden, genauer betrachtet, ist offensichtlich, dass ihnen die Tiefensuche am

meisten gefallen hat. Sie haben das Großteils damit begründet, dass ihnen diese Suchart am Anfang viele Ziel-Möglichkeiten zeigt und deshalb einen Überblick anbietet worauf sie sich konzentrieren sollten. Die schlechtere Benotung des Baseline-Algorithmus erklärten sie mit Mängeln wie fehlende Interaktivität sowie zu große Anzahl von Zielen, die sie auf einmal untersuchen sollten und keine Betonung welche dabei wichtig sind und welche nicht. Die heuristische Suche auf der anderen Seite, hat nach einer bestimmten (nicht großen) Anzahl von Zielen keine Vorschläge mehr gegeben und wurde deshalb niedriger bewertet. Das Vorschlagen von Zielen mittels der Breitensuche fanden sie auf jedem Fall interessant, waren aber durch niedrigere Anzahl der Ziele, die pro Zeile gezeigt wurden, nicht überzeugt, dass ihnen alle wichtigen Ziele auch gezeigt wurden.

Ergebnisse der quantitativen Evaluierung

In der quantitativen Evaluierung wurden die Informationen über die Länge der Benutzung, der Katalogabdeckung, die durchschnittliche Anzahl von Zielen, die Genauigkeit, die Trefferquote sowie die durchschnittliche Anzahl der Sessions nach der durchgeführten Evaluierung gesammelt.

Länge der Benutzung der Applikation

Die kürzeste Länge der Benutzung (die Zeit, die die Testperson gebraucht hat, um ihre Ziele auszudrücken) erreichte der Baseline-Algorithmus mit durchschnittlichen 142,92s, gefolgt von der heuristischen Suche mit durchschnittlichen 249,3s (siehe Abbildung 4.3 auf der Seite 88). Für das Ausdrücken von Zielen mittels der Tiefensuche brauchten die Testpersonen durchschnittlich 309s, die Benutzung dauerte bei der Breitensuche am längsten - durchschnittlich 462s.

Länge der Benutzung pro Ziel

Bei der heuristischen Suche haben die Testpersonen aber am wenigsten Ziele vorgeschlagen bekommen. Um den Einfluss der Anzahl der ausgedrückten Ziele zu verhindern, wurden die durchschnittlichen Längen der Benutzung mit der Anzahl der vorgeschlagenen Ziele dividiert - die Ergebnisse sind in der Abbildung 4.4 auf der Seite 89 ersichtlich. Dabei sieht man dass die Testpersonen durchschnittlich am wenigsten Zeit gebraucht haben, ein Ziel mittels der Tiefensuche auszudrücken (4,32s), nah gefolgt vom Baseline-Algorithmus (4,93s). Weit langsamer waren die Testpersonen beim Ausdrücken von Zielen mittels heuristischem Ansatz (durchschnittlich 8,12s) und der Breitensuche (durchschnittlich 8,70s).

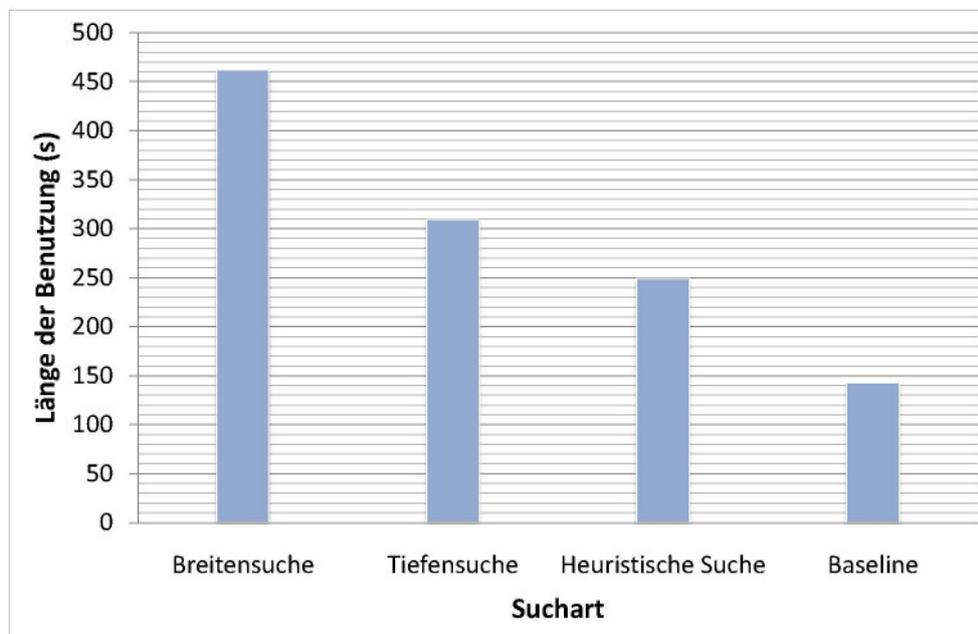


Abbildung 4.3: Länge der Benutzung (die Zeit, die die Testperson gebraucht hat, um ihre Ziele auszudrücken). Für das Ausdrücken von Zielen mittels dem Baseline-Algorithmus brauchten die Testpersonen durchschnittlich am wenigsten Zeit. Die Benutzung dauerte bei der Breitensuche am längsten.

Katalogabdeckung

Einen weiteren quantitativen Aspekt stellte die Katalogabdeckung dar, wo das beste Resultat wie erwartet der Baseline-Algorithmus (eine 100% Katalogabdeckung) lieferte, da dieser alle Ziele auf einer Liste auf einmal zeigt. Die Tiefensuche und die Breitensuche erreichten jedoch auch gute Ergebnisse (jeweils 98,65%). Die schlechteste Katalogabdeckung zeigte die heuristische Suche (eine 91,89% Katalogabdeckung). Die Ergebnisse werden graphisch auf der Abbildung 4.5 auf der Seite 90 angezeigt.

Durchschnittliche Anzahl der Ziele

Einen Vergleich der durchschnittlichen Anzahl der abgegeben und vorgeschlagenen Ziele sowie die durchschnittlichen Anzahl der Ziele, die aus dem i^* -Modell,

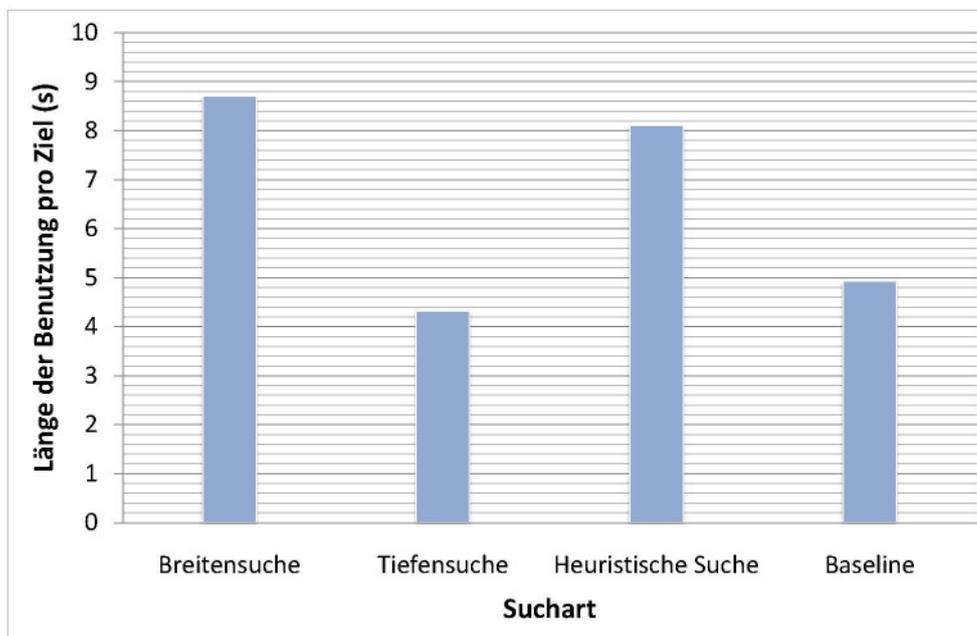


Abbildung 4.4: Länge der Benutzung pro Ziel (durchschnittlichen Längen der Benutzung wurde mit der Anzahl der vorgeschlagenen Ziele dividiert, um den Einfluss der Anzahl der ausgedrückten Ziele zu verhindern). Dabei sieht man dass die Testpersonen durchschnittlich am wenigsten Zeit gebraucht haben, ein Ziel mittels der Tiefensuche auszudrücken. Weit langsamer waren die Testpersonen beim Ausdrücken von Zielen mittels heuristischem Ansatz.

das für *Xohana* entwickelt wurde, stammen, kann man in der Abbildung 4.6 auf der Seite 91 ansehen. Am meisten wurden die Ziele bei der Baseline-Suche abgegeben (durchschnittlich 15,9 pro Session). Die Breitensuche erreichte dabei durchschnittlich 12,4 Ziele pro Session, die Tiefensuche hatte ein ähnliches Resultat (12 Ziele pro Session) und die heuristische Suche hat sich auch in diesem Aspekt der quantitativer Evaluierung mit durchschnittlich 7,6 Zielen pro Session nicht gut gezeigt.

Genauigkeit und Trefferquote

Die Vergleiche der Ergebnisse der zwei Evaluierungsmetriken - Genauigkeit und Trefferquote - der jeweiligen vier Sucharten sieht man in der Abbildung 4.7 auf

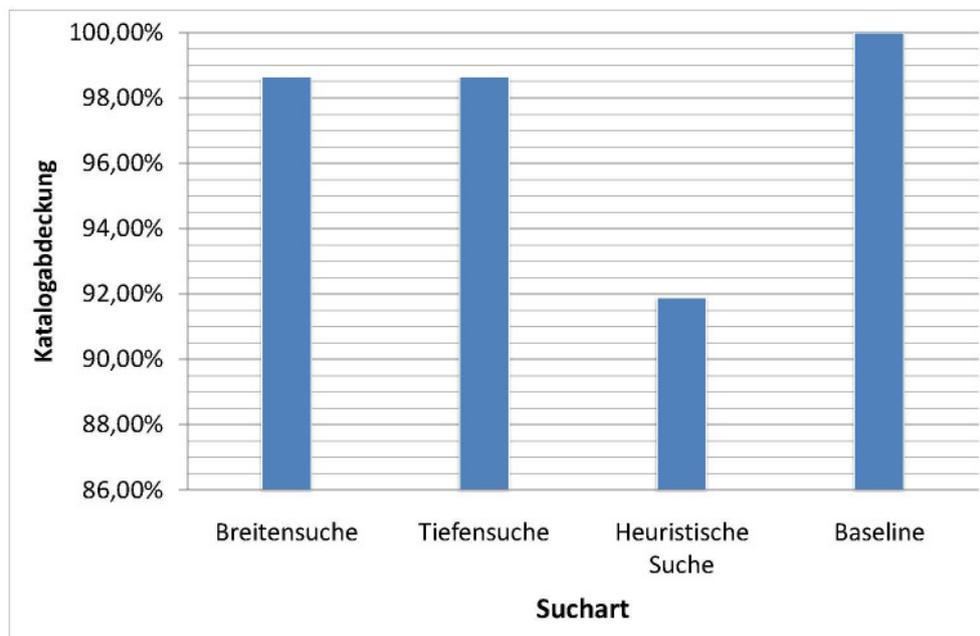


Abbildung 4.5: Katalogabdeckung. Der Baseline-Algorithmus lieferte eine 100% Katalogabdeckung, Die Tiefensuche und die Breitensuche erreichten jedoch auch gute Ergebnisse. Die schlechteste Katalogabdeckung zeigte die heuristische Suche.

der Seite 92 und in der Abbildungen 4.8 auf der Seite 93.

Die weit größte Genauigkeit hatte der Baseline-Algorithmus (0,55). Hinter ihm waren die heuristische Suche (Genauigkeit von 0,22), die Breitensuche (Genauigkeit von 0,21) und die Tiefensuche (Genauigkeit von 0,16). Ganz anders war es aber bei der Trefferquote: obwohl die Baseline-Suche wieder das beste Resultat gezeigt hat (Trefferquote von 0,55) war da die Tiefensuche im Gegensatz zu den Genauigkeitsergebnissen ganz gut (Trefferquote von 0,30). Die Breitensuche war nicht viel schlechter (Trefferquote von 0,29), die heuristische Suche hatte die weit niedrigste Trefferquote - 0,19.

Durchschnittliche Anzahl der Interaktionen

Um die Interaktion zwischen den Testpersonen und den Algorithmen schätzen zu können wurde nach der Durchführung der Evaluierung auch die durchschnitt-

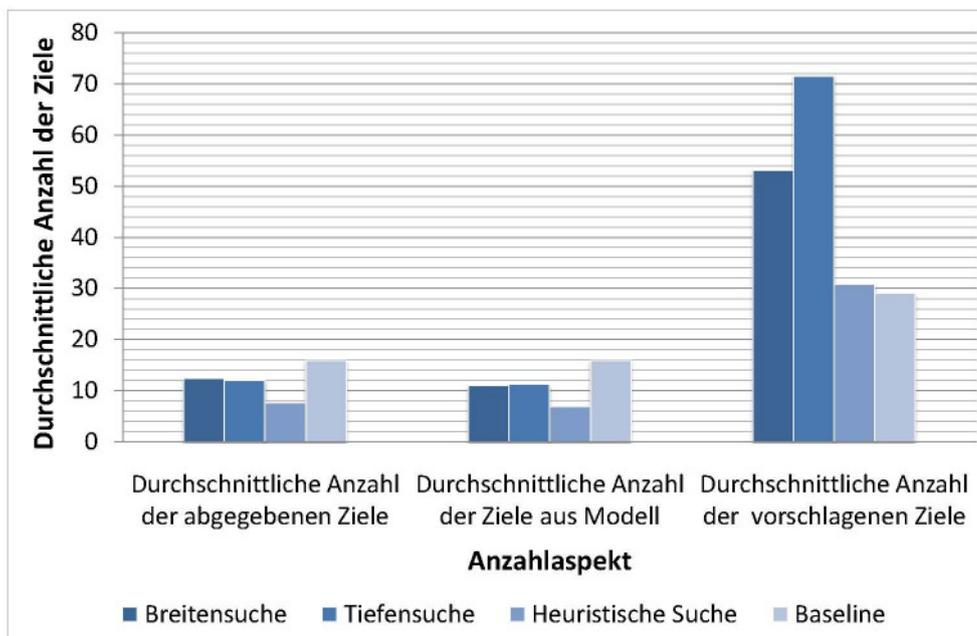


Abbildung 4.6: Vergleich der durchschnittlichen Anzahl der Zielen, die abgegeben wurden, die aus i^* -Modell stammten oder die vorgeschlagen wurden. Am meisten wurden die Ziele bei der Baseline-Suche abgegeben, gefolgt von Tiefensuche, bei der Tiefensuche wurde aber die höchste Anzahl an Zielen vorgeschlagen. Bei der Baseline-Suche stammten alle Ziele die ausgewählt wurden aus i^* -Modell.

liche Anzahl der Sessions, die eine Testperson während der Benutzung der Applikation gehabt hat, berechnet. Die Ergebnisse werden in der Abbildung 4.9 auf der Seite 94 graphisch dargestellt. Am wenigsten brauchten die Testpersonen mit der Applikation, die der heuristische Ansatz für das Vorschlagen von Zielen benutzt hat, zu interagieren (durchschnittlich 21,2 Interaktionen pro Session). Das Vorschlagen von Zielen mittels der Tiefensuche lag ziemlich entfernt - durchschnittlich 29 Interaktionen pro Session. Mit der Applikation der Breitenuche brauchten die Testpersonen am meisten Hilfe - durchschnittlich waren hier 31,5 Interaktionen pro Session notwendig. Die Interaktion mit der Baseline-Suche wurde hier nicht bewertet, da hierbei alles in einer Session entwickelt.

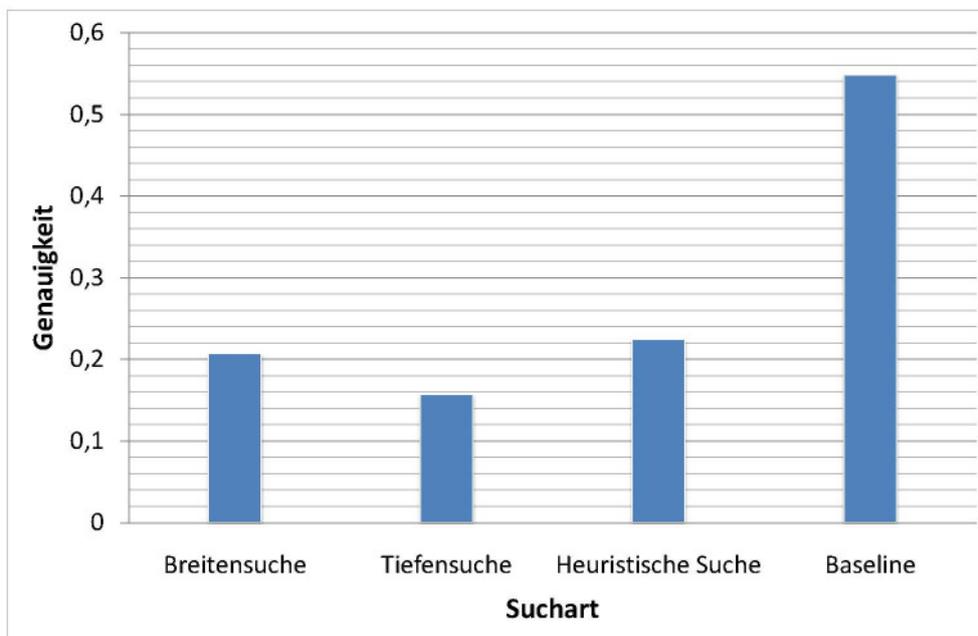


Abbildung 4.7: Genauigkeit. Die weit größte Genauigkeit hatte der Baseline-Algorithmus. Hinter ihm waren die heuristische Suche, die Breitensuche und die Tiefensuche.

Normierte durchschnittliche Anzahl der Interaktionen

Um den Einfluss der Anzahl der vorgeschlagen Zielen auch hier zu eliminieren, hat man wieder die oben genannten Zahlen durch die Anzahl der vorgeschlagen Zielen dividiert, sowie zusätzlich durch zwei dividiert (da beim Ausdrücken von Zielen zwei Felder - Kriterium und Nachfrage - ausgefüllt sein sollten). Die Ergebnisse sind in der Abbildung 4.10 auf der Seite 95 ersichtlich. Wieder haben die Testpersonen die Tiefensuche als am intuitivsten befunden - die können schon in 1,21 Interaktionen pro Feld (1/2 vom Ziel) ausdrücken. Langsamer waren sie bei der Breitensuche - 1,29 Interaktionen pro Feld und heuristische Suche - 1,34 Interaktionen pro Feld.

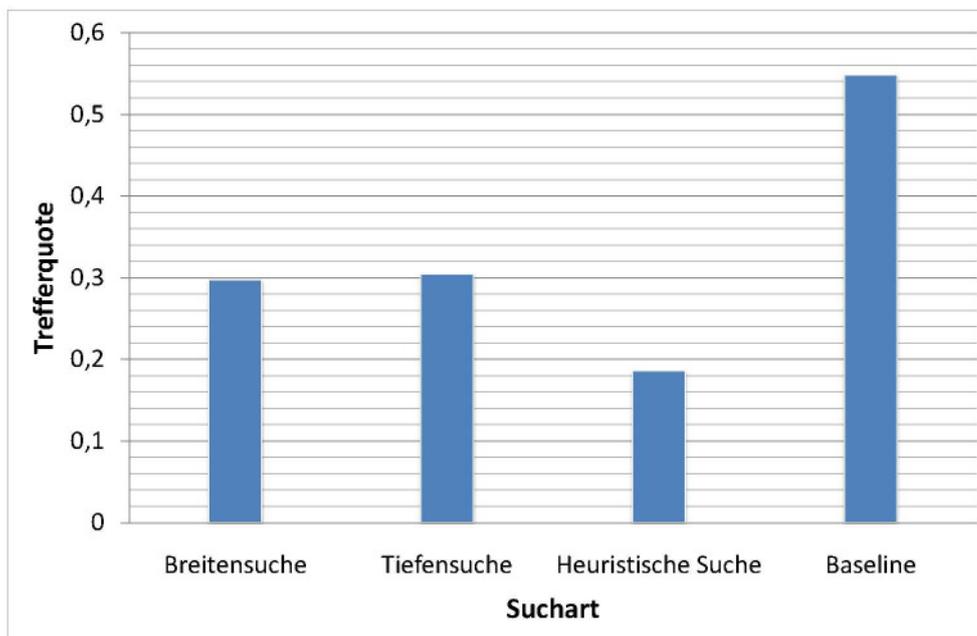


Abbildung 4.8: Trefferquote. Obwohl die Baseline-Suche wieder das beste Resultat gezeigt hat, war da die Tiefensuche im Gegensatz zu den Genauigkeitsergebnissen ganz gut. Die Breitensuche war nicht viel schlechter, die heuristische Suche hatte die weit niedrigste Trefferquote.

Interpretation der Ergebnisse der quantitativen Evaluierung

Die quantitative Evaluierung hat die Ergebnisse der qualitativen Evaluierung bestätigt - die Tiefensuche hat sich auch hier als überzeugend gezeigt. Sie hat die besten Ergebnisse schon bei den normierten Eigenschaften wie Länge der Benutzung pro Ziel und normierte durchschnittliche Anzahl der Interaktionen sowie Trefferquote erreicht. Die erreichte Katalogabdeckung war auch sehr gut und konnte nur vom Baseline-Algorithmus übertroffen werden. Die schlechteren Ergebnisse bei den anderen quantitativen Eigenschaften (insbesondere der Genauigkeit) lassen sich durch die hohe Anzahl an Zielen, die den Testpersonen vorgeschlagen wurden, (was ihnen eigentlich, wenn man die Aussage aus qualitativen Evaluierung in Betracht nimmt, gefallen hat) erklären. Auf der anderen Seite hat die Baseline-Suche auch gute Ergebnisse erreicht. Diese sind aber mit der Anzahl der Ziele, die den Testpersonen vorgeschlagen wurden, eng verbun-

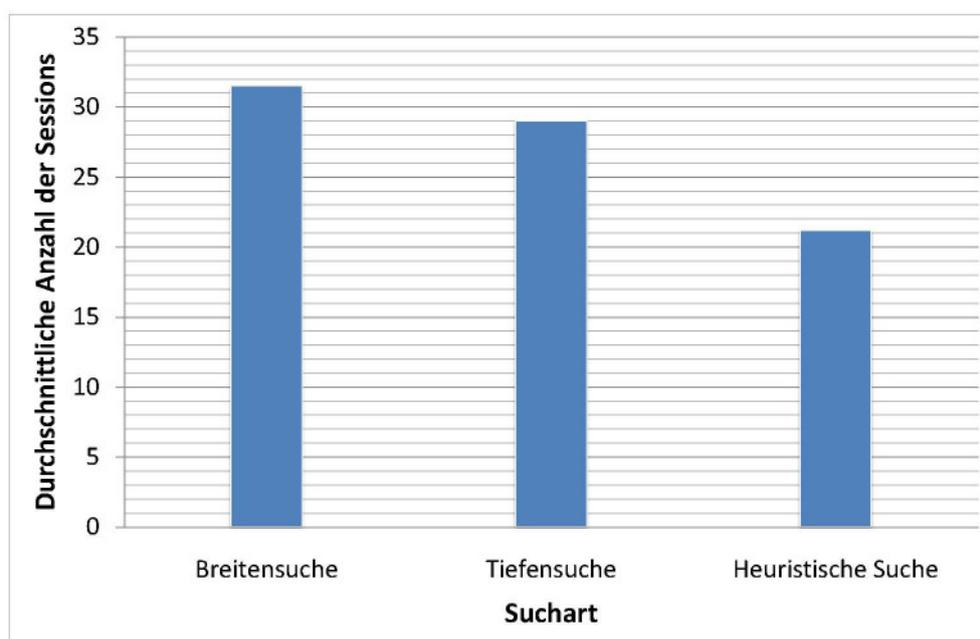


Abbildung 4.9: Durchschnittliche Anzahl der Interaktionen (d.h., die durchschnittliche Anzahl der Sessions, die eine Testperson während der Benutzung der Applikation gehabt hat). Am wenigsten brauchten die Testpersonen mit der Applikation, die der heuristische Ansatz für das Vorschlagen von Zielen benutzt hat, zu interagieren. Mit der Applikation der Breitensuche brauchten die Testpersonen am meisten Hilfe. Die Interaktion mit der Baseline-Suche wurde hier nicht bewertet, da hierbei alles in einer Session entwickelt.

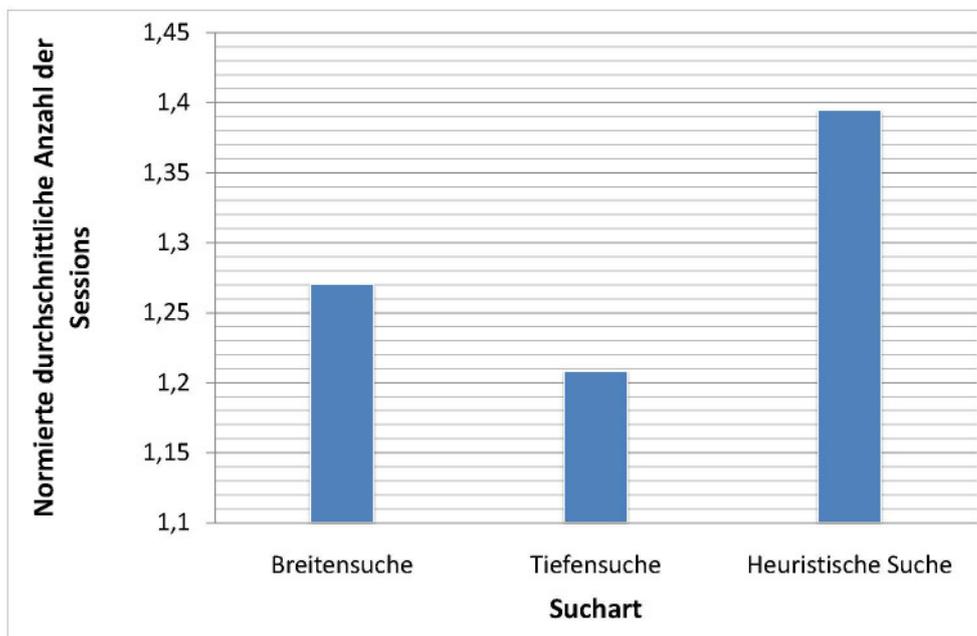


Abbildung 4.10: Normierte durchschnittliche Anzahl der Interaktionen (um den Einfluss der Anzahl der vorgeschlagen Ziele auch hier zu eliminieren, hat man wieder durch die Anzahl der vorgeschlagen Ziele, sowie zusätzlich durch zwei dividiert). Wieder haben die Testpersonen die Tiefensuche als am intuitivsten befunden.

den (in diesem Fall war die Anzahl der Ziele im Modell im Vergleich mit einer Realtime-Applikation vergleichsweise klein), was die Benutzung der Applikation mit diesem Algorithmus höchstwahrscheinlich bei einer großen Anzahl von Zielen sehr unübersichtlich und schwer nutzbar machen würde. Die Breitensuche hat auch ziemlich gute Ergebnisse erreicht, war aber nicht bestehend wie die Tiefensuche. Was wiederum sehr enttäuschend war, war die heuristische Suche, die nicht nur qualitativ sondern auch quantitativ schlechte Resultate lieferte.

Fazit

Die kombinierten Ergebnisse der beiden Evaluierungsarten haben unter Berücksichtigung der bestehenden Faktoren einen klaren Sieger gezeigt -die Tiefensuche. Diese hat auf der einen Seite den Testpersonen sehr gefallen, auf der an-

deren Seite hat sich das dann in die Ergebnisse der quantitativen Evaluierung eingespielt. Die Testpersonen konnten schnell und mit wenigen Interaktionen mit dem Empfehlungsdienst ihre Bedürfnisse ausdrücken. Dabei erzielte diese Suchart auch eine gute Trefferquote, was die Ergebnisse der Umfrage über den Entscheidungsaufwand, das Zutrauen an die Entscheidung, sowie die Absicht der Wiederbenutzung vorher vorausgesagt hatten. Die Katalogabdeckung war bei allen vier Algorithmen hoch, wobei man jedoch in Betracht nehmen muss, dass es sich hier um ein kleineres Zielmodell handelt, was auch ein schlechtes Licht auf die guten Ergebnisse des Baseline-Algorithmus wirft.

Zusammenfassung und Ausblick

5.1 Zusammenfassung

In dieser Arbeit wurde eine Applikation im Bereich von wissensbasierten Empfehlungsdiensten entwickelt, die anhand einer Ontologie der Benutzerziele, die in einem selbst erfassten i^* -Modell enthalten ist, den Benutzern mit speziellen Bedürfnissen die möglichen Urlaubsziele vorschlägt.

Nach der Einführung in die für die Arbeit wichtigen Themen wie *Xohana Software System*, *Modellierung von Zielen*, *Suchalgorithmen* sowie Empfehlungsdienste, wurde die Architektur des selbst entwickelten Ansatzes vorgestellt, gemeinsam mit dem Kern der entwickelten Applikation - 3 einfache Suchalgorithmen (*Tiefensuche*, *Breitensuche* und *heuristischer Ansatz*), die speziell für *Xohana* angepasst wurden. Durch die quantitative und qualitative Evaluierung wurden die Vorteile und Nachteile der oben erwähnten Ansätze untersucht sowie sowohl miteinander als auch mit dem Baseline-Algorithmus verglichen.

Aufgrund der durchgeführten Evaluierungen der Algorithmen kann man folgende Schlussfolgerungen ziehen:

- Sowohl quantitativ als auch qualitativ hat sich die Tiefensuche für diesen Fall im Durchschnitt als am besten geeignet gezeigt.
- Die Algorithmen sind sehr von der Anzahl der Ziele im Modell sowie von der Anzahl der vorgeschlagene Ziele abhängig. Wenn man diese Einflüsse ausschaltet, ergeben sich andere Ergebnisse.

- Den Baseline-Algorithmus haben die Testpersonen schon bei einer geringeren Anzahl von Zielen im Modell als benutzerunfreundlich empfunden.
- Den Testpersonen ist eine größere Anzahl von Vorschlägen lieber als eine geringere. Das unterstützt das Zutrauen an die Entscheidung und die Absicht der Wiederbenutzung und verringert den Entscheidungsaufwand.

5.2 Ausblick

Im Rahmen dieser Arbeit ergaben sich zahlreiche Anwendungs- aber auch Verbesserungsmöglichkeiten der angewandten Methoden. In dieser Abschnitt sollen offene Fragestellungen, die in dieser Arbeit nicht abschließend behandelt werden konnten, aufgelistet und besprochen werden.

Es ist zu überlegen, wie man das vereinfachte i^* -Modell mit anderen Beziehungstypen aus der i^* -Modellierung erweitern könnte, besonders wenn es um den negativen Einfluss der Ziele zwischen einander geht, sowie wie die Algorithmen angepasst werden sollten um diese Eigenschaften des Modells auch benutzen zu können.

Die früher beschriebene zweiteilige Evaluierung würde auch genauere Ergebnisse liefern, wenn die Anzahl der Testpersonen größer wäre als die, im Rahmen dieser Arbeit getestet.

Zusätzlich, wie am Anfang der Arbeit schon erwähnt, wenn die Benutzer die *Xohana* gefällt, wird es mit der Zeit durch die größere Anzahl der Benutzer sowie gesammeltes kollaboratives und inhaltsbasiertes Wissen möglich, ausführliche Profile über die Benutzer zu bilden, sowie implizite Informationen und das Wissen der anderen Benutzer in der Empfehlungen miteinzubeziehen. Es wäre dann auch möglich, nicht nur das Wissen über die Domäne, das durch die Experten und Wissenserforschung gesammelt werden, zu benutzen, sondern auch den Empfehlungsprozess mit kollaborativen und/oder inhaltsbasierten Empfehlungsmethoden zu erweitern und somit der bestehende Empfehlungsdienst zu verbessern.

Anwendungsfälle

A.1 Anwendungsfall 1

Gegeben: Leere Anforderung

Title:

| should be Remove

Add requirement

Abbildung A.1: Gegeben: Leere Anforderung

Breadth-First Suche

Breadth-First verwendet auch die Default-Werten wenn sie eine leere String bekommt, d.h. startet die Suche auch vom defaulten *START-ELEMENT* (siehe auch *Interface Constants*). Die Ergebnisse kann man in der Tabelle [A.1](#) auf der Seite [100](#) genauer betrachten.

Depth-First Suche

Depth-First benutzt seine Default-Werten wenn nur leere Suche-String gegeben ist, d.h. er startet die Depth-First Suche von dem ersten Element im Model - *START-ELEMENT* (siehe Tabelle [A.2](#) auf Seite [100](#)).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.1: Ergebnisse der BFS für den Anwendungsfall 1

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 0.8333333	strictly avoided 0.8333333
health 0.8333333	improved 0.8333333
sleep 0.6666666	gathered 0.6666666
stress 0.6666666	avoided 0.6666666
booking 0.6666666	extra reliable 0.6666666
assistance 0.3333333	provided 0.3333333
group 0.16666663	having competent tour guide 0.1666666
tourist 0.3333333	independent 0.3333333
journey 0.16666663	managed on my own 0.16666663
vacation 0.5	comfortable 0.5
medical care 0.5	suited for special needs 0.5
hotel 0.5	medically educated 0.3333333
hotel personal 0.3333333	connected with tourist's hospital 0.3333333
rooms 0.3333333	cleaned a lot 0.3333333
service 0.3333333	specially polite 0.3333333
selection of food 0.3333333	diverse 0.3333333
medical condition 0.16666663	suitable 0.3333333
body 0.16666663	relaxed 0.16666663
trip 0.5	adventurous 0.16666663
enough rest 0.6666666	short 0.5
holiday 0.5	medium 0.5
weather 0.6666666	long 0.5
benefits of summer 0.5	enjoyable 0.6666666
benefits of autumn 0.5	used 0.5
benefits of winter 0.5	near hospital 0.6666666
location 0.6666666	beautiful 0.6666666
trails 0.6666666	saved 0.6666666
money 0.6666666	low 0.5
price 0.5	high 0.5
health insurance 0.5	paying part of expenses 0.5

Tabelle A.2: Ergebnisse der DFS für den Anwendungsfall 1

Heuristischer Ansatz

Heuristischer Algorithmus geht auch von dem ersten Element im Baum (Tabelle A.3, Seite 101).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.3: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 1

A.2 Anwendungsfall 2

Gegeben: Teil- oder Gesamtsuchstring

Arbeitsweise

Hier werden PartString- oder ExactStringMatch angewendet, in Abhängigkeit davon, für welchen String-Matching Typ entschieden würde. Die Benutzereingaben sind auf den Abbildungen A.2 (Seite 101) und A.3 (Seite 102) und die dazugehörige Ergebnisse (die gleich für DepthFirst, BreadthFirst und heuristischer Ansatz sind) in den Tabellen A.4 (Seite 102) und A.5 (Seite 102) absehbar.

Title:

Remove

Add requirement

Abbildung A.2: Anwendungsfall 2a: Teil- oder Gesamtsuchstring

Anwendungsfall 2a

Anwendungsfall 2b

Title:

should be

Add requirement

Abbildung A.3: Anwendungsfall 2b: Teil- oder Gesamtsuchstring

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
price 0.75	low 0.75 medium 0.75 high 0.75

Tabelle A.4: Ergebnisse (gleich für alle 3 Algorithmen)

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
price 0.75	high 0.75

Tabelle A.5: Ergebnisse für den Anwendungsfall 2b (gleich für alle 3 Algorithmen)

A.3 Anwendungsfall 3

Gegeben: *Suche-String mit einem Requirement wo der beiden Felder schon ausgefüllt sind oder Suche-String mit 2 Requirements wo der letzte Requirement leer ist.*

Beide Fälle (3a und 3b, Abbildung A.4, Seite 103) geben die gleichen Ergebnisse aus, man unterscheidet nur die Ergebnisse die verschiedenen Suchalgorithmen liefern.

Breadth-First Suche

Für die Ergebnisse der Breitensuche in diesem Fall siehe Tabelle A.6 auf der Seite 103.



Abbildung A.4: Anwendungsfälle 3a und 3b

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical care 0.8	provided 0.8
hotel 0.8	suited for special needs 0.8

Tabelle A.6: Ergebnisse der BFS für den Anwendungsfall 3

Depth-First Suche

Die Tabelle A.7 auf der Seite 104 zeigt die Ergebnisse der Tiefensuche.

Heuristischer Ansatz

Man kann in der Tabelle A.8 auf der Seite 104 die Ergebnisse des heuristischen Ansatzes bei dem oben genannten Input ansehen.

A.4 Anwendungsfall 4

Gegeben: Suche-String mit zwei voll ausgefüllten Requirements, eine leere String kann auch dabei sein (s. Abbildung A.5, Seite 104).

(Anmerkung: ab hier werden diese zwei Situationen nicht unterschieden, da die die gleiche Ergebnisse ergeben.)

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical care 0.6	provided 0.6
hotel 0.6	suited for special needs 0.6
hotel personal 0.39999998	medically educated 0.39999998
rooms 0.39999998	connected with tourist's hospital 0.39999998
service 0.39999998	cleaned a lot 0.39999998
selection of food 0.39999998	specially polite 0.39999998
medical condition 0.19999999	diverse 0.39999998
body 0.19999999	suitable 0.39999998
vacation 0.19999999	improved 0.19999999
	relaxed 0.19999999
	adventurous 0.19999999

Tabelle A.7: Ergebnisse der DFS für den Anwendungsfall 3

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical care 0.75	provided 0.75
hotel 0.75	suited for special needs 0.75

Tabelle A.8: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 3

Title:

health	should be	improved	Remove
hotel	should be	suitable	Remove
	should be		Remove
Add requirement			

Abbildung A.5: Anwendungsfall 4

Breadth-First Suche

Die Ergebnisse der Breitensuche im vierten Fall sind in der Tabelle A.9 auf der Seite 105 anzusehen.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel personal 0.6	medically educated 0.6
hotel 0.6	connected with tourists hospital 0.6
rooms 0.6	cleaned a lot 0.6
service 0.6	specially polite 0.6
selection of food 0.6	diverse 0.6

Tabelle A.9: Ergebnisse der BFS für den Anwendungsfall 4

Depth-First Suche

Die Tabelle A.10 auf der Seite 105 zeigt, was für eine Ergebnisse die Tiefensuche in diesem Fall liefern wird.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical condition 0.25	improved 0.25
body 0.25	relaxed 0.25
vacation 0.25	adventurous 0.25

Tabelle A.10: Ergebnisse der DFS für den Anwendungsfall 4

Heuristischer Ansatz

Heuristischer Ansatz gibt die Ergebnisse in der Tabelle A.11 auf der Seite 105 zurück.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
vacation 0.5	comfortable 0.5
medical care 0.5	provided 0.5
hotel 0.5	suited for special needs 0.5

Tabelle A.11: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 4

A.5 Anwendungsfall 5

Gegeben: mehrere Anforderungen.

Die User-Input kann man auf der Seite 106 in der Tabelle A.6 anschauen.

Title:

health	should be	improved	Remove
hotel	must be	suited for special needs	Remove
medical care	should be	provided	Remove
vacation	should be	comfortable	Remove
	should be		Remove

Add requirement

Abbildung A.6: Anwendungsfall 5

Breadth-First Suche

Die Tabelle A.12 (Seite 106) zeigt die Ergebnisse der BFS wenn mehrere verschiedenen Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel 0.8	suited for special needs 0.8

Tabelle A.12: Ergebnisse der BFS für den Anwendungsfall 5

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.13 (Seite 107).

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.14 auf der Seite 107.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel 0.6	suitet for special needs 0.6
hotel personal 0.39999998	medically educated 0.39999998
rooms 0.39999998	connected with tourist's hospital 0.39999998
service 0.39999998	cleaned a lot 0.39999998
selection of food 0.39999998	specially polite 0.39999998
medical condition 0.19999999	diverse 0.39999998
body 0.19999999	suitable 0.39999998
vacation 0.19999999	improved 0.19999999
	relaxed 0.19999999
	adventurous 0.19999999

Tabelle A.13: Ergebnisse der DFS für den Anwendungsfall 5

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel 0.75	suitet for special needs 0.75

Tabelle A.14: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 5

A.6 Anwendungsfall 6

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.7 auf der Seite 107

Title:

price	should be	medium	Remove
benefits of summer	should be	used	Remove
selection of food	should be	diverse	Remove
trip	must be	comfortable	Remove
	should be		Remove
Add requirement			

Abbildung A.7: Anwendungsfall 6

Breadth-First Suche

Die Tabelle A.15 (Seite 108) zeigt die Ergebnisse der BFS wenn mehrere verschiedenen Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
vacation 0.6	comfortable 0.6
medical care 0.6	provided 0.6
hotel 0.6	suited for special needs 0.6

Tabelle A.15: Ergebnisse der BFS für den Anwendungsfall 6

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.16 (Seite 108).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.16: Ergebnisse der DFS für den Anwendungsfall 6

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.17 auf der Seite 108.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
booking 0.5	extra reliable 0.5

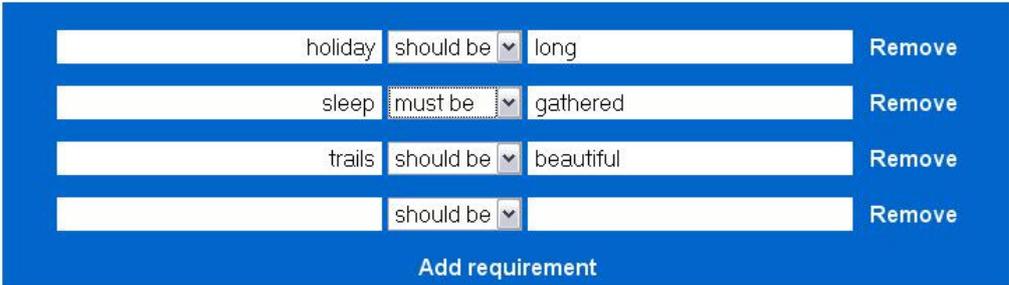
Tabelle A.17: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 6

A.7 Anwendungsfall 7

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.8 auf der Seite 109

Title:



holiday	should be	long	Remove
sleep	must be	gathered	Remove
trails	should be	beautiful	Remove
	should be		Remove

Add requirement

Abbildung A.8: Anwendungsfall 7

Breadth-First Suche

Die Tabelle A.18 (Seite 109) zeigt die Ergebnisse der BFS wenn mehrere verschieden Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
location 0.8	near hospital 0.8

Tabelle A.18: Ergebnisse der BFS für den Anwendungsfall 7

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.19 (Seite 110).

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.20 auf der Seite 110.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
location 0.3333333	near hospital 0.3333333

Tabelle A.19: Ergebnisse der DFS für den Anwendungsfall 7

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
location 0.75	near hospital 0.75

Tabelle A.20: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 7

A.8 Anwendungsfall 8

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.9 auf der Seite 110

Title:

<input type="text" value="weather"/>	<input type="text" value="should be"/>	<input type="text" value="enjoyable"/>	<input type="button" value="Remove"/>
<input type="text" value="assistance"/>	<input type="text" value="must be"/>	<input type="text" value="provided"/>	<input type="button" value="Remove"/>
<input type="text" value="service"/>	<input type="text" value="should be"/>	<input type="text" value="specially polite"/>	<input type="button" value="Remove"/>
<input type="text"/>	<input type="text" value="should be"/>	<input type="text"/>	<input type="button" value="Remove"/>
<input type="button" value="Add requirement"/>			

Abbildung A.9: Anwendungsfall 8

Breadth-First Suche

Die Tabelle A.21 (Seite 111) zeigt die Ergebnisse der BFS wenn mehrere verschieden Anforderungen gegeben sind.

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.22 (Seite 111).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
rooms 0.8	cleaned a lot 0.8
selection of food 0.8	diverse 0.8

Tabelle A.21: Ergebnisse der BFS für den Anwendungsfall 8

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.22: Ergebnisse der DFS für den Anwendungsfall 8

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.23 auf der Seite 111.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
rooms 0.75	cleaned a lot 0.75
selection of food 0.75	diverse 0.75

Tabelle A.23: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 8

A.9 Anwendungsfall 9

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.10 auf der Seite 112

Breadth-First Suche

Die Tabelle A.24 (Seite 112) zeigt die Ergebnisse der BFS wenn mehrere verschieden Anforderungen gegeben sind.

Title:

<input type="text" value="journey"/>	should be	<input type="text" value="managed on my own"/>	Remove
<input type="text" value="medical condition"/>	should be	<input type="text" value="improved"/>	Remove
<input type="text" value="price"/>	should be	<input type="text" value="low"/>	Remove
<input type="text"/>	should be	<input type="text"/>	Remove

Add requirement

Abbildung A.10: Anwendungsfall 9

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
price 0.8	medium 0.8
health insurance 0.8	high 0.8
	paying part of expenses 0.8

Tabelle A.24: Ergebnisse der BFS für den Anwendungsfall 9

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle [A.25](#) (Seite [112](#)).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.25: Ergebnisse der DFS für den Anwendungsfall 9

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle [A.26](#) auf der Seite [113](#).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
price 0.75	medium 0.75
health insurance 0.75	high 0.75
	paying part of expenses 0.75

Tabelle A.26: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 9

A.10 Anwendungsfall 10

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.11 auf der Seite 113

Title:

<input type="text" value="group"/>	should be	<input type="text" value="having competent tour guide"/>	Remove
<input type="text" value="hotel personal"/>	should be	<input type="text" value="medically educated"/>	Remove
<input type="text" value="booking"/>	must be	<input type="text" value="extra reliable"/>	Remove
<input type="text"/>	should be	<input type="text"/>	Remove

Add requirement

Abbildung A.11: Anwendungsfall 10

Breadth-First Suche

Die Tabelle A.27 (Seite 114) zeigt die Ergebnisse der BFS wenn mehrere verschieden Anforderungen gegeben sind.

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.28 (Seite 114).

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.29 auf der Seite 114.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
sleep 0.6	gathered 0.6
stress 0.6	avoided 0.6
enough rest 0.6	enjoyable 0.6
weather 0.6	near hospital 0.6
location 0.6	beautiful 0.6
trails 0.6	saved 0.6
money 0.6	

Tabelle A.27: Ergebnisse der BFS für den Anwendungsfall 10

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
assistance 0.3333333	provided 0.3333333
tourist 0.3333333	independent 0.3333333
journey 0.16666663	managed on my own 0.16666663
vacation 0.5	comfortable 0.5
medical care 0.5	suited for special needs 0.5
hotel 0.5	connected with tourist's hospital 0.3333333
rooms 0.3333333	cleaned a lot 0.3333333
service 0.3333333	specially polite 0.3333333
selection of food 0.3333333	diverse 0.3333333
medical condition 0.16666663	suitable 0.3333333
body 0.16666663	improved 0.16666663
trip 0.5	relaxed 0.16666663
	adventurous 0.16666663

Tabelle A.28: Ergebnisse der DFS für den Anwendungsfall 10

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 0.5	strictly avoided 0.5
health 0.5	improved 0.5

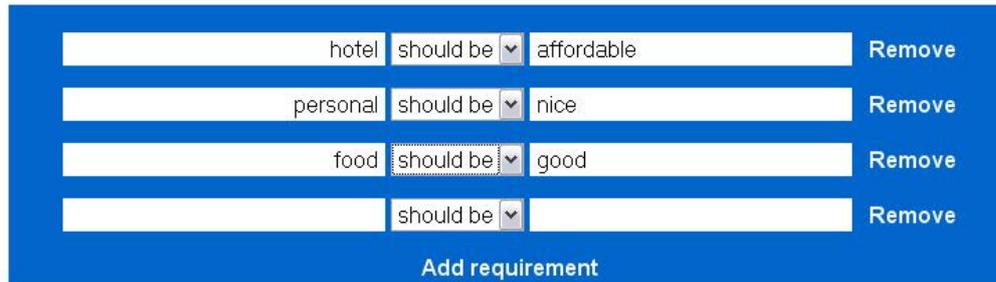
Tabelle A.29: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 10

A.11 Anwendungsfall 11

Gegeben: mehrere Anforderungen, wo keine als Ziel im i^ -Modell gefunden werden kann.*

Für die User-Input siehe Abbildung A.12 auf der Seite 115

Title:



hotel	should be	affordable	Remove
personal	should be	nice	Remove
food	should be	good	Remove
	should be		Remove

Add requirement

Abbildung A.12: Anwendungsfall 11

Arbeitsweise

Wenn der Benutzer die Anforderungen gibt, von welchen keine ein Ziel in dem i^* -Modell ist, geben alle drei Algorithmen die gleiche Ergebnisse mit gleichen Gewichtungen, und dabei handelt es sich um zwei wichtigste Ziele im Modell (die sich am höchsten Ebene im Modell befinden). Dieses kann man in der Tabelle A.30 auf Seite 115 genauer betrachten.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.30: Ergebnisse für alle 3 Algorithmensarten für den Anwendungsfall 11

A.12 Anwendungsfall 12

Gegeben: mehrere Anforderungen, wo nur die letzte als Ziel im i^ -Modell gefunden werden kann.*

Für die User-Input siehe Abbildung A.13 auf der Seite 116

Title:

<input type="text" value="hotel"/>	should be	<input type="text" value="good"/>	Remove
<input type="text" value="trip"/>	should be	<input type="text" value="nice"/>	Remove
<input type="text" value="medical care"/>	should be	<input type="text" value="provided"/>	Remove
<input type="text"/>	should be	<input type="text"/>	Remove

Add requirement

Abbildung A.13: Anwendungsfall 12

Arbeitsweise

Wenn die letzte Anforderung ein Ziel im i^* -Modell ist, liefern die Algorithmen die Ergebnisse entsprechend ihre Arbeitsweise. Falls eine von anderen als letzte Anforderung ein Ziel aus Modell ist, werden wieder die zwei wichtigste Anforderungen gegeben.

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.31 (Seite 116).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel personal 0.5	medically educated 0.5
hotel 0.5	connected with tourists hospital 0.5

Tabelle A.31: Ergebnisse der DFS für den Anwendungsfall 12

Breadth-First Suche und Heuristischer Ansatz

Wenn angewendet, geben der heuristische Ansatz und die Breadth-First die gleiche Ergebnisse mit gleicher Gewichtungen (in der Tabelle A.32 auf der Seite 117 ansehbar).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel personal 1.0	medically educated 1.0
hotel 1.0	connected with tourists hospital 1.0

Tabelle A.32: Ergebnisse des heuristischen Ansatzes und BFS für den Anwendungsfall 12

A.13 Anwendungsfall 13

Gegeben: mehrere Anforderungen, wo eine als Ziel im i^ -Modell gefunden werden kann, aber diese nicht als letzte angegeben wurde.*

Für die User-Input siehe Abbildung A.14 auf der Seite 117

Title:

<input style="width: 80%; border: none;" type="text" value="medical care"/>	should be <input style="width: 20px;" type="text" value=""/>	<input style="width: 80%; border: none;" type="text" value="provided"/>	Remove
<input style="width: 80%; border: none;" type="text" value="hotel"/>	should be <input style="width: 20px;" type="text" value=""/>	<input style="width: 80%; border: none;" type="text" value="good"/>	Remove
<input style="width: 80%; border: none;" type="text" value="trip"/>	should be <input style="width: 20px;" type="text" value=""/>	<input style="width: 80%; border: none;" type="text" value="nice"/>	Remove
<input style="width: 80%; border: none;" type="text"/>	should be <input style="width: 20px;" type="text" value=""/>	<input style="width: 80%; border: none;" type="text"/>	Remove

[Add requirement](#)

Abbildung A.14: Anwendungsfall 13

Arbeitsweise

Wenn der Benutzer die Anforderungen gibt, von welchen eine ein Ziel in dem i^* -Modell ist, aber diese nicht die letzte Anforderung ist, geben alle drei Algorithmen die gleiche Ergebnisse mit gleichen Gewichtungen, und dabei handelt es sich wie bei dem Anwendungsfall 11 um zwei wichtigste Ziele im Modell. Die Ergebnisse kann man in der Tabelle A.33 auf Seite 118 ansehen.

A.14 Anwendungsfall 14

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.15 auf der Seite 118

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.33: Ergebnisse für alle 3 Algorithmensarten für den Anwendungsfall 13

Title:

<input type="text" value="location"/>	should be	▼	<input type="text" value="near hospital"/>	Remove
<input type="text" value="rooms"/>	must be	▼	<input type="text" value="cleaned a lot"/>	Remove
<input type="text" value="hotel"/>	should be	▼	<input type="text" value="suitable"/>	Remove
<input type="text"/>	should be	▼	<input type="text"/>	Remove

Add requirement

Abbildung A.15: Anwendungsfall 14

Breadth-First Suche

Die Tabelle A.34 (Seite 118) zeigt die Ergebnisse der BFS wenn mehrere verschiedenen Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
hotel personal 0.6	medically educated 0.6
hotel 0.6	connected with tourists hospital 0.6
service 0.6	especially polite 0.6
selection of food 0.6	diverse 0.6

Tabelle A.34: Ergebnisse der BFS für den Anwendungsfall 14

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.35 (Seite 119).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical condition 0.25	improved 0.25
body 0.25	relaxed 0.25
vacation 0.25	adventurous 0.25

Tabelle A.35: Ergebnisse der DFS für den Anwendungsfall 14

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.36 auf der Seite 119.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
vacation 0.5	comfortable 0.5
medical care 0.5	provided 0.5
hotel 0.5	suited for special needs 0.5

Tabelle A.36: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 14

A.15 Anwendungsfall 15

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.16 auf der Seite 119

Title:

weather	should be	enjoyable	Remove
enough rest	should be	gathered	Remove
discrimination	should be	strictly avoided	Remove
	should be		Remove
Add requirement			

Abbildung A.16: Anwendungsfall 15

Breadth-First Suche

Die Tabelle A.37 (Seite 120) zeigt die Ergebnisse der BFS wenn mehrere verschiedenen Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
health 0.8	improved 0.8

Tabelle A.37: Ergebnisse der BFS für den Anwendungsfall 15

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.38 (Seite 121).

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.39 auf der Seite 121.

A.16 Anwendungsfall 16

Gegeben: mehrere Anforderungen, die von der Bedeutung der Zielen im i^* -Modell ähnlich sind aber nicht gleich geschrieben sind.

Für die User-Input siehe Abbildung A.17 auf der Seite 120

Title:

a lot of money	should be	▼	spent	Remove
journey	should be	▼	comfortable	Remove
hotel	should be	▼	cleaned a lot	Remove
	should be	▼		Remove

Add requirement

Abbildung A.17: Anwendungsfall 16

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
health 0.71428573	improved 0.71428573
sleep 0.57142854	gathered 0.57142854
stress 0.57142854	avoided 0.57142854
booking 0.57142854	extra reliable 0.57142854
assistance 0.28571427	provided 0.28571427
group 0.14285707	having competent tour guide 0.14285707
tourist 0.28571427	independent 0.28571427
journey 0.14285707	managed on my own 0.14285707
vacation 0.4285714	comfortable 0.4285714
medical care 0.4285714	suited for special needs 0.4285714
hotel 0.4285714	medically educated 0.28571427
hotel personal 0.28571427	connected with tourist's hospital 0.28571427
rooms 0.28571427	cleaned a lot 0.28571427
service 0.28571427	specially polite 0.28571427
selection of food 0.28571427	diverse 0.28571427
medical condition 0.14285707	suitable 0.28571427
body 0.14285707	relaxed 0.14285707
trip 0.4285714	adventurous 0.14285707
holiday 0.4285714	short 0.4285714
benefits of summer 0.4285714	medium 0.4285714
benefits of autumn 0.4285714	long 0.4285714
benefits of winter 0.4285714	used 0.4285714
location 0.57142854	near hospital 0.57142854
trails 0.57142854	beautiful 0.57142854
money 0.57142854	saved 0.57142854
price 0.4285714	low 0.4285714
health insurance 0.4285714	high 0.4285714
	paying part of expenses 0.4285714

Tabelle A.38: Ergebnisse der DFS für den Anwendungsfall 15

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
health 0.75	improved 0.75

Tabelle A.39: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 15

Arbeitsweise

Wenn der Benutzer die Anforderungen angibt, die Ziele in dem i^* -Modell ist, aber die er auch nicht in gleicher Weise ausgedrückt hat, wie sie im Modell geschrieben sind, geben alle drei Algorithmen die gleiche Ergebnisse mit gleichen Gewichtungen, und dabei handelt es sich um zwei wichtigste Ziele im Modell (die sich am höchsten Ebene im Modell befinden). Dieses kann man in der Tabelle A.40 auf Seite 122 genauer betrachten.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 1.0	strictly avoided 1.0
health 1.0	improved 1.0

Tabelle A.40: Ergebnisse für alle 3 Algorithmentypen für den Anwendungsfall 16

A.17 Anwendungsfall 17

Gegeben: mehrere Anforderungen.

Für die User-Input siehe Abbildung A.18 auf der Seite 122

Title:

discrimination	must be	strictly avoided	Remove
booking	should be	extra reliable	Remove
assistance	must be	provided	Remove
group	should be	having a competent tour guide	Remove
	should be		Remove

Add requirement

Abbildung A.18: Anwendungsfall 17

Breadth-First Suche

Die Tabelle A.41 (Seite 123) zeigt die Ergebnisse der BFS wenn mehrere verschiedenen Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
journey 0.4	managed on my own 0.4

Tabelle A.41: Ergebnisse der BFS für den Anwendungsfall 17

Depth-First Suche und heuristischer Ansatz

Für diesen Fall liefern die Tiefensuche und der heuristischer Ansatz die gleiche Ergebnisse die man in der Tabelle A.42 ansehen kann. (Seite 123).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
health 1.0	improved 1.0

Tabelle A.42: Ergebnisse der DFS und des heuristischen Ansatzes für den Anwendungsfall 17

A.18 Anwendungsfall 18

Gegeben: mehrere Anforderungen (s. Abbildung A.19, Seite 123).

Title:

<input type="text" value="location"/>	should be	▼	<input type="text" value="near hospital"/>	Remove
<input type="text" value="trails"/>	should be	▼	<input type="text" value="beautiful"/>	Remove
<input type="text" value="health"/>	should be	▼	<input type="text" value="improved"/>	Remove
<input type="text" value="sleep"/>	should be	▼	<input type="text" value="gathered"/>	Remove
<input type="text" value="vacation"/>	should be	▼	<input type="text" value="comfortable"/>	Remove
<input type="text" value=""/>	should be	▼	<input type="text" value=""/>	Remove

[Add requirement](#)

Abbildung A.19: Anwendungsfall 18

Breadth-First Suche

Die Ergebnisse der Breitensuche im vierten Fall sind auf Abbildung A.43 auf der Seite 124 anzusehen.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical care 0.8	provided 0.8
hotel 0.8	suited for special needs 0.8

Tabelle A.43: Ergebnisse der BFS für den Anwendungsfall 18

Depth-First Suche

Die Tabelle A.44 auf der Seite 124 zeigt, was für eine Ergebnisse die Tiefensuche in diesem Fall liefern wird.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical care 0.6	provided 0.6
hotel 0.6	suited for special needs 0.6
hotel personal 0.39999998	medically educated 0.39999998
rooms 0.39999998	connected with tourist's hospital 0.39999998
service 0.39999998	cleaned a lot 0.39999998
selection of food 0.39999998	specially polite 0.39999998
medical condition 0.19999999	diverse 0.39999998
body 0.19999999	suitable 0.39999998
vacation 0.19999999	improved 0.19999999
	relaxed 0.19999999
	adventurous 0.19999999

Tabelle A.44: Ergebnisse der DFS für den Anwendungsfall 18

Heuristischer Ansatz

Heuristischer Ansatz gibt die Ergebnisse in der Tabelle A.45 auf der Seite 125 zurück.

A.19 Anwendungsfall 19

Gegeben: mehrere Anforderungen.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
medical care 0.75	provided 0.75
hotel 0.75	suited for special needs 0.75

Tabelle A.45: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 18

Für die User-Input siehe Abbildung A.20 auf der Seite 125

Title:

service	should be	specially polite	Remove
discrimination	must be	strictly avoided	Remove
vacation	should be	short	Remove
health	must be	improved	Remove
money	must be	saved	Remove
hotel	should be	suited for special needs	Remove
	should be		Remove

Add requirement

Abbildung A.20: Anwendungsfall 19

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.46 (Seite 125).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
rooms 0.5	cleaned a lot 0.5
selection of food 0.5	diverse 0.5

Tabelle A.46: Ergebnisse der DFS für den Anwendungsfall 19

Breadth-First Suche und Heuristischer Ansatz

Wenn angewendet, geben der heuristische Ansatz und die Breadth-First die gleiche Ergebnisse mit gleicher Gewichtungen (in der Tabelle A.47 auf der Seite 126 ansehbar).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
rooms 1.0	cleaned a lot 1.0
selection of food 1.0	diverse 1.0

Tabelle A.47: Ergebnisse des heuristischen Ansatzes und BFS für den Anwendungsfall 19

A.20 Anwendungsfall 20

Gegeben: mehrere Anforderungen.

Die User-Input kann man auf der Seite 126 in der Abbildung A.21 anschauen.

Title:

rooms	should be	cleaned a lot	Remove
stress	should be	avoided	Remove
medical condition	must be	improved	Remove
body	should be	relaxed	Remove
price	should be	low	Remove
rooms	should be	cleaned a lot	Remove
booking	must be	reliable	Remove
	should be		Remove

Add requirement

Abbildung A.21: Anwendungsfall 20

Breadth-First Suche

Die Tabelle A.48 (Seite 127) zeigt die Ergebnisse der BFS wenn mehrere verschieden Anforderungen gegeben sind.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
sleep 0.6	gathered 0.6
enough rest 0.6	enjoyable 0.6
weather 0.6	near hospital 0.6
location 0.6	beautiful 0.6
trails 0.6	saved 0.6
money 0.6	

Tabelle A.48: Ergebnisse der BFS für den Anwendungsfall 20

Depth-First Suche

Für diesen Fall liefert die Tiefensuche die Ergebnisse in der Tabelle A.49 (Seite 127).

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
booking 0.6666666	extra reliable 0.6666666
assistance 0.3333333	provided 0.3333333
group 0.16666663	having competent tour guide
tourist 0.3333333	0.16666663
journey 0.16666663	independent 0.3333333
vacation 0.5	managed on my own 0.16666663
medical care 0.5	comfortable 0.5
hotel 0.5	suited for special needs 0.5
hotel personal 0.3333333	medically educated 0.3333333
service 0.3333333	connected with tourist's hospital
selection of food 0.3333333	0.3333333
trip 0.5	specially polite 0.3333333
	diverse 0.3333333
	suitable 0.3333333
	adventurous 0.16666663

Tabelle A.49: Ergebnisse der DFS für den Anwendungsfall 20

Heuristischer Ansatz

Wenn angewendet, gibt der heuristische Ansatz die Ergebnisse in der Tabelle A.50 auf der Seite 128.

Ergebnisse	
Feld Typ Criterium (mit Gewichtungen)	Typ Demand (mit Gewichtungen):
discrimination 0.5	strictly avoided 0.5
health 0.5	improved 0.5

Tabelle A.50: Ergebnisse des heuristischen Ansatzes für den Anwendungsfall 20

***i**-Modell für Xohana**

Die Modellierungsdateien für *Xohana*, die im Ordner «Modelling» gespeichert sind, bestehen aus einer *q7*-Datei, einer *tel*-Datei, einer XML-Datei und einem *png*-Bild. Den konkreten *q7*-Code kann man durch das Öffnen der *q7*-Datei mit einem einfachen Text-Editor ansehen, in anderen Fällen zeigt die Eclipse automatisch die visualisierte Version des Modells an, die von der *tel*-Datei geladen wird. Im *png*-Bild ist diese Version in einem übertragbaren Format gespeichert, in der XML-Datei kann man die im XML exportierte Version des Modells ansehen. Allerdings ist diese im XML nicht korrekt formatiert, deshalb wurde nach den manuellen Korrekturen die im XML akzeptierte Version des *i**-Modells im *Modell.xml* gespeichert.

Zusammensetzung aller Klassen

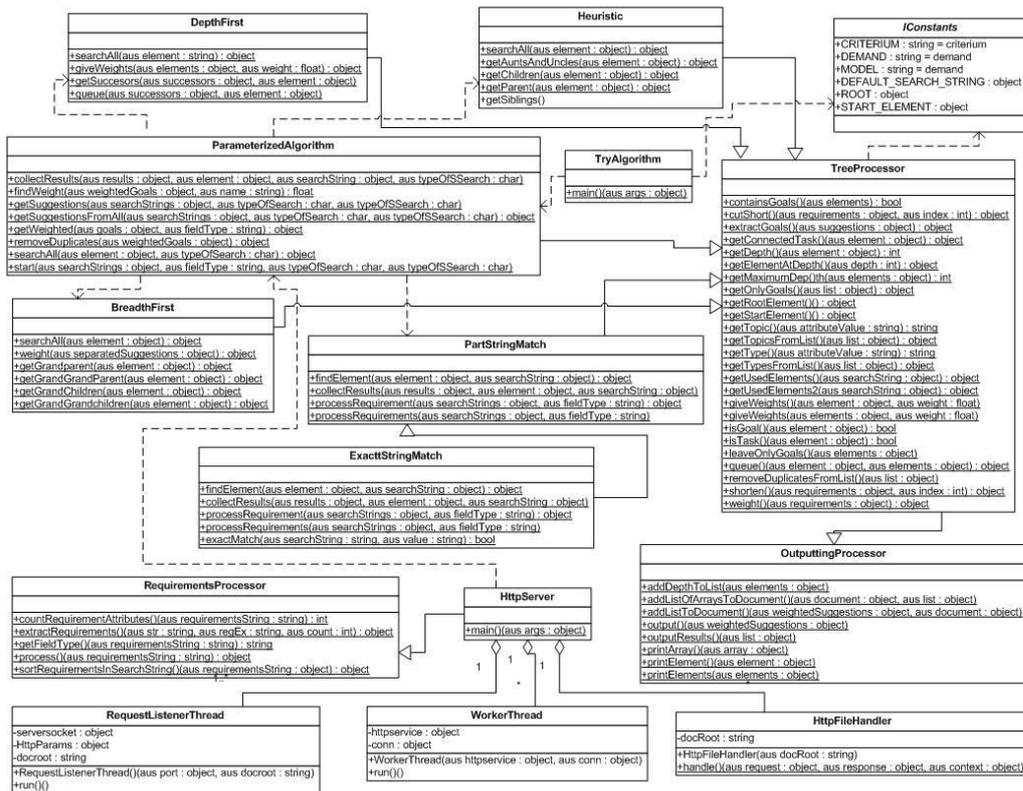


Abbildung C.1: Zusammensetzung aller Klassen

Literaturverzeichnis

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.
- [2] Robin Burke. Knowledge-based recommender systems. In *Encyclopedia of Library and Information Systems*, volume 69, pages 180–200. Marcel Dekker, 2000.
- [3] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
- [4] Robin Burke. Lecture on recommender systems. Graz University of Technology, May 2009.
- [5] Li Chen and Pearl Pu. Evaluating critiquing-based recommender agents. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 157–162. American Association for Artificial Intelligence, Menlo Park, CA 94025-3496, United States, 2006. Human Computer Interaction Group, School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland.
- [6] Li Chen and Pearl Pu. The evaluation of a hybrid critiquing system with preference-based recommendations organization. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 169–172, New York, NY, USA, 2007. ACM.
- [7] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-Functional Requirements in Software Engineering (The Kluwer International Series in Software Engineering Volume 5) (International Series in Software Engineering)*. Springer, October 1999.

- [8] William W. Cohen and Wei Fan. Web-collaborative filtering: recommending music by crawling the web. In *Proceedings of 9th International WWW Conference*, Amsterdam, Netherlands, 2000.
- [9] Felix Hernandez del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790 – 804, 2008.
- [10] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Inf. Systems*, 22(1):143–177, 2004.
- [11] Alexander Felfernig and Robin Burke. Constraint-based recommender systems: technologies and research issues. In *ICEC '08: Proceedings of the 10th international conference on Electronic commerce*, pages 1–10, New York, NY, USA, 2008. ACM.
- [12] Alexander Felfernig and Bartosz Gula. An empirical study on consumer behavior in the interaction with knowledge-based recommender applications. *IEEE Pattern Recognition and AI*, 2007.
- [13] Alexander Felfernig and Alfred Kiener. Knowledge-based interactive selling of financial services using fsadvisor. In *17th Innovative Applications of Artificial Intelligence Conference (IAAI 05)*, pages 1475 – 1482, Pittsburgh, Pennsylvania, July 2005. AAAI Press / The MIT Press.
- [14] The Eclipse Foundation. Eclipse integrated development environment, 2008. <http://www.eclipse.org>.
- [15] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *ACM Communications*, 35(12):61–70, 1992.
- [16] Peng Han, Bo Xie, Fan Yang, and Ruimin Sheng. A scalable p2p recommender system based on distributed collaborative filtering. *Expert Systems with Applications*, 27:203–210, 2004.
- [17] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems.*, 22(1):5–53, 2004.
- [18] Lun Ping Hung. A personalized recommendation system based on product taxonomy for one-to-one marketing online. *Expert Systems with Applications*, 29:383–392, 2005.

- [19] Yong Soo Kim, Bong Jin Yum, Junehwa Song, and Su Myeon Kim. Development of recommender system based on navigational and behavioral pattern of customers in e-commerce sites. *Expert Systems with Applications*, 28:381–393, 2005.
- [20] Joseph A. Konstan. Introduction to recommender systems: algorithms and evaluation. *ACM Transactions on Inf. Systems*, 22(1):1–4, 2004.
- [21] Vinod Krishnan, Pradeep Kumar Narayanashetty, Mukesh Nathan, Richard T. Davies, and Joseph A. Konstan. Who predicts better?: results from an online study comparing humans and an online recommender system. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 211–218, New York, NY, USA, 2008. ACM.
- [22] Nikos Manouselis and Constantina Costopoulou. Analysis and classification of multi-criteria recommender systems. *World Wide Web*, 10(3):415–441, 3 2007.
- [23] James March and Herbert A. Simon. *Organizations*. Wiley, New York, 1958.
- [24] Sunghwan Min and Ingoo Han. Detection of the customer time-invariant pattern for improving recommender systems. *Expert Systems with Applications*, 28:189–199, 2005.
- [25] Tim Miranda, Mark Claypool, Mark Claypool, Anuja Gokhale, Anuja Gokhale, Tim Mir, Pavel Murnikov, Pavel Murnikov, Dmitry Netes, Dmitry Netes, Matthew Sartin, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.
- [26] David M. Pennock and Eric Horvitz. Analysis of axiomatic fundatiuons of collaborative filtering. In *Proceedings of AAAI Workshop on Artificial Intelligence in E-Commerce*, Orlando, Florida, July 1999. AAAI Press.
- [27] Pearl Pu, Li Chen, and Pratyush Kumar. Evaluating product search and recommender systems for e-commerce environments. *Electronic Commerce Research*, 8(1-2):1–27, 2008.
- [28] Pearl Pu, Boi Faltings, and Paolo Viappiani. Evaluating preference-based search tools: a tale of two approaches. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pages 205–211, Boston, MA, USA, July 2006. AAAI Press.

- [29] Sven Radde, Andreas Kaiser, and Burkhard Freitag. A model-based customer inference engine. In *Proceedings ECAI 2008 - Workshop on Recommender Systems*, pages 2–7, Patras, Greece, 2008.
- [30] Maryam Ramezani, Lawrence Bermann, Rich Thompson, Robin Burke, and Bamshad Mobasher. Selecting and applying recommendation technology. In *International Workshop on Recommendation and Collaboration, in Conjunction with 2008 International ACM Conference on Intelligent User Interfaces (IUI 2008)*, Canaria, Canary Islands, Spain, 1 2008.
- [31] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Incremental critiquing. In *Research and Development in Intelligent Systems XXI. Proceedings of AI-2004*, pages 101–114, Cambridge, UK, 2004. Springer.
- [32] James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu, and Barry Smyth. Evaluating compound critiquing recommenders: a real-user study. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 114–123, New York, NY, USA, 2007. ACM.
- [33] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering. In *Proceedings of ACM CSCW*, pages 175–186. ACM, New York, 1994.
- [34] Paul Resnick and Hal R. Varian. Recommender systems. *ACM Communications*, 40(3):56–58, 1997.
- [35] Francesco Ricci, Bora Arslan, Nader Mirzadeh, and Adriano Venturini. Itr: A case-based travel advisory system. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 613–627, London, UK, 2002. Springer-Verlag.
- [36] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002.
- [37] John B. Schafer, Joseph A. Konstan, and John Riedl. E-commerce applications. *Data Mining Knowledge Discovery*, 5:115–153, 2001.
- [38] Andrew I. Schein, Alexandrin Popescul, Rin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start collaborative filtering. *Proceedings of the 25th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, August 2002.
- [39] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Croc: a new evaluation criterion for recommender systems. *Electronic Commerce Research*, 5:51–74, 2005.

- [40] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automatic "word of mouth". In *Proceedings of Conference on Human Factors in Computational Systems*, 1995.
- [41] Edward Shen, Henry Lieberman, and Francis Lam. What am i gonna wear?: scenario-oriented recommendation. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 365–368, New York, NY, USA, 2007. ACM.
- [42] Richard H. Thayer and Merlin Dorfman. *System and Software Requirements*. IEEE Computer Society Press, 1990.
- [43] University of Toronto The Knowledge Management Lab. Openome, an open-source requirements engineering tool, 2006. <http://www.cs.toronto.edu/km/openome>.
- [44] Brendon Towle and Clark Quinn. Knowledge based recommender systems using explicit user models. In *Papers from the AAAI Workshop, AAAI Technical Report WS-00-04*, pages 74–77. Menlo Park, CA: AAAI Press, 2000.
- [45] Paolo Viappiani, Pearl Pu, and Boi Faltings. Conversational recommenders with adaptive suggestions. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 89–96, New York, NY, USA, 2007. ACM.
- [46] Eric Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, Toronto, Ont., Canada, Canada, 1996.
- [47] Eric Yu, Jennifer Horkoff, Samer Abdulhadi, and Gemma Grau. i* guide. RTWH Aachen Universität, August 2007. <http://istar.rwth-aachen.de/>.
- [48] Eric Yu, Lin Liu, and Ling Li. Modelling strategic actor relationships to support intellectual property management. 2001.
- [49] Eric Yu and John Mylopoulos. Using Goals, Rules, and Methods to Support Reasoning. In *in Business Process Reengineering, International Journal of Intelligent Systems in Accounting, Finance and Management*, pages 1–13, 1996.
- [50] Eric Yu, Reza Samavi, and Thodoros Topaloglou. Strategic reasoning about business models: a conceptual modeling approach. *Information Systems and E-Business Management*, 7(2):171–198, March 2009.

- [51] Zhiyong Yu, Zhiwen Yu, Xingshe Zhou, and Yuichi Nakamura. Handling conditional preferences in recommender systems. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 407–412, New York, NY, USA, 2008. ACM.
- [52] Markus Zanker, Markus Jessenitschnig, Dietmar Jannach, and Sergiu Gordea. Comparing recommendation strategies in a commercial context. *IEEE Intelligent Systems*, 22(3):69–73, 2007.

Index

- i** Framework, 22
- Überblick über die Empfehlungsdienste, 38
- A* Suche, 37
- Akteurabgrenzung, 28
- Akteuren, 24
- Akteuren-Verbindungen-Links, 25
- Algorithms-Klassendiagramm, 73
- Anwendungsfälle, 83, 85, 86, 88, 89, 91–93, 95, 96, 98–103, 105, 106, 108, 109
- Architekturskizze, 71
- Beispiel eines *i** SR-Modells, 32
- Beitragslinks, 30
- BFS-Gewichtungsfunktion, 70
- Breitensuche, 35
- Claims, 23
- Definitionen des Empfehlungsdienstes, 38
- Dekomposition-Links, 29
- DFS-Gewichtungsfunktion, 70
- Die Klassendiagramme, 73
- Die Klassenbeschreibung, 72
- Die Klassenunterteilung, 72
- Empfehlungsdienste, 38
- Evaluationsarten, 56
- Gemischte Online-Evaluation, 63
- Gewichtungsfunktion für die heuristische Ansatz, 71
- Gewichtungsfunktionen, 69
- Gierige Beste-Erste Suche, 37
- Greedy-Algorithmus, 37
- Hintergrund, 19
- i** Framework, 24
- i**-Modell, 67
- i**-Modell für Xohana, 113
- Informierte Suchstrategien, 36
- Intentionelle Elemente, 28
- Klassendiagramm für Test-Package, 75
- Klassifikationsdimensionen, 43
- Komparative Online-Evaluation, 62
- Means-Ends-Links, 29
- Mittel-Ende-Links, 29
- Modell der strategischen Abhängigkeiten, 24
- Modell der strategischen Begründungen, 27
- Modellierung, 22
- NFR Framework, 22
- NFR-Softgoals, 23
- Nicht-funktionale Anforderungen, 22
- Offline-Evaluation, 56
- Online-Evaluation, 58

- Operationalisierte Softgoals, 23
- Processors-Klassendiagramm, 74
- Reine Online-Evaluation, 61
- SD, 24
- Server-Klassendiagramm, 75
- Softgoals, 23
- SR, 27
- Strategic Dependency Model, 24
- Strategic Rationale Model, 27
- Strategische Abhängigkeiten, 26
- Suchalgorithmen, 33
- Tiefensuche, 35
- UML Package-Diagramm, 71
- Uninformierte Suchstrategien, 34
- Unterteilung der Empfehlungsdiensten, 38
- Unterteilung des Projekts, 65
- Verwundbarkeit des i* SD-Modells, 27
- Wissensbasierte Empfehlungsdienste, 46
- Xohana, 19
- Xohana-Algorithmen, 68
- Zusammensetzung aller Klassen, 77, 115