GMD Research Series

GMD –
Forschungszentrum
Informationstechnik
GmbH

Patrick Schumacher

HCI-Aspekte von Softwareagenten

N° 3/1999

© GMD 1999

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

In der Reihe GMD Research Series werden Forschungs- und Entwicklungsergebnisse aus der GMD zum wissenschaftlichen, nicht-kommerziellen Gebrauch veröffentlicht. Jegliche Inhaltsänderung des Dokuments sowie die entgeltliche Weitergabe sind verboten.

The purpose of the GMD Research Series is the dissemination of research work for scientific non-commercial use.
The commercial distribution of this document is prohibited, as is any modification of its content.

Anschrift des Verfassers/Address of the author:
Patrick Schumacher
Institut für Angewandte Informationstechnik
GMD – Forschungszentrum Informationstechnik GmbH
D-53754 Sankt Augustin
E-mail: patrick.schumacher@gmd.de

Die Deutsche Bibliothek - CIP-Einheitsaufnahme
Schumacher, Patrick:
HCI-Aspekte von Softwareagenten / Patrick Schumacher. GMD – Forschungszentrum Informationstechnik GmbH. - Sankt Augustin : GMD – Forschungszentrum Informationstechnik, 1999
(GMD Research Series ; 1999, No. 3)
Zugl.: Bonn, Univ., Diplomarbeit, 1998
ISBN 3-88457-352-7

ISSN 1435-2699
ISBN 3-88457-352-7

Kurzfassung

Agentensysteme haben den Anspruch, Vorteile bezüglich Funktionalität und Bedienbarkeit zu bieten. Doch es zeigt sich, dass vorhandene Systeme diese Versprechungen oftmals nicht erfüllen, sondern den Anwender vor zusätzliche neue Probleme stellen. Auf der Grundlage, dass sich die Anwendung von Gestaltungsrichtlinien der Human-Computer Interaction (HCI) bei der Entwicklung traditioneller Softwaresysteme bewährt hat, greift diese Arbeit die Idee auf, HCI-Gestaltungsrichtlinien zu modifizieren und auf Agentensysteme anzuwenden. Dabei stellt sich die Frage, wie eine bestmögliche Modifikation von HCI-Gestaltungsrichtlinien aussehen könnte. Womit kann ihre Wirkung maximal verstärkt werden?

Die hier entwickelte Lösung zeigt, dass erst durch die delegationsorientierte Anwendung von HCI-Gestaltungsrichtlinien Softwareagenten entscheidend verbessert werden. Zu diesem Zweck werden zunächst geeignete HCI-Gestaltungsrichtlinien identifiziert, im Agentenkontext transformiert und auf das ebenfalls im Agentenkontext ausgearbeitete Konzept Delegation abgebildet. Das Ergebnis ist ein Leitfaden zur agentenbasierten Softwareentwicklung, mit dessen Hilfe das professionell genutze Information-Brokering-System bizzyB speziell für die Bedürfnisse und Fähigkeiten nicht-professioneller Benutzer weiterentwickelt wird.

Schlagwörter: Human-Computer Interaction, Software-Ergonomie, Agenten, Delegation, Information Brokering, Leitfaden, agentenbasierte Softwareentwicklung

Abstract

Agent systems claim to have advantages over traditional software concerning their functionality and usability. Unfortunately, it soon becomes clear that not only current systems cannot keep these promises, they even create additional new problems for the user. Based on the successful application of human-computer interaction (HCI) design guidelines in the context of traditional software, this paper picks up the idea of modifying HCI guidelines and applying them to agent systems. This raises the question of how to best modify HCI design guidelines in order to maximize their impact.

The solution developed here shows that software agents can be significantly improved only through a delegation-oriented application of HCI design guidelines. Therefore, we first identify suitable HCI design guidelines, transform them in an agent context, and later map them onto the concept of delegation which has also been elaborated in an agent context. We present a guide for agent-based software engineering which is subsequently applied to the improvement of the professional information brokering system bizzyB for the special needs and capabilities of non-professional users.

Keywords: Human-Computer Interaction, software ergonomics, agents, delegation, information brokering, guideline, agent-based software engineering

Vorwort

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als studentische Hilfskraft im Projekt COBRA an der GMD - Forschungszentrum Informationstechnik GmbH in Sankt Augustin.

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Armin B. Cremers für die Bereitschaft bedanken, diese Arbeit zu betreuen. Ich danke auch Helge Kahler für seine vielfältige Unterstützung bei der Erstellung dieser Arbeit.

Dr. Christoph Thomas verdanke ich die Möglichkeit, mich an der GMD mit den Themen HCI und Agenten auseinanderzusetzen. Er hat mich fachlich gut betreut, mir vor allem aber dieses interessante und zukunftsweisende Thema angeboten. Es hat mir großen Spaß gemacht, mit ihm und der gesamten COBRA-Projektgruppe zusammenzuarbeiten.

Weiterhin möchte ich mich bei meinen Kollegen aus der Forschungsgruppe Mensch-Maschine-Kommunikation bedanken, die mir in zahlreichen Gesprächen wertvolle Hinweise gaben und mich durch ihre konstruktive Kritik vor vielen Fehlern bewahrt haben. Mein Dank gilt hier in besonderer Weise Josef Fink, Achim Nick, Jürgen Koenemann Ph.D., Prof. Dr. Reinhard Oppermann, Anja Rockenberg, Alexander Sigel und Dr. Ramin Yasdi.

Dr. Wolfgang Dzida gab mir in fachlichen Diskussionen wertvolle Hinweise zu den Themen HCI und Evaluation, Prof. Dr. Peter Baumgartner zum wissenschaftstheoretischen Hintergrund und Günter Kniesel zum Thema Delegation. Meine Eltern nahmen die undankbare Aufgabe des Korrekturlesens auf sich. Ihnen allen sei an dieser Stelle dafür gedankt!

Hinweise

Aus Gründen der besseren Verständlichkeit werden einheitlich Funktionsbeschreibungen wie Benutzer und Leser gebraucht. Diese Ausdrücke sind geschlechtsunspezifisch gemeint.

Eine **Online-Version** enthält weiterführende Informationen zu dieser Arbeit. Sie ist abrufbar unter

http://fit.gmd.de/~patrick/diplomarbeit

und enthält neben der Arbeit selbst Kopien aller ausschließlich im WWW referierten Quellen. Dadurch sollen sie für den Leser bequem zugänglich bleiben, auch wenn die Originalquellen - wie bei WWW-Dokumenten oft der Fall - verändert, verschoben oder gelöscht wurden. Neben zusätzlicher Dokumentation der entwickelten Software in Bild und Film enthält die Online-Version Vorarbeiten zu dieser Arbeit, eine vergleichende Übersicht wichtiger Agentensysteme und -baukästen.

Inhaltsverzeichnis

Tabellenverzeichnis **9**

Abbildungsverzeichnis **11**

1 Einleitung und Aufbau der Arbeit **13**

2 Agentensysteme **17**

2.1 Einleitung 17

2.2 Überblick und Definition 17

2.3 Eine dreistufige Klassifikation von Agentensystemen 19

2.3.1 Erste Sicht: Kategorien differierender Auffassungen von Agenten 20

2.3.2 Zweite, merkmalsorientierte Sicht 21

2.3.3 Dritte, aufgabenorientierte Sicht 24

2.4 Anwendung: aufgabenorientierte Untersuchung bestätigt Schwächen von Agenten 31

2.5 Drei Hauptforderungen zur Lösung von Agentenproblemen 32

2.6 Generelle Grenzen und Probleme von Agenten 35

2.7 Zusammenfassung 37

3 Human-Computer Interaction **39**

3.1 Einleitung 39

3.2 Geschichte der HCI 39

3.3 Ausgangspunkt der Anpassung: Identifikation agentenrelevanter Gestaltungsgrundsätze in HCI-Gebieten und Normen 40

3.3.1 HCI 40

3.3.2 Softwareergonomie 43

3.3.3 Normen (DIN-Norm 66 234 und ISO-Norm 9241) 45

3.4 Vergleich vorgestellter Gebiete auf Agentenrelevanz 48

3.5 Transformation von HCI-Gestaltungsgrundsätzen zu einem agentenspezifischen Kriterienkatalog 49

3.5.1 Lernförderlichkeit 49

3.5.2 Komfort 50

3.5.3 Erwartungskonformität 51

3.5.4 Selbstbeschreibungsfähigkeit 52

3.5.5 Steuerbarkeit ... 52
3.5.6 Datenschutz/Datensicherheit ... 53
3.5.7 Conceptual Models ... 54
3.5.8 Progressive Disclosure ... 55
3.5.9 Visibility ... 56
3.5.10 Affordances und Constraints ... 57
3.5.11 Feedback ... 58
3.6 Ein System agentenrelevanter Kriterien setzt die Hauptforderungen um ... 59
3.7 Anwendung: Untersuchung Fireflys mit agentenrelevanten HCI-Kriterien ... 60
3.7.1 Lernförderlichkeit ... 62
3.7.2 Komfort ... 63
3.7.3 Erwartungskonformität und Conceptual Models ... 63
3.7.4 Selbstbeschreibungsfähigkeit und Feedback ... 66
3.7.5 Steuerbarkeit, Visibility, Affordances und Constraints ... 66
3.7.6 Progressive Disclosure ... 67
3.7.7 Datenschutz/Datensicherheit ... 67
3.7.8 Fazit: Die HCI-Hypothesen sind prinzipiell einsetzbar. ... 67
3.8 Zusammenfassung ... 68

4 Das Konzept Delegation bei Agenten ... 69
4.1 Einleitung ... 69
4.2 Delegation ist die Voraussetzung für einen Leitfaden ... 69
4.3 Aspekte der Delegation ... 70
4.3.1 Der Benutzer ... 70
4.3.2 Der Agent ... 71
4.3.3 Die Aufgabe ... 71
4.3.4 Das Verhältnis Benutzer - Agent ... 72
4.3.5 Der Prozess der Delegation ... 73
4.4 Abbildbarkeit von HCI-Gestaltungsgrundsätzen auf Delegation ... 75
4.5 Anwendung: parallele Untersuchungen Jangos ... 75
4.5.1 Allgemeines zu Jango ... 76
4.5.2 Ein Testlauf ... 77
4.5.3 Aufgabenorientierte Untersuchung von Jango ... 79
4.5.4 Untersuchung mit agentenrelevanten HCI-Gestaltungsgrundsätzen ... 80

4.5.5 Konzeptionelle Untersuchung von Jango 82
4.5.6 Zusammenfassung der Untersuchung Jangos 83
4.6 Zusammenfassung 84

5 Leitfaden zur agentenbasierten Softwareentwicklung 85
5.1 Einleitung 85
5.1.1 Motivation: „Wozu dient der Leitfaden?“ 85
5.1.2 Anwendungsbereich: „*Wann* hilft der Leitfaden?“ 86
5.1.3 Art der Hilfe: „*Wie* hilft der Leitfaden?“ 86
5.2 Designkomplexe 86
5.2.1 Komplex 1: Der Agent 87
5.2.2 Komplex 2: Der Benutzer 90
5.2.3 Komplex 3: Die Aufgabe 94
5.2.4 Komplex 4: Das Verhältnis Benutzer-Agent 95
5.3 Möglichkeiten der Evaluation 99
5.4 Zusammenfassung 99

6 Anwendung: Leitfadenorientierte Weiterentwicklung des Brokeringsystems bizzyB 101
6.1 Einleitung 101
6.2 Notwendigkeit einer verbesserten Schnittstelle für semiprofessionelle Broker 103
6.3 Aufstellung von Forderungen an die verbesserte Schnittstelle 104
6.4 Beschreibung der notwendigen Entwicklungen 104
6.4.1 Beschreibung eines vereinfachten Conceptual Models 105
6.4.2 Beschreibung der Wizards 107
6.4.3 Beschreibung weiterer Verbesserungsvorschläge zu bizzyB 111
6.5 Implementation 113
6.6 Diskussion der entwickelten Software 114
6.6.1 Überprüfung der aufgestellten Forderungen auf Umsetzung 114
6.6.2 Weitere Möglichkeiten der Evaluation 116
6.6.3 Verallgemeinerung der Ergebnisse 117
6.7 Zusammenfassung 117

7 Zusammenfassung und Ausblick 119

7.1 Zusammenfassung 119
7.2 Ausblick 120
7.2.1 Leitfaden wird erweitert und verbessert 120
7.2.2 Werkzeuge zur Agentenentwicklung verbessern sich 120
7.2.3 Lohnende Anwendungsgebiete für Agenten wachsen 121
7.2.4 Agentenbasierte Softwareentwicklung setzt sich durch 121

8 Literaturverzeichnis 123
8.1 Allgemeine Literatur 123
8.2 HCI, Softwareergonomie, Normen 123
8.3 Agenten 126

Tabellenverzeichnis

Tabelle 1: An Agentensysteme gestellte Erwartungen 31

Tabelle 2: Anspruch und (heutige) Wirklichkeit von Softwareagenten 32

Tabelle 3: Wichtige HCI- Gestaltungsgrundsätze 41

Tabelle 4: Zentrale Gestaltungsgrundsätze der Softwareergonomie (nach Oppermann u.a. 1992, 62-65) 44

Tabelle 5: Wichtige Gestaltungsgrundsätze der DIN-Norm 66 234 Teil 8 (Rödiger 1991, 15-19) 46

Tabelle 6: Prinzipen des Dialogs (ISO-Norm 9241 Teil 10) 47

Tabelle 7: Ein Testdurchlauf mit Jango 77

Tabelle 8: Jango und die Erwartungen des Benutzers 79

Tabelle 9: Jango im Agententest 81

Tabelle 10: Umsetzungshilfen zum Komplex „Agent“ 87

Tabelle 11: Umsetzungshilfen zum Komplex „Benutzer“ 91

Tabelle 12: Zusammenhang zwischen Agentenmetapher und Erwartungen des Benutzers in die Fähigkeiten des Agenten 93

Tabelle 13: Beispiele für die Verwendung von Charakteren in Agentensystemen 94

Tabelle 14: Sinnvolle Abstimmung des Ausgabemediums auf Interaktionsstil 94

Tabelle 15: Umsetzungshilfen zum Komplex „Aufgabe“ 95

Tabelle 16: Umsetzungshilfen zum Komplex „Verhältnis Benutzer-Agent“ 96

Tabelle 17: Wahl der Instruktionsmethode in Abhängigkeit von der Aufgabe des Agenten 97

Tabelle 18: Das Conceptual Model Suchmaschine in drei Komplexitätsstufen 106

Tabelle 19: Programmdurchlauf des Wizard Customer 109

Tabelle 20: Möglichkeiten zur Verbesserung der Hilfefunktion 112

Tabelle 21: Vergleich der Anzahlen nötiger Einzelschritte, um ein Profil zu generieren 115

Tabelle 22: Typische Arbeitschritte mit bizzyB und ihre Entsprechungen im Wizard Customer 116

Abbildungsverzeichnis

Abbildung 1: Übersicht und Struktur dieser Arbeit 14
Abbildung 2: Multiagentensystem mit spezialisierten Agenten 23
Abbildung 3: Der Mailfilter des Netscape-Mailprogramms 25
Abbildung 4: Resultate des Shoppingagenten BargainFinder auf die Suchanfrage nach der CD „Passion“ von Peter Gabriel. 27
Abbildung 5: Links arbeitet der Benutzer, rechts Letizia 28
Abbildung 6: Das WWW verleitet den Benutzer zu einer Tiefensuche. 28
Abbildung 7: Letizia beeinflusst die Browsingstrategie des Benutzers hin zu einer Breitensuche. 28
Abbildung 8: Clippit, in der deutschen Version Karl Klammer genannt, stellt sich vor. 30
Abbildung 9: Die Erfüllung der Hauptforderungen erhöht die Gebrauchstauglichkeit von Agentensystemen. 34
Abbildung 10: Bereiche der Ergonomie (Oppermann u.a. 1988, 5) 43
Abbildung 11: Schieberegler 57
Abbildung 12: System agentenrelevanter Kriterien 59
Abbildung 13: Der Bewertungsvorgang geht durch „submit“ in die nächste Schleife. 61
Abbildung 14: Empfehlungen myLAUNCHs regen zu weiteren Bewertungen an. 61
Abbildung 15: Details des Passports 65
Abbildung 16: Schema der Delegation: Der Benutzer delegiert eine Aufgabe an den Agenten 70
Abbildung 17: Delegation einer Teilaufgabe an den Agenten 72
Abbildung 18: Delegation und Konsultation nach Kniesel (1997) 73
Abbildung 19: Varianten der Delegation 74
Abbildung 20: Aspekte der Delegation im Überblick 75
Abbildung 21: Jango: Kategorieauswahl 77
Abbildung 22: Jango: Eingabe des Suchbegriffs 77
Abbildung 23: Jango: Visualisierung des Suchprozesses 77
Abbildung 24: Jango: Präsentation der Ergebnisse 78
Abbildung 25: Jango: Bestellformular abschicken 78
Abbildung 26: Jango: Bestätigung der Bestellung beim Anbieter erforderlich 78
Abbildung 27: Jango: Bestellformular des Anbieters 79
Abbildung 28: Unterschiedliche Erscheinungsformen des Office-Assistenten 92

Abbildung 29: Verwaltung der für bizzyB erreichbaren Datenquellen 101
Abbildung 30: Erstellung eines Suchprofils mit bizzyB 102
Abbildung 31: Übersicht und Zusammenhang der Verbesserungen bizzyBs 105
Abbildung 32: Komplexitätsstufe 1: Suchmaschinenmetapher „Simple Search“ 106
Abbildung 33: Komplexitätsstufe 2: Erweiterte Optionen „Advanced Search“ 106
Abbildung 34: Komplexitätsstufe 3: Anmeldung für „Expert Search“ 107
Abbildung 35: Startbildschirm mit Auswahl und Erläuterung der Wizards 108
Abbildung 36: Eingabe der Datenquellen 109
Abbildung 37: Wizard gibt Auskunft über bestimmte Datenquellen 109
Abbildung 38: Eingabe des Anfragetyps 109
Abbildung 39: Auswahl geeigneter Kategorien 110
Abbildung 40: Eingabe zusätzlicher Profilinformationen 110
Abbildung 41: Ausgabe der Suchergebnisse 111
Abbildung 42: Mit Hilfe des Agenten realisierte durchgängige Hilfefunktion 113

1 Einleitung und Aufbau der Arbeit

Agenten sind ein neuer Typ von Softwaresystemen, die Computerbenutzern Routinetätigkeiten abnehmen und sie bei ihrer Arbeit unterstützen sollen. Hatten bisherige Agentensysteme überwiegend einen experimentellen Charakter, ist inzwischen ein Wandel hin zu einer Dominanz professioneller Systeme zu beobachten: Mit großer Geschwindigkeit werden erfolgreiche Agentensysteme kommerzialisiert oder neu erstellt, die beispielsweise in intelligenten Hilfefunktionen oder Electronic-Commerce-Systemen zum Einsatz kommen.[1] Ein weiteres Einsatzgebiet sind Arbeitssysteme, deren Funktionsfülle soweit angewachsen ist, dass sie kaum ein Anwender mehr überblicken kann. Hier sollen Agenten einen besonders einfachen[2] Zugang bieten. Für die Zukunft ist mit einer Vielzahl unterschiedlichster Agentenanwendungen zu rechnen.

Forschung und Industrie setzen große Hoffnungen in Möglichkeiten und Marktpotential der sich entwickelnden Agententechnologie: Eine von der britischen Marktforschungsfirma Ovum erstellte Studie erwartet für das Jahr 2006 ein Marktvolumen von 4,7 Milliarden Dollar für Agentenanwendungen (Ovum 1997, 23). Eine andere Studie erwartet von Agenten revolutionäre softwaretechnische Auswirkungen, wie ihr Titel verrät: „Intelligent Agents: the New Revolution in Software“ (Ovum 1994). Vielerorts werden Agenten als Mittel zur Lösung informationstechnischer Probleme angesehen.

> Man stelle sich also einen persönlichen Softwareagenten vor, der mit seinem Anwender in natürlicher Sprache kommuniziert, der ständig aus dem Verhalten seines Benutzers lernt, der auch auf unvorhergesehene Ereignisse richtig zu reagieren vermag, der selbständig entscheidet, mit welchen Mitteln er seine Aufgaben erfüllt, und der auf diese Weise zum elektronischen Stellvertreter seines Besitzers im Netz wird.
>
> Mertens u.a. 1996, 522

Diese optimistische Vision eines zukünftigen Agenten ist allerdings noch weit von den Fähigkeiten heutiger Implementationen entfernt. Denn diese sind konventionellen Softwaresystemen hinsichtlich Funktionalität und Benutzbarkeit noch unterlegen, was Nutzwert und generelle Akzeptanz heutiger Softwareagenten relativiert. Die Unterlegenheit eines konkreten Agentensystems gegenüber konkurrierenden konventionellen Softwaresystemen verdeutlicht Tegenbos u.a. (1997, 139) am Beispiel des Agentensystems Agentware.

In der vorliegenden Arbeit wird nun versucht, die Ursachen dieser Mängel zu erkennen, die sie zu analysieren und zu demonstrieren, wie durch ihre Überwindung in einem Beispielsystem dessen Gebrauchstauglichkeit erhöht werden kann. Diese und weitere für die Arbeit wesentlichen Grundgedanken werden in dem folgenden Kasten zusammengefasst.

---

[1] Ein Beispiel für intelligente Hilfefunktionen ist der Microsoft Office Assistant (siehe Abschnitt 2.3.3.5), ein Electronic-Commerce-System ist der Online-Shopping-Agent Jango (siehe Abschnitt 4.5). Weitere Informationen zu sämtlichen, in dieser Arbeit erwähnten Agentenanwendungen finden sich in Anhang A.

[2] Die agentenspezifische Definition von Eigenschaften wie „benutzerfreundlich“ als einem Aspekt der Gebrauchstauglichkeit erfolgt in Abschnitt 2.5.

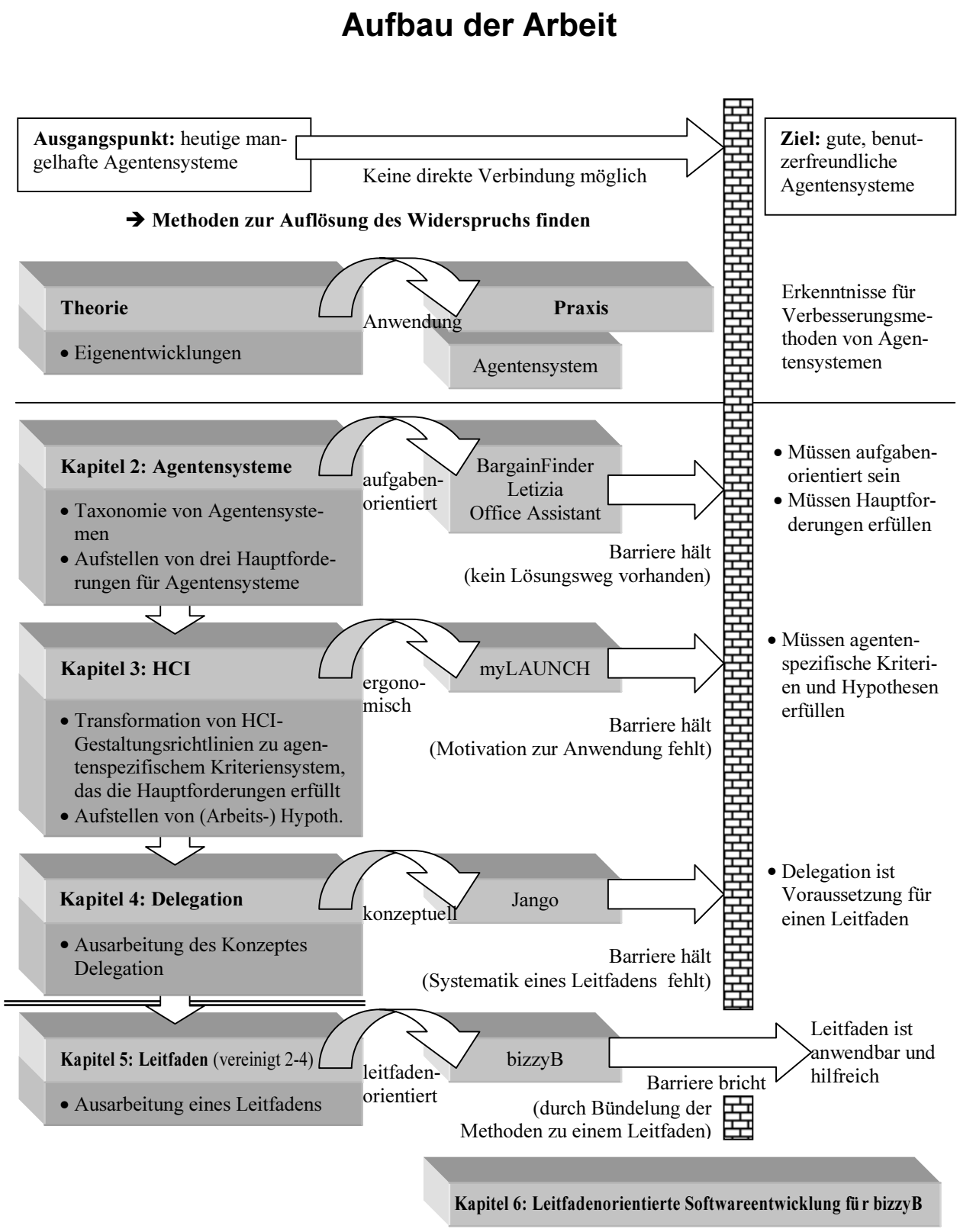


Abbildung 1: Übersicht und Struktur dieser Arbeit

Abbildung 1 zeigt eine schematische Übersicht dieser Arbeit. Im Mittelpunkt steht die Suche nach wirksamen Methoden zur Verbesserung von Softwareagenten, die schließlich in die Aufstellung eines Leitfadens mündet. Doch dazu bedarf es mehrerer unterschiedlicher Ansätze, die jeweils wirksamere Werkzeuge liefern.

Diese notwendig mehrstufige Vorgehensweise spiegelt sich in der Struktur der Arbeit wieder, die sich nicht in einen Grundlagen- und Hauptteil bzw. Theorie- und Anwendungsteil gliedern lässt. Stattdessen enthält jedes einzelne der Kapitel 2-5 sowohl einen Theorieteil mit einen neuen Ansatz zur Überwindung des Problems, als auch eine Anwendung auf Agentensysteme.

Nachdem zu Beginn der *Widerspruch zwischen Anspruch und Wirklichkeit bei Agentensystemen* dargestellt worden ist, wird nach wirksamen Mitteln gesucht, bestehende Schwächen von Agentensystemen zu überwinden bzw. Agenten von vornherein nutzerzentriert zu entwickeln.

Kapitel 2 enthält eine *aufgabenorientierte Darstellung* von Softwareagenten und ihren Problemen. Zunächst wird der Begriff Agent im Rahmen dieser Arbeit definiert, wobei Liebermans Verständnis von *Agenten als Gegensatz zu herkömmlichen direkt-manipulativen Benutzeroberflächen* zugrundegelegt wird. Es schließt sich ein *Modell von Softwareagenten* an, deren drei Stufen verschiedene Blickwinkel auf Agenten darstellen. Die erste Sicht dient gleichzeitig der *Eingrenzung der Definition von Softwareagenten auf User- und gleichzeitig Interfaceagenten***.** Die letzte Stufe dieser Übersicht stellt mit Hilfe von Beispielen typische Aufgaben von Agenten vor und leitet eine aufgabenorientierte Untersuchungsmethode von Agenten im praktischen Teil des Kapitels ein. Diese läßt zahlreiche (Funktions-) Mängel heutiger Softwareagenten erkennen und führt zur Aufstellung dreier *Hauptforderungen* für Agenten, nämlich (1) Lernförderlichkeit, (2) Kontrolloptimierung und (3) Vertrauensbildung.

Das dritte Kapitel untersucht Agentensysteme aus einem *ergonomischen Blickwinkel*. Nach Vorstellung und Vergleich der aussichtsreichsten Kandidaten zur Verbesserung von Agentensystemen - HCI[3] und Normen - werden aus diesen *agentenrelevanten Gestaltungsrichtlinien extrahiert, systematisiert und zu Hypothesen transformiert***.** Im praktischen Teil des Kapitels zeigt die Untersuchung des Agentensystems myLAUNCH die *prinzipielle Anwendbarkeit***,** aber noch mangelnde Schärfe und Rechtfertigung der Kriterien und Hypothesen.

Kapitel 4 greift *Delegation als das wichtigste, Agenten zugrundeliegende Konzept* auf, vertieft es und arbeitet seine Aspekte aus. Es entsteht ein tieferes Verständnis für den Sinn der Verbesserung von Agentensystemen - und damit der Schlüssel zur tatsächlichen Verbesserung. Im Anwendungsteil zeigt die *konzeptionelle Untersuchung* des Jango-Systems wesentliche Vorteile des integrierten Einsatzes aufgabenorientierter, ergonomischer und konzeptioneller Untersuchungsmethoden.

In Kapitel 5 werden das Wissen über Aufgaben, HCI und Delegation bei Agenten zu einem *Leitfaden* gebündelt, der die Hindernisse überwindet, die der Entwicklung agentenbasierter Software entgegenstehen.

[3] Human-Computer Interaction

Kapitel 6 demonstriert die Einsatzmöglichkeiten des Leitfadens anhand des Brokeringsystems bizzyB[4]. Bei der leitfadenorientierten Entwicklung verschiedener Systemerweiterungen *erweist sich der Leitfaden als hilfreich.*

[4] bizzyB ist die Pilotanwendung eines Brokeringsystems, eines Systems zum elektronischen Makeln von Informationen, das Einsatzmöglichkeiten und Vorteile von Agentensystemen zu zeigen soll. Es wird an der GMD entwickelt und ist Bestandteil des von der EU geförderten Projektes COBRA (**C**ommon **O**pen **Br**okerage **A**rchitecture), welches eine offene Architektur für (Information-) Brokeringsysteme entwikkelt, die mit Online-Informationen handeln.

2 Agentensysteme

> Eine Reihe von Firmen, auch Microsoft, entwickelt Agenten mit Fähigkeiten einer 'sozialen Benutzerschnittstelle'. Agenten werden grafische Benutzeroberflächen nicht ersetzen, sondern ergänzen, indem sie ein Erscheinungsbild bereitstellen, das - von Ihnen ausgewählt - Sie kennenlernt und Ihnen hilft. [...] Der Agent wird sich merken, worin Sie gut sind, was Sie in der Vergangenheit getan haben, wird Probleme versuchen vorherzusehen und Lösungen vorschlagen. (Gates 1996, 94)

Softwareagenten haben in den letzten Jahren stark an Popularität gewonnen - an der Benutzerschnittstelle von Softwaresystemen ebenso wie in WWW-typischen Bereichen der Filterung, Verarbeitung und Aufbereitung von Informationen. Dominierten zunächst der Forschung entstammende Softwareagenten, werden nun zunehmend kommerzielle Agentensysteme und Agentenaufsätze auf bestehende Systeme entwickelt. Für die Zukunft ist mit einem wachsenden Einfluss von Softwareagenten zu rechnen, die die Arbeit mit immer umfangreicher werdenden Softwaresystemen und Informationsströmen erleichtern oder gar erst möglich machen sollen.

2.1 Einleitung

Zunächst wird eine Definition von Softwareagenten im Sinne dieser Arbeit gegeben. Um einen Überblick über Klassen von Softwareagenten zu geben, wird eine Klassifikation entwikkelt, das nach Kategorien, Aspekten und Aufgaben von Agenten unterscheidet. Dabei wird der Agentenbegriff für diese Arbeit auf die Kategorie von Softwareagenten beschränkt, die sowohl User- als auch Interfaceagenten sind. Denn nur hier lassen sich HCI- Gestaltungsrichtlinien sinnvoll anwenden.

In einer aufgabenorientierten Untersuchung illustrieren beispielhaft ausgewählte aktuelle Agentensysteme Funktion und Leistung, machen aber auch Schwächen von Softwareagenten deutlich. Dennoch gibt die Unterscheidung zwischen implementierungsbedingten und grundlegenden Schwächen Anlass zur Hoffnung, zumindest erstere Kategorie von Mängeln agentenspezifisch überwinden zu können, wenn sich dafür die richtigen Methoden finden lassen.

2.2 Überblick und Definition

Trotz ihrer Aktualität ist die Idee von Softwareagenten nicht neu. Schon Mitte der 50er Jahre hatten John McCarthy und Oliver Selfridge „ein System vor Augen, das, mit einem Ziel ausgestattet, die Details der passenden Computeranweisungen ausführen konnte, und das von einem Menschen Hilfe erfragen und erhalten konnte, wenn es nicht mehr weiter wusste“ (Kay in Bradshaw 1997, 4).

In den achtziger Jahren lieferten die folgenden beiden Arbeiten aus dem Bereich der Künstlichen Intelligenz (KI) wichtige theoretische Vorarbeiten für Agenten: Newell (1982, 98ff) postuliert eine Wissensebene auf der höchsten Ebene von Computersystemen, in der Agenten nach dem Prinzip der Rationalität ihren Zielen nachgehen. In der Konzeption von Minsky (1988, 17) besitzen Agenten dagegen keine Intelligenz; Intelligenz ist vielmehr das Produkt des Zusammenwirkens zahlloser spezialisierter Agenten. Allgemein übt die KI-Forschung bis

heute einen großen Einfluss auf die Agentenforschung aus, was sich in Bezeichnungen wie „lernende“, „planende“ oder „intelligente“ Agenten niederschlägt.

Seit Beginn der neunziger Jahre ist das Interesse an Agenten und in Folge die Anzahl an Anwendungen sprunghaft gestiegen. Einige der frühen, zumeist Forschungsprojekten entstammenden Systeme wurden erfolgreich kommerzialisiert (MetaCrawler, Firefly), viele andere kommerzielle Systeme sind in Entwicklung oder bereits im Handel. Es ist anzunehmen, dass sich dieser Trend in den nächsten Jahren fortsetzen wird, wenn Agenten ihrem Anspruch gerecht werden, einen besonders einfachen Zugang zu komplexer Software zu bieten.[5]

Ein Überblick über aktuelle Agentensysteme zeigt, dass sie mehrheitlich auf dem WWW operieren. Dahinter steht zum einen die Überlegung, dass Agenten im WWW einen besonders großen Einsatzbereich haben und Forschungsergebnisse bequem weltweit zugänglich gemacht werden können. Einerseits bietet sich das WWW für gewisse Anwendungen wie die Filterung und Aufbereitung von Informationsströmen „natürlicherweise“ an. Andererseits sind viele agentenrelevante Fragestellungen erst durch das WWW aufgeworfen worden. Doch auch jenseits des WWW, etwa in den Bereichen Benutzermodellierung und Hilfefunktionen, nimmt die Bedeutung von Agenten und die Zahl ihrer Anwendungen zu.

In der Frage, was einen Agenten ausmacht und wie er definiert werden könnte, herrscht eine selbst für ähnlich expansive Forschungsrichtungen überraschende Uneinigkeit. Unterschiedliche Eigenschaften hervorhebende Begriffe wie „Agent“, „Assistent“, „Programm“ und „Werkzeug“ werden kontrovers verwendet.[6] Im Bereich der Forschung herrschen beispielsweise anspruchsvolle Agentendefinitionen vor, während etwa kommerzielle Anbieter weitaus niedrigere Anforderungen an ihre eigenen Agentensysteme stellen. Aus diesem Grund pflegen Agentenforscher wie -entwickler ihre Arbeiten stets mit einer eigenen, im Kontext ihrer Arbeit gültigen Agentendefinition einzuleiten.[7] So wird nun auch zu Beginn *dieser* Arbeit eine Definition für die im weiteren verwendete Auffassung von Agenten gegeben.

Die in dieser Arbeit vertretene Auffassung von Agenten als Systemkomponenten**,** *die dem Benutzer helfen bzw. Teile seiner Arbeit übernehmen können* orientiert sich an Liebermans (1997) Beschreibung. Lieberman ist einer der wenigen Agentenforscher, die auch auf dem Gebiet der HCI bewandert sind. Deshalb macht er sich - über die reine *Definition* von Agenten hinaus - auch Gedanken über den *Entwurf von Softwareagenten aus einem ergonomischen Blickwinkel.* Einige dieser Gedanken werden in das Ergebnis dieser Arbeit einfließen, den Leitfaden für Agentenentwickler. Liebermans Beschreibung ist auch deshalb hilfreich, weil sie explizit das Konzept Delegation aufgreift, welches das umfassendste Agenten zugrundeliegende Konzept darstellt und in Kapitel 4 genauer untersucht wird.

> Ein Agent ist jedes Programm, das vom Benutzer als Assistent oder Helfer agierend betrachtet werden kann, im Gegensatz zu einem Werkzeug im Sinne herkömmlicher direkt-manipulativer Benutzeroberflächen. Ein Agent sollte einige (aber vielleicht nicht alle) Eigenschaften aufweisen, die wir mit menschlicher Intelligenz verbinden: Lernen, Schlussfolgern, Adaptierbarkeit, Unabhängigkeit, Kreativität usw. Über den Benutzer kann man sagen, dass er eher einem Agenten eine Aufgabe delegiert als dem Agenten die Durchführung einer Aufgabe zu befehlen. (Lieberman 1997)

[5] Dieser Anspruch tritt bei Maes (1994), ja bereits im Titel „Agents that Reduce Work and Information Overload“ der Arbeit deutlich hervor.
[6] Franklins und Graessers Artikel „Is it an agent, or just a program?“ (1996) zeigt, dass sich selbst wissenschaftliche Artikel in der Frage erschöpfen können, was einen Agenten ausmacht.
[7] Als eine Vorarbeit habe ich einen detaillierten tabellarischer Vergleich zwischen ca. 20 Agentensystemen vorgenommen, der ebenso viele verschiedene Agentendefinitionen aufweist. Dieser Vergleich ist Bestandteil der Diplomarbeits-Homepage.

Man beachte, dass hier Agenten nicht über bestimmte Aufgaben oder Implementierungssprachen[8] definiert werden. Auch die bloße Bezeichnung „Agent“ macht noch kein Programm zu einem Agenten, sondern *die Möglichkeit einer Betrachtungsweise als Agent durch den Benutzer* ist ausschlaggebend. Diese flexible Handhabung erlaubt eine rasche Identifizierung von Softwaresystemen als Agenten.

Der in der Definition hervorgehobene Gegensatz zwischen Agenten und direkt-manipulativen Oberflächen ist nicht als Verbot von direkter Manipulation in Agentensystemen zu verstehen. Denn direkte Manipulation ist, geeignet umgesetzt, durchaus mit Agenten konsistent (Collins 1995, 495) bzw. nach Maes (Shneiderman u.a. 1997, 53) zur Einbettung von Agenten in eine Anwendung sogar nötig.

Mit Hilfe der obigen Definition werde ich im folgenden wichtige Kategorien, Aspekte und Aufgaben von Agentensystemen vorstellen und dabei gleichzeitig die im Kontext dieser Arbeit vertretene Auffassung von Agenten präzisieren.

2.3 Eine dreistufige Klassifikation von Agentensystemen

Obwohl der weitläufig verwendete und kontrovers definierte Agentenbegriff eine Klassifizierung von Agentensystemen erschwert, wurden von verschiedenen Seiten solche Versuche unternommen. Franklin u.a. (1996) schlagen beispielsweise eine „biologischen“ Taxonomie autonomer Agenten vor, ähnlich der linnéschen Systematik der Pflanzen und Tiere hierarchisch angelegt und unterteilt nach Eigenschaften der Agenten. Diese - nicht mehr aktuelle - Einteilung ist sehr breit angelegt und täuscht eine disjunkte Einteilbarkeit sämtlicher Agenten nach einem einzigen Kriterium vor. Müller (1996, 17-42) andererseits nimmt eine Einteilung in deliberative, reaktive und interagierende Agenten vor, deren Schwerpunkte auf der agenteninternen Repräsentation der Welt und dem Interaktionsverhalten zwischen Agenten liegen. Der Nachteil dieser Darstellung liegt darin, dass die Mehrzahl der klassifizierten Systeme dennoch zwei, wenn nicht allen der drei Einteilungsstufen zugeordnet werden muss.

Taxonomien wie die genannten eignen sich durch ihre *eindimensionalen Sichtweise* nur bedingt dazu, einen Überblick über das breite Spektrum an Agentenanwendungen zu bieten. Denn sie täuschen dem Leser eine umfassende, überlappungsfreie und zeitlich dauerhafte Einteilung von Agentensysteme vor, die sie in dieser Form nicht erzielen können: *Umfassend* können Agentensysteme prinzipiell nicht dargestellt werden, weil der Agentenbegriff derart weitläufige verwendet wird, dass eine Abgrenzung zu „normalen“ Systemen unmöglich ist. Üblicherweise führt ein solcher Versuch wie im Beispiel Franklins zu einer zu umfangreichen, sämtliche konventionellen Softwarebereiche vereinnahmenden Darstellung. Eine geeignet Abgrenzung ist lediglich bei der Beschränkung auf *eine* bestimmte Agentendefinition möglich. *Überlappungsfreiheit* ist wegen der Menge der möglichen Unterscheidungsmerkmale nicht zu verwirklichen. Auch die Hoffnung auf eine dauerhaft gültige Einteilung ist trügerisch - ändert sich doch die Funktionalität der „Agentensysteme“ noch schneller als ihre Bezeichnungen.

Dagegen habe ich für den folgenden Überblick über Softwareagenten eine Einteilung aufgestellt, die aus *drei verschiedenen Blickwinkeln* Agenten klassifiziert: nach *grundlegende Kategorien*, *Aspekten* und nach *Aufgaben* von Agentensystemen. Nach der notwendigen Abgrenzung der in dieser Arbeit vertretenen Auffassung von Agenten (erste Sicht) weckt die Ver-

[8] In der Tat werden Agenten in den unterschiedlichsten Computersprachen programmiert: In Java und C, in Skriptsprachen wie Python oder in speziellen Agenten-Entwicklungsumgebungen.

mittlung wesentlicher Eigenschaften (zweite Sicht) und Aufgaben (dritte Sicht) beim Leser ein tieferes Verständnis für das, was Softwareagenten ausmacht. Durch die Dreiteilung der Sicht wird vermieden, vom Wesen her verschiedene Aspekte in dasselbe Schema zu pressen, und der Überlappungseffekt verschiedener Aspekten wird gemildert.

2.3.1 Erste Sicht: Kategorien differierender Auffassungen von Agenten

Diese erste Einteilung unterscheidet nach fundamental unterschiedlichen Agentenkategorien, die unterschiedliche Verständnisse dessen ausdrücken, was Agenten sind. Bei jeder aufgeführten Kategorien wird angegeben, welches Verständnis auf unsere Agentendefinition zutrifft, wodurch die Definition selbst erheblich geschärft wird. Erst im Anschluss an die Kategoriensicht werden Agenten nach Eigenschaften und Aufgaben unterschieden.

2.3.1.1 Software- oder Hardwareagenten?

Ganz im Geist der Lieberman'schen Auffassung wird der Agentenbegriff für diese Arbeit **auf Softwareagenten beschränkt**, Agenten werden als eine spezielle Klasse vom Computerprogrammen aufgefasst. Andere Forscher wie Franklin u.a. (1996) dehnen dagegen den Agentenbegriff auf Roboter (auch *Hardwareagenten* genannt) und sogar Menschen aus. In der Tat können Menschen agentenähnliche Funktionen haben, beispielsweise Assistenten, Concierges und Butler. Deshalb werden Softwareagenten häufig als Roboter oder Vertreter der genannten Berufsgruppen visualisiert, was einen starken Einfluss auf die erwarteten Eigenschaften von Agenten beim Benutzer ausübt.

2.3.1.2 User- oder Systemagenten?

Liebermans Beschreibung, dass der Agent „vom Benutzer als Assistent oder Helfer" angesehen werden kann, deckt sich mit dem Begriff des Useragenten.[9] Ein Useragent ist nicht neutral eingestellt, sondern handelt parteilich zugunsten seines Nutzers. Er stellt das vorderste Glied in einer Kette wissensbasierter Systeme dar. *Systemagenten* dagegen handeln im Auftrag des Computersystems, sichern und optimieren seine Funktionsfähigkeit, wie zum Beispiel ein Garbage-Collection-Agent.[10] Die meisten Systemagenten arbeiten im Hintergrund, unbemerkt vom Benutzer, und sind oft nicht von diesem zu beeinflussen.

Der Schwerpunkt meiner Betrachtungen liegt auf **Useragenten,** da sie mit dem Benutzer interagieren, und nicht auf einer dem Anwender verborgenen Kommunikation nachgeschalteter Systemagenten.

2.3.1.3 Interface- oder Systemagenten?

Lieberman beschränkt sich auf die Beschreibung von Agenten an der Benutzeroberfläche, also *Interfaceagenten*, die mit dem Benutzer interagieren. Das Gegenstück zu Interfaceagenten sind die im letzen Punkt angesprochenen Systemagenten, die niemals mit dem Benutzer in Kontakt treten. Höchstens indirekt, nämlich über Interfaceagenten, ist eine Interaktion mit Systemagenten möglich**.** Aber die Interaktion mit dem Benutzer bleibt ein Randaspekt der Aufgabe von Systemagenten. Üblicherweise präsentieren sich Agentensysteme dem Benutzer

[9] Dieser Hilfsaspekt kommt beispielsweise bei Thomas (1996), betitelt als „To assist the User: On the Embedding of Adaptive and Agent-Based Mechanisms" zum Ausdruck.

[10] Beispiele für Systemagenten sind der Internet Connection Agent des Internet Explorers, Microsoft Network Backup Agenten, SNPT-Agenten und der WINS Proxy Agent.

über einen einzigen Interfaceagenten. Dieser übermittelt die Aufträge des Benutzers an einen oder mehrere Systemagenten und bereitet anschließend deren Ergebnisse in einer für den Benutzer verständlichen Form auf.

Mir erscheint eine **Beschränkung auf Interfaceagenten** im Rahmen dieser Arbeit, die die ergonomische Verbesserung der Mensch-Agenten-Schnittstelle zum Inhalt hat, als sinnvoll. [11]

2.3.2 Zweite, merkmalsorientierte Sicht

Die häufigste Kategorisierung von Agenten erfolgt nach gewissen besonders hervorstechenden Merkmalen wie Mobilität oder Autonomie. Im folgenden werden die wichtigsten dieser Merkmale genannt, die jeweils auch eigene Unterbereiche der Agentenforschung darstellen.

2.3.2.1 Autonomie

Ein Hauptgrund für den Einsatz von Agenten ist ihre Fähigkeit, *eigenständig Aufgaben auszuführen* , bei deren Erledigung sie gewisse *Freiheitsgrade* besitzen. Diese für konventionelle Softwaresysteme ungewöhnliche Eigenschaft erweist sich besonders bei der Erledigung wiederkehrender oder zeitaufwendiger Aufgaben als praktisch, aber auch beispielsweise beim automatischen Verkauf von Aktien, sobald der vom Agenten ständig beobachtete Börsenkurs einen bestimmten Wert unterschreitet.

Der von Lieberman genannte Gegensatz zwischen Agenten und direkt-manipulativen Oberflächen erklärt sich aus der Autonomie des Agenten: Während in konventionellen direktmanipulativen Oberflächen alle Aktionen vom Benutzer ausgehen, verhält sich ein *Agentensystem nicht rein reaktiv,* sondern kann selbständig Aktionen auslösen. Es kann beispielsweise Vorschläge machen wie Liebermans in Abschnitt 2.3.2.1 vorgestelltes Agentensystem Letizia, und unter Umständen *proaktiv*, also spontan starten.

In der Bezeichnung „autonom“ spiegelt sich ein neues Verständnis vom Verhältnis zwischen Benutzer und Agent wieder: In einem *wechselseitigen Dialog* stehend arbeiten Benutzer und Agent parallel und arbeitsteilig an derselben Aufgabe. Der autonome Agent ist weniger Befehlsempfänger als vielmehr *Vermittler, Mitarbeiter oder Assistent des Benutzers*.

2.3.2.2 Intelligenz

„Intelligente“ Aspekte von Agenten umfassen gemäß der in dieser Arbeit verwendeten Agentendefinition Liebermans *Planungsfähigkeit, Lernfähigkeit, Schlussfolgern, Kreativität und Adaptivität.* Nach Wooldridge u.a. (1995, 117) verknüpfen KI-Forscher mit intelligenten Agenten oft zusätzlich Konzepte wie *Wissen, Überzeugung, Absicht und Verpflichtung,* also Eigenschaften, die normalerweise eher mit Menschen in Verbindung gebracht werden.

Die erwähnten engen Verbindungen zwischen Agenten- und KI-Forschung haben auf Agenten Fragestellungen wie folgende übertragen: Kann Software prinzipiell intelligent sein, wo beginnt maschinelle Intelligenz, und ist nicht künstliche Intelligenz grundlegend verschieden von menschlicher Intelligenz? - Während Agenten nach Newell (1982) stets intelligent sind, besitzt bei Minsky (1988) erst der Verband von Agenten Intelligenz. Steels (1995) sieht Intelligenz als autonomen Agenten (siehe vorhergehenden Abschnitt) innewohnende Eigenschaft an, erkennbar an der Fähigkeit des Agenten, seine Strukturen anzupassen und funktionsfähig

[11] Ein Beispiel für die ebenso legitime Beschränkung auf *Systemagenten* bietet Müller, in dessen gut 200seitigem Werk „Das Design intelligenter Agenten“ (1996) die *Benutzerschnittstelle* keinerlei Erwähnung findet.

zu bleiben. Vereerstraeten (1998) weist wiederum auf den im folgenden Abschnitt diskutierten Aspekt *Mobilität* als Voraussetzung für Intelligenz hin. Er wirft die philosophische Frage auf, ob für eine statische Intelligenz prinzipiell ein Verständnis für Raum, Zeit und Bewegung - wichtige Grundlagen für Intelligenz - möglich ist.

Der Begriff Intelligenz ist mit Vorsicht zu verwenden, führt er doch oft zu im konkreten Agentensystem nicht erfüllbare Erwartungen menschenähnlicher Intelligenz. Aber dem Anwender selbst kann es unangenehm sein, ein als „intelligent“ tituliertes System zu benutzen, was nach Lanier (1995) zu Misstrauen, Ablehnung der Software oder Minderwertigkeitsgefühlen führen kann.

2.3.2.3 Mobilität

Konventionelle Softwaresysteme arbeiten ausschließlich auf dem Rechner, auf dem sie installiert bzw. gestartet wurden. Mobile Agenten haben dagegen die Fähigkeit, im Netzwerk zu „wandern“, also ihren *Programmcode vor oder während der Ausführung auf entfernte Server zu übertragen*. Mobilität kann bei auf Netzwerken operierenden Agentensystemen große Vorteile haben. Bei Suchvorgängen in großen Datenmengen kann sich die benötigte Bandbreite im Netzwerk stark reduzieren, wenn nicht sämtliche Daten zur Auswertung auf den lokalen Rechner übertragen werden, sondern der Agent die Auswertung am Ort der Datenquelle vornimmt und nur die relevanten Informationen weiterleitet.[12] Ein weiterer Vorteil ist die Möglichkeit, mit mobilen Agenten Rechnerverbände in verteilten Anwendungen gleichmäßiger auszulasten und somit Geschwindigkeitsvorteile zu erzielen. Allerdings sind alle genannten Vorteile abhängig von einer geeigneten Strategie, Organisation und Skalierung, da anderenfalls Kopiervorgänge sowie Inter-Agenten-Kommunikation die eingesparten Ressourcen wieder aufzehren können. Eine ausführliche Argumentation für den Einsatz mobiler Agenten findet sich bei Lange (1998).

Zur Zeit existieren keine echten Anwendungen mobiler Agenten im Internet (Krulwich 1998a).[13] Eine Ursache dafür ist das Fehlen allgemein zugänglichen Server, auf die sich fremde Agenten kopieren dürfen. Dagegen bieten firmeninterne Netze (Intranets) naheliegendere Anwendungsgebiete für mobile Agenten. Beispielsweise können Außendienstmitarbeiter von Wartungs- oder Transportunternehmen mit vernetzten tragbaren Arbeitsrechnern ausgestattet werden, in denen jeglicher Datentransfer durch mobile Agenten erfolgt.[14] Andere vielversprechende Einsatzbereiche für mobile Agenten sind elektronischer Handel, Telekommunikationsdienste, Netzwerkmanagement, Gruppenarbeit und Workflow-Management.

2.3.2.4 Skalierung

Agentenverbände können einzelnen Agenten gegenüber Vorteile haben bei *asynchronen, verteilten* oder *parallelisierbaren Aufgaben*. Soll beispielsweise auf mehreren Datenquellen gleichzeitig gesucht werden, können *Multiagentensysteme* viel schneller zum Ziel kommen, indem sie alle Datenquellen parallel von je einem Agenten durchsuchen lassen. Zur Verständigung untereinander benutzen Agenten spezielle Sprachen, von denen *Knowledge Interchange Format (KIF)* und *Knowledge Query and Manipulation Language (KQML)* die am weitesten verbreiteten sind.

---

[12] „Das Motto ist einfach: Bringe die Berechnungen zu den Daten, nicht die Daten zu den Berechnungen“ (Lange 1998).

[13] *Entwicklungsumgebungen* für mobile Agenten sind dagegen bereits verfügbar, zum Beispiel die Klassenbibliothek Voyager der Firma ObjectSpace.

[14] Die erste derartige Anwendung steht mit General Magic's DataRover-Familie kurz vor der Markteinführung. Die Firma ist ein Pionier auf dem Gebiet mobiler Agenten und bekannt durch die Entwicklung der Agentensprache TeleScript.

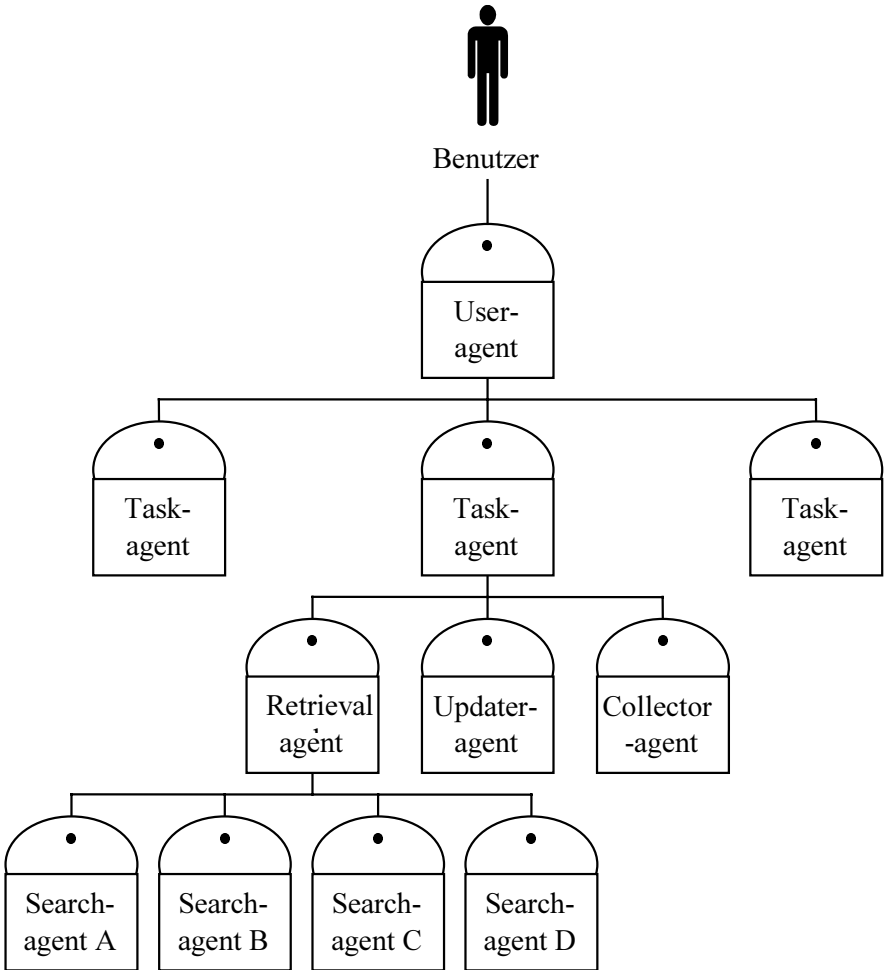


Abbildung 2: Multiagentensystem mit spezialisierten Agenten

Einen weiteren Vorteil kann die *aufgabenorientierte Spezialisierung* der Agenten innerhalb eines Multiagentensystems bieten. Abbildung 2 zeigt solch eine Architektur spezialisierter Agenten. Jeder Auftrag des Benutzers wird vom Useragenten auf einen eigenen Taskagenten übertragen. Diese teilen Teile der Aufgabe an spezialiserte Agenten ab, die wiederum (wie hier im Falle des Searchagenten) Sub-Agenten - beispielsweise für paralleles Suchen - aufrufen. Charakteristisch ist, dass der *Benutzer nur mit dem Useragenten in Verbindung* steht und von der Existenz der nachgeschalteten Agenten nichts erfährt. Der Useragent bildet die Schaltzentrale, in der Aufgaben des Benutzers verteilt und Ergebnisse für ihn aufbereitet. Für den Benutzer spielt es in diesem Fall keine Rolle, dass er in Wirklichkeit mit einem Multiagentensystem arbeitet. Die im Beispiel gezeigten Agenten arbeiten arbeitsteilig nach einem vorgegebenen Schema. Vorstellbar sind aber genauso Systeme selbstorganisierender oder konkurrierender Agenten.

Multiagentensysteme lassen sich danach einteilen, ob sich ihre Agenten lediglich selbst starten, oder auch vermehren (bzw. andere Agenten instantiieren) können. Besonders selbstreproduzierende Agentensysteme brauchen wirksame Methoden zur Verhinderung unkontrol-

lierter Vermehrung ihrer Agenten.[15] Da Agentensysteme mit zunehmender Anzahl und Vielfalt an Agenten immer unüberschaubarer, ressourcenintensiver und fehleranfälliger werden, ist es wichtig, Anzahl und Organisation der Agenten genau abzuwägen.

2.3.3 Dritte, aufgabenorientierte Sicht

Betrachten wir zuletzt Agentensysteme aus einem aufgabenorientierten Blickwinkel. Typische Aufgaben werden durch Beispielsysteme illustriert und sollen zum Verständnis für die Anwendungsvielfalt von Agentensystemen beitragen. Gleichzeitig bildet diese Sicht den Schlüssel zur ersten und direktesten Untersuchungsmethode von Agentensystemen, nämlich der aufgabenorientierten Analyse.

Maes (1994) spricht zwei Hauptziele von Agenten an, nämlich die Informations-Überbelastung (Information Overload) des Benutzers zu verhindern und lästige oder routinemäßige Aufgaben zu verringern. Ersteres trifft auf die im folgenden angesprochenen Filter und Recommender zu, letzteres auf Shoppingagenten und Hilfefunktionsagenten.

2.3.3.1 Filter

Eine traditionelle Aufgabe von Useragenten ist die Filterung von Informationsströmen. Schon seit Mitte der achtziger Jahre werden Mailfilter eingesetzt, beispielsweise im E-Mail-System X.400. Heute bieten zahlreiche E-Mail-Softwareprogramme die Möglichkeit, mit Hilfe von Filtern eingehende E-Mails vorzusortieren. Diese Filter sind als Agenten zu betrachten, weil sie dem Benutzer ermöglichen, unter großen Mengen von E-Mails schnell die besonders wichtigen zu identifizieren. Sie reduzieren also den Information Overload des Benutzers. Einmal aktiviert, laufen Mailfilter kontinuierlich im Hintergrund und werden selbständig aktiv, sobald eine neue E-Mail eintrifft. Somit stehen sie im Gegensatz zu traditionellen direktmanipulativen Oberflächen.

[15] Die Bedeutsamkeit dieser Forderung zeigt die Geschichte des ersten Virus. Konzipiert als Kontrollstruktur, zur optimalen Auslastung eines Rechnerverbundes, brachte sie durch unkontrollierte Vermehrung ihrer Elemente und eine in Folge exponentiell anwachsende Zahl von Anfragen binnen kurzer Zeit sämtliche Rechner zum Stillstand.

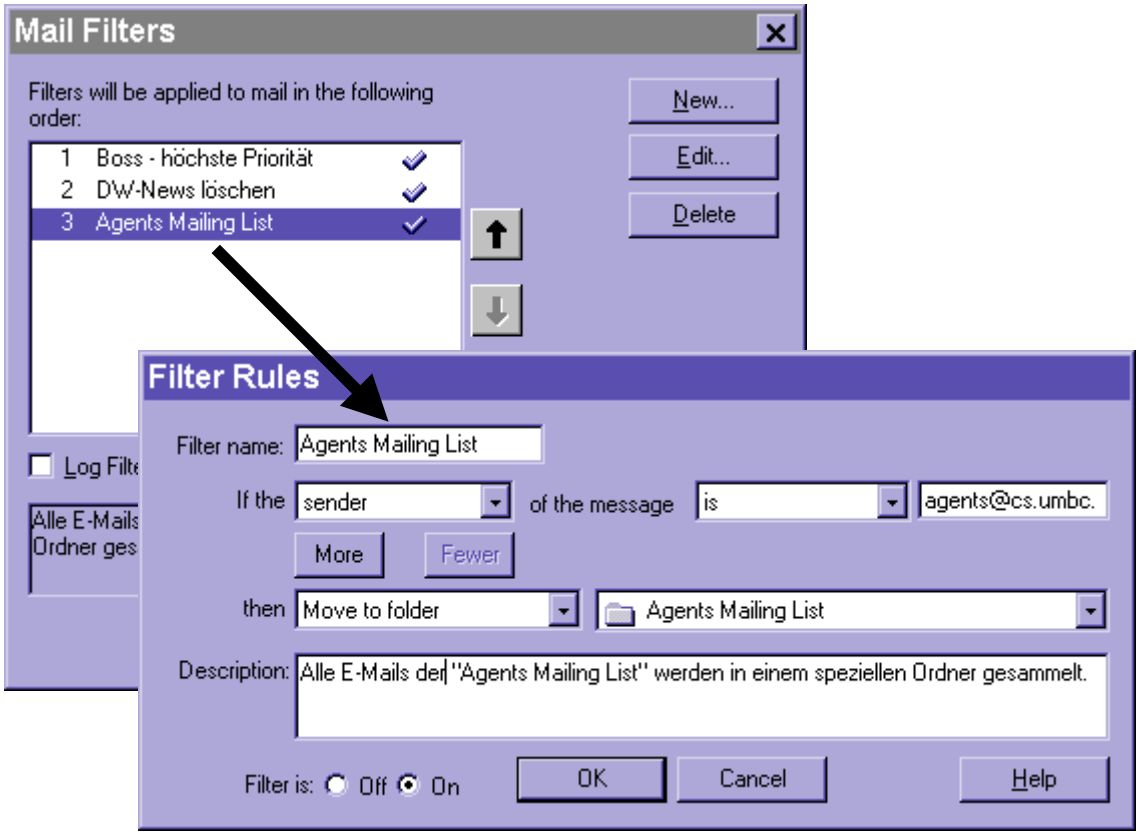


Abbildung 3: Der Mailfilter des Netscape-Mailprogramms

Als Beispielanwendung habe ich die E-Mail-Komponente des Netscape Communicators gewählt. Abbildung 3 zeigt drei verschiedene Filter sowie die Details eines von ihnen, des „Agent Mailing List"-Filters. Diesen Filter habe ich so eingestellt, das alle E-Mails, die von der unter der Adresse „agents@cs.umbc.edu" operierenden Agenten-Mailingliste eingehen, automatisch in einem separaten Ordner gesammelt werden. Ein anderer Filter erhöht die Priorität der E-Mails des Arbeitgebers, wodurch diese an oberster Stelle in der E-Mail-Liste erscheinen. Ein weiterer Filter löscht automatisch eingehende E-Mails einer bestimmten (als uninteressant empfundenen) Mailingliste. Jeder beliebige Teil einer eingehenden E-Mail kann analysiert werden und darauf basierend eine Vielzahl von Aktionen ausgelöst werden. Einmal gestartete Filter bleiben dauerhaft aktiviert. Dies kann den Nachteil haben, dass die Existenz von Filtern in Vergessenheit gerät, wenn die Auswirkungen ihrer Aktionen nicht für den Benutzer sichtbar sind, wie im Falle des Löschfilters.

Einen weiteren E-Mail-Agenten beschreibt Maes (1994), der vergleichbar zu dem vorgestellten Filter arbeitet, aber nicht „programmiert" wird. Dieser Agent beobachtet den Benutzer bei der Arbeit und führt seine Filteraktionen autonom auf Basis der dabei gewonnenen Daten durch. Andere Agenten mit Filterfunktion durchsuchen das WWW (Agentware, MetaCrawler), das Usenet (vgl. Maes 1994) oder Datenbanken.

2.3.3.2 Stellvertreter

Benutzer können Agenten Aufgaben übertragen, die diese *stellvertretend* für sie im WWW ausführen. Zur Zeit sind dies kleine, mechanische Aufgaben, aber mit Verbesserung von Agentenaspekten wie Lern- und Planungsfähigkeit werden diese Aufgaben an Umfang und Komplexität zunehmen. Mertens u.a. (1996, 522) vertreten die Ansicht, dass der Agent schließlich „zum elektronischen Stellvertreter seines Besitzers im Netz wird."

Eine zugegebenermaßen triviale, aber verbreitete Umsetzung eines Stellvertreters beantwortet während der Urlaubszeit des Benutzers sämtliche am Arbeitsplatz eingehenden E-Mails automatisch mit einer Kurzmeldung in der folgenden Art: „Bin bis zum 5. Mai in Urlaub und beantworte erst dann wieder E-Mails."

In Systemen wie Internet Chats oder Online-Mehrbenutzer-Spielen[16] präsentieren sich die Benutzer oft über längere Zeit den übrigen Nutzern in einer konstanten Erscheinungsform, sei sie nun realistisch oder fiktiv gewählt. Allerdings ist diese Erscheinungsform nur dann sichtbar, wenn der Benutzer mit dem System verbunden ist. Dagegen existieren in beiden Systemen dauerhafte „künstliche" Charaktere, die zumeist nützliche Aufgaben wie die Einweisung von Neulingen oder den automatischen Ausschluss von Störenfrieden aus dem System haben. Es ist gut vorstellbar, dass sich Systeme etablieren, in denen sich auch *menschliche Benutzer dauerhafte Repräsentationen wählen* können. Immer, wenn der Benutzer das System verlässt, würde er die *Kontrolle seines Charakters an einen Agenten übergeben*. Dieser Agent müsste über ein Modell des Benutzers verfügen und für eine gewisse Menge von Aktionen vom Benutzer autorisiert sein. Gute Agenten würden den Benutzer so „lebensecht" vertreten, dass andere Benutzer diesen Wechsel gar nicht bemerken. Der Agent könnte den Charakter schlafen lassen, bis der Benutzer wieder erscheint, oder begonnene Aufgaben während dessen Abwesenheit fortführen.

Innerhalb weniger Jahre werden sich die beschriebenen Systeme von ihrer heutigen textorientierten Form zu lebensecht animierten 3D-Systemen wandeln. Mit diesem Wandel werden Möglichkeiten und Bedeutung der dauerhaften Präsenz der Avatare, der gewählten Erscheinungsformen der Benutzer, durch Stellvertreteragenten deutlich steigen.[17]

2.3.3.3 Shoppingagenten

Ein *Shoppingagent* sucht im Auftrag des Benutzers im WWW systematisch nach Angeboten für ein bestimmtes, möglichst genau definiertes Produkt wie CDs oder Bücher. Anlaufstellen sind unter anderem Elektronic Malls oder WWW-Seiten der Produkthersteller. Hat der Agent ein „günstiges" Angebot gefunden, meldet er sich beim Benutzer zurück bzw. kauft das Produkt, sofern er dazu autorisiert und mit Geldmitteln ausgestattet ist.

[16] sogenannte Multi-User Domains (MUDs)
[17] Erste 3D-Systeme mit durch Avatare repräsentierten Benutzern existieren bereits, so zum Beispiel das System blaxxun.

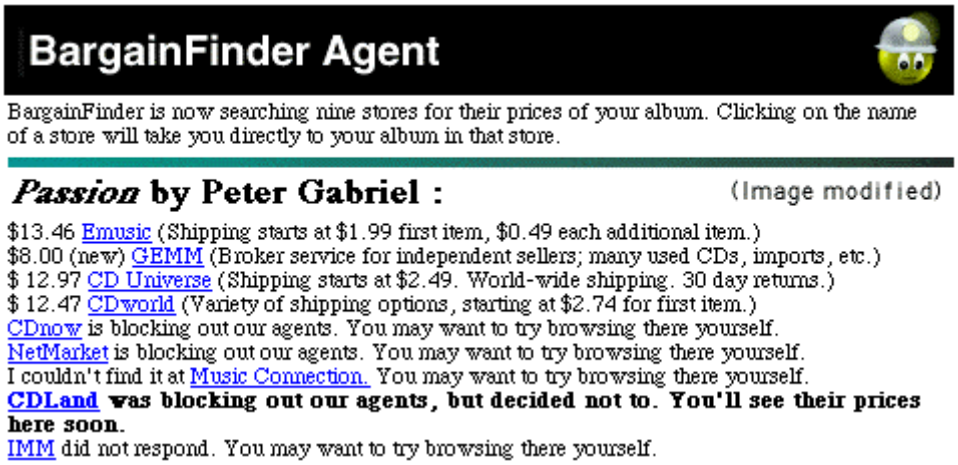


Abbildung 4: Resultate des Shoppingagenten BargainFinder auf die Suchanfrage nach der CD „Passion“ von Peter Gabriel.

Zur Illustration von Shoppingagenten zeigt Abbildung 4 das System BargainFinder, den nach Angaben seines Entwicklers Bruce Krulwich ersten intelligenten Shoppingagenten im WWW. BargainFinder sucht bei einer Reihe festgelegter Anbieter simultan nach CDs. Geordnet nach Anbietern gibt BargainFinder in einer Liste zurück, ob die gewünschte CD verfügbar ist und was sie kostet.

Schon dieses einfache Beispiel zeigt, dass ein Shoppingagent zu durchaus brauchbaren Resultaten kommen und dem Benutzer viel Zeit ersparen kann.[18]

Sicherlich besitzen Shoppingagenten auch filtertypische Aspekte. Wesentlich ist hier jedoch die Stellvertreterrolle, die der Agent einnimmt, denn er arbeitet im Auftrag und Namen des Benutzers. Es ist gedanklich kein großer Schritt zu einer *Autorisierung* des Agenten durch den Benutzer, unter bestimmten Bedingungen ein gesuchtes Produkt zu kaufen. Voraussetzung ist, dass der Agent Zugriff auf elektronisches Geld oder Kreditkarteninformationen hat. Es ist klar, das solche Zusatzfunktionen nur bei weitreichendem Vertrauen des Anwenders in den Agenten akzeptiert werden. Für viele Stellvertreteragenten bietet sich eine *Benutzermodellierung* an, die vor allem zu besseren Suchergebnissen des Agenten bei unpräzisen Vorgaben des Benutzers führt.

2.3.3.4 Recommender

Recommender sind Agentensysteme, die den Benutzer bei seiner Tätigkeit beobachten und - selbständig oder auf Verlangen - Vorschläge machen oder Anregungen geben. Als Beispiel möchte ich ein Agentensystem vorstellen, das dem Benutzer Vorschläge zum effektiven Browsen im WWW macht.

Letizia ist ein akademisches Agentensystem, erdacht und prototypisch entwickelt von Henry Lieberman. Da keine Testversion von Letizia erhältlich ist, muss ich mich bei der Beschreibung auf die von Lieberman (1997) bereitgestellten Informationen stützen. Während der Benutzer einen normalen Browser bedient, wird er von dem Agenten beobachtet. Dieser versucht nun, das Verhalten des Nutzers zu analysieren und vorherzusagen. Er lädt autonom alle vom aktuell dargestellten Dokument aus erreichbaren Seiten und untersucht sie auf mögliche

[18] In Kapitel 4 wird Jango, ein neuerer und mächtigerer Shoppingagent, eingehend analysiert.

Relevanz für den Benutzer. Die „interessantesten" der auf diese Weise in Breitensuche erhaltenen Fundstücke schlägt Letizia dann dem Benutzer zum Lesen vor.

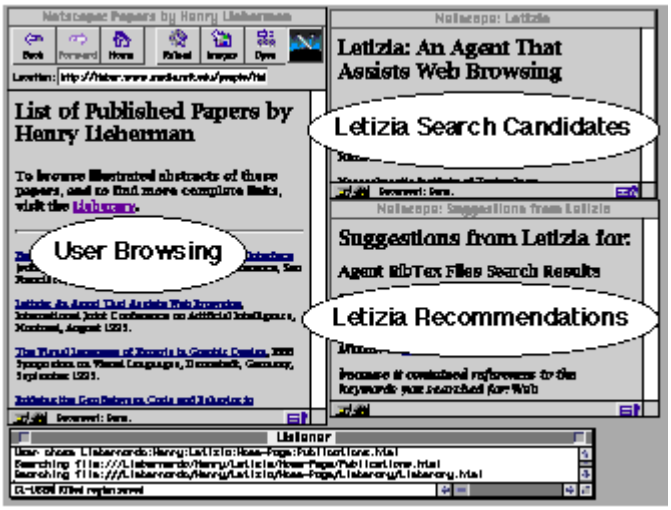


Abbildung 5: Links arbeitet der Benutzer, rechts Letizia

Letizia lässt dem Benutzer die Wahl, unbeeinflusst vom Agenten im linken Browserfenster zu arbeiten oder die Link-Vorschläge im rechten mittleren Fenster zu berücksichtigen(siehe Abbildung 5). Die einzige, aber wichtige Funktion der beiden übrigen Fenster ist es, dem Benutzer Feedback zu geben über gerade untersuchte und noch zu untersuchende Seiten. Auffällig sind die platzintensiven Kontroll- und Anzeigefenster Letizias, die mehr Raum einnehmen als das eigentliche Browsingfenster selbst. Unklar ist, ob das System auch auf Benutzer eingerichtet ist, die simultan mehrere Ziele verfolgen und darum mehrere Browserfenster geöffnet haben.

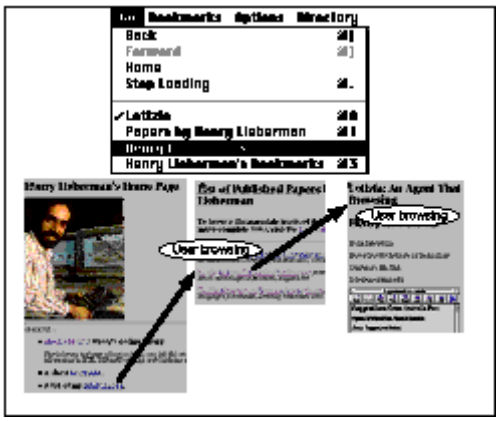


Abbildung 6: Das WWW verleitet den Benutzer zu einer Tiefensuche.

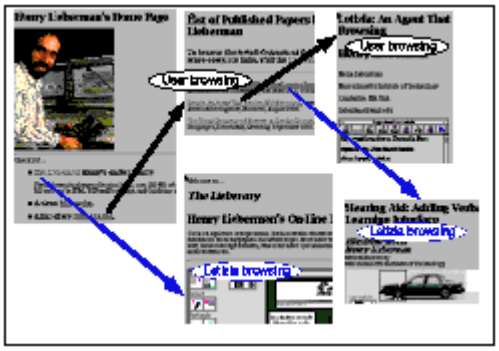


Abbildung 7: Letizia beeinflusst die Browsingstrategie des Benutzers hin zu einer Breitensuche.

Von der aktuellen Position des Benutzers aus führt Letizia eine Breitensuche durch. Dadurch soll der Benutzer weg von der ineffektiven Tiefensuche, zu der man im WWW leicht verleitet wird, und hin zu einer Breitensuche geführt werden (Abbildung 6 und Abbildung 7).

Andere Recommender-Systeme sind beispielsweise Firefly, LifestyleFinder und Alexa. Das in Kapitel 4 ausführlicher untersuchte Firefly lernt die Vorlieben des Benutzers und macht darauf basierend Vorschläge, empfiehlt beispielsweise zum Musikgeschmack des Benutzers passende CDs. LifestyleFinder dagegen ermittelt indirekt den Lebensstil des Benutzers und empfiehlt ihm zu diesem Stil inhaltlich passende WWW-Adressen. Alexa ist eine Hilfssoftware, die unter anderem dynamisch Vorschläge in Form von WWW-Adressen macht und damit Letizia am nächsten kommt. Im Unterschied zu Letizia untersucht Alexa nicht die von der aktuellen Seite ausgehenden Links, sondern gewinnt seine Seitenvorschläge mit Hilfe einer Ähnlichkeitssuche in seiner WWW-Datenbank. Das führt beispielsweise zu folgender interessanten Fehleinschätzung Alexas: Liest man mehrere deutschsprachige WWW-Seiten, so empfiehlt Alexa Server (vor allem in den USA) mit deutschsprachiger Literatur oder germanistisch relevanten Adressen. Offenbar hält Alexa den Benutzer nicht für einen lokale Informationen abrufenden Deutschen, sondern fälschlicherweise für einen Amerikaner mit besonderem Interesse an der deutschen Sprache.

Im Zusammenhang mit Recommendern wichtige Techniken sind *Benutzerbeobachtung* und *Benutzermodellierung*: Durch Beobachtung kann der Agent seine Vorschläge auf die aktuelle Situation des Benutzers zuschneiden, angepasst an Faktoren wie Aufgabe, Zeitdruck und Aufmerksamkeit. Durch Benutzermodellierung können längerfristige Aspekte wie Präferenzen, Wissen und Interessen des Benutzers berücksichtigt werden.

2.3.3.5 Hilfefunktion

In der Rolle von Hilfefunktionen liefern Agenten einen neuer Zugang zu komplexen Systemen. Angesichts immer komplexer und umfangreicher werdender Software besteht die Aufgabe eines Agenten primär darin, deren Benutzbarkeit zu erhöhen bzw. erst zu ermöglichen. Hilfefunktionen sind ein Beispiel für nicht primär auf das WWW orientierte Agentenaufgaben.

In großen Softwaresystemen haben sich über die letzten Jahre hinweg eine Vielzahl von Hilfefunktionen angesammelt:

- Alphabetischer Index mit Stichwörtern
- Thematische, aufgabenorientierte Hilfe, oft in Kapitelform
- *Direkthilfe*, die Symbole oder aktuelle Wahlmöglichkeiten erläutert
- *Tip des Tages*: Diese zufällig ausgewählte Kurzhilfe wird automatisch beim Programmstart oder auf Verlangen gegeben. Nach dem Schema „Um A zu erreichen, tue B." erhöht sie kontinuierlich das Problemlösungswissen des Benutzers.
- *Demo-Modus, Beispiele und Templates*: demonstrieren, was mit dem System erreicht werden kann und auf welche Weise.
- *Wizards* führen gängige Aufgaben prototypisch mit dem Benutzer durch, der lediglich die gewünschten Eingaben tätigen muss.

Teile dieser Hilfsmöglichkeiten haben bereits Ansätze von Agentenfunktionalität, insbesondere die Direkthilfe sowie Wizards.

Trotz dieser Bemühungen sind Softwaresysteme allgemein schlecht bedienbar: Wichtige Funktionen sind nicht auffindbar, während Menüs immer zahlreichere irrelevanter Funktionen

enthalten, ein als „schleichenden Featurismus“ bezeichnetes Phänomen (Norman 1988, 172; Nelson 1990, 236). Die Benutzeroberfläche präsentiert sich inkonsistent (Erickson 1990, 1), Fehlermeldungen sind nichtssagend. Solche und ähnliche Klagen durchziehen die gesamte HCI-Literatur. Nach Shneiderman (1992, 1) führen Bedienungsprobleme zu Frust, Angst und im schlimmsten Fall zur generellen Meidung von Computern.

Durch den Einsatz von Agenten könnte sich ein völlig neuer Zugang zu komplexen Systemen eröffnen, wobei der Agent ausführt - wie eine Sekretärin - was der Benutzer beabsichtigt. Zumindest könnte ein Agent eine den Nutzer beobachtende und kontextsensitve Hilfefunktion darstellen.

Microsofts ersten Schritt in diese Richtung stellt der in der aktuelle Office-Version eingeführte Assistent dar, wohl der zur Zeit am weitesten verbreitete Softwareagent.[19] Das Office-Paket ist durch den Einsatz des Assistenten kein Agentensystem geworden, sondern hat lediglich einen Agentenaufsatz erhalten. Dieser präsentiert sich als lustig animierter Helfer in einem separaten kleinen Fenster. Seine Erscheinungsform lässt sich unter einer Reihe unterschiedlich gestaltete Charaktere auswählen. Trotz der im Charakter-Auswahlmenü beschriebenen Individualität (siehe die Beschreibung in Abbildung 8) beschränkt sich diese auf die Animation und Geräusche des Assistenten, erstreckt sich aber nicht auf das Interaktionsverhalten mit dem Benutzer.

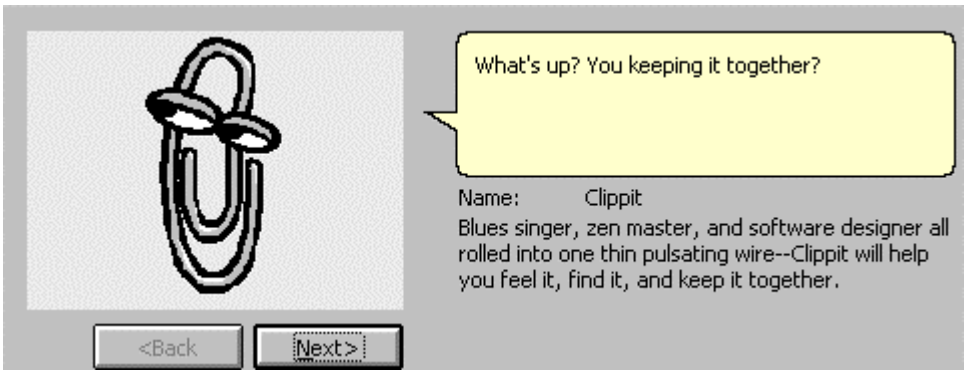


Abbildung 8: Clippit, in der deutschen Version Karl Klammer genannt, stellt sich vor.

Inhaltlich stellt dieser Agent einen neuen Zugang zur von jeher mit Schwachstellen behafteten Hilfefunktion des Programms dar. Nun kann der Benutzer eine Frage in Klartext eingeben, und der Agent liefert eine Liste möglicherweise hilfreicher Seiten der konventionellen Hilfefunktion zurück. Zur Bearbeitung der Benutzerfrage wertet der Agent sowohl die bisherigen Aktionen des Benutzers als auch dessen Frage mit einem IntelliSense genannten, aber nicht weiter dokumentierten Verfahren aus. Die angebotene Hilfe des Assistenten ist durchaus sinnvoll: Mich selbst hat sie oft schneller zur gesuchten Hilfeseite geführt als manuelles Suche in der konventionellen Hilfefunktion. Zusätzlich hat der Agent, der sich im übrigen abschalten und umfassend konfigurieren lässt, die Aufgabe übernommen, beim Programmstart und auf Verlangen einen zufällig gewählten „Tip des Tages“ auszugeben.

Es lässt sich vermuten, dass die Entwickler des Assistenten drei Ziele im Auge hatten. Sie wollten zunächst Schwächen der bisherigen Hilfefunktion ausmerzen, was teilweise gelungen

[19]Das vor einigen Jahren entwickelte erste Agentensystem Microsofts, genannt Bob, führte durch mangelnde Kundennachfrage gepaart mit heftiger Kritik von Agentenforschern zu großen Imageverlusten und finanziellen Einbußen. Insofern lässt die Tatsache, dass nun das (nach Windows) zentrale Softwarepaket mit Agenten ausrüstet wird, auf die Bedeutung schließen, die die Firma Agenten beimisst.

ist. Weiterhin sollte die Akzeptanz von Agenten in Microsoft-Systemen mit diesem in seinem Funktionsumfang recht beschränkten Vertreter getestet werden. Das wichtigste Ziel scheint eine durch die Einbringung eines Spaßfaktors (siehe Abschnitt 5.2.3) und Adaptierbarkeit gesteigerte (emotionale) Bindung des Benutzers an das Office-Paket zu sein.

2.4 Anwendung: aufgabenorientierte Untersuchung bestätigt Schwächen von Agenten

Abschnitt 2.3.3 hat anhand typischer Beispiele illustriert, dass Softwareagenten sehr unterschiedliche Aufgaben haben. Somit können Agenten nur unter Berücksichtigung ihrer speziellen Aufgabe untersucht und kritisiert werden. Dazu müssen zunächst eigens für das zu untersuchende System aufgabenorientierte, messbare Kriterien aufgestellt werden, die anschließend überprüft werden. Diese weniger konstruktive als analytische Methode hat, wie sich zeigen wird, den HCI-Gestaltungsgrundsätzen gegenüber Vorteile hinsichtlich *Quantifizierbarkeit und Operationalisierbarkeit.* Außerdem ist es mit geringem Aufwand möglich, Agentensysteme etwa mit einer Checkliste auf vorher aufgestellte Forderungen zu testen.

Tabelle 1: An Agentensysteme gestellte Erwartungen

Klassen von Aufgaben	Beispiel	Erwartungen des Benutzers	Umsetzung
Shoppingagent	BargainFinder	• Zeitersparnis beim Suchen	+
		• Auffinden des günstigsten Angebots	−
		• Vorschlag von interessanten Produkten und Sonderangeboten, die dem Benutzer unbekannt sind	−
		• Selbständige Durchführung von Einkäufen	−
		• Berücksichtigung von Sicherheitsaspekten	o
Recommender	... beim Browsen: Letizia	• Vorschau: Benutzer weiß schon vor dem Aufruf eines Links, was sich dahinter verbirgt.	−
		• Optische Hervorhebung interessanter Links	+
		• Anzeige thematisch verwandter und „interessanter“ Links	−
		• Bessere Orientierungshilfen zu „Wo bin ich?“, „Was kann ich hier tun?“, „Wo komme ich her?“, „Wohin kann ich gehen?“, die das „Lost in Hyperspace“-Phänomen verhindern.	−
		• Geschwindigkeitsvorteil durch Preloading	−
Hilfe-Funktion	Office Assistant	• Hilfe im Kontext	o
		• Lösung eines aktuellen Problems, ohne das Problem ausführlich beschreiben zu müssen	o
		• Proaktivität: Programm beobachtet Benutzer und macht selbständig Vorschläge zur Aufgabenoptimierung	−
		• Lernfunktion: für verschiedene Kompetenzgrade und Aufgabengebiete	−
		• Vollständiges Nachschlagewerk	−

Tabelle 1 stellt aus Abschnitt 2.3.3 entnommene Klassen von Aufgaben und dazu passende Beispielsysteme vor. Die aufgestellten Erwartungen des Benutzers sind abgeleitet von

- vermuteten Erwartungen eines mit Agentensystemen im Allgemeinen vertrauten Benutzers
- (Werbe-) Beschreibungen des aktuellen Agentensystems

- vermuteten Schlussfolgerungen, die dem Benutzer durch die Art der Repräsentation des Agentensystems nahegelegt werden

In der letzten Spalte der Tabelle ist vermerkt, ob das Beispielsystem die jeweilige Annahme erfüllt (+), nicht erfüllt (–) oder mit Einschränkungen erfüllt (o).

Es zeigt sich, dass die überwiegende Anzahl der Annahmen von den konkreten Systemen nicht erfüllt werden. Dies bestätigt die Untersuchungen von Tegenbos u.a. (1997, 139), die bei der Untersuchung von Agentware, einer kommerziellen intelligenten Suchmaschine der Firma Autonomy, wesentliche Mängel hinsichtlich der Gebrauchstauglichkeit feststellen. In der obigen Untersuchung ergaben sich Mängel unabhängig davon, ob es sich um einen Shoppingagenten, einen Recommender oder eine Hilfe-Funktion handelt.

Ein Ergebnis der Untersuchung ist der Nachweis ihrer *Wirksamkeit beim Identifizieren von Mängeln bei Agenten*. Es fehlt allerdings ein Werkzeug, um geeignete Vorschläge zur Überwindung dieser Mängel zu machen. Das enttäuschende Fazit der Untersuchung macht die Notwendigkeit deutlich, Mittel zu entwickeln, die *Agentensysteme bereits in ihrer Entstehungsphase verbessern*. Denn wollte man BargainFinder, Letizia und den Office Assistenten im nachhinein mit den Erwartungen des Benutzers in Einklang bringen, müsste man die Systeme weitgehend neu erstellen.

Tabelle 2: Anspruch und (heutige) Wirklichkeit von Softwareagenten

Anspruch	Wirklichkeit
+ Lästige Routinetätigkeiten werden übernommen. + Information Overload beim Benutzer wird verhindert. + Die Komplexität von Systemen und Aufgaben verringert sich.	– Agenten lösen Probleme schlechter als herkömmliche Software. – Agenten sind weniger gebrauchstauglich als herkömmliche Software. – Softwareagenten werfen neuartige Bedienungsprobleme auf.

Tabelle 2 zieht ein verallgemeinerndes Fazit der bisherigen Untersuchungsergebnisse von Agentensystemen. Es existiert ein *Widerspruch zwischen Anspruch und Wirklichkeit* von Softwareagenten.[20] Übertriebene Erwartungen, Umsetzungsmängel sowie neuartige Probleme wie die ungewohnt erscheinende mittelbare Steuerung kompensieren einen Gutteil der Vorteile von Agenten.

2.5 Drei Hauptforderungen zur Lösung von Agentenproblemen

Wie kann mit einem im vorausgegangenen Abschnitt geschärften Problembewusstsein die Lösung der Schwierigkeiten von Agenten angegangen werden?

- Nicht weiter verwunderlich ist, dass sich bereits durch normale softwaretechnische Maßnahmen einige Schwächen schlecht programmierter oder schlecht in bestehende Systeme integrierter Agentensysteme vermeiden lassen. Doch der Anspruch ist, nicht allgemeine, sondern *agentenspezifische Probleme zu lösen, und zwar mit agentenspezifischen Mitteln.*
- Bereits angesprochen wurde die der Nutzen einer *aufgabenorientierten Analyse* des Systems und die Notwendigkeit, bereits in ihrer Entstehungsphase zu verbessern. Weiterhin

[20] Diese Ergebnisse werden auch durch spätere Untersuchungen (siehe Abschnitte 3.7 und 4.5) bestätigt.

ist wichtig, *für Agenten geeignete Problem zu erkennen* und nur solche durch Agenten zu lösen (siehe Abschnitt 5.2.1.1).

- Eine Forderung nach *maximierter Gebrauchstauglichkeit*, was die Bereiche *Benutzerfreundlichkeit, Effektivität und Zumutbarkeit* umfasst, ist sinnvoll. Diese Forderung beschränkt sich auf Arbeitssysteme, da sich Gebrauchstauglichkeit erst im harten, professionellen Arbeitseinsatz auszahlt. Es ist dagegen völlig legitim, dass sich experimentelle Systeme wie Letizia und BargainFinder auf die bloße Demonstration der Machbarkeit einer Agententechnologie beschränken. Deshalb beschränken sich alle weiteren Untersuchungen (und auch der Leitfaden selbst) auf professionelle Arbeitssysteme.
- Ergonomische Methoden haben sich bei der Verbesserung der Gebrauchstauglichkeit von Softwareanwendungen bewährt. Da gerade diese einen Schwachpunkte heutiger Agentensysteme ausmacht, stellt sich die Frage, ob erfolgreiche *ergonomische Mittel* hier hilfreich sein können.

Um der letzten Frage nachzugehen, werden im folgenden Kapitel diejenigen ergonomischen Gebiete untersucht, die sich bei Agentensystemen nutzbringend anwenden lassen. Beispielsweise könnten die Stärken von Agentensystemen (nämlich die beim Anspruch von Agenten formulierten Punkte) unterstützt werden. Wesentlicher für die Gebrauchstauglichkeit ist jedoch die *Vermeidung von Usability-Nachteilen*, da die gesamte Gebrauchstauglichkeit einer Anwendung nach dem Prinzip des schwächsten Glieds einer Kette durch seine Schwachstellen begrenzt wird. In Agentensystemen werden die folgenden Problemkomplexe als die Hauptschwachstellen identifiziert::

1. Agentensysteme können für den Benutzer ungewohnt und erklärungsbedürftig sein. Besonders mobile und Multiagentensysteme sind in diesem Zusammenhang problematisch: Ihre neuartigen Konzepte sind wie ihre Funktionsweise *schwer durchschaubar*, und eine parallele Kontrolle und Steuerung mehrerer Agenten stelle *hohe Anforderungen an den Benutzer*.[21] Dem steht der Anspruch von Softwareagenten entgegen, dem steigenden Anteil nicht-professioneller Computerbenutzer einen einfachen Zugang zu immer komplexeren Softwaresystemen anzubieten. Folglich müssen Agenten die Belange gerade von computerunerfahrenen Benutzern berücksichtigen, die weder mit Softwaresystemen im allgemeinen vertraut sind, noch mit Agentenkonzepten im speziellen. Also müssen Agenten *Lernprozesse beim Benutzer unterstützen* - bezüglich des Systems wie bezüglich ihrer selbst.
2. Laut Maes (1994, 146) baut die Agentenschnittstelle keine Schicht zwischen Benutzer und Anwendung auf, weil der Benutzer stets alternativ, den Agenten umgehend, direkt mit der Anwendung arbeiten kann. Tatsächlich aber übernehmen Agenten Systemfunktionen restlos (vor allem in als Agentensystemen konzipierten Anwendungen) oder setzen auf ihnen auf. In beiden Fällen wird, funktional oder technisch, eine Schicht aufgebaut, die die Distanz zwischen Benutzer und System erhöhen und zu einer *Mittelbarkeit der Steuerung* führen kann.[22] In manchen Fällen ist es gerade beabsichtigt, durch den Einsatz von Agenten die Kontrollmöglichkeiten des Benutzers zu beschränken. In öffentlichen Systemen, beispielsweise Info-Kiosks können Zeit- und Funktionsbeschränkungen notwendig sein.

[21] Beispielsweise genügt bei der Suchmaschine Agentware nicht die Eingabe von Suchbegriffen Stattdessen muss der Benutzer pro Suchthema jeweils einen Agenten erzeugen, „trainieren“ und die Suche selbst initiieren.
[22] Microsofts Bob ist Negativbeispiel dafür, keine Alternativen zur Agentensteuerung der Anwendung (hier: des Betriebssystems) zuzulassen. Der Office Assistant dagegen, vorgestellt im vorausgehenden Abschnitt, setzt technisch auf vorhandener Funktionalität der Hilfefunktion auf, überträgt dadurch allerdings auch deren Unzulänglichkeiten.

Ähnlich verhält es sich in tutoriellen oder sicherheitskritischen Systemen. In beiden Fällen liegt die Lösung in der *Optimierung des Kontrollaspekts* mit HCI-Methoden.

3. Kritikern wie Lanier (1995), Norman (1997) und Shneiderman u.a. (1997, 56) zufolge lösen Agenten beim Benutzer **Misstrauen** und Angstgefühle aus - insbesondere, wenn sie als „intelligent“ bezeichnet werden. Viele Benutzer bezweifeln, dass Agentensysteme kostspielige (wie bei Shoppingagenten) oder sicherheitskritische Aktionen tatsächlich absolut zuverlässig ausführen. Dieser Effekt wird durch die beiden zuletzt genannten Problembereiche noch verstärkt. Die Korrektheit der Systeme vorausgesetzt, muss aus diesem Grund versucht werden, mit HCI-Mitteln dem Benutzers unbegründete Ängste vor Agentensystemen zu nehmen**,** so dass er *dem Softwareagenten mit Recht vertrauen* kann. Vor allem soll verhindert werden, dass sich der Agentenbegriff zu einem negativen Schlagwort entwickelt, wie es - berechtigt oder nicht - den Begriffen ActiveX und Cookie widerfahren ist.

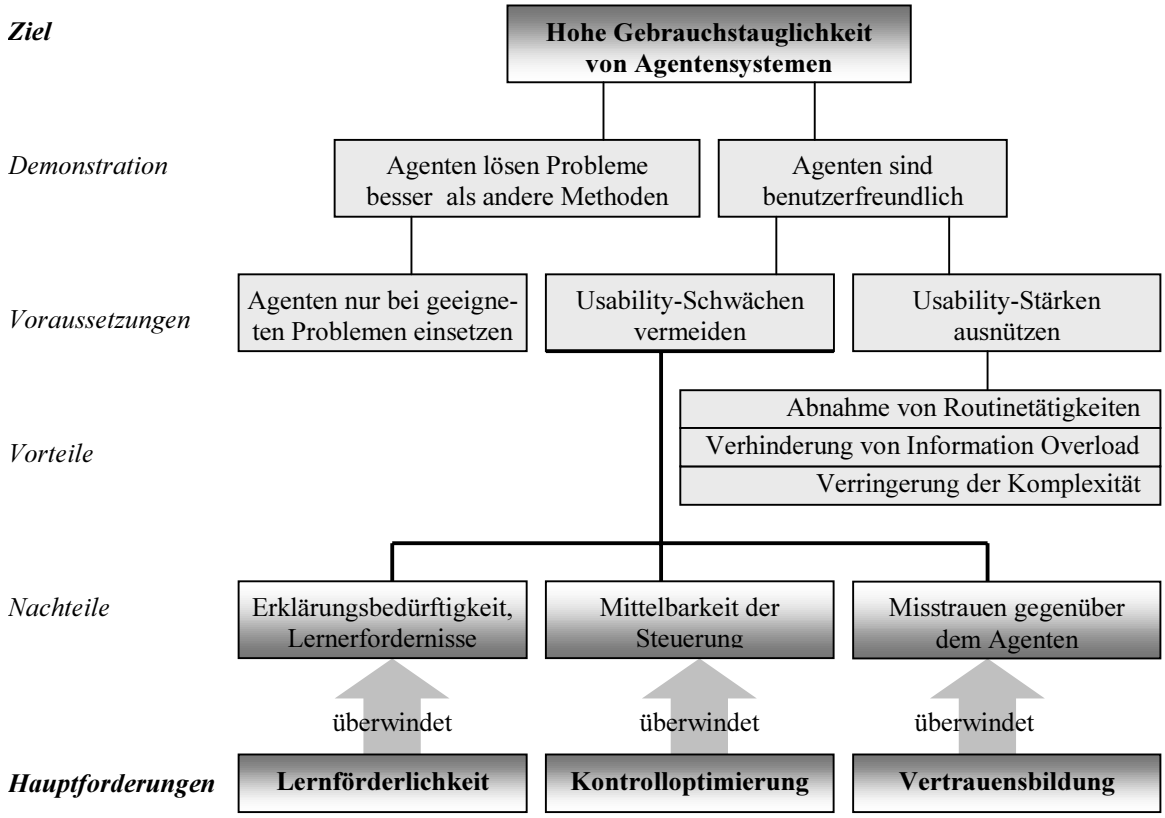


Abbildung 9: Die Erfüllung der Hauptforderungen erhöht die Gebrauchstauglichkeit von Agentensystemen.

Die vorausgegangenen Argumentationsstränge werden in obiger Abbildung 9 grafisch veranschaulicht. Ausgangspunkt aller Überlegungen ist das Ziel, die Gebrauchstauglichkeit von Softwareagenten zu maximieren. Dieses Ziel steht in der Grafik an oberster Stelle. Ein Weg zum Ziel ist die Demonstration, wann und inwiefern Agenten konventionellen Anwendungen überlegen sind. Diese Demonstration wird am ehesten gelingen, wenn der Einsatz von Agen-

ten auf Bereiche fokussiert wird, in denen Agenten tatsächlich konzeptionelle Vorteile gegenüber anderen Methoden haben.

Eine weiteres Mittel zum Ziel ist die Demonstration der Benutzerfreundlichkeit von Agenten, was durch Verstärkung der Usability-Vorteile und Vermeidung von Usability-Nachteilen möglich ist. Usability-Stärken können auf relativ unproblematische Weise maximiert werden, nämlich im wesentlichen durch eine geeignete Wahl der Agentendomäne. Die großen Usability-Nachteile Erklärungsbedürftigkeit, Mittelbarkeit der Steuerung und Misstrauen verlangen dagegen nach speziellen, durch die Hauptforderungen ausgedrückte Gegenmaßnahmen, Also werde ich im folgenden diejenigen Gestaltungsrichtlinien auswählen und für Agenten anpassen, die speziell die genannten Hauptforderungen nach (1) Lernförderlichkeit, (2) Kontrolloptimierung und (3) Vertrauensbildung unterstützen und damit letztlich die Gebrauchstauglichkeit von Softwareagenten erhöhen. In Abschnitt 3.6 ist dieser Schritt vollendet, in welchem mit Abbildung 12 die obige Abbildung nach unten hin durch ein System von Kriterien vervollständigt wird.

Wir halten fest: Die Aufgabe an HCI ist, Methoden zu liefern, um die Schwachpunkte der Benutzbarkeit von Agentensystemen zu beheben. Gegen die Hauptmängel stelle ich drei Hauptforderungen:

1. Lernförderlichkeit
2. Kontrolloptimierung
3. Vertrauensbildung

Die HCI wird also im folgenden Kapitel von vornherein darauf untersucht, welche ihrer Gestaltungsrichtlinien diese drei Hauptforderungen unterstützen.

2.6 Generelle Grenzen und Probleme von Agenten

Außer den bisher angesprochenen Beschränkungen von Softwareagenten existieren noch andere *grundlegende* Limitierungen von Softwareagenten. Ohne sie im Rahmen dieser Arbeit lösen zu können, möchte ich sie dennoch erwähnen, um die Erwartungen in den zu entwikkelnden Agenten-Leitfaden in realistischen Grenzen zu halten.

Setzt sich die immer breitere Verwendung des Agentenbegriffs für unterschiedliche, zum Teil konträre Konzepte fort, so könnte sich der Begriff zu einem *inhaltslosen Schlagwort* entwikkeln. Auf dieselbe Weise sind Attribute wie „interaktiv“ und „generisch“ Opfer ihres eigenen Erfolgs geworden. Längst haben sie ihren einstigen Inhalt verloren, schmücken aber weiterhin die Beschreibungen zahlreicher Softwareprodukte.[23] Auch der umgekehrte Fall ist denkbar, dass nämlich ein Modewort wie „Agent“ auch einmal *aus der Mode kommt* und völlig verschwindet. Zwei der erfolgreichsten Agentensysteme, Netbots Jango und Microsofts Office-Assistent, werden von den Herstellern bewusst nicht als „Agenten“ bezeichnet, was mit der Missverständlichkeit des Begriffs begründet wird.

Man kann Parallelen ziehen zwischen vergangenen Wellen der Euphorie über „Elektronengehirne“, künstliche Intelligenz und das der aktuellen Internet-Euphorie, in dessen Fahrwasser

[23] Beim Kunden wecken sie zwar keine spezifischen Erwartungen mehr, können aber dennoch nicht einfach weggelassen werden. Da alle anderen Anbieter ihre Produkte mit ihnen schmücken, würde in diesem Fall optisch etwas „fehlen“, was in Folge die Kunden vom Kauf abbringen könnte. Eine Analogie kann hier die Lebensmittelwerbung dienen, die die meisten ihrer Produkte als „wohlschmeckend“ und „gesund“ anpreist, obwohl diese Attribute keine speziellen Erwartungen beim Kunden wecken.

Agenten ihren großen Aufschwung genommen haben.[24] Bisher wurde jede dieser Wellen von einer Phase der Ernüchterung abgelöst, und diese Entwicklung könnten in ähnlicher Weise Agenten durchmachen. Denn in der augenblicklichen Euphorie um Agenten werden, wie bereits erwähnt, verschiedentlich *übertriebene optimistische Erwartungen* an die Fähigkeiten zukünftiger Agentensysteme geweckt. Sobald klar wird, dass viele Behauptungen unhaltbar sind, könnte dies zu großen Enttäuschungen und zu einer allgemein negativen Besetzung des Agentenbegriffs führen. Auch (Nwana 1996, 36f) sieht dieses Problem, in dessen Folge er ein Verschwinden des Interesses sowie der Fördermittel befürchtet. In jedem Fall sollte die Geschichte der „Künstlichen Intelligenz" ein mahnendes Beispiel sein für die Gefahren, eine neue Technologie in den Rang eines Allheilmittels zu erheben.

Einzelne Kritiker (Lanier 1995; Stoll 1995, 95ff, Shneiderman u.a. 1997, 57) *lehnen Softwareagenten aus prinzipiellen Erwägungen ab.* Sie haben große konzeptuelle oder Sicherheitsbedenken, weshalb sie von der Verwendung von Agenten ganz abraten und stattdessen reine direkt-manipulative Oberflächen favorisieren. Es ist wenig aussichtsreich, diesen grundlegenden Bedenken durch bloße Verbesserung von Softwareagenten begegnen zu können.

Als konkretes *Beispiel für prinzipielle Limitierungen von Agentensystemen* möchte ich nun Shoppingagenten herausgreifen. Dieser Typ von Agent sucht für den Benutzer im WWW nach Produkten und kann sie unter bestimmten Bedingungen auch kaufen.[25] Solche Systeme können nur bei Markttransparenz optimal arbeiten. Optimistische Stimmen glauben, dass Käufer durch „electronic shopping" binnen weniger Jahre „die Sortiments- und Preistransparenz zurückgewonnen [haben werden]" (Burger u.a. 1996, 157). Doch dieser Glaube ist illusorisch, kann doch ein Agent für ein bestimmtes Produkt niemals eine vollständige oder über den Preisfaktor hinausgehende vergleichende Marktübersicht liefern.

- **Agenten liefern keine vollständige Angebotsübersichten**. Dazu ist das WWW zu unübersichtlich, und die Menge potentieller Anbieter befindet sich in einem andauernden Wandel. Heutige Shoppingagenten sind auf eine (entwicklerseitig fest definierten) Teilauswahl von Anbietern beschränkt.[26] Und selbst unter diesen handverlesenen Anbietern sind Agenten nicht überall willkommen. Da von einem Preisvergleich nur Billiganbieter, sogenannte *price leaders* wirklich profitieren, schließen eine Reihe von Anbietern Agenten vom Zugriff auf ihre Angebotsseiten aus (siehe BargainFinder, Abbildung 4).
- **Shoppingagenten unterscheiden Produkte ausschließlich nach dem Faktor Preis.** Damit beschränkt sich der Einsatzbereich von Shoppingagenten automatisch auf Massenprodukte und bestimmte Dienstleistungen, die bis auf den Preis identisch sind. Weiterhin sind nicht-neuwertige Güter auszuschließen, da Wertverluste höchstens formelhaft von Agenten berechnet werden können, in der Praxis aber der subjektive Restwert gebrauchter Güter für den Kunden den Ausschlag gibt. Weiterhin kann der Agent nicht wissen, welcher Abnutzungsgrad eines Produkt für den Benutzer noch akzeptabel ist.[27]
- **Mehrwerte verhindern die Vergleichbarkeit von Produkten.** Da die meisten Online-Anbieter in der Tat eine Markttransparenz fürchten, werden sie - genau wie herkömmliche

[24] Auf die Parallelen zwischen diesen Wellen der Euphorie und die mangelnde Seriosität ihrer (oft bemerkenswert gleichlautenden) Versprechungen weist beispielsweise Stoll (1995, 94f) hin.
[25] Die Entscheidung, hier Shoppingagenten als Beispiel herauszugreifen, begründet sich aus der detaillierten Untersuchung des Shoppingagenten Jango in Abschnitt 4.5. Bereits jetzt soll der Leser ein Verständnis für Funktionen und Grenzen dieses Agententyps gewinnen.
[26] Eine Ausnahme bildet hier der experimentelle Shoppingagent ShopBot (Dorenboos u.a. 1997), der - allerdings mit zahlreichen Beschränkungen - selbständig neue Anbieter und Angebote finden kann.
[27] Man denke nur an die weitaus besseren Verkaufsaussichten für ein- und denselben, leicht verbeulten Gebrauchtwagen in Frankreich als in Deutschland.

Anbieter - diese effektiv zu verhindern wissen.[28] Denn ohne großen Aufwand können sie Mehrwerte wie Geschenkpackungen, Mengen- bzw. Kombinationsrabatte oder eine kostenlose Hotline anbieten - die Liste lässt sich beliebig fortsetzen. Der Agent kann *im besten Fall umfassend informieren* über Verfügbarkeit, Sonderangebote, Garantien, Gebühren etc. des gewünschten Produkts. Aber die Entscheidung für ein bestimmtes Produkt, und vor allem das Aufstellen von Entscheidungskriterien („was ist mir hier am wichtigsten?") bleibt die Aufgabe des Menschen.

- **Viele Güter können nicht hinreichend genau spezifiziert werden**. Für die im Kontext von Shoppingagenten beliebten CDs reicht die Angabe von Titel und Interpret zur hinreichend genauen Spezifikation aus. Dagegen umfasst die Menge notwendiger Spezifikationen bei der Buchung eines Urlaubs oder beim Kauf eines Autos immens viele Details. Oft hat eine Person, die ein Auto kaufen will, genaue Vorstellungen über Marke und Modell des Wunschfahrzeugs, kennt aber die möglichen Motorisierungs- und Ausstattungsvarianten, Farbgebungen und Extras selbst nicht. Da sie diese zum Kauf notwendigen Informationen dem Agenten nicht mitteilen kann, bleibt der mögliche Einsatzrahmen des Agenten hier auf das Einholen von Informationen beschränkt.
- **Die Domäne „Shopping" ist für Agenten nur bedingt geeignet**. Grundsätzlich lässt sich gegen Shoppingagenten anführen, dass die Kaufpräferenzen von Menschen zeitlich und situativ stark variieren, unscharf und widersprüchlich sind. Gerade durch die gedankliche Beschäftigung mit dem zu kaufenden Objekt ändern, verblassen oder konkretisieren sich die Kaufabsichten. Oft entsteht ein Kaufwunsch spontan oder erst situativ im Geschäft.

In anderen Anwendungsbereiche von Softwareagenten lassen sich in vergleichbarem Ausmaß generelle Probleme und Grenzen finden, doch soll die Domäne Shoppingagent als Beispiel hier genügen.

2.7 Zusammenfassung

Ausgehend von einer Begriffsbestimmung „Agent" wurde eine überblicksartige Einteilung von Agenten(-systemen) in Kategorien, Aspekte und Aufgaben vorgenommen. Beispielhaft ausgewählte aktuelle Agentensysteme illustrieren Funktion und Leistung, machen aber auch Schwächen dieser Technologie deutlich. Als Methode zur Problemerkennung wird eine aufgabenorientierte Analyse vorgeschlagen, als Mittel zur Überwindung der Probleme die Hauptforderungen *Lernförderlichkeit, Kontrolloptimierung* und *Vertrauensbildung* aufge-

[28] Krulwich (1998b) stellt gegen diese Argumentation eigene Erfahrungen mit seinem CD-Shoppingagenten BargainFinder. Demnach unterstützen Online-Verkäufer Shoppingagenten, weil der Vorteil der erhöhten Publizität ihres Angebots den Nachteil der Preistransparenz übersteigt. Weiterhin beobachtet Krulwich, dass der typische Online-Besucher zwei CDs erwirbt, ohne für die zweite CD einen erneuten Preisvergleich mit BargainFinder anzustellen. Daraus leitet Krulwich einen generell geringen Stellenwert der Markttransparenz ab, übersieht dabei aber, dass der Gesamtpreis für Bestellungen bei zwei verschiedenen Anbietern doppelte Versandkosten zur Folge hätte. Diese Tatsache ist dem Käufer jedoch bewusst. Gegen Krulwichs erstes Argument lässt sich einwenden, dass das Entgegenkommen der Händler nur einen momentanen Vorteil in der Anlaufphase eines Shoppingagenten darstellt. Spätestens zu dem Zeitpunkt, an dem weiter entwickelte Shoppingagenten über bessere Methoden der autonomen Anbietersuche verfügen, wird diese Möglichkeit der Einflussnahme verschwinden.

stellt. Der vorausgegangene Abschnitt hat darüberhinaus prinzipielle Grenzen der Verbesserung von Agenten aufgezeigt.

3 Human-Computer Interaction

Noch bis in die siebziger Jahre hinein waren Computersysteme große, langsame und immens teure Maschinen, die von dafür ausgebildeten Spezialisten bedient wurden. Als Computer allmählich für größere Bevölkerungsschichten erschwinglich wurden, nahmen der Anreiz und die Notwendigkeit zu, auch für Nichtspezialisten bedienbare Modelle zu entwickeln. Im Fahrwasser dieser Entwicklung erlebte eine Reihe von wissenschaftlichen Feldern, die sich mit dem Verhältnis von Mensch und Computer beschäftigen, einen Aufschwung.

3.1 Einleitung

Das letzte Kapitel beschäftigte sich mit Softwareagenten. Eine erste Analyse ergab, dass der Erfolg von Agenten von einer Verbesserung ihrer Nutzbarkeit abhängt. Es wurden die Hauptproblembereiche (1) Lernerfordernisse/Erklärungsbedarf, (2) Mittelbarkeit/Indirektheit und (3) Misstrauen identifiziert, denen die positiven Hauptanforderungen (1) Lernförderlichkeit, (2) Kontrolloptimierung und (3) Vertrauensbildung entgegengestellt wurden. Dieses Kapitel stellt nun die wesentlichen Gebiete vor, die die Gebrauchstauglichkeit von Softwaresystemen untersuchen, nämlich HCI, Softwareergonomie und Normen. Agentenrelevante Grundsätze und Kriterien, die die drei Hauptanforderungen unterstützen, werden eingehend analysiert, im Agentenkontext passend umdefiniert und durch Hypothesen präzisiert. Diese Hypothesen werden im sich anschließenden Anwendungsteil an dem Agentensystem myLAUNCH untersucht.

3.2 Geschichte der HCI

Schon die Pioniere der Datenverarbeitung wie Vannevar Bush und Alan Turing haben sich in den vierziger Jahren dieses Jahrhunderts „über das Verhältnis der Computer zur menschlichen Arbeit Gedanken gemacht“ (Hellbardt 1997). Dennoch beschäftigten sich Entwickler bis in die siebziger Jahre fast ausschließlich mit der *Hardware* und Methoden, diese kleiner, schneller und billiger zu machen. Um diese Zeit wurden allmählich neue Anforderungen an Programme gestellt: Sie sollten nicht nur das gestellte Problem lösen, sondern zusätzlich gut bedienbar sein. Erstmals wurden Begriffe wie „Benutzerschnittstelle“ und „Man-Machine Interface (MMI)“ für Systemdesigner und Forscher wichtig (Preece u.a. 1994, 7). Diese Begriffe wurden definiert als „diejenigen Aspekte des Systems, mit denen der Benutzer in Kontakt kommt“ (Moran in Preece u.a. 1994, 7). 1969 war ein bedeutendes Jahr für die HCI, denn in diesem Jahr fand die erste große HCI-Konferenz[29] statt, und das erste internationale Journal[30] wurde gegründet (Shackel 1997, 973).

Bald wurde der Softwareindustrie klar, dass eine gute Benutzerschnittstelle ein verkaufsförderndes Argument für ihre Produkte sein kann. Dies ging mit dem inflationären Gebrauch des Schlagwortes „*benutzerfreundlich*“ einher, was aber kaum mehr bedeutete als einen aufgeräumten und dadurch ästhetisch ansprechenderen Bildschirm. Auf der anderen Seite begannen sich Forscher für die Möglichkeiten und Grenzen der Menschen zu interessieren, die Computer bedienen. Zunächst beschränkte sich dieses Interesse auf psychologische Prozesse der Benutzer, erweiterte sich jedoch bald um Aspekte wie Arbeits- und Trainingsmethoden, Or-

[29] International Symposium on Man-Machine Systems
[30] International Journal of Man-Machine Studies

ganisationsformen und Gesundheitsrisiken. Diese Strömungen wurden Mitte der 80er im Begriff *Human-Computer Interaction* zusammengefasst. Damit wurde zum Ausdruck gebracht, dass nicht nur die Benutzerschnittstelle, sondern sämtliche Aspekte der Interaktion zwischen Benutzern und Computern von Interesse sind. Das damalige Selbstverständnis kommt in der Definition von Baecker u.a. in (Preece 1994, 7) von 1987 gut zum Ausdruck, dass HCI „eine Menge von Prozessen, Dialogen und Aktionen [ist], mit denen ein menschlicher Benutzer einen Computer beschäftigt und mit ihm interagiert".

Inzwischen hat sich das Verständnis dieser Wissenschaft erweitert und umfasst Design, Evaluation und Implementierung interaktiver Computersysteme für Menschen und die „Erforschung der sie umgebenden Hauptphänomene" (ACM SIGCHI, 1992). Mit dieser Erweiterung haben auch die Anleihen von HCI bei anderen Wissensgebieten stark zugenommen. Die *Hauptquellen sind Informatik, Kognitive Psychologie, Sozial- und Organisationspsychologie* und *Ergonomie*. Weitere Beiträge kommen aus Gebieten wie Anthropologie, Design, Industriedesign, Ingenieurswesen, Künstliche Intelligenz, Linguistik, Medizin, Philosophie, Physiologie und Soziologie (Shackel 1997, 970; Preece u.a. 1994, 38). Somit ist HCI heute stark interdisziplinär geprägt.

Für die Zukunft ist mit einer weiter zunehmenden Bedeutung der HCI zu rechnen. Da immer weitere Bevölkerungskreise Computer benutzen (oder benutzen müssen), wird der Bedarf nach für „jedermann" bedienbarer Soft- und Hardware wachsen. Außer den offensichtlichen HCI-Wachstumsgebieten Internet und PC nennt (Shackel 1997, 983) als zukunftsträchtige Bereiche unter anderem Smartcards, Computersicherheit, theoretische HCI und Virtual-Reality-Systeme.

3.3 Ausgangspunkt der Anpassung: Identifikation agentenrelevanter Gestaltungsgrundsätze in HCI-Gebieten und Normen

Nun werden wichtige Gebiete der HCI vorgestellt und ihre speziellen Werkzeuge, im wesentlichen *Gestaltungsgrundsätze und Kriterien*, auf die Anwendbarkeit in Agentensystemen untersucht.[31]

3.3.1 HCI

Eine Übersicht über grundlegende HCI-Gestaltungsgrundsätze zeigt Tabelle 3, die sich im wesentlichen nach den von Donald Norman, einem Pionier des nutzerzentrierten Designs, aufgestellten Gestaltungsgrundsätzen richtet (Norman 1988). Hinzugefügt habe ich das neuere Progressive Disclosure (vgl. Apple 1993), weil es zugleich die Forderungen nach Lernförderlichkeit und Kontrolloptimierung unterstützt und deshalb im Agentenkontext wesentlich ist.

[31] An dieser Stelle ist eine Unterscheidung zwischen „weichen" Gestaltungsgrundsätzen und „harten" Kriterien sinnvoll. Es ist unstrittig, dass in der HCI *Gestaltungsgrundsätze* aufgestellt werden. Bei Normen wie der DIN 66 234, Teil 8 ist sogar „ausdrücklich von 'Grundsätzen' der Dialoggestaltung" (Dzida 1994, 287) die Rede. Im Bereich der Softwareergonomie dagegen stellt Dzida eine verbreitete, aber missverständliche Verwendung des Begriffs „Kriterium" für Gestaltungsgrundsätze fest - eine Bezeichnung, die nach der Aufstellung konkretisierter, messbarer Anforderungen verlangt. Aus Gründen der Einheitlichkeit spricht diese Arbeit deshalb stets von Gestaltungsgrundsätzen. Diese werden erst dann als Kriterien bezeichnet, wenn sie von geeignet formulierten, mess- und überprüfbaren Forderungen begleitet werden.

Tabelle 3: Wichtige HCI- Gestaltungsgrundsätze[32]

Visibility	Visibility ist der Grundsatz, jederzeit erkennen zu können, in welchem Zustand sich ein System befindet. Weiterhin müssen sämtliche augenblicklichen Eingabemöglichkeiten sichtbar sein.
Affordances	Affordances ermuntern den Benutzer, automatisch die augenblicklich „richtige“ Eingabe zu tun. Objekteigenschaften wie Form, Farbe oder Plazierung geben dem Benutzer unterschwellige Signale, welche Aktionen sinnvoll sind.
Constraints	Constraints sind das Gegenteil von Affordances, nämlich bewusste Beschränkungen, die dem Benutzer Falscheingaben erschweren oder unmöglich machen sollen.
Mapping	Mapping beschreibt die Beziehung zwischen Manipulationsobjekten und manipulierten Objekten. Gute Mappings sind 'natürlich', haben also Analogien zwischen der Art der Eingabe und der ausgelösten Aktion.
Feedback	Feedback ist die Rückmeldung der Handlungen des Benutzers und deren Auswirkungen
Forgiveness	Ein ergonomisch gutes System verzeiht Eingabefehler und lässt sie rückgängig machen.
Conceptual Models	Vorstellungsmodelle veranschaulichen, wie Systeme funktionieren, und legen ihre unsichtbaren Funktionsabläufe offen.
Progressive Disclosure	Progressive Disclosure ist die Methode, die Komplexität von Systemen vor dem ungeübten Benutzer zu verbergen, ihm stattdessen nur die wesentlichen Eingabemöglichkeiten zu präsentieren. Erst später wird die gesamte Funktionsvielfalt, u.U. in mehreren Schritten, dem nun kompetenteren Benutzer „enthüllt“ und zugänglich gemacht.

Agentensysteme können dem Benutzer zunächst fremd und undurchschaubar erscheinen, verlangen sie doch von ihm ein Umdenken hinsichtlich seines Verständnisses, auf welche Weise ihm Anwendungen bei der Arbeit helfen: Während werkzeugbasierte Hilfe nur auf die Initiative des Benutzers erfolgt, kann agentenbasierte Hilfe auch vom System initiiert werden (siehe Thomas 1996, 3). Traditionelle Software mit direkt-manipulativer Oberfläche war passiv, wurde wie ein Werkzeug benutzt. Softwareagenten dagegen verhalten sich wie Assistenten, können proaktiv handeln, nichtdeterministisch und lernfähig sein. Aus diesen Gründen ist die Forderung von **Visibility** (auch Transparenz genannt), Systemzustand und Eingabemöglichkeiten jederzeit erkennen zu können, für Agentensysteme von größter Wichtigkeit. Es wäre jedoch ein Fehler, Visibility allein mit visuellen Mitteln erreichen zu wollen. Auch mit akustischen und haptischen Mitteln lässt sich die Systemtransparenz entscheidend unterstützen. In der Praxis ist es allerdings schwierig, den Zustand eines mobilen Agenten, der sich autonom im Netzwerk bewegt, plastisch zu machen, vor allem, wenn der Agent über einen langen Zeitraum hinweg arbeitet. Schwierig ist auch die übersichtliche Darstellung des Systemzustands komplexer Agentenpopulationen.

Buttons in einer Benutzerschnittstelle tragen **Affordances** in sich, ermuntern zum Beispiel zum drücken, nicht aber zum verschieben (wie Schieberegler) oder drehen (wie Drehregler). Gute Affordances sind in Agentensystemen wichtig, um neuen Benutzern Hinweise auf Zweck und Funktionsweise von Agenten zu geben.

Computerprogramme benutzen **Constraints** in Form von Sicherheitsabfragen, um zum Beispiel unumkehrbare Eingaben (Datei löschen, Festplatte formatieren) zu erschweren. In Agentensystemen haben Constraints dagegen die Aufgabe, den Benutzer vor dem Auslösen geld- oder bandbreitenintensiver Aktionen zu warnen. Beides, Affordances und Constraints, sind also gleichermaßen für Agentensysteme relevante Gestaltungsgrundsätze.

Dagegen hat **Mapping**, das eine große Bedeutung beim Entwurf von Schnittstellen physischer Manipulationsobjekte hat, keine überragende Bedeutung im Agentenkontext.

[32] Leider gibt es keine anerkannte deutsche Terminologie der HCI-Gestaltungsgrundsätze. Deshalb verwende ich die englischen Wendungen.

Nur eingeschränkt durchsetzbar ist die Forderung nach **Forgiveness** in Agentensystemen. In vielen herkömmlichen, nicht vernetzten Systemen können theoretisch sämtliche Aktionen umkehrbar gestaltet sein. Eingaben über eine UNDO-Funktion widerfuhr sein, gelöschte Dokumente aus dem Papierkorb gerettet werden. Der Benutzer hat die völlige Kontrolle über installierte und laufende Anwendungen in seinem System. In dynamischen Systemen mit mehreren, gleichberechtigten Akteuren wie dem WWW dagegen ist Umkehrbarkeit nur bis zu einem gewissen Grad möglich. Einmal erfolgte Transaktionen wie Käufe, Verkäufe oder der Informationsaustausch mit anderen Agenten lassen sich jedoch nur begrenzt und im Konsens mit allen anderen Beteiligten umkehren, sollen nicht ihre Rechte verletzt werden[33]. Dennoch bleibt die Forderung bestehen, alle prinzipiell *umkehrbaren* Aktionen widerrufen zu können, beispielsweise durch Versionskontrollen und Vorabexplorationen.

Feedback, Conceptual Models und Progressive Disclosure sind allesamt für Agenten wichtige Aspekte. Gerade wenn Agenten länger dauernde Aufgaben bearbeiten, ist ein kontinuierliches **Feedback** über den Fortschritt der Arbeit nötig. Im Idealfall ist Feedback:

- unmittelbar, also ohne merkliche Verzögerung.
- umfassend: informiert über *jede* getätigte Eingabe.
- vollständig: informiert über *sämtliche* Auswirkungen einer Aktion

Der Shoppingagent Jango (siehe Abschnitt 4.5) verfügt über ein sehr gutes Feedback. Geeignet umgesetztes Feedback unterstützt wiederum Visibility, was auch ein Beispiel für die gegenseitigen Abhängigkeiten der HCI-Gestaltungsgrundsätze ist.

Ein wichtiges Hilfsmittel von **Conceptual Models** sind Metaphern, mit denen Neues durch den Rückgriff auf ähnliches Altes erklärt wird. Grafische Benutzeroberflächen benutzen beispielsweise die Desktopmetapher. Genauso tragen Conceptual Models beispielsweise durch die Assistentenmetapher viel zur Veranschaulichung von Agenten bei. Die geeignete Wahl einer Agentenmetapher Abschnitt kann zu einem tieferen Verständnis des Benutzers führen.[34]

Ein nützlicher Gestaltungsgrundsatz ist auch **Progressive Disclosure**. Es erlaubt, die Benutzerschnittstelle sehr einfach zu halten, was neuen Benutzern hilft, die wichtigsten Eingabemöglichkeiten schnell zu erkennen. Gleichzeitig kann durch wiederholtes Einblenden erweiterter Optionen eine präzise Kontrolle sämtlicher Agentenkomponenten ermöglicht werden. Weiterhin können Benutzer durch *eingeblendete Programmtips* oder *Beispielläufe* zum selbständigen Explorieren und Lernen animiert werden.[35]

[33] Hier kann ein Vergleich mit der Börse hilfreich sein. An der Börse werden Aktien ausgetauscht, im WWW Informationen, Dienstleistungen und Güter. Der Verkauf von Aktien an der Börse lässt sich, zumindest meistens, durch den Kauf der gleichen Aktien wieder rückgängig machen. Doch der Preis kann inzwischen gestiegen sein, und es gibt keine Möglichkeit zum Rückkauf nach dem alten Kurs. Ähnlich verhält es sich im WWW mit Rückforderungen verkaufter Güter, und noch schwieriger ist es bei gelieferten Informationen, eine vollständige Rückgabe durchzusetzen.

[34] Zahlreiche Beispiele für Agentenmetaphern bietet der Leitfaden (Abschnitt 5.2.2.1, Tabelle 12).

[35] Zu diesem Zweck wird in den Weiterentwicklungen des bizzyB-Systems in Abschnitt 6.4 Progressive Disclosure mehrfach eingesetzt.

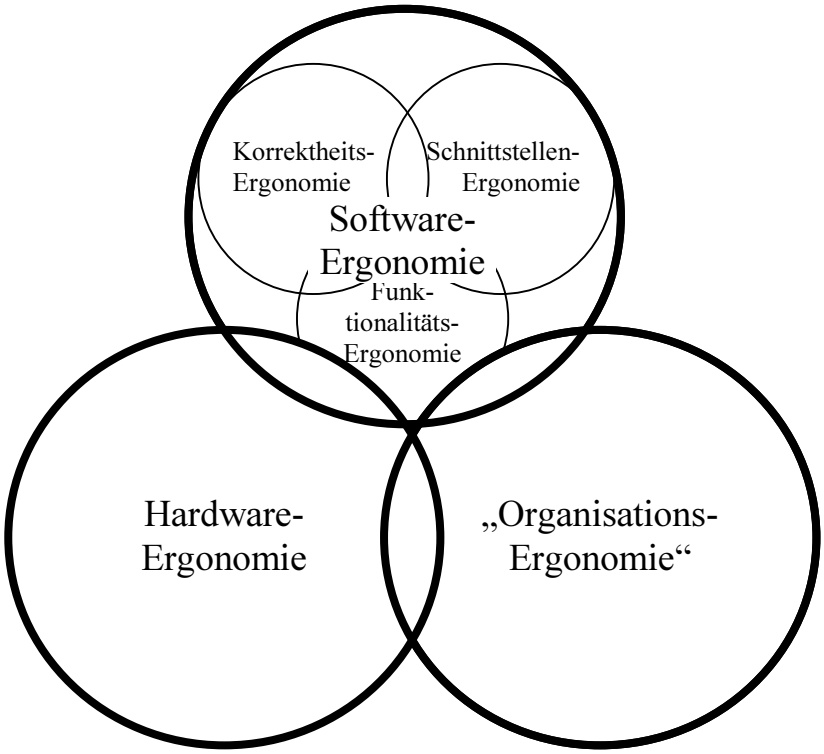


Abbildung 10: Bereiche der Ergonomie (Oppermann u.a. 1988, 5)

HCI umschließt die Bereiche Software-, Hardware- und Organisations-Ergonomie, die sich, wie Abbildung 10 zeigt, überschneiden.[36] Diese drei Bereiche untersuchen die Leistungsmöglichkeiten des *arbeitenden Menschen* und beschäftigen sich mit der Anpassung von Arbeitsmitteln und Arbeitsumgebung an die Eigenschaften und Bedürfnisse des Benutzers. Während die Hardwareergonomie technische Geräte wie Monitore und Tastaturen untersucht, entwickelt die Softwareergonomie insbesondere *Grundsätze und Methoden zur Gestaltung interaktiver Programmsysteme*. Aus diesem Grund kann die Softwareergonomie einen wichtigen Beitrag zur Untersuchung und Verbesserung von Softwareagenten leisten. Die Organisations-Ergonomie wiederum beschäftigt sich mit der Einbettung menschlicher Arbeit in den organisatorischen Kontext.

3.3.2 Softwareergonomie

Im Unterschied zur HCI, die sich auch mit technologiegesteuerten Innovationen und Machbarkeitsstudien befasst, steht bei der Softwareergonomie stets die *Gebrauchstauglichkeit* von *Arbeitssystemen* im Vordergrund. Ziel ist die möglichst *problemnahe Arbeitsweise* des Benutzers. Das Softwaresystem darf ihn also nicht daran hindern, sich ganz seiner eigentlichen Aufgabe zu widmen. Eine ausführliche Aufstellung softwareergonomischer Gestaltungsgrundsätze findet sich bei Oppermann u.a. (1992, 64-65) (siehe Tabelle 4). Sie wurde im Hinblick auf die Evaluation von Dialogsystemen erstellt, ist also potentiell nützlich für die spätere

[36] Die Anwendbarkeit von HCI-Gestaltungsgrundsätzen ist jedoch - entgegen ihres Namens - nicht auf die Interaktion von Menschen mit auf *Computern* beschränkt. So demonstriert Norman (1988; 1992) ihre Anwendung an einer Vielzahl von Gegenständen des täglichen Lebens, zeigt beispielsweise Affordances von Türklinken, Mappings bei Autositzverstellungen und Conceptual Models von Kühlschränken.

Untersuchung von Mensch-System-Dialogen in Agentensystemen (siehe Abschnitte 4.3.4 und 5.2.4). Einige Gestaltungsgrundsätze sind der Norm ISO 9241 entnommen, die im folgenden Abschnitt vorgestellt wird.

Tabelle 4: Zentrale Gestaltungsgrundsätze der Softwareergonomie (nach Oppermann u.a. 1992, 62-65)

Verfügbarkeit	„Unter *Verfügbarkeit* wird verstanden, dass der Benutzer nicht durch Störungen (z.B. Systemabstürze) oder zu lange Antwortzeiten bei der Benutzung behindert wird.“
Nützlichkeit	Der Aspekt *Nützlichkeit* verlangt im Wesentlichen, dass eine Software ihren Zweck erfüllt.
Komfort	Dieser Grundsatz bezieht sich auf die „komfortable Benutzung des Bürosystems durch den Benutzer“, das ihn bei der Arbeitsaufgabe unterstützen, nicht aber unnötig belasten soll.
Übersichtlichkeit	„Die *Übersichtlichkeit* bezieht sich auf die Darstellung und Anordnung der Informationen auf dem Bildschirm. [...] Das Ziel besteht in der Entlastung des menschlichen Kurzzeitgedächtnisses.“
Selbstbeschreibungsfähigkeit	„Ein Dialog ist *selbstbeschreibungsfähig*, wenn dem Benutzer auf Verlangen Einsatzzweck sowie Leistungsumfang des Dialogsystems erläutert werden können oder der Benutzer auf Verlangen dem jeweiligen Dialogschritt entsprechende Erläuterungen erhalten kann.“
Fehlerrobustheit	„Ein Dialog ist *fehlerrobust*, wenn trotz erkennbar fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis mit minimalem oder ohne Korrekturaufwand erreicht wird. Dazu müssen dem Benutzer die Fehler zum Zwecke der Behebung verständlich gemacht werden.“
Erlernbarkeit	Das System soll mit minimalem Aufwand *erlernbar* sein, wofür „die Reduzierung der Komplexität und Erhaltung der Konsistenz“ Voraussetzungen sind.
Individualisierbarkeit	*Individualisierbarkeit* fordert hier die Anpassbarkeit des Systems „an die individuellen Bedürfnisse und Fähigkeiten des Benutzers.“
Steuerbarkeit	„Ein Dialog ist steuerbar, wenn der Benutzer die Geschwindigkeit des Ablaufs sowie die Auswahl und Reihenfolge von Arbeitsmitteln oder Art und Umfang von Ein- und Ausgaben beeinflussen kann.“
Kooperations- und Kommunikationsförderlichkeit	Dieser Aspekt beschäftigt sich mit der Unterstützung von Gruppenarbeit und insbesondere der Kommunikation zwischen den Arbeitenden.
Datenschutz / Datensicherheit	Der rechtliche Aspekt Datenschutz umfasst Forderungen nach dem Schutz des Arbeitenden vor Leistungs- und Verhaltenskontrolle und nach der Kontrolle des Arbeitenden über die Weitergabe bestimmter Daten. Der technische Aspekt Datensicherheit fordert beispielsweise die Vermeidung von Datenverlusten.

Komfort ist ein für Softwaresysteme wesentlicher Gestaltungsgrundsatz, weil er nach Oppermann (1992, 64) Aspekte der *Aufgabenangemessenheit* enthält, welches „als eines den übrigen Kriterien - mit Ausnahme der Erlernbarkeit - übergeordnetes Kriterium angesehen [wird]“ (Triebe nach Oppermann 1992, 64). Besondere Bedeutung für Agenten gewinnt Komfort hier zusätzlich durch die im Vergleich zu konventionelleren Anwendungen vorherrschenden Komfortdefizite von Softwareagenten.

Selbstbeschreibungsfähige und *in minimaler Zeit bzw. einem Minimum an Operationen zu erlernende* Agentensysteme sind für eine reibungslose Gewöhnungsphase agenten-unerfahrener Benutzer wichtig, unterstützen also die Forderung nach Lernförderlichkeit. Die Aspekte **Erwartungskonformität, Steuerbarkeit** und **Datenschutz** sind weiterhin unerlässlich für den Aufbau von Vertrauen beim Benutzer, was wiederum eine Voraussetzung für die Delegation von Aufgaben an den Agenten ist. Datenschutz ist insbesondere wichtig, wenn beispielsweise ein Shoppingagent, mit Geld oder Kreditkartendaten versehen, Aufträge ausführen soll. Diese Daten müssen so verschlüsselt sein, dass ein „Aushorchen“ des Agenten durch fremde

Programme unmöglich wird. **Nützlichkeit, Übersichtlichkeit, Individualisierbarkeit** sowie **Kooperationsförderlichkeit** sind bei Agenten genauso wichtig wie bei herkömmlicher Software.

Problematisch dagegen ist die Forderung nach **Verfügbarkeit** des Agenten, die sich im WWW bewegen. Denn dort können Faktoren wie Datendurchsatz und Geschwindigkeit entfernter kooperierender Komponenten gleichermaßen unvorhersehbar wie entscheidend für die Laufzeit des Agenten sein. Auch eine völlige **Fehlerrobustheit** ist nicht erreichbar, man denke nur an das Auffinden mobiler Agenten oder die Aufgabe, vom Agenten fälschlich getätigte finanzielle Transaktionen wieder rückgängig zu machen.

3.3.3 Normen (DIN-Norm 66 234 und ISO-Norm 9241)

Aus mehreren Gründen bietet sich eine Untersuchung softwareergonomischer Normen an: Ihre Gestaltungsgrundsätze sind immer auf Arbeitssysteme bezogen, sind also von daher potentiell geeignet für einen auf Arbeitssysteme beschränkten Leitfaden (nach Abschnitt 2.5). Auch wenn ergonomischen Normen die Quantifizierbarkeit und Präzision technischer Normen fehlt (Dzida 1993, 1), haben sie dennoch eine normative Komponente. Das ist wichtig für eine spätere Operationalisierung der aus ihnen im Agentenkontext zu gewinnenden Kriterien. Normen kann man sich als das geronnene Ergebnis softwareergonomischer Forschung und politischen Konsenses vorstellen. Bezüglich Operationalisierbarkeit erweisen sich softwareergonomische Normen - wie sich zeigen wird - geeigneter als HCI und Softwareergonomie. Von diesen Normen möchte ich nun zwei wichtige Vertreter kurz vorstellen.

3.3.3.1 DIN-Norm 66 234 Teil 8

Als erste deutschsprachige softwareergonomische Norm entstand zwischen 1977 und 1988 die DIN-Norm 66 234 Teil 8, „Grundsätze ergonomischer Dialoggestaltung“ (vgl. Rödiger 1991, 13). Tabelle 5 stellt ihre fünf Gestaltungsgrundsätze für ergonomische Software vor, nämlich **Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität** und **Fehlerrobustheit**. Ihre Verwandtschaft mit der Softwareergonomie wird durch die teilweise (inhaltliche wie namentliche) Anlehnung an Gestaltungsrichtlinien der Softwareergonomie deutlich.

Tabelle 5: Wichtige Gestaltungsgrundsätze der DIN-Norm 66 234 Teil 8 (Rödiger 1991, 15-19)

Aufgaben-angemessenheit	„Ein Dialog ist aufgabenangemessen, wenn er die Erledigung der Arbeitsaufgabe des Benutzers unterstützt, ohne ihn durch Eigenschaften des Dialogsystems unnötig zu belasten.“
Selbstbeschreibungsfähigkeit	„Ein Dialog ist selbstbeschreibungsfähig, wenn dem Benutzer auf Verlangen Einsatzzweck sowie Leistungsumfang erläutert werden können und wenn der einzelne Dialogschritt unmittelbar verständlich ist oder der Benutzer auf Verlangen dem jeweiligen Dialogschritt entsprechende Erläuterungen erhalten kann.“
Steuerbarkeit	„Ein Dialog ist steuerbar, wenn der Benutzer die Geschwindigkeit des Ablaufs sowie die Auswahl und Reihenfolge von Arbeitsmitteln oder Art und Umfang von Ein- und Ausgaben beeinflussen kann.“
Erwartungs-konformität	„Ein Dialog ist erwartungskonform, wenn er den Erwartungen der Benutzer entspricht, die sie aus Erfahrungen mit bisherigen Arbeitsabläufen oder aus der Benutzerschulung mitbringen sowie den Erfahrungen, die sie sich während der Benutzung des Dialogsystems und im Umgang mit dem Benutzerhandbuch bilden.“
Fehlerrobustheit	„Ein Dialog ist fehlerrobust, wenn trotz erkennbar fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis mit minimalem oder ohne Korrekturaufwand erreicht wird. Dazu müssen dem Benutzer die Fehler zum Zwecke der Behebung verständlich gemacht werden.“

An dieser Stelle soll nun auf die DIN-Norm 66 234 Teil 8 nicht weiter eingegangen werden, da ihre Gestaltungsgrundsätze inzwischen in der neueren und im folgenden Abschnitt vorgestellten ISO-Norm 9241 aufgegangen sind.

3.3.3.2 ISO-Norm 9241

Die Norm ISO 9241 hat ebenso wie die bekannte ISO 9000-Reihe eine *qualitätssichernde Ausrichtung*, ist aber auf Bildschirmarbeit bezogen. Sie trägt den Titel „Ergonomic Requirements for Office Work with Visual Displays“ und enthält umfangreiches softwareergonomisches Wissen. Die ISO Norm 9241 ist der internationale Nachfolger der DIN-Norm 66 234. Dessen im vorausgegangenen Abschnitt vorgestellter Teilbereich ging in Teil 10: Dialogprinzipien der ISO-Norm auf. Ein anderer Teil thematisiert die Präsentation von Information (Teil 12). Bezogen auf das „look and feel“ von Programmen beschreibt Teil 12 den „look“, also statische Elemente, während Teil 10 das dynamische „feel“ enthält. Wieder andere Teile (13-17) sind Spezialisierungen der in Teil 10 genannten Dialogprinzipien. Für die Interaktion mit Agenten ist hier Teil 10 wesentlich, dessen Prinzipien[37] in Tabelle 6 zusammengefasst sind.

[37] Neben den gängigen deutschen Bezeichnungen der Gestaltungsrichtlinien habe ich die offiziellen englischen Begriffe aufgeführt.

Tabelle 6: Prinzipen des Dialogs (ISO-Norm 9241 Teil 10)

Aufgabenangemessenheit (suitability for the task)	Ein Dialog ist in dem Maße aufgabenangemessen, wie er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.
Selbstbeschreibungsfähigkeit (self-descriptiveness)	Ein Dialog ist in dem Maße selbstbeschreibungsfähig, wie jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer erklärt wird, wenn er die entsprechenden Informationen verlangt.
Steuerbarkeit (controllability)	Ein Dialog ist in dem Maße steuerbar, wie der Benutzer in der Lage ist, den gesamten Dialogablauf bis zu dem Punkt, an dem das Ziel erreicht ist, zu beeinflussen.
Erwartungskonformität (conformity with user expectations)	Ein Dialog ist in dem Maße erwartungskonform, wie er den Kenntnissen aus bisherigen Arbeitsabläufen, der Ausbildung und der Erfahrung des Benutzers sowie den allgemein anerkannten Übereinkünften entspricht.
Fehlerrobustheit (error tolerance)	Ein Dialog ist in dem Maße fehlerrobust, wie das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben mit minimalem oder ohne Korrekturaufwand erreicht wird.
Individualisierbarkeit (suitability for individualization)	Ein Dialog ist in dem Maße individualisierbar, wie er Anpassungen an individuelle Benutzerbelange und Benutzerfähigkeiten im Hinblick auf eine gegebene Arbeitsaufgabe zuläßt.
Lernförderlichkeit (suitability for learning)	Ein Dialog ist in dem Maße lernförderlich, wie er dem Benutzer während des Erlernens Unterstützung und Anleitung gibt.

Gewisse Ähnlichkeiten mit den Gestaltungsgrundsätzen der Softwareergonomie sind unverkennbar. **Aufgabenangemessenheit** und **Selbstbeschreibungsfähigkeit** sind im Agentenkontext aus denselben Gründen wie die gleichlautenden Gestaltungsgrundsätze der DIN 66234 relevant.

Der Punkt **Aufgabenangemessenheit** umfasst die Aspekte Funktionalität und Nützlichkeit. Funktionalität setzt voraus, dass der Benutzer keinesfalls unnötig belastet wird und zum Beispiel nötige Informationen in der richtigen Reihenfolge erhält. Nützlichkeit bezieht sich auf den Unterstützungseffekt des Benutzers bei seiner Arbeit.

Die Definition von **Selbstbeschreibungsfähigkeit** geht auf die Wichtigkeit der **Rückmeldung**, des Feedback, ein. Gerade Agentensysteme sollten ein umfassendes Feedback bieten, um für den Benutzer verständlich und durchschaubar zu sein.

Aspekte der **Steuerbarkeit -** zum Beispiel die Geschwindigkeit oder der Verlauf des Dialogs - sollten nicht vom Dialogsystem vorgeschrieben werden, sondern immer unter Kontrolle des Benutzers stehen. Und Optimierung der Kontrolle ist ja eine der Hauptforderungen von Agentensystemen. Es ist eine verständliche Forderung, Agentensysteme, deren Verhalten noch nicht restlos verstanden oder erklärbar ist, besser kontrollieren zu können. Sicherheitskritische, teure oder unumkehrbare Aktionen sollten sogar restlos kontrollierbar sein. Einschränkend muss gesagt werden, dass bei steigender Komplexität der Aufgabe eine detaillierte Kontrolle durch den Benutzer immer weniger durchführbar wird. Wichtig ist aber die *Möglichkeit*, sämtliche Funktionen des Agenten bei Bedarf kontrollieren zu können.

Auch **Erwartungskonformität** ist eine im Agentenkontext wichtige Forderung und eine der Voraussetzungen für die Hauptforderung nach Vertrauensbildung. Selbstverständlich sollten generell Dialogverhalten und Informationsdarstellung von Softwaresystemen *einheitlich* sein. Auch sollten unregelmäßige Antwortszeiten minimiert und durch kontinuierliches Feedback verständlich gemacht werden.

Das System sollte verhindern, dass irgendeine Benutzereingabe zu undefinierten Systemzuständen oder zu Systemabbrüchen führen kann. **Fehlerrobustheit** ist - wie schon beim

gleichlautenden Softwareergonomie-Grundsatz erwähnt - eine wichtige Forderung. Allerdings kann insbesondere bei WWW-Agenten eine Fehlerrobustheit nie garantiert werden. In einem Netzwerk, in dem Datenstaus und verlorengegangene Datenpakete alltägliche Probleme sind, unterliegen auch Agenten diesen Unwägbarkeiten.

Individualisierbarkeit ist kein rein auf den Agentenkontext beschränkter Gestaltungsgrundsatz. Er verlangt nach der Möglichkeit, nach individuellen Gesichtspunkten zwischen alternativen Darstellungsformen oder Arbeitsabläufen wählen zu können.

Sehr wichtig ist dagegen die mit der dritten Hauptforderung gleichnamige **Lernförderlichkeit**, die im vorletzten Abschnitt unter derselben Definition bereits bei der Softwareergonomie (unter dem Namen „Erlernbarkeit“) als besonders wichtige Gestaltungsrichtlinie erkannt wurde. Das System soll *mit minimalem Zeitaufwand erlernbar* sein, wofür „die Reduzierung der Komplexität und Erhaltung der Konsistenz“ Voraussetzungen sind.

Die DIN-Norm 66 234 wurde von der ISO-Norm 9241 vollständig umschlossen und ersetzt. Diese ist auch neuer als die bereits 1988 verabschiedete DIN-Norm: Ihre meisten Teile sind bereits endgültig als Industriestandard verabschiedet, einige Teile werden noch bearbeitet. Zu einer Zeit, in der Normen wie die ISO 9000-Reihe in der Industrie stark an Bedeutung gewinnen, steigen auch der Bekanntheitsgrad und das Ansehen der ISO-Norm 9241. Dass sie auch Anhaltspunkte zur Operationalisierbarkeit bietet, zeigen beispielsweise der EVADIS-Leitfaden oder der ISO 9241-Evaluator, welche Softwaresysteme einer Konformitätsprüfung hinsichtlich dieser Norm unterziehen. Es ist möglich, mit Hilfe dieses Evaluationsverfahrens Software bereits während der Entwicklung auf Konformität zu testen (Oppermann 1992, 13).

Sicherlich lassen sich auch in anderen Bereiche Methoden zur Untersuchung von Agenten finden. An weiteren relevanten Normen wäre beispielsweise die Qualitätsnorm ISO 8402 zu nennen. Zahlreiche Styleguides beschäftigen sich mit ergonomischer Programmgestaltung, wie die renommierten Apple Style Guides (Apple 1993, 1997), denen der Gestaltungsgrundsatz Progressive Disclosure entnommen ist. Auch rein auf das WWW bezogene Styleguides bieten eine Fülle von Hinweisen, die für im WWW operierende Agenten wertvoll sein können. Doch auf eine ausführliche Betrachtung weiterer Bereiche soll an dieser Stelle verzichtet werden. Denn die untersuchten Bereiche decken bereits ein breites Methodenspektrum ab, und ihre als agentenrelevant identifizierten Methoden stellen ein ausreichend solides Fundament zur Evaluation und insbesondere Verbesserung der Mensch-Agenten-Schnittstelle von Agentensystemen dar. Eine Untersuchung weiterer Bereiche würde wenig neue, sondern einen immer höheren Anteil redundanter Methoden liefern. Weisen doch bereits die Gestaltungsrichtlinien der untersuchten Normen erhebliche Überlappungen untereinander sowie mit softwareergonomischen Grundsätze auf.

3.4 Vergleich vorgestellter Gebiete auf Agentenrelevanz

Es folgen grundsätzliche Überlegungen zur Eignung der bisher vorgestellten Gebiete, agentenrelevante Kriterien zu liefern.

Ein unbestreitbarer Vorteil gerade internationaler Normen ist ihr *hohes Ansehen*, das wie im Fall der ISO-Norm 9241 zu breiter Akzeptanz und Anwendung geführt hat. Andererseits entwickeln sich Normen nur langsam, erstarren und veralten somit schnell. Sowohl DIN-Norm 66 234 als auch ISO-Norm 9241 wurden vor dem Durchbruch von Agentensystemen ausgear-

beitet. Offenbar hatten die Mitglieder der Normungsausschüsse *konventionelle Programme* vor Augen, *nicht aber Agentensysteme*. Deshalb erwähnt keine einzige ihrer Ergonomierichtlinien Agentensysteme oder geht auf diesen Programmtyp ein. Eine Folge ist ihr in meiner Untersuchung aufgetretene geringer Anteil an der Gesamtheit agentenrelevanter Richtlinien.

Doch das Haupthindernis agentenspezifischer Normen ist der *Allgemeingültigkeitsanspruch* vieler Normen. So nimmt die ISO-Norm 9241 für sich lediglich einen *Richtliniencharakter* in Anspruch und geht bewusst *nicht* auf eine bestimmte technische Implementation ein (Dzida 1993, 2). Dadurch kann sie Agenten, die nicht nur Schnittstelle sind, sondern insbesondere auch eine eigenständige Technologie darstellen, nicht vollständig erfassen. Aber auch bei einer Beschränkung auf eine Schnittstellensicht müssen Aspekte von Normen stärker auf den Agentenkontext angepasst werden.

Dagegen verwandeln sich die praxisorientierten und vergleichsweise „weichen" Gestaltungsgrundsätze der HCI in einen Vorteil bei der Untersuchung und Verbesserung von Agentensystemen. Denn sie haben eine größere Flexibilität und sind dadurch oft auf mehrere der Hauptforderungen von Agentensysteme gleichzeitig anwendbar.

Die softwareergonomischen Gestaltungsgrundsätze liegen bezüglich ihrer Relevanz für Agenten auf halbem Weg zwischen HCI und Normen, und gut die Hälfte der untersuchten Grundsätze weist einen Agentenbezug auf.

Folglich verfügen sämtliche vorgestellten Gebiete über Grundsätze, die sich als „Rohmaterial" für agentenspezifische Kriterien eignen. Da die Gestaltungsgrundsätze in ihrer jetzigen Form nicht oder nicht ausreichend die besonderen Anforderungen von Softwareagenten berücksichtigen, muss im nächsten Abschnitt eine agentenspezifische Anpassung erfolgen.

3.5 Transformation von HCI-Gestaltungsgrundsätzen zu einem agentenspezifischen Kriterienkatalog

Die agentenrelevanten Gestaltungsgrundsätze werden nun agentenspezifisch analysiert und operationalisiert. Das Ergebnis ist ein *Katalog von Kriterien*, der operationalisierte *Hypothesen* enthält, Neuformulierungen der Gestaltungsgrundsätze unter besonderer Berücksichtigung der in Kapitel 2 aufgestellten Hauptforderungen Lernförderlichkeit, Kontrolloptimierung und Vertrauensbildung. Der sich daran anschließende Abschnitt 3.6 stellt diese Kriterien systematisch dar und zeigt Evaluationsmöglichkeiten auf.

3.5.1 Lernförderlichkeit

Der ISO 9241-Grundsatz Lernförderlichkeit sagt aus, dass „Ein Dialog [...] in dem Maße lernförderlich [ist], wie er dem Benutzer während des Erlernens Unterstützung und Anleitung gibt" (ISO 9241). Ein System sollte mit minimalem Zeitaufwand erlernbar sein, wofür die Reduzierung der Komplexität und die Erhaltung der Konsistenz Voraussetzungen sind.

Methoden zur Erhöhung der Lernförderlichkeit in Softwaresystemen sind beispielsweise spezielle Lernmodi und Hilfefunktionen. Aber auch weniger explizite Techniken wie Progressive Disclosure verbessern durch anfängliches Verbergen von Systemfunktionalität die benutzergerechte Dosierung der Lernerfordernisse. Ebenso ist Selbstbeschreibungsfähigkeit hier ein wichtiges Mittel, um Software „selbsterklärend" und mit minimalem Aufwand erlernbar zu gestalten.

Lernförderlichkeit berücksichtigt exakt die Forderungen der gleichnamigen ersten Hauptforderung, weshalb sie ein im Agentenzusammenhang wichtiger Gestaltungsgrundsatz ist. Durch Lernförderlichkeit wird für computerunerfahrene oder agentenunerfahrene Benutzer die Lernphase verkürzt und die Eingewöhnungsphase erleichtert.

Für Agentensysteme stelle ich deshalb die Hypothese auf:

> **Lernförderlichkeit ist das Ausmaß, in dem die Belange agentenunerfahrener Benutzer berücksichtigt sind und der Agent das Erlernen des Systems fördert.**

Dabei liegt der Schwerpunkt weniger auf der Dialoggestaltung als auf der Vermittlung von Einsatzzweck und Konzepten des Agenten und den Unterschieden in ihrem Verhalten gegenüber anderen Systemen.

Operationalisieren lässt sich Lernförderlichkeit mit Hilfe von Fragen wie

- Deckt die Hilfefunktion die gesamte Funktionalität der Anwendung ab?
- Berücksichtigt die Schnittstelle speziell agentenunerfahrene Benutzer?
- Existieren unterschiedliche Zugänge zur Systemhilfe, die die Verschiedenheit der Fragebedürfnisse und Kompetenzstufen des Benutzers berücksichtigen? - Eine Liste verschiedener Hilfsmethoden findet sich in Abschnitt 2.3.3.5.
- Wurden die Aspekte Selbstbeschreibungsfähigkeit, Conceptual Model und Progressive Disclosure (siehe entsprechende Abschnitte) in einer Weise berücksichtigt, die den Benutzer zum explorativen Erlernen der Anwendung animieren?

3.5.2 Komfort

Dieser Gestaltungsgrundsatz bezieht sich nach Oppermann u.a. (1992, 64) auf eine „komfortable Benutzung des Bürosystems durch den Benutzer“, das ihn bei der Arbeitsaufgabe unterstützen, aber nur minimal belasten soll. Es umfasst Teilaspekte der in DIN 66 234 Teil 8 enthaltenen Forderung nach Aufgabenangemessenheit: „Ein Dialog ist aufgabenangemessen, wenn er die Erledigung der Arbeitsaufgabe des Benutzers unterstützt, ohne ihn durch Eigenschaften des Dialogsystems unnötig zu belasten.“. Aus der herausragender Bedeutung der Aufgabenangemessenheit (siehe Abschnitt 3.3.2) erklärt sich die übergeordnete Stellung des Komfortaspekts.

Die Notwendigkeit von Komfort ergibt sich direkt aus dem Anspruch von Agentensystemen, einen *einfacheren* Zugang zu Softwaresystemen zu bieten. Konsequenterweise sollte sich dieser Komfortgedanke auch durch die Gestaltung der Benutzerschnittstelle ziehen. Eine komfortable Kontrolle über den Agenten verlangt beispielsweise, jederzeit in seinen Handlungsablauf eingreifen zu können. Eine komfortable Lernfunktion setzt das Ziel Lernförderlichkeit um.

Meine Hypothese für Komfort bei Agentensystemen lautet deshalb:

> **Komfort ist das Ausmaß, in dem der Benutzer bei seiner Aufgabe und bei der Interaktion mit dem System von einem Agenten unterstützt wird.**

Sie greift den Anspruch auf, mit Agenten (a) sehr vieles auf einfache Weise und (b) spezielle Aufgaben besonders gut zu erledigen. Die Definition berücksichtigt weiterhin die vielfach

schlechte Benutzbarkeit von Agentensystemen und bietet eine Möglichkeit, bei gleichbleibender Agenten-Funktionalität eine verbesserte Benutzbarkeit zu registrieren.

Einige Möglichkeiten, Komfort zu operationalisieren sind:

- Der Zeitfaktor wird gemessen, um den sich - bei gleichen Resultaten - eine Aufgabe durch den Einsatz von (unterschiedlichen) Agenten verkürzt.
- Anzahl und Komplexität der nötigen Interaktionschritte mit und ohne Agent wird gemessen.
- Der Qualitätssprung bei derselben Arbeitsaufgabe, wenn der Benutzer sie mit (unterschiedlichen) Agenten anstatt manuell erledigt.
- Testbenutzer tragen ihre subjektiven Eindrücke in einer Skala ein.

3.5.3 Erwartungskonformität

Die ISO-Norm 9241 definiert Erwartungskonformität wie folgt: „Ein Dialog ist in dem Maße erwartungskonform, wie er den Kenntnissen aus bisherigen Arbeitsabläufen, der Ausbildung und der Erfahrung des Benutzer sowie den allgemein anerkannten Übereinkünften entspricht."

Dialogverhalten und Informationsdarstellung sollten innerhalb eines Dialogsystems einheitlich sein. Änderungen des Dialogzustandes sollten auf einheitliche Art und Weise herbeigeführt werden. Brüche in der Erwartungskonformität führen zu Bedienungsfehlern und in Konsequenz zu einem generellen Misstrauen des Benutzers in das System.

Auf Agenten bezogen unterstützt Erwartungskonformität alle drei aufgestellten Forderungen: Verhält sich ein System erwartungskonform, kann es schneller erlernt werden, weil der Benutzer beispielsweise vom Systemverhalten in einer bestimmten Situation auf das Systemverhalten in ähnlichen Situationen schließen kann. Weiterhin übt der Benutzer eine größere Kontrolle über ein sich konsistent verhaltendes System aus als über eines mit unvorhersagbaren Reaktionen. Erwartungskonformität bestärkt das Vertrauen des Benutzers in das System, während ein Mangel daran - wie bereits angesprochen - zu Misstrauen führt. Conceptual Models sind das mächtigste Mittel, das isoliert ganze Systeme von Erwartungen beim Benutzer auszulösen vermag. Umso wichtiger ist also bei Agentensystemen die Wahl eines geeigneten Conceptual Models - nämlich die Wahl des Erscheinungsbilds des Agenten.

Für Agentensysteme stelle ich deshalb die Hypothese auf:

> **Erwartungskonformität ist das Ausmaß, in dem das Verhalten des Agenten dem durch seine Erscheinungsform beim Benutzer suggerierten Verhalten entspricht.**

Das bedeutet, dass der Agent sich entweder „agentenkonform" oder aber seinem Erscheinungsbild entsprechend verhalten soll. Zum Beispiel sollte das Erscheinungsbild Hund Rückschlüsse auf Aufgaben und Fähigkeiten des Agenten zulassen. Sollte es für die Interaktion vorteilhaft sein, kann auf einen Agenten-Charakter auch ganz verzichtet werden. Zu „agentenkonform" ist die Forderung zu ergänzen, dass im WWW operierende Agenten eventuelle unregelmäßige Antwortzeiten dem Benutzer durch umfassendes Feedback verständlich und transparent machen.

Ähnlich schwer wie die Operationalisierung des wichtigsten verwandten Kriteriums Conceptual Model ist die Operationalisierung von Erwartungskonformität, da die tatsächlichen indi-

viduellen Erwartungen des Benutzers an die Anwendung nicht vorhersehbar sind. In diesem Fall bietet sich allenfalls eine Nutzerbefragung vor und nach Einsatz der Anwendung an.

3.5.4 Selbstbeschreibungsfähigkeit

„Ein Dialog ist in dem Maße selbstbeschreibungsfähig, wie jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer erklärt wird, wenn er die entsprechenden Informationen verlangt." (ISO 9241)

Der Benutzer soll über Änderungen im Zustand des Dialogsystems, die wichtig für seine Aufgabe sind, informiert werden Beispiele für Selbstbeschreibungsfähigkeit sind „sprechende" Fehlermeldungen, die anstatt einer wenig aussagekräftigen Fehlernummer eine Beschreibung des Fehlers liefern, verbunden mit der vermuteten Ursache und einer möglichen Abhilfe. Auch Kontexthilfe[38] und -beschreibung[39], die die normale Hilfefunktion vieler Softwaresysteme ergänzen, gehören dazu.

Auf Agentensysteme angewendet unterstützt Selbstbeschreibungsfähigkeit die Forderung nach Lernförderlichkeit. Der Aspekt verlangt nach einer sich selbst erklärenden Oberfläche, guten und vollständigen Hilfefunktionen sowie einem ebensolchen Feedback. Für Agenten mit länger dauernden Aufgaben muss ein kontinuierliches oder regelmäßiges Feedback angeboten werden.

Für Agentensysteme stelle ich deshalb die Hypothese auf:

> **Selbstbeschreibungsfähigkeit ist das Ausmaß, in dem ein Agent ein vollständiges und kontinuierliches Feedback sowie Hilfestellungen anbietet.**

Dabei ist mit „vollständigem" Feedback eine Rückmeldung gemeint, die Zustand, Tätigkeit, Aufenthaltsort und andere relevante Punkte mit einschließt.

Folgende Fragen helfen bei der Operationalisierbarkeit des Kriteriums:

- Inwieweit geben Verhalten und Oberfläche des Agenten Hinweise auf dessen Fähigkeiten und Funktionen? Und inwiefern besteht zumindest die *Möglichkeit*, Erklärungen zu erhalten?
- Wie umfassend sind die Hilfefunktionen?
- Ist das Feedback vollständig und selbsterklärend? Welche Bereiche werden abgedeckt?

3.5.5 Steuerbarkeit

Steuerbarkeit ist ein softwareergonomischer Gestaltungsgrundsatz, der nach Oppermann u.a. (1992, 65) folgendermaßen definiert ist:

„Ein Dialog ist steuerbar, wenn der Benutzer die Geschwindigkeit des Ablaufs sowie die Auswahl und Reihenfolge von Arbeitsmitteln oder Art und Umfang von Ein- und Ausgaben beeinflussen kann."

---

[38] Gemeint ist der Modus, in dem das Anwählen von Bildschirmelementen nicht die normale Funktion auslöst, sondern eine Erläuterung der Funktionsweise erscheint.

[39] Immer mehr Systeme zeigen, sobald der Mauszeiger über einem Bildschirmelement verharrt, eine Kurzbeschreibung des Elements in einem kleinen gelben Rechteck an. (Auf Macintosh-Rechnern wird diese Funktion „Bubble-Help" genannt.)

Hier ist also zum einen eine Flexibilität des Systems bezüglich der Arbeitsweise des Benutzers vonnöten, zum anderen (und viel wesentlicher für Agenten) werden starke Kontrollmöglichkeiten von Seiten des Benutzers gefordert. Kontrollierbarkeit ist aber die zweite Hauptforderung an Agentensysteme, weshalb Steuerbarkeit im Agentenkontext gut anwendbar ist. Auch der dritten Hauptforderung, Vertrauensbildung, kann Steuerbarkeit entsprechen. Hinter der Forderung nach Kontrolle steht häufig ein Mangel an Vertrauen gegenüber dem nur teilweise verstandenen Agenten. Durch starke Kontrollmöglichkeiten (auch die UNDO-Funktion gehört dazu) verringert sich die Furcht vor Falscheingaben und unverständlichen Aktionen des Agenten. Und damit wird die Akzeptanz von Agenten insgesamt erhöht. Sicherlich können nicht *alle* Aktionen von WWW-Agenten kontrollierbar gemacht werden - eine einmal abgeschickte E-Mail kann auch nicht mehr zurückgeholt werden. Trotzdem ist eine weitestgehende *Möglichkeit* der Kontrolle von Agenten eine berechtigte Forderung, baut sie doch auch die Mittelbarkeit zwischen Benutzer und Agent ab.

Für Agentensysteme stelle ich in diesem Licht folgende Hypothese auf:

> **Steuerbarkeit ist das Ausmaß, in dem die gesamte Interaktionsgestaltung mit dem Agenten durch den Benutzer kontrolliert und flexibel gestaltet werden kann.**

Unter „kontrollierbar“ ist hier „einzeln vom Benutzer ausführbar“ und „umkehrbar“ zu verstehen.

Die Operationalisierbarkeit von Steuerbarkeit im Agentenkontext orientiert sich am Extremfall der unbeschränkten Kontrollmöglichkeit sämtlicher Aspekte des Agenten durch den Benutzer:

- Wie hoch ist der Prozentsatz der kontrollierbaren an den möglichen Aktionen? Welcher Anteil der Eingaben kann rückgängig gemacht werden? Für welchen Anteil an seiner Arbeitszeit ist ein Agent „unerreichbar“ (weil er zum Beispiel im WWW unterwegs ist)?
- Inwieweit gestaltet sich die Dialoggestaltung flexibel?
- Inwieweit wird Steuerbarkeit geeignet dargestellt?

Weitere Anhaltspunkte für Operationalisierungen bieten die mit Steuerbarkeit im Zusammenhang stehenden HCI-Prinzipien Visibility, Constraints und Affordances. Sie alle sind Bestandteile von Kontrollierbarkeit; steuern und beeinflussen die Fähigkeiten und Interaktionsmöglichkeiten von Agenten

3.5.6 Datenschutz/Datensicherheit

Diese letzte softwareergonomische Forderung verlangt nach Oppermann u.a. (1992, 63) den Schutz des Arbeitenden vor Leistungs- und Verhaltenskontrolle, zum anderen soll die Weitergabe bestimmter Daten der Kontrolle durch den Arbeitenden unterliegen.

In Agentensystemen sind Datenschutz und Datensicherheit essentiell für die dritte Hauptforderung nach Vertrauensbildung,. Die Leistungs- und Verhaltenskontrolle durch den Arbeitgeber (zum Beispiel durch das Zählen der Anzahl der Tastaturanschläge) spielt in Agentensystemen keine primäre Rolle. Aber die Weitergabe persönlicher Daten durch Agenten, unfreiwillig oder freiwillig, darf nur mit Einwilligung des Besitzers möglich sein.

Um diese Punkte zu berücksichtigen, lauten meine Hypothesen zu Datenschutz und Datensicherheit in Agentensysteme folgendermaßen:

Datenschutz ist das Ausmaß, in dem der Benutzer Umfang, Art und Empfänger seiner Daten kontrollieren kann. Datensicherheit ist das Ausmaß, in dem der Agent feindlichen Agenten oder anderen Objekten gegenüber resistent ist.

Der letzte Teilaspekt ist nur mit Einschränkungen erfüllbar und unter anderem an die Güte des Verschlüsselungscodes geknüpft. Eine weitere Gefahr droht (zukünftigen) mobilen Agenten, die zum Beispiel mit Kaufaufträgen und Geld im WWW unterwegs sind. Sie könnten mit scheinbaren Sonderangeboten auf einen fremden Rechner gelockt, vom Netz getrennt und „gefangengenommen" werden.

Die Operationalisierung von Datenschutz wird dadurch erschwert, da das Gefühl des Benutzers, seine Daten seien ausreichend „geschützt", nicht auf rein rationalen Grundlagen beruht und starken Schwankungen unterworfen sein kann. ein individuelles Bewusst stark unterschiedliche Schutzbedürfnisse. unterschiedliche ist nicht einfach.

- Werden sensible Daten übertragen? Werden sie verschlüsselt?
- Welchen Aufwand würde eine unautorisierte Entschlüsselung erfordern?
- Wurden Schutzvorkehrungen gegen das Ausspähen von Daten getroffen? Welcher Art sind diese Vorkehrungen?

3.5.7 Conceptual Models

„*Conceptual Models* ist der generische Ausdruck, der die unterschiedlichen Arten beschreibt, wie Systeme von unterschiedlichen Leuten verstanden werden. Sie bestehen hauptsächlich aus (1.) der Weise, in der *Benutzer* ein System konzeptualisieren und verstehen und (2.) der Weise, in der *Designer* ein System konzeptualisieren und verstehen (Preece 1994, 151)." Conceptual Models sind Gedankenmodelle, die uns unter anderem dabei helfen, Neues zu verstehen. Die obige Definition deutet aber auch die Probleme an, die entstehen, wenn sich Designer und Benutzer dasselbe System unterschiedlich *vorstellen*.

Die wichtigste Klasse von Conceptual Models sind dabei Metaphern, mit denen Neues durch die demonstrierte Analogie zu etwas Bekanntem verständlich gemacht wird. Beispielsweise verwenden grafische Benutzeroberflächen heute allgemein die Desktopmetapher. Mit dem Vergleich Benutzeroberfläche zu Schreibtisch können unerfahrene Computerbenutzer sehr schnell die Grundfunktionen der Benutzeroberfläche lernen und ein Verständnis für die möglichen Aktionen entwickeln. Dagegen halten Kritiker wie Nelson (1990, 236f) Metaphern prinzipiell für schlechte „Krücken" und lehnen insbesondere die Desktopmetapher ab. Trotz des unbestrittenen Erfolgs der Desktopmetapher muss zugegeben werden, dass jede Metapher Grenzen hat, wie das Papierkorbsymbol als Ablage für zu löschende Dateien zeigt: Der Papierkorb steht in Wirklichkeit *neben* dem Schreibtisch, nicht auf ihm, was aber der Nützlichkeit der Metapher keinen Abbruch tut.

Norman (1986, 46) weist darauf hin, dass sich die Bedeutung von Conceptual Models auf die Lern- und Fehlersuchphase beschränkt, dort allerdings „essentiell" ist. Folglich sind Conceptual Models besonders für die Lernförderlichkeit von Agentensystemen geeignet und können auf folgende Weise zum grundlegenden Verständnis des Benutzers von Agentensystemen beitragen: Durch die Präsentation eines geeigneten Agentencharakters kann der Benutzer wichtige Rückschlüsse über Komplexität, Arbeitsweise oder Grad an Autonomie des Agenten

ziehen.[40] Umgekehrt kann der Benutzer durch schlechte Agenten-Charaktere sehr leicht zu Fehleinschätzungen und Überschätzungen verleitet werden, insbesondere bei der Wahl menschlicher oder „übermenschlicher“ Charaktere.

Für Agentensysteme stelle ich deshalb folgende Hypothese auf:

> **Conceptual Models bezeichnen das Ausmaß, in dem ein gewähltes Gedankenmodell ein dazugehöriges Agentensystem konzeptualisieren und verstehen hilft.**

Beispielsweise erwartet Negroponte (1990, 350), dass durch Delegation (siehe Kapitel 4) die Desktopmetapher sich zu einer Theaterbühne wandelt, deren Akteure Agenten sind.

Es ist ein Mangel, dass die Agentenmetapher keine allgemein akzeptierte Visualisierung oder Erscheinungsform kennt.[41] Deshalb existieren die unterschiedlichsten Erscheinungsformen von Agenten nebeneinander her, ohne dass man angeben könnte, wie ein „typischer“ Agent aussieht. Dies ist neben der geringen Präzision des Begriffs ein Grund für die auch hier wieder schlechte Operationalisierbarkeit. Als Versuch, die Angemessenheit der Agentenmetapher *selbst* im Einzelfall zu beurteilen, sind die folgenden beiden Fragen aufzufassen.

- Inwieweit hat die Konzeption als Agentensystem für den Benutzer Verständnis- oder Performanzvorteile?
- Inwieweit deckt sich das Verhalten des Agenten mit dem verwendeten Conceptual Model?

3.5.8 Progressive Disclosure

Apple (1993) gibt Progressive Disclosure als eine der beiden Hauptmethoden an, um die Komplexität der Benutzerschnittstelle zu reduzieren.[42] Diese Methode ist auch deshalb so wichtig, weil viele Anwendungen heute über eine Menge von Funktionen verfügen, die bei weitem zu umfangreich ist, um gleichzeitig dargestellt zu werden.

Ein häufige und wenig aufwendige Methode ist, Progressive Disclosure durch Buttons der Art „*Weitere Optionen“* in Dialogboxen umzusetzen, hinter denen sich feinere Einstellmöglichkeiten verbergen. Progressive Disclosure verlangt im weitesten Sinne dagegen nach Anpassung des gesamten Systems an die Belange unterschiedlich kompetenter Nutzergruppen. Eine solche Aufteilung, etwa in *Erstbenutzer*, *kompetenter Benutzer* und *Experte*, macht allerdings erst ab einer gewissen Systemkomplexität Sinn. Der Übergang zwischen den Kompetenzstufen kann fließend sein und für jede Systemkomponente individuell erfolgen. Weiterhin muss die Aufteilung in Gruppen nicht notwendigerweise explizit zu sein. Sie kann auch implizit geschehen, beispielsweise durch eine adaptive Benutzerschnittstelle oder einfach durch das Anbieten von „*Weitere Optionen“* und *Profi-Optionen*.

Die verhüllende Tendenz von Progressive Disclosure scheint im Widerspruch zu Visibility zu stehen, erreicht vom Effekt her aber dasselbe, nämlich eine verringerte Systemkomplexität sowie eine Optimierung der Kontrolle. Damit unterstützt Progressive Disclosure zwei der für Agentensysteme wichtigen Hauptforderungen, *Lernförderlichkeit* und *Kontrolloptimierung*.

---

[40] Gegen diese Forderung verstößt der Microsoft Office-Assistant in besonders eklatanter Weise, da er bei gleichbleibender Funktionalität neun unterschiedliche Charaktere zur Auswahl stellt.

[41] Auch der Ausdruck „Agent“ selbst ist eine Metapher, die das Verhalten oder die Programmierung bestimmter Softwaresysteme veranschaulicht.

[42] Als zweite Methode zur Verringerung von Komplexität wird das Setzen von Benutzerpräferenzen genannt.

Progressive Disclosure ist das Ausmaß, in dem Agentensysteme geeignete, unterschiedlich komplexe Systemzugänge anbieten.

Diese Hypothese unterstützt die Forderung nach Kontrollierbarkeit von Agenten. Dem Anspruch von Softwareagenten, besonders gebrauchstauglich zu sein, sowie dem hohen Anteil agentenunerfahrener Benutzer kommt die Forderung entgegen, mehrere durchgängige Komplexitätsstufen anzubieten, sobald es die Systemkomplexität zulässt, und die am stärksten Komplexität verbergende Stufe voreinzustellen.

Zur Operationalisierung von Progressive Disclosure helfen folgende Fragen:

- Verbirgt das System in geeigneter Weise weniger wichtige Optionen vor dem Benutzer?
- Wie groß ist der (durchschnittliche oder maximale) Aufwand, verborgene Optionen zu erkennen und zu finden? Wie viele Stufen besitzen beispielsweise die Untermenüs?
- Werden unterschiedlich komplexe Zugänge zum System angeboten? Ist die voreingestellte Schnittstelle auf Anfängerniveau?

Eine Möglichkeit ist die Aufteilung in zwei Zugänge für Benutzer mit bzw. ohne Agentenerfahrung.

3.5.9 Visibility

Visibility ist der Grundsatz, jederzeit den Systemzustand erkennen zu können. Weiterhin müssen sämtliche augenblicklichen Eingabemöglichkeiten sichtbar sein. Dieses Schlüsselprinzip der HCI geht auf Norman (1988) zurück. Das Prinzip steht insofern im Gegensatz zu dem soeben erläuterten Progressive Disclosure, als dass es mit entgegengesetzten Mitteln arbeitet, also die unmittelbare Systemtransparenz erhöht, nicht vermindert. Dennoch kommt Visibility genau denselben Hauptforderungen an Agentensysteme entgegen, nämlich Lernförderlichkeit und Kontrollierbarkeit. Dabei trägt Visibility zur Kontrollierbarkeit insofern bei, indem sie die Distanz zwischen Benutzer und System verringert, was gemeinsam mit Affordances und Constraints zu einer besseren Steuerbarkeit führt. Mit Selbstbeschreibungsfähigkeit hat Visibility Überschneidungen, berücksichtigt allerdings weniger stark. Visibility hilft insbesondere dabei, Interna des Agenten zu veranschaulichen, die zum Verständnis seiner Funktions- und Arbeitsweise beitragen.

Für Agentensysteme stelle ich deshalb die Hypothese auf:

Visibility ist das Ausmaß, in dem für den Benutzer jederzeit Zustand, Aufenthaltsort, Arbeitsaufgabe, Arbeitsfortschritt, Arbeitsergebnis und Arbeitsmethode des Agenten erkennen zu können.

Sicherlich wäre es utopisch, eine völlige Visibility all dieser Aspekte in ein- und demselben Agentensystem zu fordern. Dennoch enthält die Hypothese gerade die zum Verständnis beitragenden, aber in der Regel vor dem Benutzer verborgenen Aspekte, beispielsweise die Frage, wie der Agent arbeitet und wofür er zuständig ist.

Ansatzpunkte zur Operationalisierung von Visibility sind:

- In welchem Anteil seiner Arbeitsspanne ist der Zustand des Agenten erkennbar? Mit welcher Verzögerung werden Zustandsänderungen übermittelt?

- Inwieweit sind die Aspekte Aufenthaltsort, Arbeitsaufgabe, Arbeitsfortschritt, Arbeitsergebnis und Arbeitsmethode des Agenten erkennbar?

3.5.10 Affordances und Constraints

Affordances sind Hinweise, die dem Benutzer Aufschluss über die Bedienungs- und Manipulationsweise von Objekten geben. Dieser - wie Visibility grundlegende - HCI-Begriff geht ebenfalls auf Norman (1992, 19) zurück, der Affordances als einen technischen Begriff definiert, „der sich auf die Eigenschaften von Objekten bezieht - welche Arten von Operationen und Manipulationen auf ein spezielles Objekt angewendet werden können." Durch geeignete Gestaltung können Objekte ihre Bedienungsweise dem Benutzer unterschwellig vermitteln, so dass sich die Effizienz der Interaktion erhöht und Fehlbedienungen minimiert werden.

Beim Design von Benutzeroberflächen spielen Affordances eine wichtige Rolle. Eigenschaften wie Größe, Form und Farbe signalisieren dem Benutzer den Verwendungszweck von Desktopelementen. Buttons signalisieren durch ihr Erscheinungsbild, dass sie gedrückt, nicht aber verschoben werden können. Ein wie in Abbildung 11 dargestellter Schieberegler dagegen macht dem Benutzer klar, dass hier eine Einstellung (stufenlos) regelbar ist, und dass jede Einstellungsänderung durch *Schieben* erfolgt, nicht aber durch Drehung oder Drücken. In Formularen oder Optionsfeldern signalisieren Kreise oder Kästchen auf Texthöhe, dass sie „angekreuzt" bzw. ausgewählt werden können.

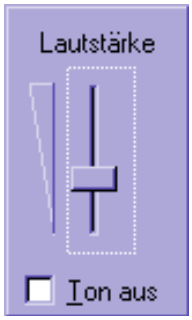


Abbildung 11: Schieberegler

In Bezug auf die Hauptforderung Kontrollierbarkeit sind Affordances im Zusammenhang mit Constraints und Mapping zu sehen. Das ergänzende Gegenstück zu Affordances bildet der HCI-Aspekt Constraints, der die Anzahl und Art von Eingabemöglichkeiten beschränkt. Damit sollen zum Beispiel unerwünschte oder gefährliche Aktionsmöglichkeiten verhindert werden. Beide, Affordances und Constraints, verstärken sich wechselseitig in ihrer Wirksamkeit, den Benutzer zu leiten. Daher sind Affordances und Constraints wesentliche Bausteine der Steuerbarkeit. Auch Mapping trägt wegen seiner Fähigkeit, die Beziehung zwischen Eingabeobjekten und manipulierten Objekten „natürlich" zu machen, zur Wirksamkeit von Affordances erheblich bei. Dies kann das Beispiel eines Software-Schiebereglers zur Lautstärkeeinstellung veranschaulichen: Der optisch hervorgehobene Schieber wirkt als Affordance, gerade ihn zu bewegen, und zwar - wie die hinter dem Schieber liegende Rille andeutet - in vertikaler Richtung. Software-Constraints verhindern auf der anderen Seite eine Bewegung in einer anderen als der vertikalen Richtung oder ein Bewegen anderer Elemente. Das Mapping zwischen Aufwärtsbewegung und kontinuierlicher Erhöhung der Lautstärke korrespondiert mit der Erfahrung des Benutzers von anderen Lautstärkereglern.

Im Bereich Lernförderlichkeit sind Affordances und Constraints für die Belange der Computer- sowie Agenteneinsteiger wichtig. Gerade die Einstiegsphase wird ihnen durch den Einsatz nonverbaler, keine bewussten Lernprozesse erfordernder Mittel erleichtert, da sie die Aufmerksamkeit des Benutzers nicht belasten, der gerade ein Verständnis für Agentensysteme entwickelt.

Für Agentensysteme stelle ich deshalb die Hypothese auf:

> **Affordances sind diejenigen Eigenschaften, die die auf den Agenten sinnvoll anwendbaren Operationen erkennen lassen.**

Diese Hypothese ist *positiv*; Affordances betrachten nur die Eigenschaften, die die sinnvolle Steuerung des Agenten verdeutlichen. Eine größere Einschränkung ist die Festlegung auf „sinnvolle“ Operationen. Doch dies erlaubt mir, Agenteneigenschaften abzudecken, die beispielsweise durch ihre Signalwirkung den Benutzer auf günstige Aktionen aufmerksam machen. Diese Eigenschaften würden sonst nur - allerdings mit umgekehrtem Vorzeichen, von Constraints erfasst. Problematisch ist der Versuch einer Operationalisierbarkeit von Affordances. Denn es müsste die Art und Stärke von Affordances meßbar gemacht werden bzw. der Unterschied zwischen Affordances und der Abwesenheit von Affordances.

Da der Aspekt Steuerbarkeit bereits die Kontrollierbarkeit des Agenten selbst regelt, kann die Beschreibung von Constraints als einem wichtigen Zulieferer von Steuerbarkeit für die Hypothese im Agentenbereich im Wesentlichen unverändert übernommen werden:

> **Constraints sind das Ausmaß bewußter Beschränkungen der Anzahl und Art von Interaktionsmöglichkeiten mit einem Agenten.**

Der Aspekt Constraints ist wie Affordances nicht ausreichend operationalisierbar. Da Constraints Aktionen des Benutzers *verhindern*, kann ihr Einfluss - anders als bei Affordances - nicht isoliert ermittelt werden, sondern nur im direkten Vergleich mit einer Anwendung, die diese Constraints nicht enthält, sonst aber mit ihr identisch ist.

3.5.11 Feedback

Feedback gibt nach Norman (1988, 27) dem Benutzer Rückmeldung darüber, welche Aktionen das System vorgenommen hat und was die Ergebnisse waren. Diese Rückmeldungen können auf unterschiedlichste Weise erfolgen: durch Text, Grafiken, Klänge, Animationen, ja sogar taktiles Feedback ist möglich.

In Agentensystemen trägt Feedback als allgemeiner HCI- Gestaltungsgrundsatz zu allen drei Hauptforderungen bei. Lernförderlichkeit ist auf Feedback angewiesen, um dem Benutzer Rückmeldung über richtige und falsche Aktionen geben zu können. Von Feedback sind Aspekte wie Affordances, Constraints und Steuerbarkeit abhängig. Deshalb ist eine Kontrolle des Systems ohne die Möglichkeit, Feedback zu erhalten, nicht praktikabel. Feedback trägt weiterhin zur Visibility einer Anwendung bei, indem es dem Benutzer die Auswirkungen seiner Aktionen erkennbar macht. Wegen seiner grundlegenden Bedeutung ist Feedback auch eine Voraussetzung für Vertrauen in einen Agenten.

Auf Agentensysteme bezogen muss Feedback idealerweise unmittelbar, kontinuierlich und vollständig sein. Durch Unmittelbarkeit und Vollständigkeit kann Feedback die Mittelbarkeit von Agentensystemen überwinden. Ein kontinuierliches Feedback hält den Benutzer bei langandauernden Agentenaufgaben informiert. Speziell in Agentensystemen sind zwei Ebenen von Feedback zu unterscheiden, das Feedback des Systems und das Feedback des auf ihm operierenden Agenten. Dies fließt auch in die folgende Hypothese von Feedback bei Agentensystemen ein:

> **Feedback ist das Ausmaß, in dem der Benutzer eine unmittelbare, kontinuierliche und vollständige Rückmeldung über Aktionen von Agenten und System sowie über die Auswirkungen dieser Aktionen erhält.**

Ansatzpunkte zur Operationalisierung sind Messungen, inwieweit beide Arten des Feedbacks tatsächlich unmittelbar, kontinuierlich und vollständig sind. Es kann beispielsweise untersucht werden, für welchen Prozentsatz der Zeit der Status des Agenten rückgemeldet wurde.

3.6 Ein System agentenrelevanter Kriterien setzt die Hauptforderungen um

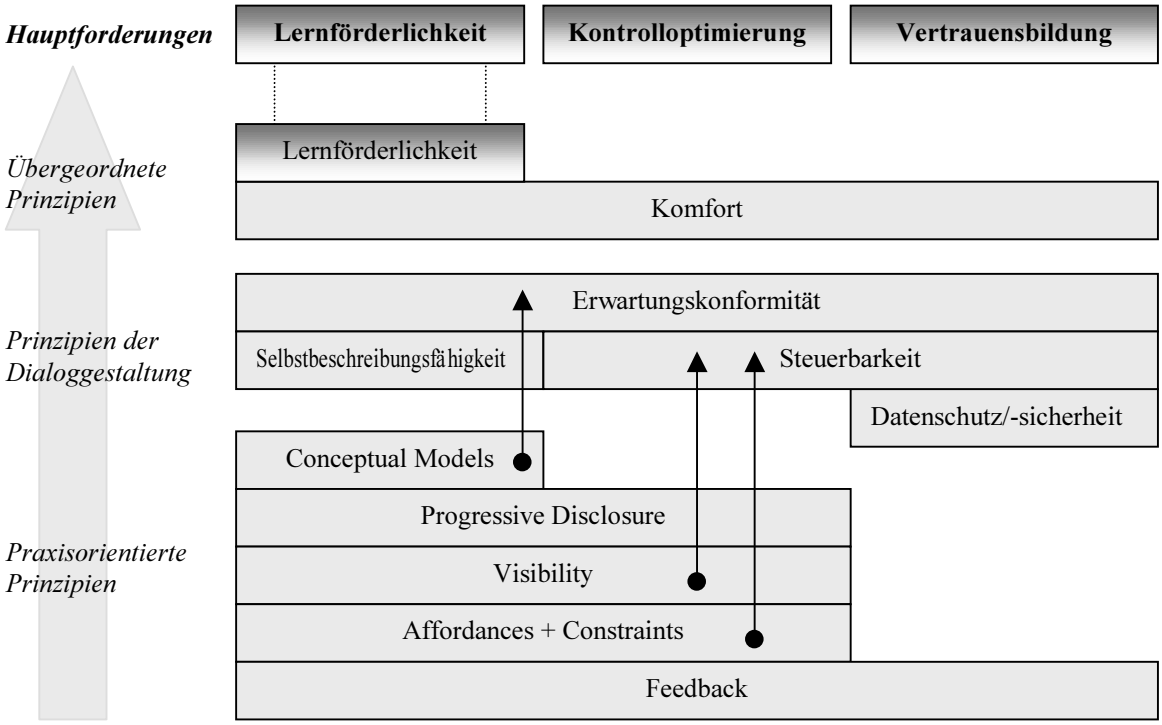


Abbildung 12: System agentenrelevanter Kriterien

Abbildung 12 stellt die agentenrelevanten Kriterien systematisch dar. Die oberste Zeile enthält die drei Hauptforderungen an die Kriterien, also die Mittel, um die Hauptschwächen von Agentensystemen zu beheben. Somit ergänzt die Abbildung inhaltlich wie optisch die in Kapitel 0, Abbildung 9 veranschaulichte Argumentationskette von der Gebrauchstauglichkeit von Agentensystemen zu den Hauptforderungen. Auf der vertikalen Achse finden sich in absteigender Komplexität die Gruppen *übergeordnete Prinzipien, Prinzipien der Dialoggestaltung und Praxisorientierte Prinzipien*. Jedes Prinzip trägt zu denjenigen Hauptforderungen bei, deren Spalten es überdeckt. Pfeile geben besonders starke Beiträge von Kriterien untereinander an.

Vorteile dieser Darstellung sind die schnelle Zuordnungsbarkeit von Kriterien zu Hauptforderungen und Komplexitätsgruppen. Wird beispielsweise ein *praxisorientiertes Prinzip* gesucht, das die Hauptforderung *Vertrauensbildung* unterstützt, so kann in der Tabelle *Feedback* als Kanditat abgelesen werden.

Das System agentenrelevanter Kriterien stellt somit meine aus der Menge an HCI-Grundsätzen getroffene *Vorauswahl* dar, auf die sich alle weiteren Untersuchungen stützen.

Auch wenn sich über die Auswahl *einzelner Kriterien* streiten lässt, war der Selektionsprozess (und die Transformation) notwendig für eine Erleichterung weiterer Untersuchungen.

Bei einigen der untersuchten Kriterien ist eine *Operationalisierung schwierig*. Allgemein ließen sich jedoch zumindest Prüffragen finden, mit deren Hilfe ich Hypothesen gebildet habe. Diese sollen die im Agentenzusammenhang angepasste Bedeutung des Kriteriums widerspiegeln, etwas in der Art: „Wenn die Hypothese zutrifft, dann erfüllt der untersuchte Agent das aktuelle Kriterium." Zusätzlich lassen sich - wie in der Abbildung dargestellt - einige Kriterien im Zusammenhang betrachten, sie können beispielsweise durch den Rückgriff auf die mit ihnen in Verbindung stehenden, untergeordneten Richtlinien ergänzt werden.

Die soweit operationalisierten Kriterien haben zwei Funktionen: Zum einen helfen sie in der Entwurfsphase des Agentensystems, den Faktor Benutzbarkeit im Auge zu behalten, also besser bedienbare Agenten zu bauen. Zum anderen können sie in einer komparativen Evaluation eingesetzt werden, um beispielsweise einen nach HCI-Richtlinien optimierten Agenten gegen seinen nicht-optimierten Gegenpart zu evaluieren.

Als Evaluationsmethode erscheint mir angesichts der aufgestellten Hypothesen eine *subjektive Evaluation* (vgl. Oppermann 1988, 323ff) geeignet, die auf einem mit Hilfe der Hypothesen und der in Abbildung 12 dargestellten Zusammenhänge ausgearbeiteten, auf die zu untersuchende Anwendung angepassten Fragenkatalog basiert. Dadurch kommt diese Form der Evaluation einem leitfadenorientierten Prüfverfahren nahe. Den Versuch einer Evaluation mit diesen Methoden werde ich im Folgenden an dem Agentensystem Firefly durchführen.

3.7 Anwendung: Untersuchung Fireflys mit agentenrelevanten HCI-Kriterien

Firefly ist als eines der ältesten Agentensysteme im WWW bekannt. Seine Anfänge liegen in einem studentischen Projekt am Massachusetts Institute of Technology (MIT). Das Ziel war, durch Filtertechniken dem Benutzer zu helfen, Musikgruppen zu finden, die seinem persönlichen Geschmack entsprechen. Firefly Network, eine von Pattie Maes, einer der „Pioniere der Agentenforschung" (Franklin u.a. 1996) mitbegründete Firma, vermarktet inzwischen die Firefly-Technologie und entwickelt sie weiter. Heutzutage werden mit Firefly unterschiedlichste Informationsangebote personalisiert, von Nachrichten über Software bis hin zu Sportartikeln.

Aus mehreren Gründen ist diese Technologie ein günstiger Kandidat für den Test der im letzten Abschnitt erarbeiteten Kriterien: Firefly ist ursprünglich eine *Forschungsentwicklung*, die die Möglichkeiten von Agenten aufzeigen sollte. Gleichzeitig ist Firefly kommerziell erfolgreich und besitzt einen hohen Bekanntheitsgrad. Als generische Technologie hat sich Firefly bereits in verschiedenen Domänen bewährt, wodurch auch seine zukünftige Bedeutung vielversprechend aussieht.

Meine Untersuchungen konzentrieren sich auf die Weiterentwicklung eines Musik-Recommenders, der ersten und oft mit dem Namen Firefly gleichgesetzten Anwendung Fireflys. Diese war so erfolgreich, dass sie vor kurzem von Launch, einem CD-Magazin, gekauft wurde und seitdem unter dem Namen myLAUNCH firmiert. Sich bei der Untersuchung auf eine spezielle Implementation von Firefly zu beschränken hat den Vorteil, andere Implementationen als Vergleiche heranziehen zu können. Auf der anderen Seite wird dadurch deutlich, dass viele der Gestaltungselemente *implementationsabhängig* sind, also keine direkte Folge

der darunterliegenden *Technologie* sind. Durch diesen Umstand ist es prinzipiell möglich, das Wunschbild eines optimal benutzbaren Firefly-Systems zu entwerfen, da ein auf seine Technologie abstrahiertes System wesentlich größere Freiheitsgrade bei einer (gedanklichen) idealen Implementation erlaubt als ein konkretes, nur geringe Modifikationen erlaubendes Agentensystem.

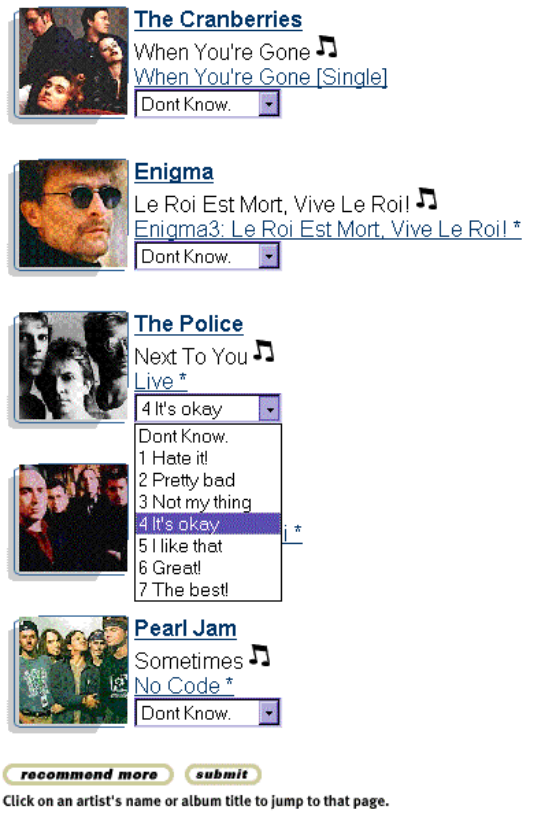


Abbildung 13: Der Bewertungsvorgang geht durch „submit“ in die nächste Schleife.

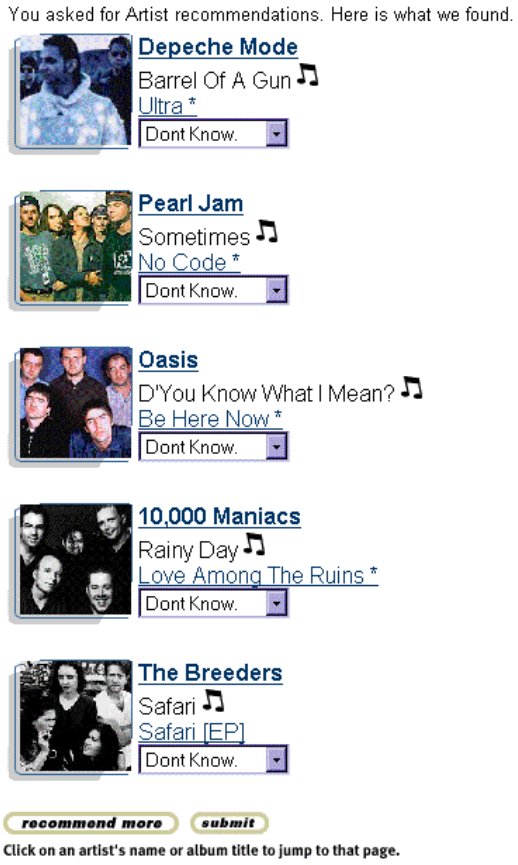


Abbildung 14: Empfehlungen myLAUNCHs regen zu weiteren Bewertungen an.

Die allen Firefly-Anwendungen zugrundeliegende Technik ist das „Collaborative Filtering“. Die Idee dabei ist, dass dem System die Vorlieben sämtlicher Benutzer (hier also ihr Musikgeschmack) bekannt und in Interessensprofilen gespeichert sind. Nun schlägt der Agent dem Benutzer bestimmte Titel als hörenswert vor, die zu den Favoriten anderer Benutzer mit ähnlichen Interessenprofilen gehören. Benutzervorlieben ermittelt das System dabei, indem es den Benutzer wiederholt Musiktitel bewerten lässt und schließlich dazu „passende“ Titel als Anregung vorschlägt. Der Effekt soll ähnlich dem sein, wenn der Benutzer einen Freund mit ähnlichem Musikgeschmack um Musikempfehlungen bittet.

myLAUNCH lässt den Benutzer zunächst zehn CDs auf einer siebenstufigen Skala bewerten. Danach legt es eine Liste von mehreren CDs vor, die dem Benutzer möglicherweise gefallen (Abbildung 13). Nach der Bewertung auch dieser Titel erscheint eine neue Kurzliste von CDs (Abbildung 14). In jedem Durchlauf werden die Vorschläge den Vorlieben des Benutzers besser angepasst. Jede der dargestellten CDs kann bei Bedarf direkt gekauft werden, womit die

Einnahmequelle von myLAUNCH enthüllt wäre. Typisch für Recommender-Systeme ist auch hier, dass die Empfehlungen umso präziser werden, je ehrlicher die Angaben des Benutzers sind.[43]

Einschränkend muss gesagt werden, dass, wie ich aus erster Hand erfuhr, nicht alle Benutzer mit der Güte der von myLAUNCH gemachten Musikvorschläge zufrieden sind. Daraus könnte man den Schluss ziehen, dass der von Firefly verfolgte Ansatz, Benutzereinstellungen zu ermitteln, *prinzipiell* falsch ist. Da ich selbst jedoch auch in ähnlichen, weniger weit entwickelten Recommendersystemen oft sehr brauchbare Empfehlungen erhalten habe,[44] stellen folgende, auf die spezielle *Umsetzung* myLAUNCHs bezogene Hypothesen eine wahrscheinlichere Erklärungsmöglichkeit[45] dar:

- Die Auswahl an vorhandenen CDs ist zu begrenzt. Dies begrenzt sowohl die Profilbildung als auch die Bandbreite möglicher Vorschläge durch das System.
- Benutzer werden unabhängig von ihren Vorlieben zu speziellen Titeln geleitet, an deren Verkauf myLAUNCH ein besonderes Interesse hat. Allerdings stünde dies im Widerspruch zu myLAUNCHs strengen Datenschutzregeln (Abschnitt 3.7.7).
- Benutzer mit heterogenen Vorlieben oder Vorlieben für wenige, in unterschiedlichen Genres singenden Künstlern (Frank Sinatra, Bob Dylan) machen es dem System schwer, den Geschmack des Nutzers zu erkennen.

Weitere Funktionen von myLAUNCH sind eine Reihe von unterschiedlichen Musikstilen zugeordnete Chatrooms.[46] Auch die Möglichkeit, die Namen anderer Benutzer mit ähnlichem Musikgeschmack aufgelistet zu bekommen, besteht. Durch Rückgriff auf die von Firefly angebotenen persönlichen Benutzer-Homepages ist es möglich, auf sehr einfache Weise Gleichgesinnte zu identifizieren, weitere Informationen zu erhalten und mit ihnen in Kontakt zu treten.

Nun werde ich myLAUNCH nach den im vorausgegangenen Kapitel aufgestellten Hypothesen untersuchen. Dabei werden auch die Hypothesen selbst auf Anwendbarkeit in einem realen Agentensystem untersucht.

3.7.1 Lernförderlichkeit

Hypothese: Lernförderlichkeit ist das Ausmaß, in dem die Belange agentenunerfahrener Benutzer berücksichtigt sind und der Agent das Erlernen des Systems fördert.

Lernprozesse des Benutzers werden von myLAUNCH generell schlecht unterstützt. Die einzige Art von „Hilfefunktion“ besteht aus einer statischen Informationsseite mit Kurzerläuterungen der Hauptfunktionen sowie der Beantwortung einiger häufiger Fragen. Eine nach Stichworten abfragbare Hilfefunktion oder eine Kontexthilfe existiert nicht, ebensowenig ein Lern- oder Demonstrationsmodus. Unterschiedliche Benutzergruppen werden nicht berücksichtigt.

Dabei ist einzuräumen, dass myLAUNCH über keine große Komplexität oder Funktionsvielfalt verfügt. Die geringe Komplexität macht den Verzicht auf die Personifizierung des Agen-

[43] Dies wirft die interessante Frage auf, inwieweit Menschen es akzeptieren, dass eine Maschine ihnen Ehrlichkeit abverlangt.
[44] Gute Erfahrungen habe ich mit Buchempfehlungen des BookMatchers von amazon.com (Online-Buchversand) sowie mit Musikempfehlungen des Album Advisors von CDnow (Online-CD-Versand) gemacht.
[45] Leider bin ich hier auf Spekulationen angewiesen, da keine internen Informationen über myLAUNCH zugänglich sind.
[46] In einem Chatroom können sich mehrere Benutzer gleichzeitig austauschen, indem sie zeilenweise Text eingeben, der für alle lesbar angezeigt wird.

ten akzeptabel. Dennoch dürfte myLAUNCHs ungewöhnlich reiterative Methode der Profilbildung gerade ungeübten Benutzern gewöhnungsbedürftig vorkommen.[47] Für die wichtige Gruppe der ungeübten Benutzer fehlt an dieser Stelle eine spezielle Unterstützung seht.

Dennoch hat der Prozess des Präferenzensammelns durch seine Wiederholungen lern- und verständnisfördernde Nebeneffekte. Täglich wechselnde, personalisierte Artikel aus der Musikbranche motivieren, sich mit dem System zu beschäftigen und dabei Funktionen „nebenbei“ zu lernen. Eine letzte Wissensquelle ergibt sich aus der Möglichkeit, mit anderen eingeloggten Benutzern auf einfache Weise in Kontakt zu treten und sie um Rat zu fragen.

Insgesamt erzielt myLAUNCH im Bereich Lernförderlichkeit schlechte Resultate, da dieser Aspekt kaum aktiv, sondern allenfalls indirekt unterstützt wird oder als Nebeneffekt auftritt. Auch Richtlinien wie Conceptual Models, Selbstbeschreibungsfähigkeit und Progressive Disclosure sind schwach ausgeprägt. Eine konsequente Umsetzung von Lernförderlichkeit verlangt eine Reihe von Hilfefunktionen und eine bessere Erklärung und Anleitung für unerfahrene Benutzer, wie sie beispielsweise der Firefly-Cousin Filmfinder anbietet: Hier enthält der Startbildschirm eine ausführliche Einleitung sowie eine Hilfefunktion.

3.7.2 Komfort

Hypothese: Komfort ist das Ausmaß, in dem der Benutzer bei seiner Aufgabe und bei der Interaktion mit dem System von einem Agenten unterstützt wird.

Beschränkt man die Aufgabe von myLAUNCH auf das Auffinden unbekannter, möglicherweise interessanter CDs, so ist das System sehr komfortabel, da es eine für den Benutzer nicht nur langwierige, sondern schlicht unmögliche Arbeit übernimmt. In der aktuellen Implementation wird der dadurch prinzipiell hohe Komfortfaktor allerdings durch die zu kleine Auswahl an CDs wesentlich gemindert. Im Test erhielt ich ab dem fünften Empfehlungsdurchgang stets gleichlautende Empfehlungen, später dann gar keine mehr. Das scheint im Zusammenhang mit geringen Musikkenntnissen oder überwiegend negativen Bewertungen sehr schnell auftreten zu können.

Dem zweiten Aspekt von Komfort, der Reduktion der Interaktionsmenge mit dem System, kommt myLAUNCH durch die stark den Benutzer integrierende Benutzerführung nicht entgegen. Auch bezogen auf die Hauptforderungen Kontrolloptimierung und Vertrauensbildung schneidet myLAUNCH schwach ab: Denn eine *konsequente Umsetzung von Kontrollmöglichkeiten* umfasst beispielsweise die Offenlegung und Editierbarkeit der vom System ermittelten Präferenzen. *Vertrauen* kann dagegen erst entstehen, wenn der Benutzer zu Recht annehmen kann, Empfehlungen aus einer großen, repräsentativen Datenbank zu erhalten.

3.7.3 Erwartungskonformität und Conceptual Models

Hypothesen: Erwartungskonformität ist das Ausmaß, in dem das Verhalten des Agenten dem durch seine Erscheinungsform beim Benutzer suggerierten Verhalten entspricht. Conceptual Models bezeichnen das Ausmaß, in dem ein gewähltes Gedankenmodell ein dazugehöriges Agentensystem konzeptualisieren und verstehen hilft.

[47] Und das kann zu Bedienungs- und Verständnisfehlern führen! Bei meinem ersten Kontakt mit dem damals neuen Firefly-System fand ich den Weg aus der Bewertungsschleife nicht - in der irrigen Annahme, es müsse (analog zum LifestyleFinder) eine *festgelegte* Anzahl von CDs bewertet werden, bevor Firefly „richtig“ starten würde. Wieder und wieder wurden mir Variationen derselben mir unbekannten Titel zur Bewertung vorgelegt (die Auswahl an verschiedenen CDs war noch sehr begrenzt) - bis ich schließlich aufgab. Diese Gefahr wird heute durch eine klarere Benutzerführung verhindert.

myLAUNCH verfügt über kein explizites Conceptual Model, und auch Firefly bietet keines an. Dabei würde ein geeignetes Conceptual Model helfen, Brüche in der Erwartungskonformität zu verhindern. Eine Visualisierung des myLAUNCH-Agenten in der Rolle eines Discjockeys, Plattenverkäufers oder Musikers bietet sich an. Beispielsweise könnte der Discjockey in der Rolle präsentiert werden, ein Musikprogramm, speziell für den Benutzer zusammenstellen zu wollen. In dieses Szenario ließen sich alle wesentlichen Funktionen von myLAUNCH auf natürliche Weise integrieren:

- Der Discjockey kennt den Benutzer zu Anfang nicht, kann sich aber im Dialog mit dem Benutzer über dessen musikalische Vorlieben informieren. Dieser Dialog spiegelt den iterativen Bewertungsvorgang von Musik durch den Benutzer auf natürliche Weise wieder, stellt an diesem kritischen Punkt die Erwartungskonformität wieder her.
- Als Musikexperte weiß der Discjockey, was sich aktuell in der Musikszene ereignet und „erzählt“ für den Nutzer interessante Neuigkeiten bei Bedarf. Er integriert an dieser Stelle die personalisierten Musiknachrichten.
- Der Discjockey verfügt selbstverständlich über eine große Sammlung an Musikstücken, in die er den Benutzer gerne „reinhören“ lässt. Dadurch wird die Funktion integriert, gewisse (aber nicht alle) Musikstücke kurz zur Probe hören zu können. Denn die Sammlung des Discjockeys ist zwar groß, aber nicht umfassend.
- In einer fortgeschrittenen Umsetzung könnte der Discjockey sogar eine Erklärung liefern, warum zur Zeit der Zugriff auf sämtliche Musikdaten verweigert wird und an dieser Stelle erneut die Erwartungskonformität des Benutzers wiederherstellen. Er könnte dies durch einen verschlossenen Plattenschrank gleichzeitig motivieren und visualisieren.

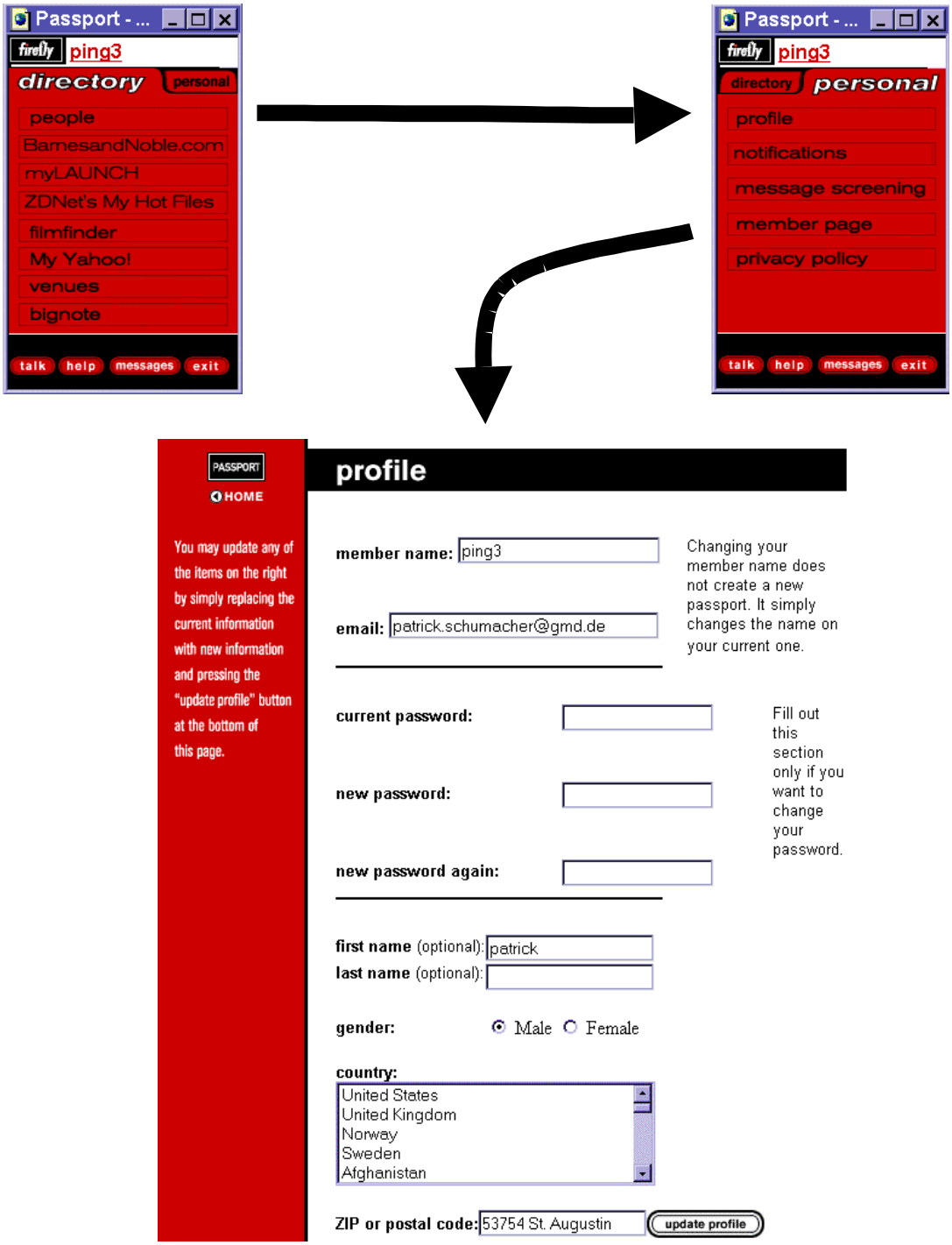


Abbildung 15: Details des Passports

Ein Conceptual Model verwendet Firefly jedoch, nämlich das der Pass-Metapher (Abbildung 15). Der Firefly-Pass, visualisiert durch ein kleines Browserfenster, das an einen Reisepass erinnern soll, enthält die persönlichen Daten des Benutzers, Zugänge zu sämtlichen Firefly-Sites und Chatrooms. Angeblich enthält er auch die persönlichen Präferenzen für alle besuchten Firefly-Sites. Zumindest im letzten Punkt muss die Beschreibung fehlerhaft sein, da die lokalen Präferenzen in Wirklichkeit von den einzelnen Sites gespeichert werden, ohne Möglichkeit der Einsichtnahme durch den Benutzer. Dieser Beschreibungsfehler muss allerdings nicht zwangsläufig zu Verständnisproblemen beim Benutzer führen.

Zusammenfassend lässt sich sagen, dass myLAUNCH die Erwartungen des Benutzers an mehreren Stellen verletzt. Viele dieser Brüche lassen sich durch ein geeignetes Conceptual Model verhindern. Die an anderer Stelle gewählte Pass-Metapher hat Vorteile, ist aber wegen ihrer Überfrachtung mit nicht-passtypischen Inhalten zweifelhaft.

3.7.4 Selbstbeschreibungsfähigkeit und Feedback

Hypothesen: Selbstbeschreibungsfähigkeit ist das Ausmaß, in dem ein Agent ein vollständiges und kontinuierliches Feedback sowie Hilfestellungen anbietet. Feedback ist das Ausmaß, in dem der Benutzer eine unmittelbare, kontinuierliche und vollständige Rückmeldung über Aktionen von Agenten und System sowie über die Auswirkungen dieser Aktionen erhält.

Wie schon bei „Lernförderlichkeit“ erwähnt, hat die Hilfefunktion von myLAUNCH Schwächen. Als WWW-Anwendung verwendet das System die typischen WWW-Fehlermeldungen, wenn beispielsweise Datenzugriffe misslungen sind. Doch selbst diese Fehlermeldungen lassen sich, soweit sie auf lokale Daten bezogen sind, durch Änderungen am Server verständlicher gestalten. Dem Grundsatz Selbstbeschreibungsfähigkeit zuwider läuft auch die fehlende Einsichtsmöglichkeit in seine eigenen Profildaten.

In den meisten Fällen besteht das Feedback auf Benutzereingaben in myLAUNCH aus dem Aufbau einer neuen WWW-Seite. Bis auf die Musikdaten selbst existiert kein akustisches Feedback.

Zusammenfassend lässt sich sagen, dass das Mittel Feedback zufriedenstellend eingesetzt wurde, während bei der Selbstbeschreibungsfähigkeit Mängel bleiben. Eine konsequente Umsetzung von Selbstbeschreibungsfähigkeit würde hier eine Reihe erweiterter oder neuer Hilfefunktionen verlangen sowie die Möglichkeit der Einsicht und Editierbarkeit der eigenen Musik-Präferenzen.

3.7.5 Steuerbarkeit, Visibility, Affordances und Constraints

Hypothesen:

- *Steuerbarkeit ist das Ausmaß, in dem die gesamte Interaktionsgestaltung mit dem Agenten durch den Benutzer kontrolliert und flexibel gestaltet werden kann.*
- *Visibility ist das Ausmaß, in dem für den Benutzer jederzeit Zustand, Aufenthaltsort, Arbeitsaufgabe, Arbeitsfortschritt, Arbeitsergebnis und Arbeitsmethode des Agenten erkennen zu können.*
- *Affordances sind diejenigen Eigenschaften, die die auf den Agenten sinnvoll anwendbaren Operationen erkennen lassen.*
- *Constraints sind das Ausmaß bewußter Beschränkungen der Anzahl und Art von Interaktionsmöglichkeiten mit einem Agenten.*

Der myLAUNCH-Agent ist nicht proaktiv und verfügt über keine Benutzermodellierung. Wird er aufgerufen, gibt er entweder eine Liste von potentiell interessanten oder von zu bewertenden CDs zurück, was auf dasselbe hinausläuft. Danach fällt er sofort in einen Zustand der Inaktivität. Diese Aufgaben verlangen nach keiner verbesserten Steuerbarkeit, mit der Ausnahme, dass Bewertungsvorgänge umkehrbar sein müssen.

Zum Thema Visibility ist die übersichtliche und konsistente Art der Informationsaufbereitung zu loben, die nicht einsehbaren Nutzerpräferenzen erneut anzumahnen. Sämtliche Eingabealternativen werden angezeigt.

Auf effektive Weise regen Affordances zum Bewerten von CDs an: Die Bewertungsschleife kann wieder und wieder durchlaufen werden, wobei die neu zu bewertenden Titel bereits Empfehlungen des Systems darstellen. Auch der Modus, in dem das myLAUNCH selbst Vorschläge macht, gibt dem Benutzer die Möglichkeit, eigene Empfehlungen abzugeben. Ein gutes Zusammenwirken von Affordances und Constraints weist die Suchseite auf, die verschiedene, gut gestaltete Formulare zum Bewerten, Suchen und Blättern bereitstellt. Zusammenfassend lässt sich sagen, dass der weitere Bereich Steuerbarkeit gut umgesetzt ist.

3.7.6 Progressive Disclosure

Hypothese: Progressive Disclosure ist das Ausmaß, in dem Agentensysteme geeignete, unterschiedlich komplexe Systemzugänge anbieten.

myLAUNCH setzt die Technik des Progressive Disclosure nicht konsequent um, sondern beschränkt sich im Wesentlichen auf das Verbergen von Einstellungen im Firefly-Pass. Es existieren keine unterschiedlichen Systemzugänge für verschieden kompetente Benutzergruppen. Eine konsequente Umsetzung von Progressive Disclosure würde zumindest einen Einsteigermodus und einen professionellen Modus beinhalten. Weiterhin würde sie sämtliche vom System gespeicherten thematischen Präferenzen (Musikvorlieben) sichtbar und editierbar machen.

3.7.7 Datenschutz/Datensicherheit

Hypothese: Datenschutz ist das Ausmaß, in dem der Benutzer Umfang, Art und Empfänger seiner Daten kontrollieren kann. Datensicherheit ist das Ausmaß, in dem der Agent feindlichen Agenten oder anderen Objekten gegenüber resistent ist.

An prominenter Stelle weist myLAUNCH auf seine „Privacy Policy“ hin, die besonders strenge Datenschutzstandards vorsieht. Alle Firefly-Systeme sind prinzipiell passwortgeschützt, so dass nur der Benutzer selbst seine Einstellungen ändern kann. Besonders komfortable Einstellungen können für die Homepage des Benutzers gemacht werden, auf der jede einzelne Angabe wahlweise lesbar oder lesegeschützt gemacht werden kann. Firefly plant für die Zukunft die Einrichtung einer elektronischen Geldbörse, was die Frage nach sicheren Übertragungsprotokollen aufwerfen wird.

Ein wichtiger Aspekt des Datenschutzes ist die aus der Privacy Policy entnommene Zusicherung, dass Produktempfehlungen *ausschließlich* aufgrund des eigenen Profils und insbesondere nicht aufgrund von Anbieterinteressen zustande kommen.

Fazit: Während sich der Bereich Datensicherheit (mangels verfügbarer Informationen) nicht beurteilen lässt, ist Datenschutz ein integraler Bestandteil des Systems.

3.7.8 Fazit: Die HCI-Hypothesen sind prinzipiell einsetzbar.

Trotz ihres Erfolgs und hohen Beliebtheitsgrades weist die Firefly-Anwendung myLAUNCH Mängel und zahlreiche Ansatzpunkte zur Verbesserung auf, die bei der Untersuchung mit agentenrelevanten Kriterien zutage getreten sind.

Dieses Ergebnis demonstriert gleichzeitig die *prinzipielle Anwendbarkeit der aufgestellten Hypothesen* bzw. Kriterien. Das hier wieder auftretende Problem der Quantifizierbarkeit und Operationalisierbarkeit könnte durch das Vorschalten einer aufgabenorientierten Analyse wie in Kapitel 2.4 verhindert werden.

3.8 Zusammenfassung

Einem Rückblick auf die Geschichte der HCI folgte die Vorstellung wichtiger Spezialgebiete und ihrer Gestaltungsgrundsätze. Als agentenrelevante Grundsätze wurden (angegeben in der aus Abbildung 12 entnommenen Reihenfolge) *Lernförderlichkeit, Komfort, Erwartungskonformität, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Datenschutz, Conceptual Models, Progressive Disclosure, Visibility, Affordances, Constraints und Feedback* identifiziert. Diese Gestaltungsgrundsätze wurden eingehender analysiert, verglichen und zu agentenrelevanten Kriterien transformiert. Dabei sollen Hypothesen die Operationalisierbarkeit der Kriterien erleichtern. Mit einer Systematisierung der angepassten Kriterien wurde ein Ansatz zur Umsetzung der in Kapitel 0 aufgestellten Hauptforderungen zu einem vorläufigen Abschluss gebracht.

Dieser Ansatz wird im sich anschließenden Anwendungsteil anhand des Agentensystems myLAUNCH untersucht. Dabei zeigt sich zum einen die bedingte Anwendbarkeit der Hypothesen, zum anderen im Widerspruch zur Beliebtheit von myLAUNCH stehende Mängel bei der Benutzbarkeit.

4 Das Konzept Delegation bei Agenten

Bisher wurden Agentensysteme nach verschiedenen Methoden eingeteilt, untersucht und beurteilt. Was fehlt, ist eine Untersuchung der Delegation, die von Lieberman als charakteristisch für das Verhältnis von Anwender und Agent angesehen wird. Delegation ist ein umfassendes und wichtiges Konzept, das Agentensystemen zugrunde liegt. Darum bildet es eine gute Ergänzung bei der agentenspezifischen Anpassung von HCI-Gestaltungsgrundsätzen.

4.1 Einleitung

In der dieser Arbeit zugrunde liegenden Definition von Agenten beschreibt Lieberman die Instruierung des Agenten durch den Benutzer als „Delegation“ und macht den Gegensatz zu normalen „Befehlen“ deutlich (siehe Abschnitt 2.2). Der Benutzer führt eine Aufgabe nicht unmittelbar selbst aus, sondern überträgt die Ausführung einem Agenten, gibt ihm gewisse Rechte, Pflichten, Mittel und Autorisierungen. Die Delegation ist auch deshalb ein für Agenten wichtiges Konzept, da sie sowohl den Benutzer als auch den Agenten umschließt.

Im folgenden wird zunächst die Entwicklung eines Leitfadens für Softwareagenten mit Aspekten der Delegation motiviert. Eine anschließende Ausarbeitung der Delegation schärft den Blick für ihre zahlreichen Konstrukte, auf die geeignete HCI-Gestaltungsgrundsätze abgebildet werden. Im darauffolgenden Anwendungsteil wird der Shoppingagent Jango eingehend nach ergonomischen, aufgaben- und delegationsorientierten Methoden untersucht.

4.2 Delegation ist die Voraussetzung für einen Leitfaden

Die Forderung nach Delegation ist anwendungsgesteuert, geht also auf professionelle Anwender selbst und die Erfordernisse ihrer Arbeit zurück. Die Entwicklung von Agenten ist dagegen bisher technologiegesteuert. Und dennoch ist das Konzept Delegation so stark in Agenten verankert, dass die Verkürzung der Agentendefinition auf „ein Programm, das im Namen des Benutzers handelt“, als eine Umschreibung von Delegation gesehen werden kann.

> **Das Besondere ist hier, dass sich eine *anwendungsgesteuerte* Forderung mit einem *technologiegesteuerten* Mittel umsetzen lässt.**

Und das hat mehrere wichtige Folgen:

1. Durch die Umsetzung von Delegation *bekommen Agenten mit einem Mal einen anwendungsgesteuerten Sinn*. Der Machbarkeitsaspekt tritt gegenüber der Arbeitsaufgabe in den Hintergrund.
2. Damit überschreiten Agenten an dieser Stelle die Stufe von der Technikentwicklung zur Softwareentwicklung!
3. Mit diesem Schritt wird gleichzeitig eine konkrete *Operationalisierung der Gestaltungsgrundsätze von Agentensystemen nötig.*
4. Dies aber ist die *Voraussetzung für die sinnvolle Anpassung von HCI- Gestaltungsgrundsätzen.* Denn in technologiegesteuerten Systemen ist - wie bereits erwähnt - eine gute Gebrauchstauglichkeit zweitrangig.

5. In der Konsequenz macht nun die Entwicklung von Evaluationsmethoden für Agentensysteme mit einem Mal Sinn. Und *damit wird letztlich ein softwareergonomischer Leitfaden nötig*, um schon in der Entwurfsphase Agentensysteme zu optimieren.

4.3 Aspekte der Delegation

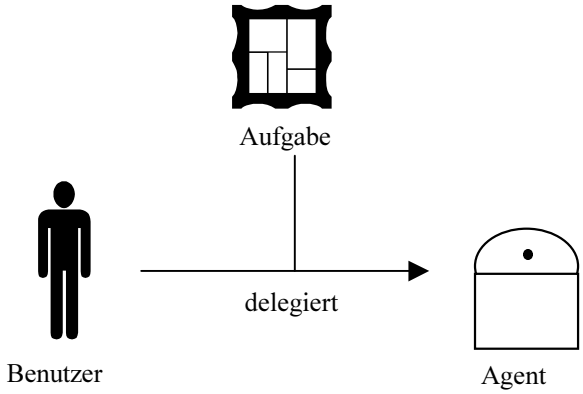


Abbildung 16: Schema der Delegation: Der Benutzer delegiert eine Aufgabe an den Agenten

Abbildung 16 zeigt die an der Delegation beteiligten und im folgenden genauer untersuchten Elemente: *Benutzer, Agent, Aufgabe* und den *Vorgang der Delegation* selbst.

4.3.1 Der Benutzer

Die natürlichste den Benutzer betreffende Frage ist die nach der **Motivation**: Warum sollte der Benutzer Aufgaben an einen Agenten delegieren, anstatt sie selbst auszuführen? Der häufigste Grund ist die als Delegation Model (vgl. Thomas 1996, 156) bezeichnete *Delegation typischer WWW-Aufgaben*, die nicht weiter anspruchsvoll sind, aber Zeit kosten, wie das Vorsortieren von E-Mails. Insbesondere regelmäßig wiederkehrende Aufgaben, wie das tägliche Suchen der Börsenkurse oder des Wetterberichts, oder das Preloading wichtiger WWW-Seiten sind hier gute Aufgabenbereiche für Agenten. Den Benutzer entlastende Agenten präsentieren sich in der Regel als Assistent und haben kein eigenes Agenten-Erscheinungsbild. Ein weiteres Motiv für Delegation ist die *Übertragung komplexer Aufgaben*, deren Ausführung für den Benutzer schwer bis unmöglich ist, an einen dafür spezialisierten Agenten. Oder der Agent sucht als Shoppingagent im gesamten WWW nach bestimmten Produkten und handelt (mit anderen Agenten) günstige Preise aus.

Für den Umfang der Delegation wesentlich ist das **Vertrauen**, dass der Benutzer in den Agenten setzt. Denn ein misstrauischer Benutzer wird Agenten gar nicht benutzen, ihnen zumindest aber keine sicherheitskritischen oder geldintensiven Aufgaben anvertrauen. Grundlage des Vertrauens ist das Verständnis in die Funktionsweise des Agenten und seine absolute Zuverlässigkeit. Norman (1997) argumentiert, dass sich das Vertrauen des Benutzers erst langsam entwickeln wird – bedingt durch die Unzuverlässigkeit von Computersystemen und das generelle Misstrauen von Menschen gegenüber technischen Neuerungen.

4.3.2 Der Agent

Für den Benutzer ist es wesentlich, die hier als **Kompetenz** bezeichnete Mächtigkeit eines Agenten richtig einzuschätzen. Entzieht sie sich der Kenntnis des Benutzers, ergeben sich Gefahren für die Delegation. Aus Furcht vor ungewollten Resultaten wird der Benutzer möglicherweise in diesem Fall überhaupt nichts an den Agenten delegieren. Dies ist mit einer Verhandlungssituation vergleichbar, die zu eskalieren droht, sobald ein Verhandlungspartner Zugeständnisse nachträglich zurücknimmt, weil sie seine Kompetenzen übersteigen. Maes (1994, 149) nennt vier Methoden, mit denen lernfähige, autonome Agenten Kompetenz erwerben können, nämlich das „dem-Benutzer-über-die-Schulter-schauen", Feedback vom Benutzer, vom Benutzer gegebene Beispiele und Ratschläge anderer Agenten. Doch das Problem bleibt, die Kompetenz eines Agenten zu erfahren und nicht durch ausprobieren herausfinden zu müssen.

Delegation an Agenten hat wesentliche Unterschiede zur Delegation an einen Menschen. Einer Sekretärin können mit gutem Wissen Aufgaben übertragen werden, die sie in ähnlicher Form schon einmal gelöst hat. Bei Agenten geht das nicht. Der Grund: Eine Sekretärin lernt stetig dazu und erwirbt *Metawissen* über ihre Aufgaben. Mit der Zeit kann sie ihren Chef *richtig einschätzen*, erwirbt sich eine *Übersicht* über ihre Arbeit und gewinnt an *Erfahrung*. Im Gegensatz dazu erwirbt ein Agent, vorausgesetzt er ist lernfähig, hauptsächlich Faktenwissen. Wohl mag er Regelmäßigkeiten und Muster in seinen Aufgaben erkennen und diesbezüglich Verbesserungsvorschläge machen, aber den Sinn seiner Arbeit kann er nicht begreifen. Selbst ein über den Benutzer eventuell gebildetes Modell hilft dem Agenten, anders als der Sekretärin, in unbekannten Situationen nicht. Also: Ein Agent gewinnt an **Wissen**, eine Sekretärin an Erfahrung. Ist nun das Vorhandensein von Metawissen beim Agenten für dessen erfolgreiches Arbeiten notwendig (nach Laurel (1990, 362) ist dies *immer* der Fall), kann die Lösung nur darin liegen, dem Agenten dieses Metawissen rechtzeitig zu vermitteln.

Menschen dürfen Fehler machen, Agenten nicht. Der Aspekt **Zuverlässigkeit** fordert vom Agenten, mögliche Falscheingaben in den ihm delegierten Aufgaben zu erkennen und zu melden. Gleichzeitig muss der Agent fehlerfrei „funktionieren" und seine Aufgabe erledigen. Delegation ist immer auch eine Vertrauenssache, und nur zuverlässigen Agenten kann der Benutzer vertrauen.

Im Zusammenhang mit Zuverlässigkeit ist der Aspekt **Robustheit** zu sehen. Er bezieht sich hauptsächlich auf die Interaktion des Agenten mit anderen Agenten oder Systemen. Nur an zuverlässige Agenten darf der Useragent Aufgabenteile weiterdelegieren. Sonst könnte er unüberprüfbare fehlerhafte Daten verwenden oder von „bösartigen" Agenten „ausgeraubt" oder auf andere Weise in seiner Tätigkeit behindert werden.

Eine der Hauptaufgaben von Agenten ist die **Reduktion der Komplexität** des Systems. Der Agent verhält sich wie eine Maske oder ein Übersetzer, was dem Benutzer einen einfachen Zugang zu einem an sich komplexen System erlaubt. In diesem Sinne ist Delegation die Übersetzung der Benutzereingaben in die Systemsprache.

4.3.3 Die Aufgabe

Aus der Arbeitspsychologie stammt die Forderung nach der Ganzheitlichkeit menschlicher Arbeit (vgl. Ulich 1992, 162ff).[48] Im Gegensatz zur im tayloristischen System vorherrschen-

[48] Diese Forderung wurde in der Softwareergonomie beispielsweise mit dem Ganzheitlichkeitsaspekt des EVADIS II-Leitfadens (siehe Oppermann u.a. 1992, 62) aufgegriffen.

den „unvollständigen Arbeit“, insbesondere Fließbandarbeit, führt das Gefühl, „ein ganzes Stück“ Arbeit getan zu haben, zu besserer Identifikation mit der Arbeit, Motivation und letztlich zu besseren Arbeitsergebnissen. Die Ursache ist, dass der Arbeitende Bedeutung und Stellenwert bei einer ganzheitlichen Arbeit besser erkennen kann als bei der Bearbeitung sinnlos erscheinender Einzelteile einer Aufgabe.

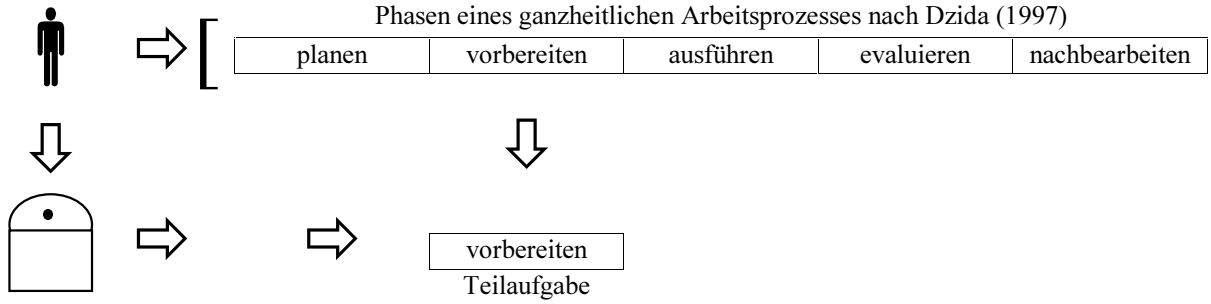


Abbildung 17: Delegation einer Teilaufgabe an den Agenten

Soll nun ein Agent in einen ganzheitlichen Arbeitsprozess eingefügt werden, ist darauf zu achten, daß die beim Menschen verbleibende Arbeit dennoch ganzheitlich bleibt. Dies wird durch Agenten in der Form eines dem Benutzer zuarbeitenden Helfers unterstützt, an welchen lästige oder mechanische **Teilaufgaben** delegiert werden (Abbildung 17). Es ist nicht Aufgabe des Agenten, abgeschlossene Arbeiten auszuführen, sondern eher mit dem Benutzer und anderen Systemkomponenten (auch der Agent ist nur eine Systemkomponente) zu kommunizieren und zwischen ihnen zu vermitteln, beispielsweise als Mailfilter. Hier kann wieder die Analogie zur Sekretärin aufgegriffen werden: Wenn die Sekretärin Dokumente vor- oder nachbearbeitet und Termine für ihren Chef plant, erledigt sie stets nur ausgelagerte Teilaufgaben. Initiative und Koordination gehen nach wie vor vom Chef aus. Im Agentenkontext sind Aufgaben, die immer beim Benutzer bleiben sollten, zum Beispiel die Zielrichtung der Arbeit.

4.3.4 Das Verhältnis Benutzer - Agent

Aus der objekt-orientierten Programmierung stammt die Auffassung von Delegation als **Kapselung** (vgl. Lieberman 1986), also der weitgehenden Trennung unabhängig voneinander arbeitender Programmteile. Im Agentenkontext kann die Abtrennung von Arbeitsteilen (siehe vorausgehender Abschnitt) und Übergabe an den Agenten als Analogie zur Kapselung gesehen werden. Durch diese Trennung von Benutzer und Agent verursacht Delegation eine Indirektheit oder *Mittelbarkeit* des Arbeitsprozesses, was zu Problemen führen kann: Es ist jeweils ein Zwischenschritt nötig, um einen Arbeitsschritt durchzuführen, und jegliche Rückmeldung muss eine zusätzliche Station durchlaufen, bevor sie den Benutzer erreichen kann.

Ersteres verlangt nach einer guten **Steuerbarkeit** des Agenten, so dass für den Benutzer eine komfortable und problemnahe Arbeitsweise möglich ist. Letzteres Problem lässt sich durch vollständiges **Feedback** des Agenten beheben. Für den Benutzer ist Feedback über jeden Schritt des Agenten potentiell wichtig, ebenso über Zustandsänderungen des Systems. Unter Umständen muss eine Rückmeldung *kontinuierlich* erfolgen, um hilfreich zu sein. Beide Me-

thoden, Relation und Feedback, wurden bereits in Abschnitt 3.6 als wesentlich zur Umsetzung der dritten Hauptforderung, der optimierten Steuerbarkeit, erkannt.

Bisher noch nicht betrachtet wurde die **Relation** zwischen Agent und Benutzer. Der waagerechte Pfeil in Abbildung 16 soll ein gleichwertiges Verhältnis zwischen beiden Parteien andeuten. Tatsächlich aber steht der Agent in der Regel auf einer niedrigeren Stufe als der Benutzer, zum Beispiel in der Form Assistent oder Befehlsempfänger. Andererseits kann der Agent komplexe Aufgaben durchführen oder den Zugriff auf eine Datenbank mit Spezialwissen darstellen, also dem Benutzer gegenüber die Position eines Lehrers einnehmen.

4.3.5 Der Prozess der Delegation

Delegation ist eine wichtige **Technik der objekt-orientierten Programmierung**. Daraus kann man mit Recht folgern, dass sich gerade objektorientierte Sprachen gut zur Implementation von Agentensystemen eignen. Für Agenten wichtig ist diese Technik, weil sie es erlaubt, den Agenten neben seiner Aufgabe auch mit einer Strategie ihrer Lösung auszustatten. Lieberman (1986), der den Begriff in die objekt-orientierte Programmierung einführte, sieht Delegation als (bessere) Alternative zur **Vererbung**. Kniesel (1996) bezeichnet Delegation darauf aufbauend als dynamische Vererbung zur Laufzeit und zieht eine Trennung zur oft missverständlich als Delegation bezeichneten **Konsultation** (Abbildung 18). Beiden Konzepten gemeinsam ist, dass eine Aufgabe an ein Objekt übertragen und im Namen des Delegierenden ausgeführt wird. Der Unterschied besteht in der Art der Ausführung. Vereinfacht gesagt soll im Fall Delegation die Aufgabe nach Art des Auftraggebers ausgeführt werden, im Fall Konsultation jedoch auf dem Objekt eigene Weise. Im Agentenkontext bedeutet Delegation hier, dass der Benutzer Aufgaben an einen Agenten delegiert, die er selbst nicht lösen will oder kann. Der Agent soll die Aufgabe prinzipiell selbständig lösen, im Fall von Unklarheiten oder fehlendem Wissen aber stets mit dem Benutzer Rücksprache halten.

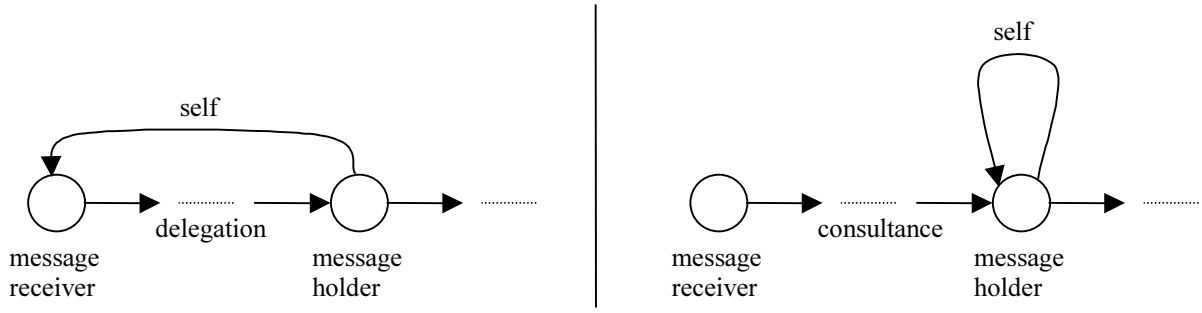


Abbildung 18: Delegation und Konsultation nach Kniesel (1997)

Delegation kann ein mehrstufiger **transitiver** Vorgang sein, bei dem der Agent ihm übertragene Aufgaben an verschiedene spezialisierte Agenten weiterdelegiert, die ihrerseits wieder untergeordnete Agenten aufrufen.

Abbildung 19 zeigt links die übliche **Variante der Delegation**: Nur ein einzelner Agent, nämlich der in Abschnitt 2.3.1.2 beschriebene Useragent, übernimmt die Kommunikation zwischen Benutzer und System (vergleiche Abbildung 2). Alternativ können, wie auf der rechten Seite dargestellt, unterschiedliche (spezialisierte) Agenten direkt mit dem Benutzer kommunizieren.

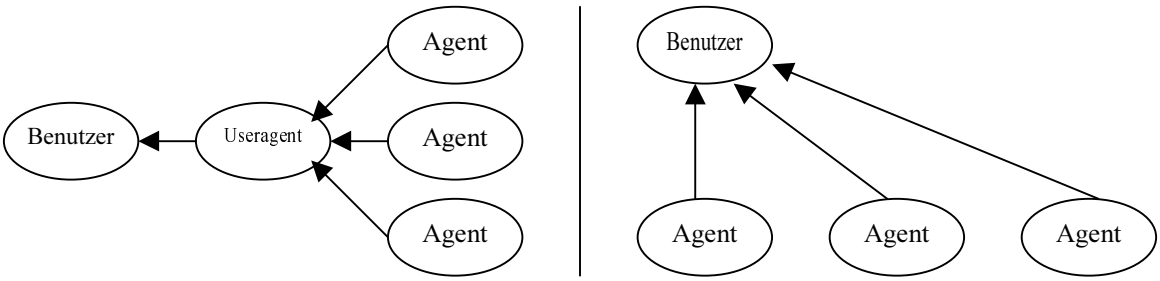


Abbildung 19: Varianten der Delegation

Wer trägt die *Verantwortung*, wenn ein Agent Fehler macht, zum Beispiel eine teure Fehlbuchung vornimmt? Man ist geneigt, diese Verantwortung in bestimmten Fällen dem ausführenden Objekt, hier also einem fremden oder eigenen Agenten zuzuschieben. Andererseits sind Agenten künstliche Objekte, Artefakte. Schon aus rechtlichen Erwägungen heraus kann die Verantwortung daher nur bei Menschen liegen. Agentensysteme sind nicht haftbar, haben kein Bewusstsein und wissen nichts über den Sinn ihrer Aufgabe. Also kann einem Agenten lediglich eine begrenzte Erlaubnis ausgesprochen werden, etwas zu tun. Diesen Vorgang nenne ich **Autorisierung**.

Ein letzter Delegationsaspekt unterscheidet zwischen den aus dem WWW bekannten Techniken **Push und Pull.** Das Pull-Modell der Delegation bedeutet, dass der Agent im Normalfall inaktiv ist und vom Benutzer für jede Aufgabe von neuem aktiviert werden muss. Diese Methode ist typisch für konventionelle Anwendungsprogramme und auch typisch für das WWW, in dem jede neue Seite explizit vom Benutzer angefordert werden muss. Im Gegensatz dazu gehen die Aktionen beim Push-Modell vom Agenten aus, der, einmal gestartet, im Hintergrund wartet und im geeigneten Moment selbständig Aktionen durchführt. Agenten eignen sich besonders für langfristige Aufgaben nach dem Push-Modell, bei denen sie zum Beispiel auf E-Mails warten oder den Benutzer beobachten und ihm proaktiv Vorschläge machen. Doch auch die Pull-Delegation eignet sich für Agenten, wie der im nächsten Abschnitt besprochene Shoppingagent beweisen wird.

In Abbildung 20 werden noch einmal sämtliche bei der Delegation relevanten Aspekte dargestellt.

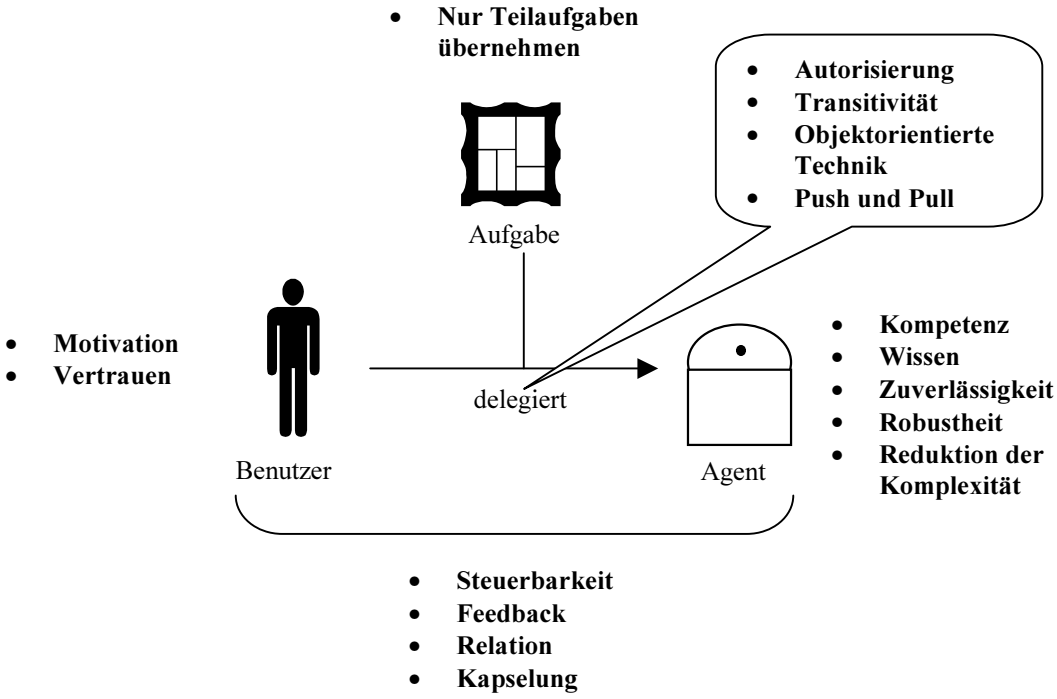


Abbildung 20: Aspekte der Delegation im Überblick

4.4 Abbildbarkeit von HCI-Gestaltungsgrundsätzen auf Delegation

Delegation spielt, ähnlich wie Agenten, in den bisher untersuchten Gestaltungsgrundsätzen und keine Rolle. Allenfalls indirekt greifen softwareergonomische Gestaltungsgrundsätze die Delegation auf, wie zum Beispiel mit den Forderungen nach Steuerbarkeit und Transparenz. Steuerbar sollen beispielsweise Einsatzgebiet, Periodizität und Strategien des Agenten sein. Gleichzeitig müssen Punkte wie Sinn und Zweck des Agenten, Status und prinzipielle Funktionsweise transparent bleiben.

Aus diesem Grund werde ich nun eine konzeptorientierte Untersuchung der ausgearbeiteten Aspekte der Delegation am Agentensystem Jango durchführen. Der Schwerpunkt der Untersuchung liegt auf der Umsetzung von Delegation, die erst sekundär durch ergonomische Mittel unterstützt oder verwirklicht wird. Dadurch wird zugleich eine Verbindung von HCI und Delegation geschaffen, die sich als „Erfolgsformel“ für gute Agentensysteme herausstellt.

4.5 Anwendung: parallele Untersuchungen Jangos

Jango ist ein Shoppingagent der 1996 gegründeten Firma Netbot. Die Firma wurde von zwei Professoren, Oren Etzioni und Dan Weld gegründet, die an der Washington University Agentenforschung betreiben. Mitte Oktober 1997 wurde Netbot von der für ihre Suchmaschine bekannten Firma Excite für 35 Millionen Dollar übernommen, um Jango in den Excite Shopping Channel zu integrieren (Netbot 1997). Jango ist eine komfortablere Weiterent-

wicklung von ShopBot (Doorenbos 1997), der wiederum durch den populären Shoppingagenten BargainFinder inspiriert wurde (Doorenbos u.a. 1997).

Aus folgenden Gründen habe ich Jango zur Untersuchung ausgewählt:

- Seine Ursprünge liegen in der Agentenforschung, nicht in der Industrie. Also sollte das System Agentenkonzepte umsetzen und sich nicht nur aus Marketinggründen Agentensystem nennen.
- Jango wird kommerziell eingesetzt, muss also höheren softwareergonomischen Ansprüchen genügen als rein experimentelle Systeme.
- Jango ist leistungsfähiger als vergleichbare Systeme wie BargainFinder oder Kasbah.
- Jango verfügt bereits über eine gebrauchstaugliche Oberfläche. Deshalb kann sich die Untersuchung im Wesentlichen auf Delegation beschränken.
- Jango wird weiterentwickelt und gepflegt. Also kann eine eingehende Untersuchung von Jango Vorschläge zur Produktverbesserung ergeben, die für die Weiterentwicklung nützlich sind.

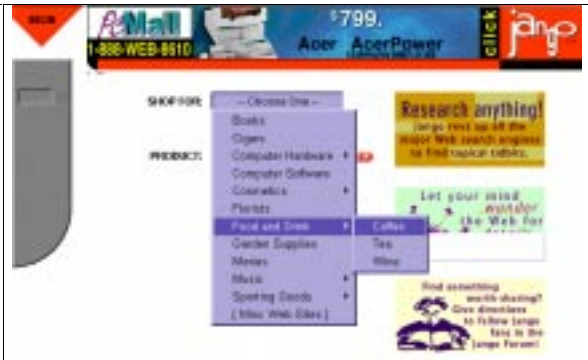
4.5.1 Allgemeines zu Jango

Jango verspricht, den Benutzer in zweierlei Hinsicht zu entlasten: Vom mitunter *komplizierten Prozess des Findens* von Anbietern eines gesuchten Produkts im WWW zum einen, zum anderen aber vom lästigen und *zeitaufwendigen routinemäßigen Suchen* nach (günstigen) Angeboten bei bereits bekannten Anbietern.

Jango ist eine Windows-Applikation, die wahlweise den Netscape Navigator oder den Internet Explorer als Benutzerschnittstelle verwendet. Nach dem Programmstart wählt der Benutzer aus einer vorgegebenen Auswahl von Produktkategorien die gewünschte aus und sucht darauf mit einer Stichwortsuche nach dem gewünschten Produkt.

4.5.2 Ein Testlauf

Tabelle 7: Ein Testdurchlauf mit Jango

Jango ist graphisch ansprechend gestaltet. Es ist kein freies Suchen nach Produkten möglich, wie man vermuten könnte, sondern die Auswahl beschränkte sich auf elf Kategorien (zum Beispiel Computer, Zigarren, Floristen) und 13 Unterkategorien. Die zwölfte Kategorie „Misc Web Sites“ führt zu einer Art Meta-Suchmaschine, die lediglich nach WWW-Seiten sucht, aber sonst keinen Bezug zum Online-Shopping aufweist.	 **Abbildung 21: Jango: Kategorieauswahl**
Ich habe in der Kategorie „Food and Drink“ die Verfeinerung „Coffee“ ausgewählt. Als Suchbegriff habe ich „Java“ gewählt, eine im angelsächsischen Bereich gängige Kaffeesorte. Davor habe ich bereits Jango meine Anschrift einschließlich meiner Kreditkarteninformationen anvertraut, um einen Kaufwunsch auch in die Tat umsetzen zu können.	 **Abbildung 22: Jango: Eingabe des Suchbegriffs**
Jango sucht. Sobald eine Quelle erreicht wurde, wird ihr Name schwarz hinterlegt und mit einem Häkchen oder Kreuz für Erfolg bzw. Misserfolg versehen. Gleichzeitig ertönt ein Trommelschlag. Während der Suchphase wächst eine Art Fortschrittsindikator, eine gepunktete Linie, im oberen Bildbereich. Am linken Bildrand läuft die Restzeitanzeige rückwärts, und an mehreren Stellen gleichzeitig werden die Trefferzahlen in den verschiedenen Suchbereichen laufend aktualisiert. Ein Trommelsolo läutet das Ende der Suchphase ein.	 **Abbildung 23: Jango: Visualisierung des Suchprozesses**

Automatisch wird - wie bei einer Suchmaschine - eine Ergebnisseite generiert, die in diesem Beispiel gefundene Java-Quellen tabellarisch und um Zusatzinformationen angereichert auflistet. Durch unterschiedliche Produktnamen und Quantitäten wirkt die Aufstellung recht ungeordnet.

Aber Jango bietet auch andere Informationen an:

1. **More Info**: Eine oder mehrere gefundene Definitionen oder Umschreibungen des Suchbegriffs
2. **Directory**: Eine alphabetische Liste von Links zu allen zur aktuellen Anfrage passenden Anbietern.
3. **Misc Hits**: Eine Metacrawler-ähnliche Liste von Links, zusammengestellt aus den Treffern der Suchbegriffs bei den gängigen weltweiten Suchmaschinen.
4. **Specials**: Bislang eine leere Dummy-Seite, soll aber einmal Sonderangebote enthalten.

Abbildung 24: Jango: Präsentation der Ergebnisse

Ich habe mich für die Mischung „Jumping Java Estate“ zum Pfundpreis von $12.79 entschieden. In einer Sicherheitsabfrage muss ich Jango die Erlaubnis zur Übermittlung meiner Adress- und Kreditkartendaten an den Anbieter bestätigen. Wenn, wie in diesem Fall, der Anbieter ein verschlüsseltes „sicheres“ Protokoll zur Kaufabwicklung anbietet, wird es von Jango angezeigt und verwendet.

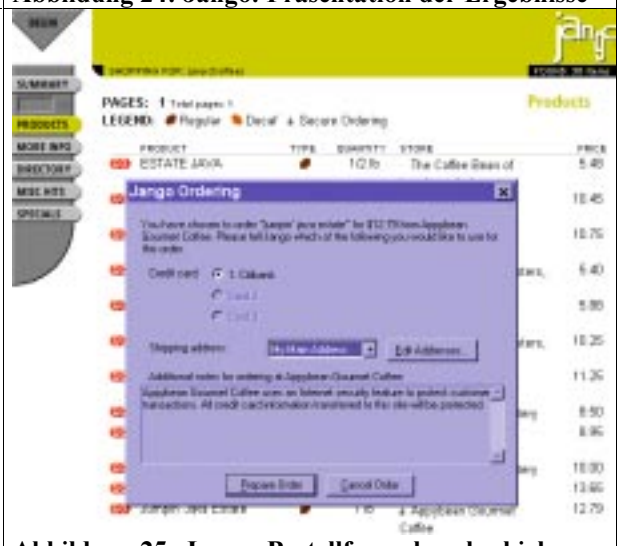


Abbildung 25: Jango: Bestellformular abschicken

Zum Erstaunen des Benutzers klinkt sich Jango an dieser Stelle aus dem laufenden Kaufprozess aus und übergibt die Kontrolle an den Anbieter. Ich werde ein zweites Mal zur Bestätigung des Kaufs aufgefordert. Obwohl es bisher keine Möglichkeit gab, die gewünschte Menge an Kaffee zu spezifizieren, ist hier bereits die Menge ein Pfund (entspricht einer Packung) voreingestellt.

Abbildung 26: Jango: Bestätigung der Bestellung beim Anbieter erforderlich

Offenbar wurden die persönlichen Daten wider Erwarten doch nicht an den Anbieter übermittelt, denn unvermittelt erscheint diesbezüglich ein leeres Formular. Alle Informationen müssen erneut eingegeben werden. Die Kaufwille sinkt, und der Testkauf wird abgebrochen.
Fairerweise muss gesagt werden, dass in einem späteren Testlauf die persönlichen Daten vollständig zum Anbieter (in diesem Fall einem Videohändler) gelangten. Ein Manko bleibt in diesen Fällen das Länder-Auswahlfeld, das bei Jango nur die Selektion „USA“, „Canada“ und „other“ zulässt. Somit muss vom Käufer, der in einem mit „other“ bezeichneten Teil der Erde wohnt, *bei jedem Einkauf* das Bestimmungsland manuell eingegeben werden.

Abbildung 27: Jango: Bestellformular des Anbieters

4.5.3 Aufgabenorientierte Untersuchung von Jango

Hier soll untersucht werden, inwiefern Jango typische Erwartungen an einen Shoppingagenten und seine Aufgaben erfüllt. Als Methode wird die in Abschnitt 2.4 vorgeschlagene aufgabenorientierte Untersuchung verwendet. Die in Tabelle 8 aufgeführten Benutzererwartungen sind weder vollständig noch repräsentativ, setzen sich aber aus Leistungsbeschreibungen Jangos und typischen, an einen Shoppingagenten explizit oder implizit gestellten Erwartungen zusammen.

Tabelle 8: Jango und die Erwartungen des Benutzers

Erwartung des Benutzers	Tatsächliche Aktionen bzw. Fähigkeiten Jangos	Kritik	Umsetzg.
Suchen nach beliebigen Produkten und Anbietern	Suche nach einer stark eingeschränkten, voreingestellten Menge von Produkten.	Kein „freies Suchen“ möglich.	–
Zeitersparnis beim Suchen	Tatsächliche beträchtliche Zeitersparnis durch parallele Suchanfragen.		+
Bereitstellung einer Marktübersicht	Eine teilweise Marktübersicht wird hergestellt.	Nur die Angebote der in Jango hartkodierten Anbieter werden aufgelistet. Keine Selektion oder Verbreiterung der Anbieter möglich.	o
Auffinden des günstigsten Angebots	Gesamtübersicht über mit dem Suchstring „matchende“ Einträge	Keine „intelligente“ Suche, keine intelligente Auflistung der Produkte.	–
Vorschlag von interessanten Produkten und Sonderangeboten, die dem Benutzer unbekannt sind	Vorgesehen, der Link führt allerdings auf eine „Baustellenseite“. „Misc Hits“ führt möglicherweise zu interessanten Angeboten.	Führt zu enttäuschter Erwartung beim Benutzer	–
Selbständig Einkäufe tätigen	Jango kauft nicht tatsächlich ein, sondern gibt nur Empfehlungen. Er kann aber unter Umständen die zum Kauf nötigen persönlichen Daten teilweise an Anbieter weitergeben.	Kaufen muss der Kunde nach wie vor selbst, auch wenn ihm die dazu notwendigen Schritte erleichtert werden.	–
Sicherheitsaspekte werden berücksichtigt	Passwortschutz von Jango, Hinweis auf „sichere“ Abrechnungsmöglichkeiten einzelner Anbieter.		+

Wohl kann Jango Suchvorgänge beschleunigen und automatisieren, allerdings sind die vorgegebenen möglichen Suchbereiche eine starke Einschränkung. Zum einen erweist sich Jango hier als zu starr und unflexibel, zum anderen enttäuscht er die in einen Shoppingagenten gesetzten Erwartungen.

Im Rahmen seiner auf voreingestellte Anbieter begrenzten Suchmöglichkeiten findet Jango auch günstige Angebote. Allerdings verfügt er weder über eine über Stringmatching hinausgehende intelligente Suchfunktion, noch listet er Produkte auf intelligente Weise auf. Jango sortiert seine Suchresultate streng alphabetisch und wirft dabei zum Beispiel CDs, Singles und Videos durcheinander. Die Unstrukturiertheit von Jangos Antworten steht dem Benutzerwunsch nach Übersichtlichkeit und schnellem Auffinden der preiswertesten Angebote entgegen.

Die wie eine vollwertige Funktion erscheinende „Misc Hits“-Option führt zu einer Seite, die bis auf die Ankündigung zukünftiger Angebote leer ist. Dies steht im Widerspruch zu der ansonsten recht guten und ansprechenden Gestaltung von Jango. Das Ergebnis ist eine enttäuschte Erwartung beim Benutzer, obwohl die Option mit Leichtigkeit hätte deaktiviert oder verborgen werden können.

Im Grunde genommen verdient Jango die Bezeichnung *Shopping*agent nicht, sondern ist in seiner jetzigen Form eher als intelligente Suchmaschine oder Recommender anzusehen. Denn er kann Einkäufe nicht selber tätigen, sondern nur auf (teilweise vorausgefüllte) Bestellseiten des Anbieters verzweigen. In der nächsten Version soll allerdings „richtiges“ Shopping möglich sein.

Sicherheitsaspekte berücksichtigt Jango in ausreichendem Maße, beispielsweise mit dem durch Passwortabfrage sicherbaren Programmstart. Jango verfügt zwar über die Möglichkeit der verschlüsselten Übermittlung sensibler Daten, ist dabei aber vom Server des Anbieters abhängig. Bietet dieser kein sicheres Übertragungsprotokoll an, dann kann auch Jango die Kreditkartendaten nur unverschlüsselt übertragen.

Fazit: Trotz seiner Überlegenheit gegenüber anderen Shoppingagenten enttäuscht Jango die in ihn gestellten Erwartungen. Ursachen sind für Agentensysteme typische Mängel bei der Umsetzung, hier möglicherweise verursacht durch Zeit- und Geldmangel, Neuheit der Materie sowie Beschränkungen auf Anbieterseite. Da die meisten Schwächen von Jango Verletzungen von HCI- Gestaltungsgrundsätzen sind, wird im nächsten Schritt Jango an den in Kapitel 3 als wesentlich herausgefundenen HCI- Gestaltungsgrundsätzen geprüft.

4.5.4 Untersuchung mit agentenrelevanten HCI-Gestaltungsgrundsätzen

Das in Abschnitt 3.6 aufgestellte System agentenrelevanter Grundsätze wird nun auf Jango abgebildet.

Tabelle 9: Jango im Agententest

Grundsatz	Eingehalten	Verletzt
Lernförderlichkeit	Leichte Bedienung, einfaches System	
Komfort	Komfortable Bedienbarkeit (der eingeschränkten Funktionen)	
Erwartungskonformität		• Jango „kauft“ nicht wirklich • Anbieter und Suchmaschinen sind nicht konfigurierbar • Kein „freies Suchen“ möglich
Selbstbeschreibungsfähigkeit	Stets eingeblendete Hilfefunktion	Keine weitere Information zu voreingestellten Anbietern und Suchmaschinen
Steuerbarkeit	Abbruch und Einstellung der Suchzeit möglich.	Keine Änderung der voreingestellten Auswahl von Datenquellen möglich.
Datenschutz	Passwortschutz von Jango, Hinweis auf eventuelle „sichere“ Abrechnungsmöglichkeiten einzelner Anbieter.	
Conceptual Models	Erweiterte Suchmaschine (nicht explizit gemacht)	
Progressive Disclosure	Präferenzen einstellbar	Nur ein Modus für alle, kein Fortgeschrittenen- oder Expertenmodus
Visibility	Klare Hinweise, wo sich der Benutzer befindet und welche Aktionen er ausführen kann.	Die „wirklichen Anfragen“ an Anbieter und Suchmaschinen werden verborgen.
Affordances	Die Jango-Fernsteuerung ist als dauerhaftes Top-Fenster voreingestellt.	
Constraints	Fehlerhafte Server-Anfragen sind ausgeschlossen	
Feedback	Deutliches akustisches Feedback bei jeder Aktion Jangos oder des Benutzers einstellbar Während der Suche: Fortschrittsindikator, Restsuchzeitanzeige und Erfolgsanzeige der einzelnen Server	

Stärken von Jango sind neben übergeordneten vor allem praxisorientierte Prinzipien wie Affordances, Constraints und Erlernbarkeit. Besonders gut gelungen ist das stets umfassende und kontinuierliche Feedback. Insgesamt macht die Oberfläche von Jango einen aufgeräumten, durchdachten und ansprechenden Eindruck. Die Visibility ist gut und dem Funktionsumfang angepasst. Auch das Conceptual Model einer Suchmaschine passt zu dem Aufgabentyp. Kaum umgesetzt wurde dagegen Progressive Disclosure, was bei der aktuell geringen Komplexität des Programms noch entschuldbar ist.

Schwächen finden sich bei der Dialoggestaltung, zum Beispiel der Selbstbeschreibungsfähigkeit, die Suchmaschinen und Anbieter bei Jango nicht erfasst. Dies wäre besonders wünschenswert in Verbindung mit einer individuellen Auswahl von Datenquellen, die ebenfalls nicht umgesetzt ist. Größter Kritikpunkt bleibt die gleich mehrfach gebrochene Erwartungskonformität. Diese ließe sich durch eine geringfügige Änderungen in Layout und Beschreibung des Programms sowie durch die Erweiterung um eine echte Kauf-Funktion wiederherstellen.

Die Hauptforderungen nach Lernförderlichkeit, Kontrolloptimierung und Vertrauensbildung (siehe Abschnitt 2.5) betrachtend erfüllt Jango erstere Forderung gut, die beiden weiteren Forderungen zumindest ausreichend.

4.5.5 Konzeptionelle Untersuchung von Jango

Wohl ist die Motivation für Jango ursprünglich *technologiegesteuert*, also die Demonstration der Leistungsfähigkeit eines Shoppingagenten. Doch tritt – zumindest optisch – der Agentenaspekt in der aktuellen Umsetzung vor der Aufgabe, Produkte in WWW zu finden, zurück. Für den normalen Benutzer finden sich keine Hinweise darauf, dass es sich bei Jango überhaupt um ein Agentensystem handelt, weder in der Bezeichnung („Assistent"), noch in der Beschreibung oder dem Programm selbst.

Delegation im Sinne einer Übertragung von Aufgaben auf Jango findet statt. Der Benutzer gibt lediglich Suchzusammenhang und Suchstring vor, den Rest erledigt Jango. Somit ist die Wahl der *Implementation Jangos als Agentensystem angebracht*.

4.5.5.1 Jango aus Benutzersicht

Jango führt langwierige und für neue Benutzer schwierige Suchtätigkeiten in einem Bruchteil der bei manueller Ausführung nötigen Zeit aus. Damit erfüllt er eine der an Agentensysteme gestellten Hauptaufgaben, nämlich die *Entlastung des Benutzers von Routinetätigkeiten***.**

Schwächen zeigt Jango dabei, *Vertrauen* beim Benutzer aufzubauen, was besonders wichtig ist bei einer Anwendung, die sowohl Name und Adresse als auch die Kreditkartendaten des Benutzers erfragt. Erfreulicherweise funktioniert Jango alternativ auch ohne die Preisgabe persönlicher Informationen. Die Zuverlässigkeit und gute Handhabbarkeit sind Punkte für das Vertrauen in Jango, aber schwerer wiegen die mehrfachen Inkonsistenzen und enttäuschten Erwartungen in das System. („Warum will Jango meine Kreditkarteninformationen, wenn er sie dann doch nicht weiterleitet?)

Fazit: Jango schneidet aus Benutzersicht mittelmäßig ab.

4.5.5.2 Jango aus Agentensicht

Jango lässt den Benutzer in dem Glauben, er kaufe für ihn ein. Die Folge ist, dass der Benutzer Jangos *Kompetenz* falsch einschätzt, was er erst beim ersten ernsthaften Kaufversuch herausfindet. Da Jango weder lernfähig noch autonom ist, hat er keine Möglichkeit, diese Kompetenz zu erwerben. Daraus ergibt sich, dass Jango auch keine Benutzermodellierung durchführen kann, da er auch weder über den Benutzer, noch über Besonderheiten der Anbieter-Sites Wissen erwerben kann.

Jango arbeitet zuverlässig, besitzt allerdings die erwähnten Inkonsistenzen und wird in Kombination mit manchen anderen Programmen instabil. Da wichtige Daten nur aus festgelegten „zuverlässigen" Quellen kommen, gibt es keine nennenswerten Angriffsmöglichkeiten auf die Robustheit Jangos.

Jango verbirgt seine Komplexität und (für den Benutzer irrelevante) Details seiner Arbeit, wie zum Beispiel Details von Serveranfragen.

Insgesamt schneidet Jango in diesem Komplex ebenfalls durchschnittlich ab.

4.5.5.3 Jango aus Aufgabensicht

Nicht der gesamte Komplex des Kaufens, sondern nur vorbereitende, klar abgegrenzte *Teilaufgaben* werden ausgeführt. Jango erfüllt also vollständig die Forderung nach einem Agentensystem, das nur zuarbeitet.

4.5.5.4 Jango aus der Sicht des Verhältnisses Benutzer - Agent

Von den Hauptforderungen, die sich aus der Kapselung ergeben, setzt Jango nur den Bereich *Feedback* in sehr guter Weise um. Dagegen weist die *Steuerbarkeit* starke Mängel auf. Weder die Auswahl der Quellen, noch die Bildschirmdarstellung ist veränderbar.

Jango verfügt über ein begrenztes Spezialwissen, so dass die Aufgabe, die er in 90 Sekunden erledigt, den Benutzer mehrere Stunden kosten würden, wobei das nicht am Umfang der Arbeit liegt, sondern tatsächlich am mangelnden Wissen um die richtigen Anbieterquellen. Andererseits kann Jango keine Dialoge führen, darf nicht einmal Vorschläge machen. Aufgabe und Art der Bewältigung sind von vornherein festgelegt. Jango reagiert auf Knopfdruck.

Aus diesen Gründen würde ich das **Verhältnis** von Jango zum Benutzer als das eines kompetenten Befehlsempfängers bezeichnen.

4.5.5.5 Jango aus Sicht des Delegationsprozesses

Da die *programmiertechnischen Grundlagen* von Jango nicht zugänglich sind, bleibt die Frage nach der Umsetzung von *Delegation bei der Programmierung* offen.

Offensichtlich fragt Jango sämtliche Datenquellen parallel ab, benutzt also die *Transitivität* der Delegation, um Einzelanfragen an untergeordnete Agenten weiterzugeben. Die Delegation oder Weitergabe von Kreditkarteninformationen an Anbieter zum schnelleren Ausfüllen des Bestellzettels ist fehlerhaft.

Jango ist lediglich *autorisiert*, Informationen zu sammeln und aufzubereiten. Er tätigt keine Käufe, führt aber vorbereitende Aufgaben durch.

Jango ist nicht autonom in dem Sinne von Freiheitsgraden bei der Erledigung seiner Aufgabe. Das System ist ebenfalls nicht proaktiv, sondern muss vom Benutzer jedes Mal von neuem gestartet werden. Somit verwendet es also das *Pull-Modell der Delegation*.

4.5.6 Zusammenfassung der Untersuchung Jangos

Die Konzeption von Jango als Agentensystem wurde durch die konzeptionelle Untersuchung nachträglich gerechtfertigt.

Der Agent tritt nicht in Erscheinung. Ein normaler Benutzer merkt also nicht, dass er es mit einem Agentensystem zu tun hat. Das ist in der jetzigen Umsetzung von Jango durchaus angebracht, weil ein Agentencharakter die Interaktion mit dem jetzigen System nicht nachhaltig verbessern würde. Sollte Jango allerdings um die Funktionen wie „periodisches Suchen“ und „automatisiertes Einkaufen“ erweitert werden, könnte dies durch einen geeigneten Agenten-Charakter unterstützt werden. An der Interaktion mit Jango ist wenig auszusetzen, umso mehr dafür an der bislang nur sehr eingeschränkt möglichen Steuerbarkeit.

Über die Qualität der Agententechnologie lässt sich wenig sagen, da keine Implementierungs- oder Entwurfsdetails vorliegen. Jango selbst präsentierte sich im Test als stabil laufendes System, aber die Zuverlässigkeit des Jango-Browser-Gesamtsystems wird durch den Browser bestimmt. Unter bestimmten Umständen können durch den Einfluss anderer, den Browser ebenfalls fernsteuernder Systeme, sehr instabile Systeme entstehen. In der Untersuchung trat ein solcher Fall beispielsweise dann auf, wenn zusätzlich zu Jango das Recommendersystem Alexa aktiviert war. Der Schutz sensibler Daten ist vom jeweiligen Anbieter abhängig und davon, ob er dasselbe sichere Übertragungsprotokoll wie Jango unterstützt. Zum Thema Ro-

bustheit ist zu sagen, dass Jango durch die festgelegte Auswahl der Anbieter keine neutralen Informationen liefert und so auch keine Markttransparenz herstellen kann. Eine Robustheit in bezug auf andere Agenten dürfte noch keine Rolle spielen.

Fazit: Die Mischung unterschiedlicher Untersuchungsverfahren (aufgaben-, HCI-basierte und konzeptorientierte Untersuchung) hat sich als für Agentensysteme vielversprechend erwiesen. Erstmals können auch Agentenkonzepte in die Untersuchung einfließen. Diese ist aber in der jetzigen Form noch zu inhomogen und zeitaufwendig. Aus diesem Grund wird vorgeschlagen, die bisherigen Punkte in einem Leitfaden zu bündeln. Das würde die Vorteile vereinen und die Anwendung auf die Entwurfsphase von Agentensystemen ausdehnen. Dies werde ich im folgenden versuchen.

4.6 Zusammenfassung

Delegation vereinigt Benutzerwünsche nach Arbeitserleichterung mit dem zentralen - Agenten zugrundeliegenden - Konzept. Dadurch wird die Bedeutsamkeit agentenbasierter Softwareentwicklung deutlich, was den Schlüssel zur Entwicklung eines Leitfadens für Agentensysteme darstellt. Eine Ausarbeitung der Delegation schärft den Blick für ihre zahlreichen Aspekte und deren Umsetzung mit HCI-Mitteln.

Im Anwendungsteil wurde der Shoppingagent Jango eingehend nach ergonomischen, aufgaben- und delegationsorientierten Methoden untersucht, was unter anderem die Anwendbarkeit dieser Methoden bewies. Es bleibt die Aufgabe, sie in Form eines Leitfadens zu bündeln und zu vereinheitlichen.

5 Leitfaden zur agentenbasierten Softwareentwicklung

Wie lassen sich die für Agenten gefundenen Untersuchungsmethoden optimal verwerten? Bisher wurden sie lediglich getrennt auf einzelne Agentensysteme angewendet. Doch weitaus nützlicher wäre eine Bündelung dieser Methoden in einer möglichst allgemein anwendbaren Form, die Designer zukünftiger Agentensysteme bei ihrer Arbeit unterstützt. Dies soll nun mit dem „Leitfadens zur agentenbasierten Softwareentwicklung“ versucht werden.[49] Dieser Leitfaden ist die meines Wissens erste Anleitung, um speziell Agentensysteme gebrauchstauglich zu entwickeln.

5.1 Einleitung

Nun können die Ergebnisse der vorausgegangenen Kapitel nutzbringend verwertet werden. Damit der Leitfaden nicht auf das Format einer Checkliste reduziert bleibt, werden zusätzlich, Werkzeuge, Beispiele sowie weiterführende Literatur angegeben.

5.1.1 Motivation: „Wozu dient der Leitfaden?“

Zunächst soll die Frage nach der Zielrichtung des Leitfadens beantwortet werden, seiner Relevanz und den zugrundeliegenden Überlegungen: Drei übergeordnete Motive bilden sozusagen der „Geist“, der hinter dem Wortlaut der einzelnen Gestaltungsgrundsätze steht.

1. **Die Benutzerfreundlichkeit von Agentensystemen wird verbessert** (siehe Abschnitt 2.5). Dazu müssen die Hauptforderungen nach *Lernförderlichkeit, Kontrollierbarkeit und Vertrauensbildung* erfüllt werden. Gleichzeitig müssen natürliche Usability-Stärken (*Abnahme von Routinetätigkeiten, Verhinderung von Information Overload, Verringerung der Komplexität des Systems*) unterstützt werden.
2. **Agenten setzen das Konzept Delegation um** (siehe Kapitel 4). Dabei ist ein zentraler Punkt im Verhältnis Mensch - Agent das *Vertrauen gegenüber dem Agenten.* Oft ist es das für den Benutzer allein entscheidende Kriterium für Ablehnung oder Akzeptanz. Aspekte wie Erscheinungsbild, Steuerbarkeit oder Zuverlässigkeit sind daher in erster Linie als vertrauensbildende Maßnahmen zu verstehen.
3. **Softwareagenten werden effektiv eingesetzt.** Dazu ist es notwendig, Agenten nur in solchen Domänen und für solche Aufgaben einzusetzen, wo sie traditionellen Systemen gegenüber *grundlegende Vorteile* aufweisen. Dies ist eine Voraussetzung für den langfristigen Erfolg und die Akzeptanz von Agentensystemen, eine der in Abschnitt 2.5 aufgestellten Hauptforderungen.

[49] Der Begriff „agenten*basiert*“ wurde wegen seiner stärkeren Betonung des Agentenaspekts im Gegensatz zu „agenten*orientiert*“ gewählt. Der Begriff der „agentenorientierten Programmierung“ wurde von Shoham (1993, 51f) eingeführt, worunter dieser eine Spezialisierung der objektorientierten Programmierung versteht. Shoham betrachtet „mentale Zustände“ von Agenten wie Überzeugungen und Verpflichtungen, vertritt aber keine Schnittstellensicht wie in unserem Fall.

5.1.2 Anwendungsbereich: „*Wann* hilft der Leitfaden?“

Der Leitfaden bezieht sich in erster Linie auf *Arbeitssysteme* und weniger auf Lernsysteme, Spiele und Agenten aus dem Bereich der Forschung. In Lernsystemen dürfen Agenten motivierenden Eigenschaften haben, in Spielen rein der Unterhaltung dienen, und in Forschungsanwendungen gibt es keinerlei Beschränkungen für Agenten. In Arbeitssystemen dagegen haben Agenten fest definierte Aufgaben (beispielsweise Belastungsoptimierung), unterliegen zahlreichen Beschränkungen und müssen vergleichbaren Anwendungen gegenüber deutliche Vorteile aufweisen.

Der Agentenbegriff wird im Sinne der Definition Liebermans (Abschnitt 2.2), eingeschränkt auf gleichzeitige User- und Systemagenten, verwendet. Deshalb beziehen sich die folgenden Gestaltungsgrundsätze rein auf die *Schnittstelle* zwischen Agent und Benutzer, nicht aber auf Organisation und Kommunikation nachgeschalteter Agenten.

Dieser Leitfaden enthält keine operationalisierten Kriterien, sondern „weiche“ Gestaltungsgrundsätze, was nach Dzida (1994, 291) legitim ist, da der Leitfaden beim Entwurf der Benutzerschnittstelle *neuer* Softwareagenten hilft. Ist allerdings eine anschließende Evaluierung der Schnittstelle beabsichtigt, so müssen bereits jetzt spezielle und überprüfbare *Anforderungen*, sprich Kriterien, an die zu entwickelnde Software formuliert werden. Hinweise dazu liefert Abschnitt 5.3.

5.1.3 Art der Hilfe: „*Wie* hilft der Leitfaden?“

Zu beachten ist, dass dieser Leitfaden nur eine *allgemeine Hilfestellung* geben kann. Deshalb wird nur vereinzelt auf verschiedene Typen von Agentensystemen eingegangen. Weil der Leitfaden – ähnlich wie ein Styleguide – auf einer abstrakten Ebene formuliert ist, ist er prinzipiell *programmiersprachen-unabhängig*, enthält dafür aber auch *keine Implementierungdetails*. Effizienz und Güte der Programmierung werden nicht erfasst. Ebenfalls nicht Thema des Leitfadens sind Optimierungen des Agenten nach softwaretechnischen Gesichtspunkten wie Geschwindigkeit oder ökonomischer Umgang mit der Bandbreite des Netzwerks. Sorgfältig vom Programmierer zu wählen sind Hilfsmittel und Datenquellen für Agenten, beispielsweise eine geeignete Ontologie für einen Information-Broker-Agenten.

5.2 Designkomplexe

Das in Abschnitt 4.3 ausgearbeitete Modell der Delegation benutzt eine Einteilung in die Bereiche Benutzer, Aufgabe, Agent, Verhältnis Benutzer-Agent sowie Prozess der Delegation. Da der Leitfaden in besonderem Maße Delegation umsetzen soll, habe ich die Einteilung der Delegation für den Leitfaden übernommen.[50] Jeder Komplex beginnt mit einer überblicksartigen Zusammenfassung relevanter Fragen und Methoden, ergänzt um praktische Beispiele und abgeschlossen durch Verweise auf bereits in Abschnitt 3.5 behandelte agentenrelevante Gestaltungsgrundsätze.

[50] Dem Leser mag auffallen, dass der Designkomplex „Prozess der Delegation“ des Modells der Delegation nicht weiter berücksichtigt wird. Die Ursache ist, dass er mit Ausnahme des Punktes Authorisierung nur Implementierungstechniken enthält, und diese werden im Leitfaden nicht behandelt. Der Punkt Autorisierung selbst bedarf keiner weiteren Erläuterung, da in Abschnitt 4.3.5 geklärt wurde, dass die Verantwortung für Aktionen des Agenten in jedem Fall beim Benutzer verbleibt.

5.2.1 Komplex 1: Der Agent

Dieser Komplex gibt Hilfestellung bei der Frage, *wann der Einsatz von Agenten lohnend ist.* Weitere Themen sind die zuverlässige und *fehlerfreie Arbeitsweise* sowie *robuste Interaktionen* mit anderen Agenten (vgl. Abschnitt 4.3.2). Zunächst sind in Tabelle 10 Umsetzungshilfen zum aktuellen Komplex zusammengefasst.

Tabelle 10: Umsetzungshilfen zum Komplex „Agent“

	Ist die **Umsetzung als Agentensystem** im konkreten Fall angebracht? (A. 5.2.1.1)
Fragen	1. Kann das Problem durch ein Agentensystem wirksam gelöst werden? → siehe Agentendefinition in Abschnitt 2.2 2. Ist ein Agentensystem hier konventionellen Programmen überlegen? → Dann setzt es Delegation (Kapitel 4) um. Wenn 1. und 2. erfüllt sind, ist die Umsetzung als Agentensystem angebracht.
Indizien für Agenteneignung einer zu entwickelnden Anwendung	• *Asynchrone* Aufgaben existieren. • Anwendung soll zu gewissem Grad *autonom* vom Benutzer sein. • Verteilte, parallele Retrieval-Aufgaben existieren.
Beispiele für agentengeeignete Software	• Aufgaben: Filter, Stellvertreter, Shoppingagenten, Recommender, Hilfesysteme (siehe typische Agentenaufgaben in Abschnitt 2.3) • Zu typischen Eigenschaften siehe Ende dieses Abschnitts

	Zuverlässigkeit (Abschnitt 5.2.1.2)	**Robustheit** gegenüber der Umwelt (A. 5.2.1.3)
Gestaltungsrichtl. (A. 3.5)	Aspekte der Hauptforderung 3 nach Vertrauensbildung: Komfort, Erwartungskonformität, Steuerbarkeit, Feedback	Datenschutz, Datensicherheit
Methoden	• Benutzermodellierung • CASE-Tools • Design Patterns	• Gegen Lauschangriffe → starke Verschlüsselungstechniken einsetzen • Gegen Manipulation → Agent sollte „gesundes Misstrauen“ besitzen • Gegen Zerstörung → Autonomie des Agenten erhalten
Beispiele	• Aktionen sollten umkehrbar sein (UNDO-Funktion) • Sicherheitsabfragen vor wichtigen Aktionen • Seriöse Funktionsbeschreibung des Agenten • Agent zertifizieren lassen (z.B. bei VeriSign)	• Interaktionsfähigkeit des Agenten limitieren • Keine Beschränkung auf ein Angebot/eine Quelle
Weiterführende Lit.	Zu Design Patterns siehe Kendall (1997) und Aridor u.a. (1998)	Lanier (1995) warnt vor Gefahren für Softwareagenten

Zu agentenbezogenen *Delegationsaspekten* siehe Abschnitt 4.3.2.: **Kompetenz, Wissen, Zuverlässigkeit, Robustheit**, und **Reduktion der Komplexität**.

5.2.1.1 Entscheidung für oder gegen Agenten

Folgende entscheidenden Fragen müssen *vor* dem Beginn der Softwarekonzeption beantwortet werden:

1. Kann das Problem durch ein Agentensystem wirksam gelöst werden?
2. Ist ein Agentensystem hier konventionellen Softwaresystemen überlegen?

Die erste Frage lässt sich relativ leicht mit Blick auf die Agentendefinition (Abschnitt 2.2) lösen. Ergänzend können die Konkretisierungen und Aufgaben aus dem Agentenmodell in Abschnitt 2.3 herangezogen werden. Auch die in Tabelle 10 genannten Indizien lassen die Agenteneignung einer zu entwickelnden Anwendung erkennen:

Nun kommt die zweite Frage ins Spiel, mit der nicht nur die *Möglichkeit* der Agentenumsetzung, sondern deren *Vorteile* betrachtet werden. Denn wenn der Einsatz von Agenten keine klaren Vorteile hat, trägt er weder zur Ergonomie des Programms noch zu einer Verdeutlichung des Agentenkonzepts bei. Eher werden Benutzer verwirrt oder in ihren Annahmen über die Eigenschaften und Fähigkeiten des Systems getäuscht.

Der Vorteil gegenüber konventionellen Programmen sind die Konzepte von Agentensystemen, die in Kapitel 4 unter dem Begriff Delegation untersucht wurden. *Nur dann ist die Umsetzung einer Anwendung durch ein Agentensystem sinnvoll, wenn sie konventionellen Umsetzungen gegenüber konzeptionell überlegen ist.* Denn dann trägt die Wahl von Agenten nicht nur zum Erfolg des Programms, sondern auch zum Erfolg der Agententechnologie (siehe Abbildung 9) insgesamt bei.

Folgende, aus Abschnitt 4.5.5 entnommenen Punkte deuten auf die einer Anwendung zugrunde liegenden Agentenkonzepte hin. Sollten mehrere von ihnen im konkreten Fall zutreffen, so ist die Umsetzung als Agentensystem vorteilhaft.

- Der Benutzer möchte lästige oder schwierige Aufgaben vom System erledigen lassen.
- Die Hauptaufgabe des Systems ist die Reduktion der Komplexität eines bestehenden Systems.
- Das System übernimmt Teilaufgaben, arbeitet dem Benutzer zu.
- Die Aufgabenbereiche des Systems und des Benutzers sind klar (oder bewusst) voneinander getrennt.
- Der Benutzer autorisiert das System zu bestimmten Aktionen.

5.2.1.2 Zuverlässigkeit

Dieser Punkt deckt die Bereiche Korrektheit, Sicherheit und Datenschutz ab, setzt die Hypothese zu Datenschutz/Datensicherheit (Abschnitt 3.5.6) um und ist eine wichtige Voraussetzung für die Herstellung von Vertrauen, wie gefordert in Abschnitt 2.5.

Systemfehler können bei Agenten schlimme (und auch teure) Folgen haben. Dies kann durch beweisbar korrekte Software verhindert werden. Doch da heutige Entwicklungsumgebungen die Korrektheit der mit ihr entwickelten Software nicht beweisen können, sollten zumindest die Techniken der Softwareerstellung von Agenten hohen Standards genügen. Insbesondere die traditionelle Software-Entwicklungsmethode wie der Test-Debug-Zyklus ist wegen seiner Fehlerträchtigkeit kein guter Ansatz. Wünschenswert ist dagegen die Verwendung von Agenten-Baukästen, CASE-Tools[51] und geeigneter Design Patterns[52], allesamt Maßnahmen, die zur Entwicklung fehlerarmer Systeme beitragen.

Mögliche Falscheingaben von Benutzerseite müssen abgefangen oder umkehrbar gemacht werden, beispielsweise durch Sicherheitsabfragen und, wo möglich, durch UNDO-Funktionen. Es muss dem Agenten weiterhin unmöglich sein, gegen den Willen des Benutzers zu handeln.

[51] CASE (Computer-Aided Software Engineering)-Tools sind integrierte Entwicklungsumgebungen, die eine komfortable und professionelle Erstellung auch größerer Softwaresysteme ermöglichen sollen.
[52] Design Patterns sind informelle Methoden des Software-Engineerings, die die Form eines Problem-Lösungs-Paares haben. Sie sind in den letzten Jahren in objekt-orientierten Programmiersprachen populär geworden.

Der Agent sollte ein Mindestmaß an Zeit und Aufmerksamkeit des Benutzers beanspruchen. Dies kann durch Methoden wie Benutzermodellierung und konsequentes Feedback angestrebt werden. Größte Vorsicht muß der Agent beim Umgang mit Geldmitteln des Benutzers walten lassen. Hier sind Constraints wie Sicherheitsabfragen nötig. Auch die Beschreibung der Fähigkeiten und des Leistungsumfangs eines Agenten muss seriös sein.

Der Schutz der Daten des Benutzer sollte eine Selbstverständlichkeit sein. Sowohl der unautorisierte Zugriff von Dritten, als auch die aktive Weitergabe an Dritte ist auszuschließen. Abzuwägen ist das Interesse der Justiz an Nutzerdaten zum Zweck der Strafverfolgung. Methoden für den Datenschutz sind die starke Verschlüsselung aller sensiblen Daten, die der Agent gespeichert hat bzw. kommuniziert.

Eine wichtige Rolle bei all diesen Punkten können *Zertifizierungsbehörden* wie VeriSign spielen, die schon heute Webseiten und über sie angebotene Programme „zertifizieren“. Das bedeutet nicht mehr, als dass der Benutzer vor dem Abruf der Daten einen Hinweis auf den Autor bekommt und darauf, dass kein dritter diese Daten manipuliert hat. Es trägt aber enorm zum Vertrauen auf der Benutzerseite bei. Analog könnten dem Benutzer unbekannte Agenten vor dem ersten Start einen Hinweis über ihre Herkunft, Unbedenklichkeit und Zuverlässigkeit geben.

In der letzten Zeit ist weiterführende Literatur zu Design Patterns für Agentensysteme erschienen: Kendall u.a. (1997) zeigt den Einsatz von Design Patterns in Multiagentensystemen, Aridor u.a. (1998) entwirft Patterns für mobile Agenten.

5.2.1.3 Robustheit

Mit Robustheit bezeichne ich die Fortsetzung des Vertrauens in den Agenten. Die vorausgegangen Aspekte hatten das Ziel, dass der Agent selbst zuverlässig arbeitet und seine Schnittstelle mit dem Benutzer optimiert ist. Die sich nun anschließenden Punkte verlassen die direkte Interaktionsumgebung und gehen auf die besonderen und für die Zukunft absehbaren Gefahren für Agenten in Netzwerken ein. Ich halte es für wesentlich, sie zu erwähnen, weil durch sie die bisher erreichte Funktionsfähigkeit von externer Seite zunichte gemacht werden kann. Es sind daher Maßnahmen nötig, um Agenten gegen Gefahren abzusichern, denen sie in einer potentiell feindlichen Netzwerkumgebung ausgesetzt sind.

Mehrere Gefahrenquellen sind denkbar: Lauschangriff, Manipulation, Raub und Zerstörung.

- Die internen Daten des Agenten werden „abgehört“, was sich durch starke *Verschlüsselungstechniken* erschweren lässt.
- Der Agent wird von destruktiven Agenten oder Personen „ausgeraubt”, bekommt also zum Beispiel minderwertige oder gar keine Ware für sein Geld. Der Agent muss zwischen seriösen und unseriösen Anbietern unterscheiden können sowie zwischen realistischen und unrealistischen Angeboten.
- Ein Informationsagent liefert keine neutrale Anbieterübersicht, sondern wird von einzelnen Anbietern unzulässig beeinflusst. Lanier (1995) weist darauf hin, dass bei einer von Agenten dominierten Sicht auf Informationsquellen „Werbung zu der Kunst [wird], Agenten zu kontrollieren”. Und es bestehe die Gefahr, diese Kontrolle auch durch Bestechung, Hacken oder sonstige illegale Methoden zu erlangen. Der Agent muss eine prinzipielle *Autonomie gegenüber seinen Partnern* bewahren.

- Agenten sollen kooperieren und sich austauschen können. Es muss aber verhindert werden, dass ein Agent durch sehr schnelle Serien externer Anfragen außer Kraft gesetzt wird. Durch diese bösartige, aber im WWW durchaus verbreitete Technik kann ganze Webserver und Computer dazu gebracht werden, den Dienst zu verweigern. Umso wichtiger ist eine *Immunität des Agenten* gegen solche Methoden.

Agenten müssen also robust genug sein, solchen möglichen Attacken zu widerstehen. Dies kann zum Beispiel durch eine vom Umfang her streng quotierte Interaktionsfähigkeit des Agenten geschehen, durch eine gegenseitige Authentifizierung von Agenten mit Hilfe der erwähnten Zertifizierungsschlüssel. Sinnvoll können auch „Faustregeln" des täglichen Lebens sein wie diese, erst nach dem Einholen dreier konkurrierender Angebote einen Kauf zu tätigen. Sehr wünschenswert, aber schwerer zu realisieren ist ein eingebautes Misstrauen des Agenten gegenüber „unmöglichen", zum Beispiel unrealistisch günstigen Angeboten.

Es empfiehlt sich eine Lektüre von Lanier (1995), der zwar Agenten generell ablehnt, aber wichtige Hinweise auf Gefahren durch fremde Agenten gibt.

5.2.2 Komplex 2: Der Benutzer

Die für den Benutzer wesentlichen Aspekte der Delegation sind *Motivation und Vertrauen* (siehe Abschnitt 4.3.1). Der Benutzer braucht eine Motivation zur Benutzung des Agenten, und er muss ihm mit Recht vertrauen können. Beides hängt zunächst vom Erscheinungsbild des Agenten ab, das in Anschluss an die Umsetzungshilfen des Komplexes „Benutzer" in Tabelle 11 thematisiert wird.

Tabelle 11: Umsetzungshilfen zum Komplex „Benutzer“

	Erscheinungsbild des Agenten (Abschnitt 5.2.2.1)
Gestaltungsrichtl. (Abschnitt 3.5)	Erwartungskonformität, Conceptual Models
Hinweise	• Alternativen zu vorgegebenen Erscheinungsbildern anbieten. • Mögliche kulturspezifische Konnotationen von Charakteren berücksichtigen.
Fragen	Hat das Wissen des Benutzers darüber, dass das System ein Agentensystem ist, Vorteile für sein Verständnis oder Vertrauen? Verbessert es die Arbeitseffizienz des Benutzers? • Nein → keine Visualisierung. • Ja →Visualisierung angebracht. • Teilweise → nur geeignete Teile visualisieren
Methoden	• Bei der Wahl des Agenten-Charakters Benutzererwartungen berücksichtigen (Tabelle 12), besonders bei anthropomorphen Erscheinungsbildern. • Charakter in Abhängigkeit vom Interaktionsziel auswählen
Beispiele	• Erscheinungsformen: Zauberer, Einstein, Engel, Mensch, Diener, Butler, Hund, Roboter, Insekt (Tabelle 13; Abbildung 28) • Frei programmierbare Präsentationsagenten: Microsoft Agent (flexibel, kostenlos), Persona, togglethis
	Komfortable Ein- und Ausgabemedien (Abschnitt 5.2.2.2)
Gestaltungsrichtl. (Abschnitt 3.5)	Komfort, Steuerbarkeit, Visibility, Feedback
Hinweise	Interaktion mit dem Agenten sollte *mühelos* sein.
Methoden	Wahl der Ein- und Ausgabemedien ist abhängig vom Interaktionsstil und -ziel. • Typische Eingabemedien: Tastatur (schnelle umfangreiche Eingaben möglich), Maus (bequeme Auswahl von Alternativen) Spracheingabe (Natürlichkeit, Vorteile in dialogorientierten Systemen und bei Teamarbeit • Typische Ausgabemedien: Text, Töne, Grafik, Animation, Sprachausgabe (siehe Tabelle 14)
Beispiele	Microsoft Agent: erlaubt Spracheingabe/-ausgabe, Text, Animationen, Töne

Zu benutzerbezogenen *Delegationsaspekten* siehe Abschnitt 4.3.1: **Motivation** und **Vertrauen**.

5.2.2.1 Erscheinungsbild des Agenten

Soll die Tätigkeit des Agenten eine direkte Repräsentation auf der Benutzeroberfläche haben? Konkreter, ist eine Visualisierung des Agenten sinnvoll, und welche Erscheinungsform ist gegebenenfalls am besten geeignet? Der folgende Abschnitt gibt diesbezüglich Hinweise.

Hat das Wissen des Benutzers darüber, dass das System ein Agentensystem ist, Vorteile für sein Verständnis oder Vertrauen? Verbessert es die Arbeitseffizienz des Benutzers?

1. „Nein.“- In diesem Fall sollte der *Agent nicht als solcher in Erscheinung treten*, sondern beispielsweise hinter einer quasi-direktmanipulativen Oberfläche verborgen bleiben.
2. „Ja.“ - Wo eine *Visualisierung des Agenten* im Nutzungskontext von Vorteil ist, da sollte sie auch erfolgen, da ein geeignet gewähltes Erscheinungsbild dem Benutzer unmittelbar Hinweise auf Aufgaben und Funktionsweise des Agenten geben kann.
3. „Teilweise.“ - Es besteht auch die Möglichkeit, *nur besonders geeignete Teile der Agentenfunktionalität durch einen Charakter zu visualisieren* und andere nicht. Beispielsweise kann ein als „Experte oder Lehrer“ dargestellter Agent die Funktion eines Moderators zwischen dem Benutzer und dem eigentlichen Agentensystem übernehmen.

Abbildung 28: Unterschiedliche Erscheinungsformen des Office-Assistenten

Die Wahl einer geeigneten Erscheinungsform des Agenten ist von den aktuellen *Erwartungen der Benutzer* an den präsentierten Agenten und von dessen Einsatzgebiet abhängig Dabei existiert nach Thomas u.a. (1997) keine „beste Metapher“, sondern verschiedene nützliche Möglichkeiten. Die adaptive Leistung des Systems liegt darin, dem Benutzer eine die *tatsächlichen Fähigkeiten des Agenten vermittelnde Erscheinungsform* zu präsentieren.

Beispielsweise ist die Komplexität der vom Agenten zu übernehmenden Aufgabe ein wichtiges Indiz. Denn Menschen tendieren dazu, vom Äußeren auf das Innere zu schließen, in diesem Fall also in Abhängigkeit vom Erscheinungsbild dem Agenten mehr oder weniger „Intelligenz“ zuzuschreiben. Erfordert die vom Agenten zu lösende Aufgabe nun eine geringe „Intelligenz“, ist also wenig komplex, dann ist eine Metapher für den Agenten angebracht, die ebenfalls geringe Intelligenz suggeriert.

Als Negativbeispiel ist hier der Office Assistent zu nennen, dessen neun (!) zur Auswahl stehende Erscheinungsformen in Abbildung 28 dargestellt sind. Die Abbilder sind nicht nur höchst unterschiedlich, sondern geben vor, jeweils eine eigene Persönlichkeit zu besitzen. Beides suggeriert dem Benutzer unterschiedliche Fähigkeiten und Aufgaben der verschiedenen Chararaktere, die aber (wie in Abschnitt 2.3.3.5 erwähnt) von Aufgabenbereich und Interaktionsverhalten völlig identisch sind. Tabelle 12 gibt nun Empfehlungen für die Wahl geeigneter Charaktere für verschiedene Aufgaben von Agenten. Zu beachten ist, dass diese Hinweise unverbindlich sind, denn, wie bereits gesagt, die Erwartung des Benutzers ist ausschlaggebend für die Wahl der Erscheinungsform.

Tabelle 12: Zusammenhang zwischen Agentenmetapher und Erwartungen des Benutzers in die Fähigkeiten des Agenten

Metapher	Erwartung des Benutzers	Geeignete Anwendungsbereiche
Zauberer	Verfügt über für unmöglich gehaltene, überraschende Fähigkeiten	Makrorecorder, Hilfefunktion (Problemlösungswissen)
Einstein	Geniale rationale Leistungen	Computeralgebrasysteme, komplexe naturwissenschaftliche Systeme
Engel	Schutzengel schützen auf unauffällige Weise vor Gefahren (beispielsweise vor Falscheingaben oder deren Folgen), helfen bei Problemen.	Unübersichtliche oder (mit Einschränkungen) sicherheitskritische Systeme
Mensch	Kann komplexe Aufgaben intelligent ausführen, ist Arbeitskollege, Partner.	Benutzermodellierung, intelligente Hilfe-Funktion
Diener, Butler, Hund	Führt Anweisungen aus, befolgt Befehle.	Einfache periodische Retrievalaufgaben
Roboter, Insekt	Führt simple Anweisungen aus.	Terminwarner, Mailfilter

Weiterhin sollte das Erscheinungsbild der Relation zwischen Mensch und Agent (siehe Abschnitt 4.3.4 der Delegation) entsprechen. Es ist unnatürlich, von einer Biene einen Expertenrat zu erhalten oder einen Zauberer zum Filtern von Mail abzustellen.

Selbstverständlich sollte das *Erscheinungsbild unaufdringlich und optisch ansprechend* sein. Der Einsatz vollständig animierter Präsentationsagenten kann geeignet sein, um dem Benutzer durch Sprache, Gestik oder Mimik innere Zustände und Tätigkeiten des Agenten auf anschauliche Weise zu vermitteln. Im Interesse von Benutzern, die visualisierten Agenten kritisch gegenüberstehen, sollte die *Darstellung des Agenten-Charakters ausblendbar* sein.

Erwartungen des Benutzers von Charakteren können kulturspezifisch differieren: Unter Umständen sind bestimmte Vertreter, beispielsweise Engel, schlicht unbekannt oder lösen negative Reaktionen aus: Zauberer können mit Okkultismus in Zusammenhang gebracht werden, einzelne Tierarten als „unrein“ oder heilig gelten. In jedem Fall sollte eine hohe Intelligenz suggerierende Erscheinungsform vermieden werden, wenn die Fähigkeiten und Aufgaben des Agenten verhältnismäßig wenig komplex sind. Menschen tendieren sehr leicht dazu, die Aktionen von Programmen (oder von Tieren) als „intelligent“ zu interpretieren und sie zu vermenschlichen (Norman 1997). Besonders im Fall anthropomorpher Agenten führt dies leicht dazu, dass der Benutzer dem Agenten menschenähnliche Fähigkeiten zuschreibt, eine Überschätzung, die nach kurzer Zeit enttäuscht werden muss. Norman weist darauf hin, dass die Wahl menschlicher Erscheinungsformen zu moralischen Problemen führen kann. Andererseits kann ein über entsprechende Fähigkeiten verfügender anthropomorpher Agent einen in hohem Maße natürlichen und nuancierten Dialog mit dem Benutzer führen (siehe auch Laurel 1990, 358f). Als Ausweg bieten sich - erneut - *alternativ angebotene Erscheinungsformen* des Agenten an, beispielsweise Tierarten wie Eulen oder Delphine, die Intelligenz symbolisieren. Problemlos einsetzbare Charaktere sind Roboter und Flaschengeister, da sie die Konnotationen niedrige Intelligenz und umfangreiche Fähigkeiten kombinieren.

Tabelle 13: Beispiele für die Verwendung von Charakteren in Agentensystemen

Charakter	Agentensystem
Zauberer	Microsoft Agent, LifestyleFinder
Einstein, Shakespeare	Microsoft Office Assistant
Engel	Virtual-Reality-System[53]
Mensch	BargainFinder, Persona
Flaschengeist	Microsoft Agent
Roboter	Microsoft Agent, Microsoft Office Assistant
Hund	Agentware, Microsoft Office Assistant, Webdoggie
Programm-Logo, Ball, Katze, Erdkugel, Büroklammer	Microsoft Office Assistant
Grashüpfer	LiveAgent

Abschließend zeigt Tabelle 13 eine Übersicht über in aktuellen Agentensystemen verwendete Charaktere. Diese können als Anschauungsmaterial für die Entwicklung eigener Agentencharaktere dienen.

5.2.2.2 Komfortable Ein- und Ausgabemedien

Der Agent sollte *mehrere gängige Eingabeformen unterstützen*, beispielsweise *text-, maus- und sprachbasierte Eingaben*. Sind zahlreiche Anfragen des Agenten durch Auswahl zwischen vorgegeben Optionen zu bearbeiten, bietet sich eine mausgesteuerte Eingabe für den Benutzer an. Eine Sprachsteuerung unterstützt dagegen besonders gut das Konzept der Delegation, wenn der Agent nur wenige Informationen zum Arbeiten benötigt. Weiterhin bietet sich Spracheingabe für durch Dialoge gekennzeichnete Teamarbeit von Benutzer und Agent an. Auf der Ausgabeseite kann der Agent zwischen *Text, Tönen, Grafik, Animationen und Sprachausgabe* wählen (siehe Tabelle 14).

Tabelle 14: Sinnvolle Abstimmung des Ausgabemediums auf Interaktionsstil und -ziel

Medium	Sinnvolle Anwendungsgebiete
Text	Retrievalagenten, Recommender: Übersicht über große Mengen von Daten
Animation, Grafik	Warnungen, Warten auf Eingabe
Ton, Geräusch	(dringende) Warnungen, Feedback
Sprache	Erklärung, Zusammenarbeit mit dem Benutzer, Dialog

Die am besten geeignete Methode kann nur im Kontext der Aufgabe bestimmt werden. Akustische (Alarm-)Signale können eine weitaus stärkere Wirkung haben als optische Signale. Sie eignen sich allerdings nicht zur asynchronen Benachrichtigung, sind ungeeignet für datenintensive Meldungen, und besonders sprachlichen Meldungen können vom Benutzer nicht gleichzeitig, sondern nur nacheinander verarbeitet werden. Dem Komfortideal nahe kommt hier der Microsoft Agent, der animiert ist, Texte und Töne ausgibt sowie über Spracheingabe und -ausgabe verfügt.

5.2.3 Komplex 3: Die Aufgabe

Dieser Komplex schließt den Bogen zur aufgabenorientierten Untersuchungsmethode von Agenten aus Abschnitt 2.4. Zunächst gibt Tabelle 15 eine Zusammenfassung zu aufgabenbezogenen Umsetzungshilfen bei Softwareagenten.

[53] Im Unterschied zu den anderen, in realen Systemen eingesetzten Charakteren erscheint ein Engel in dem Film „Enthüllung“ als Agentenaufsatz auf ein Virtual-Reality-System. Dort demonstriert er auf überzeugende Weise seine schützenden und helfenden Fähigkeiten.

Tabelle 15: Umsetzungshilfen zum Komplex „Aufgabe“

	Arbeitsaufteilung	**Spaßfaktor**
Richtlinien	• Benutzer muss verantwortungsvolle Aufgabenanteile behalten. • Aufgabenanteile Benutzer und Agent sollen klar getrennt sein.	• In Arbeitssystemen sind Spaßfaktoren nicht angebracht. • Spaßfaktoren müssen abschaltbar sein. • Computerspiele, Lernsoftware dürfen Spaßfaktoren besitzen
Fragen	Behält der Benutzer verantwortungsvolle Aufgaben?	
Methoden	• Voruntersuchung, ob der Benutzer über- oder unterfordert ist. • Belastungsoptimierung	Untersuchung, inwieweit ein Spaßfaktor motivierend auf den Benutzer wirkt.
Beispiele	• Aufgabe in Bereiche wie planen / vorbereiten / ausführen / evaluieren / nachbearbeiten teilen. Wer übernimmt welchen Teilbereich? • Agent bereitet vor / bereitet nach /arbeitet zu → gute Arbeitsteilung	Softwareagenten mit Spaßfaktor : Microsoft Agent, Office Assistant, Persona, togglethis interactive characters. (Alle genannten Beispiele sind *animierte* Agenten.)

Zum aufgabenbezogenen *Delegationsaspekt* „**Nur Teilaufgaben an Agenten abgeben“** siehe Abschnitt 4.3.3.

Der zentrale Aspekt der „Aufgabe“ ist es, auf *eine ausgewogene Arbeitsteilung zwischen Agenten und Benutzer* zu achten (siehe Abschnitt 4.3.3). Der Agent soll nur geeignete Teile der Arbeitsaufgabe bearbeiten. Weder soll er die gesamte Aufgabe alleine, noch dem Benutzer lediglich die „Reste“ überlassen.

Ob Software einen „Spaßfaktor“ haben darf, also ein humorvolles Erscheinungsbild oder Verhalten des Agenten, hängt vom ihrem Einsatzgebiet ab.[54] Ein Spaßfaktor kann bei Lernsoftware angebracht sein, wenn dadurch der Benutzer auf spielerische Weise motiviert wird. Ist das System ein Computerspiel, dann steht der Aspekt Gebrauchstauglichkeit erst an zweiter Stelle, und Späße sind generell erlaubt. In Arbeitssystemen dagegen steht Gebrauchstauglichkeit im Vordergrund, und dazu können lustige Elemente nur sehr wenig beitragen. Allenfalls in der Lernphase können sie motivierende Vorteile bringen. Das Ziel, den Benutzer auf Dauer Freude bei seiner Arbeit erfahren zu lassen, wird wesentlich dagegen durch dessen Arbeitsumfeld bestimmt, seine Aufgaben, Kollegen und Arbeitsorganisation. Die Verbindung zwischen Arbeitssystemen und Spaß (Freude) kann allenfalls eine indirekte sein: Der Benutzer freut sich über das einwandfreie Funktionieren seiner Anwendung.

5.2.4 Komplex 4: Das Verhältnis Benutzer-Agent

In diesem abschließenden Komplex, dessen Umsetzungshilfen in Tabelle 16 zusammengefasst sind, werden Aspekte der Delegation aus Abschnitt 4.3.4 unter dem Oberpunkt „Gestaltung der Interaktion“ aufgegriffen.

[54] Animierte Agenten wie der Office Assistant oder vor allem der Microsoft Agent zeichnen sich durch 'lustige' Erscheinungsformen aus.

Tabelle 16: Umsetzungshilfen zum Komplex „Verhältnis Benutzer-Agent“

	Komfortable Interaktion mit dem Agenten (Abschnitt 5.2.4.2)	**Steuerbarkeit** (Abschnitt 5.2.4.3)
Gestaltungsrichtlinien (Abschnitt 3.5)	Komfort, Erwartungskonformität, Steuerbarkeit, Conceptual Models, Progressive Disclosure, Visibility, Feedback	Steuerbarkeit sowie weitere Aspekte der Kontrolloptimierung (Hauptforderung 2): Komfort, Erwartungskonformität, Conceptual Model, Progressive Disclosure, Visibility, Affordances + Constraints
Richtlinien	Instruktionsmethode des Agenten muss dessen Aufgabe angepasst sein (siehe Tabelle 17).	• Möglichkeit der Kontrolle ist wichtiger als Autonomie des Agenten. • Benutzer soll „Gefühl der Kontrolle“ behalten.
Beispiel	• Programming-by-Example • Abstraktes Verhalten vorgeben • Anpassung von Templates: Autonomy • Proaktive Agenten: Mailfilter	„Kontroll“-Modus
Weiterführende Literatur	• Programming-by-Example: Cypher (1993) • Abstraktes Verhalten vorgeben: SodaBot (1997)	Norman (1997): „Gefühl der Kontrolle“
	Gestaltung der Interaktion (Abschnitt 5.2.4.1)	**Feedback** (Abschnitt 5.2.4.4)
Gest.Richtl. (A. 3.5)	Komfort	Feedback, Visibility
Richtlinien	• Problemnahes Arbeiten soll ermöglicht werden. • Agentenunfahrene Benutzergruppen sollen berücksichtigt werden.	• Benutzer immer über Aktionen und Zustände des Agenten informieren. • Feedback soll anpassbar, vollständig, kontinuierlich und selbsterklärend sein können. • Agent soll Vorhersagen über seinen Ablauf machen können.
Beispiel	Keine Bevormundung des Benutzers (siehe Steuerbarkeit).	Progress Bar, simulierte Testläufe, Vorabinformationen

Zu *Delegationsaspekten* bezüglich des Verhältnisses Benutzer zu Agent siehe Abschnitt 4.3.4: **Steuerbarkeit, Feedback, Relation** und **Kapselung**

5.2.4.1 Gestaltung der Interaktion

Die optimale Interaktionsform eines Agenten hängt primär von seiner Aufgabe ab. Dennoch gibt es allgemeine Anforderungen an den Interaktionsstil des Agenten:

Softwareagenten sollen ein *problemnahes Arbeiten* ermöglichen. Deshalb darf der Benutzer nicht das Gefühl haben, am Agenten zu arbeiten, sondern gemeinsam mit dem Agenten am Problem. Dazu ist es erforderlich, all diejenigen internen Prozesse des Agenten auszublenden, die für den Benutzer nicht im Sinne eines erhöhten Verständnisses zur Transparenz des Agenten beitragen.

Die Dialoggestaltung sollte besonders *agentenunerfahrene Benutzergruppen berücksichtigen,* was eine im Vergleich zu konventionellen Programmen deutlich erhöhte Behutsamkeit, Intuitivität und Selbsterklärungsfähigkeit verlangt. Erwartungskonformität ist ein wichtiger Punkt. Wesentlich ist die Nachvollziehbarkeit der Aktionen des Agenten.

Der Interaktionsstil des Agenten sollte einerseits seinem Erscheinungsbild entsprechen (siehe letzter Abschnitt), darf aber andererseits *niemals bevormundend* sein. Diese Forderung wird im anschließenden Punkt *Steuerbarkeit* konkretisiert. Die Interaktion mit dem Agenten ist mit-

unter *inhaltlich* gewöhnungsbedürftig. Umso wichtiger ist eine *komfortable Interaktion* (siehe folgender Abschnitt) mit dem Agenten *an der Oberfläche.*

5.2.4.2 Komfortable Interaktion mit Agenten

Tabelle 17: Wahl der Instruktionsmethode in Abhängigkeit von der Aufgabe des Agenten

Methode der Instruierung	Sinnvolle Anwendungsgebiete
Programming-by-Example	Delegation, Macros
Abstraktes Verhalten vorgeben	Suchmaschine
Anpassung von Templates (direktmanipulative Oberfläche)	Filter
Keine Programmierung	Proaktive Agenten

Die Methode, mit der der Benutzer den Agenten steuert, muss sorgfältig und in Abhängigkeit von der Aufgabe gewählt werden. Tabelle 17 zeigt eine Reihe komfortabler Instruktionsmethoden und dazu passende Agenten. Bewusst weggelassen wurde die Methode der Programmierung durch zeilenweise Eingabe von Programmcode, wie sie in den meisten Agenten-Entwicklungsumgebungen[55] noch üblich ist. Denn nicht nur lässt sie jeglichen Komfort vermissen, sondern ist trotz des hohen Aufwands für den Benutzer, eine Agenten-Programmiersprache erlernen zu müssen, fehlerträchtig, wenig problemnah und kontraintuitiv. Besser ist *Programming-by-Example*, wobei der Benutzer dem Agenten die Aufgabe erst einmal vormacht, also quasi ein Macro aufzeichnet, dass vom Agenten nur noch wiederholt werden muss. Dies ist Delegation in der Form „Tu es so, wie ich es vorgemacht habe." Eine Fülle von Beispielen - auch von Agentensystemen - finden sich hierzu bei Cypher (1993).

Die Vorgabe des abstrakten Verhaltens als Instruktionsmethode benutzt SodaBot (1997), eine am MIT entwickelte, inzwischen allerdings eingestellte Agenten-Entwicklungsumgebung. Die Anpassung von Templates auf einer direktmanipulative Oberfläche ist dagegen bei dem Suchagenten Agentware möglich, der das Abrichten von Hunden als Metapher für den Prozess der Anfragespezifikation verwendet.

Besonders komfortabel (im Sinne der in Abschnitt 3.5.2 definierten Gestaltungsrichtlinie für Komfort) sind proaktive Agenten, die nie instruiert werden, sondern sich bei Problemen selbsttätig und mit einer Reihe von Lösungsvorschlägen melden, aus denen nur noch ausgewählt werden muss.

5.2.4.3 Steuerbarkeit[56]

Dieser Komplex kommt der in Abschnitt 2.5 aufgestellten Hauptforderung nach Kontrolloptimierung sowie der Hypothese aus 3.5.5 entgegen, die bereits eine Kontrolle „sämtlicher Aktionen eines Agenten" verlangt. Im Interesse all derer, die Agenten gerade erst kennenlernen oder ihnen nur bedingt trauen, muss einer solchen Forderung eine hohe Priorität eingeräumt werden, beispielsweise durch einen zusätzlichen „Kontroll"-Modus. Auch dass dabei der

[55] Beispiele für Entwicklungsumgebungen sind IBMs Aglets Workbench und AgentBuilder, Programmiersprachen wie D'Agents und TeleScript oder Klassenbibliotheken wie JATlite. Als eine Vorarbeit habe ich einen detaillierten tabellarischen Vergleich zwischen ca. zehn unterschiedlichen Agenten-Entwicklungsumgebungen vorgenommen. Dieser Vergleich ist Bestandteil der Diplomarbeits-Homepage.

[56] Dieser sowie der folgende Aspekt *Feedback* wurden wegen ihrer Wichtigkeit aus dem vorangegangenen Komplex *Interaktion* ausgegliedert.

Agent viel von seiner Autonomie und seinen Vorteilen insgesamt verliert und sich eher wie ein ferngesteuerter Schrittmotor verhält, muss in Kauf genommen werden.

Gestaltungsrichtlinien (siehe Abschnitt 3.5), um komfortable und weitestgehend steuerbare Agenten zu entwerfen, sind:

- Ein gutes (natürliches) Conceptual Model sorgt für ein gutes Verständnis des Benutzers davon, was der Agent tut und in welcher Weise er wohl steuerbar ist.
- Durch Progressive Disclosure werden Schritt für Schritt alle Steuermechanismen des Agenten zugänglich gemacht.

Hinter dem Ruf nach Steuerbarkeit steht erneut der Mangel an Vertrauen in das System. Der Benutzer will ein „Gefühl der Kontrolle“ behalten, ein nach Norman (1997) für Menschen generell wichtiger psychologischer Aspekt. Auch wenn kaum jemand diese Kontrollmöglichkeiten ausschöpft, trägt doch ihre schiere Existenz stark zur Beruhigung der Benutzer bei, geben sie ihm ein Gefühl der Sicherheit und Nachprüfbarkeit. Auf lange Sicht bildet sich dadurch beim Benutzer das in Hauptforderung 3 (Abschnitt 2.5) geforderte Vertrauen.

Steuerbarkeit und das Verbergen von Komplexität sind nicht zwangsläufig Gegensätze. Zunächst einmal sollen gerade diese Agenten einfach zu bedienen sein, deren Zweck das *Verbergen von Komplexität* ist. Alles andere stünde im Gegensatz zu ihrer Hauptaufgabe. Gleichzeitig sollten mit dem mächtigen Mittel *Progressive Disclosure* Zugänge zu einer schrittweise stärkeren Kontrolle des Agenten angeboten werden.

5.2.4.4 Feedback

Dieser Komplex ergänzt gleichermaßen die vorhergehenden Komplexe *Gestaltung der Interaktion* und *Steuerbarkeit*.

Niemals darf ein Agent den Benutzer unnötig über seine Aktionen und Zustände im unklaren lassen. Das wäre vor allem bei mit Kreditkarten oder Geld des Benutzers ausgestatteten Agenten inakzeptabel. Deshalb muss - ganz im Einklang mit der Hypothese zu Feedback aus Abschnitt 3.5.11 - das Feedback von Agenten *vollständig, kontinuierlich und selbstbeschreibungsfähig* (siehe Abschnitt 3.5.4) sein. Zu den beiden ersten Punkten ist einschränkend zu erwähnen, dass Feedback unter diesen Umständen umfangreicher als vom Benutzer gewünscht werden kann und daher möglichst hinsichtlich *Quantität, Auswahl und Periodizität anpassbar* sein soll.

Ein selbsterklärendes Feedback führt zu transparenten Handlungen: Der Agent soll nicht nur Auskunft geben, was er *getan* hat, sondern auch *warum*. Mit anderen Worten, er soll auf Verlangen seine Entscheidungsstrukturen und über den Nutzer gebildeten Persönlichkeitsprofile zugänglich und editierbar machen. Damit werden seine Handlungen für den Anwender transparent und nachvollziehbar.

Als in diesem Zusammenhang wichtige Agenten-Fähigkeit ist die der präzisen Abschätzungen zu fordern: Der Agent soll möglichst genaue Vorhersagen über Dauer, Geldeinsatz und Aufwand machen können - schon bevor er mit der Aufgabe beginnt.

Eine gute Feedback-Methode ist hier „Progress Bars“, die Zeitdauer, Fortschritt und Mitteleinsatz des Agenten kontinuierlich visualisieren. Durch simulierte Testläufe und Vorabdialoge kann sich der Benutzer ein Bild über Aufwand und Machbarkeit einer Aufgabe machen.

5.3 Möglichkeiten der Evaluation

Dieser Leitfaden dient zur *Neuentwicklung von Softwareagenten*. Wie bereits in Abschnitt 5.1.2 erwähnt, sind dazu Gestaltungsrichtlinien, wie sie der Leitfaden enthält, das richtige Mittel. Diese Richtlinien müssen *nicht notwendigerweise gut operationalisierbar* sein. Als Beispiel dient die aus Abschnitt 3.5.10 entnommene Forderung nach Constraints:

> Constraints sind die bewussten Beschränkungen der Anzahl und Art von Eingabemöglichkeiten bei Agenten.

Ist eine Evaluation des Agentensystems wesentlich, so müssen solche „weichen" Gestaltungsgrundsätze zu konkreten Kriterien umgewandelt werden (siehe auch Abschnitt 3.3, Fußnote 31). Dies könnte folgendermaßen geschehen**:**

> Während der Arbeit muss der Agent alle 60 Sekunden ein optisches und akustisches Feedback über seinen Status liefern.

Methoden, die den *Entwurf* unterstützen, sind in erster Linie die in Abschnitt 2.5 aufgestellten Hauptforderungen nach Lernförderlichkeit, Kontrolloptimierung und Vertrauensbildung und das in Abbildung 12 dargestellte System agentenrelevanter Gestaltungsgrundsätze.

Um harte Kriterien aufzustellen, ist die Orientierung an den in Abschnitt 3.5 aufgestellten Hypothesen sinnvoll, die aber im aktuellen Kontext geschärft und angepasst werden müssen. Der Aufwand für diese Arbeit darf nicht unterschätzt werden, denn - die Erkenntnis aus dem Bereich softwareergonomischer Normen (vgl. Dzida 1993, 16) hat auch hier Gültigkeit - das Aufstellen von guten Kriterien ist schwieriger als deren anschließende Überprüfung. Die Formulierung von Anforderungen an ein Agentensystem bereits in der Entwurfsphase ist die Voraussetzung für eine sinnvolle spätere Evaluation. Auf dieser Grundlage bietet sich als Evaluationsmethode beispielsweise die in Abschnitt 3.6 vorgeschlagene subjektive Evaluation an.

5.4 Zusammenfassung

Dieses Kapitel hat den Weg gewiesen, wie die bisherigen Erkenntnisse gebündelt werden können, um die Entwicklung agentenbasierter Software zu unterstützen: In einem Leitfaden, der das Konzept Delegation bei Agentensystemen umsetzt (aus Kapitel 4), der nach Delegationsaspekten aufgebaut ist, der im Agentenkontext transformierten HCI-Gestaltungsrichtlinien folgt (aus Kapitel 3) und der die einer Untersuchung von Agentensystemen entspringenden Hauptforderungen umsetzt (aus Kapitel 0).

Im folgenden Kapitel wird sich zeigen, ob sich dieser Leitfaden auch in der Praxis sinnvoll anwendbar ist.

6 Anwendung: Leitfadenorientierte Weiterentwicklung des Brokeringsystems bizzyB

Dieses Kapitel behandelt den ersten Einsatz des Leitfadens für Agentensysteme. Das Brokeringsystem bizzyB wird mit seiner Hilfe unter besonderer Berücksichtigung des Konzeptes Delegation weiterentwickelt und so für einen weiteren Nutzerkreis zugänglich gemacht.

6.1 Einleitung

bizzyB ist ein flexibles System zum Makeln von Online-Informationen, das an der GMD im Rahmen des COBRA-Projektes entwickelt wird. Ein Prototyp wurde für den Einsatz an der Mailänder Handelskammer durch professionellen Informationsmakler entwickelt. Diese Makler haben die Funktion von Dienstleitern, deren Kunden Unternehmen sind, die Firmenkontakte suchen. Sei es, dass ein regionales Unternehmen internationale Investoren sucht oder ein nicht-italienisches Unternehmen neue Absatzmärkte in der Mailänder Region erschließen will - in beiden Fällen können die Informationsmakler weiterhelfen. Dazu greifen sie auf die umfangreichen Firmeninformationen verschiedener externer und hauseigener Datenbanken zurück.

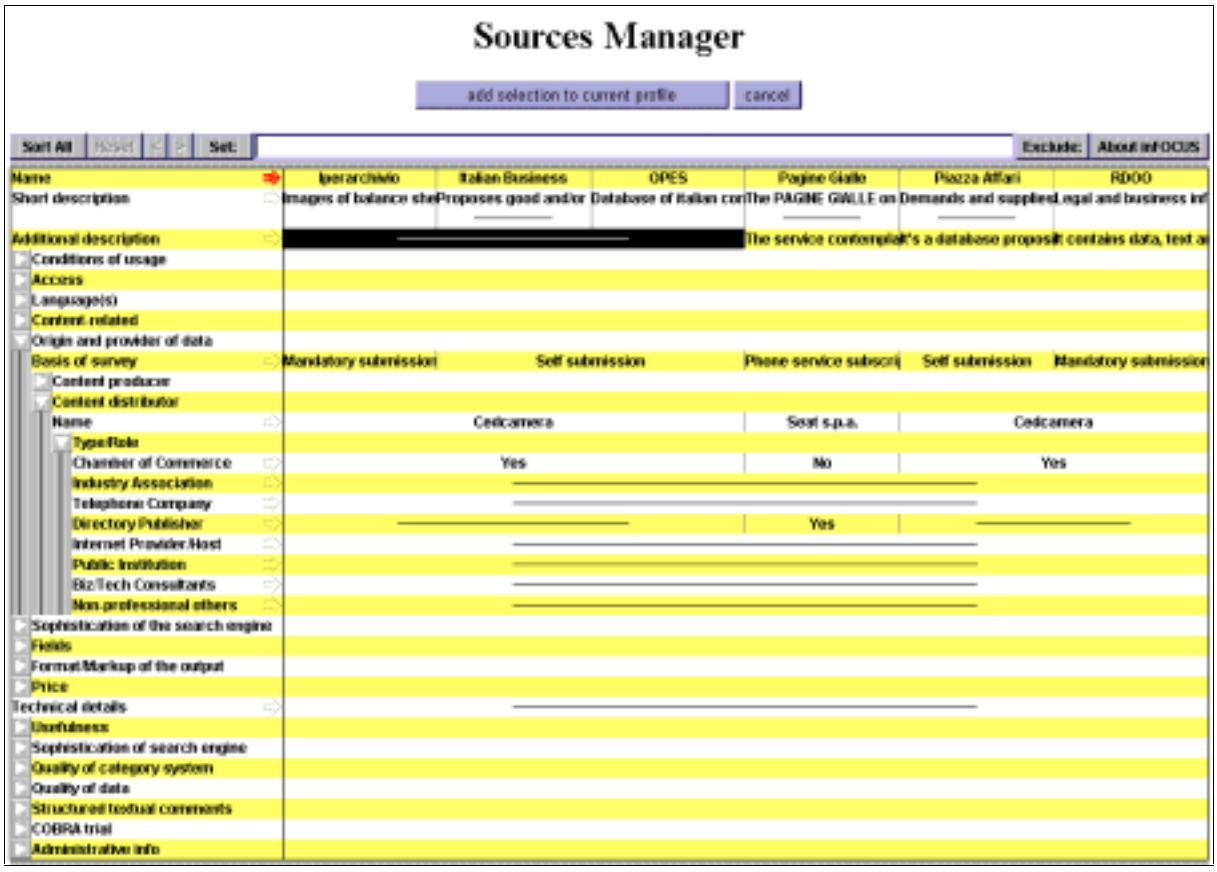


Abbildung 29: Verwaltung der für bizzyB erreichbaren Datenquellen

Bei ihrer Arbeit begegnen diese Makler einigen wiederkehrenden Schwierigkeiten:

- Schriftliche Anfragen, die per Post, Telefax oder E-Mail eintreffen, sind häufig schlecht formuliert, enthalten unpräzise formulierten Anfragen und informieren nur ungenügend über den Nachfrager.
- Es existiert eine Vielzahl von Datenquellen, deren Informationen unterschiedlich umfangreich, präzise und aktuell sind. Einige Datenquellen sind kostenpflichtig, viele verwenden eigene Anfragesprachen. Es ist viel Erfahrung auf Maklerseite nötig, um qualitativ gute Antworten auf externe Anfragen zu liefern.
- Die verwendeten Datenquellen lassen sich nach Kategorien abfragen, die entweder die Form von Codes oder Stichwörtern haben. Unglücklicherweise lassen viele Datenquellen nur ein einziges (oft ihr eigenes) proprietäres System von Kategorien als Anfrage zu. Da die Datenquellen nicht vernetzt sind, ist paralleles Suchen in mehreren Quellen nicht möglich. Zudem muss für jede Anfrage in einem neuen System die entsprechende Kategorie vom Broker (teilweise in Büchern) nachgeschlagen werden.

bizzyB erleichtert nun die Aufgabe der Broker insofern, indem es eine integrierte Schnittstelle anbietet, um parallel in mehreren Datenquellen suchen zu können, keine Abfragesprachen und nur noch eine einzige Klassifikation erlernen zu müssen. In einer anderen Systemkomponente können Broker Zusatzinformationen über sämtliche verfügbare Datenquellen erhalten und geeignete für eine Anfrage auszuwählen (siehe Abbildung 29).

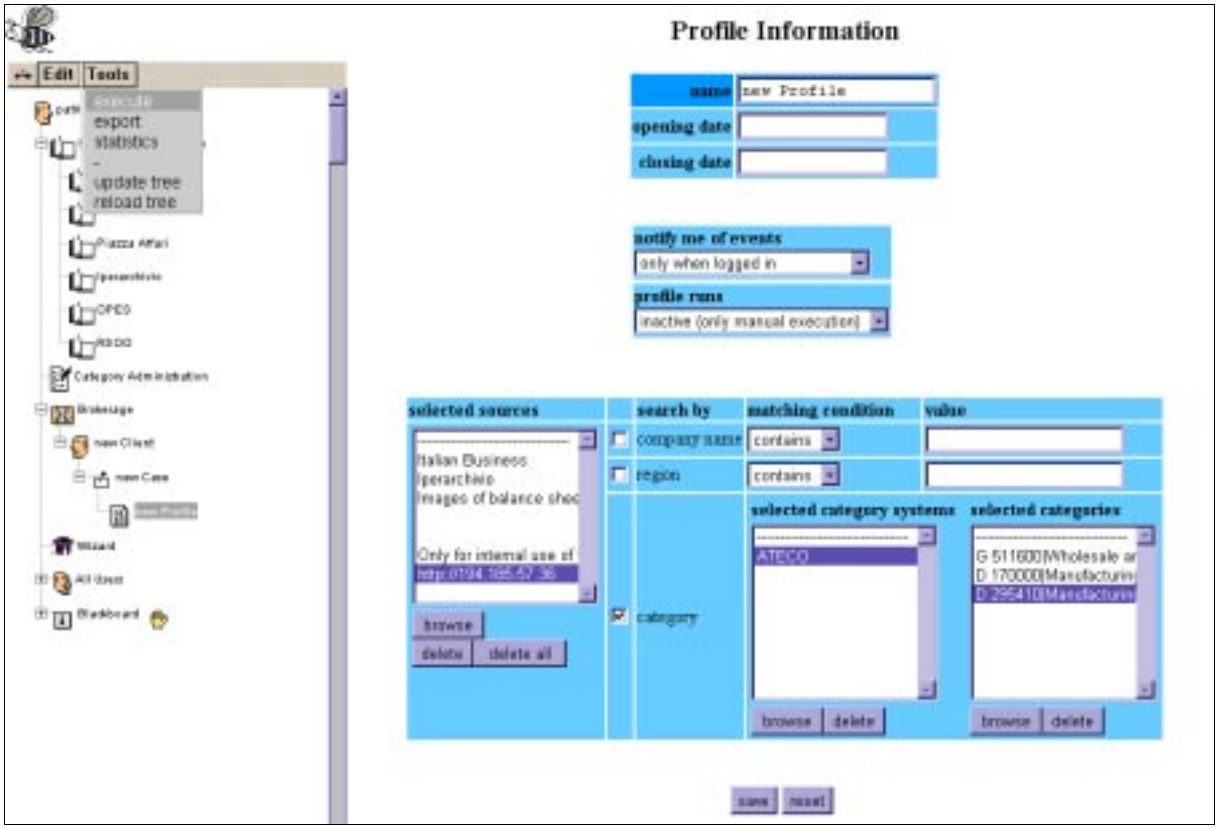


Abbildung 30: Erstellung eines Suchprofils mit bizzyB

Information Broker legen für jeden Kunden ein *client*-Objekt an, das mehrere, als *cases* bezeichnete annotierbare Interessensfelder des Kunden enthalten kann. Die eigentlichen, den Interessensfeldern zugeordneten Suchanfragen (*profiles*) sind, vereinfacht ausgedrückt, eine Kombination relevanter Datenquellen, Suchkategorien und Suchstrings (siehe Abbildung 30), die bizzyB in vom Broker festgelegten Abständen periodisch ausführt. Suchergebnisse legt das System gebündelt in sogenannten *Dossiers* ab, welche an den Kunden weitergeleitet werden können. Zusätzliche Funktionen erlauben es, andere Broker um Rat zu fragen und Teile ihrer (niedergeschriebenen) Expertise anderen Brokern zugänglich zu machen.

Für die Zukunft ist eine *Erweiterung des Nutzerkreises von bizzyB vorgesehen, so dass die Endversion folgende Nutzergruppen umfassen* wird:

Anbieter ◄——► **Broker** ◄——► **Kunde**

- Da bizzyB keine besseren Informationen liefern kann, als zuvor von **Anbietern** eingegeben wurden, ist die vollständige, präzise und korrekte Informationserfassung auf Anbieterseite vonnöten. Weil sie auch eigene Datenbestände an Firmeninformationen aufbaut, sind Mitarbeiter der Mailänder Handelskammer über ihre Brokerrolle hinaus auch Anbieter von Informationen. In dieser Funktion würden sie von einem für die Datenerfassung erweiterten und optimierten bizzyB besonders profitieren.
- **Informationsbroker** sind die Hauptbenutzer von bizzyB, arbeiten professionell und intensiv mit dem System. Für sie ist ein robustes, adaptierbares und schnelles System wichtiger als eine einfache Bedienung.
- Der **Kunde** ist Laie, bestenfalls gelegentlicher kompetenter Nutzer von bizzyB. Für ihn ist eine intuitive und einfache Benutzerführung ausschlaggebend. Auf keinen Fall soll er durch ein Übermaß an Funktionen verwirrt werden. Wichtiger ist, ihn zunächst – durch Constraints sowie das Weglassen weniger wichtiger Funktionen – schrittweise durch das Programm zu führen. So lange, bis er das komplexe Conceptual Model verstanden hat. Zur Zeit haben Kunden jedoch keinen Zugang zu bizzyB.

6.2 Notwendigkeit einer verbesserten Schnittstelle für semiprofessionelle Broker

Eine Erweiterung des Nutzerkreises von bizzyB wird angestrebt. Zukünftig sollen außer professionellen Brokern an der Mailänder Handelskammer, die täglich mit dem System arbeiten, auch *gelegentliche Nutzer* eingebunden werden. Beispielsweise sollen auch Sekretärinnnen einfache Anfragen mit dem bizzyB-System bearbeiten können. Vorteile erhofft man sich einerseits durch die Entlastung der bisherigen Broker von weniger anspruchsvollen Anfragen, andererseits aber durch die Erhöhung des potentiellen Anwenderkreises. Um diese neuen Benutzer auch ohne Hilfe erfahrener bizzyB-Broker optimal an das System heranzuführen, *muss die Lernförderlichkeit erhöht werden***.**

Die bisherige Benutzeroberfläche ist auf die Bedürfnisse professionelle Broker abgestimmt, was die Komplexität der Schnittstelle rechtfertigt. Die neue Zielgruppe wäre damit allerdings überfordert. Bei ihr können Fertigkeiten im Umgang mit Computern vorausgesetzt werden,

die etwa denen von Geschäftsleuten entsprechen, die über eine gewisse Vertrautheit mit Computern im allgemeinen und dem WWW im speziellen verfügen.

Ein erfolgversprechender Ansatz ist zunächst die wenig aufwendige Vereinfachung des aktuellen Systems, z.B. durch ein *intuitives Conceptual Model*, um das Verständnis besonders des unerfahrenen Benutzers entscheidend zu steigern. Ein zweiter, ergänzender Ansatz ist die verbesserte Unterstützung des Benutzers. Also sind Verbesserungen der Schnittstelle nötig, die *das existierende System vereinfachen* und den *Benutzer zusätzlich unterstützen*. Während beide Ansätze durch Delegation umzusetzen sind, zeichnet sich der erste Ansatz durch Weglassen, der zweite durch Hinzufügen von Systemelementen aus. Unter Verwendung des Leitfadens wird nun diese Aufgabe angegangen.

6.3 Aufstellung von Forderungen an die verbesserte Schnittstelle

Das Ziel ist, den aktuellen Mailänder bizzyB-Prototypen für die Bedürfnisse der neuen Benutzergruppe semiprofessioneller Broker zu optimieren, welche das System erlernen bzw. *erneut* erlernen. Zu diesem Zweck folgt nun eine Aufstellung (möglichst messbarer) Forderungen an die neue Oberfläche.[57]

1. **Benutzer müssen einen intuitiven, möglichst einfachen Zugang zum System erhalten.** Vor allem in der Lernphase ist es wichtig, die Einstiegshürde für neue Benutzer zu minimieren. Dies soll durch ein alternativ angebotenes vereinfachtes Conceptual Model geschehen. Denn das Conceptual Model des Business-to-Business-Brokeringsystems ist zu komplex für normale Benutzer. Selbst die Entwickler des aktuellen bizzyB-Systems hat es viel Zeit gekostet, dieses Modell zu internalisieren**.**
2. **Hilfs- und Orientierungsmöglichkeiten müssen verbessert werden**. Denn es ist anzunehmen, dass die neuen Nutzer das System niemals vollständig erlernen und deshalb immer wieder Funktionen erneut lernen müssen
3. **Arbeitsanteile sollen an einen Agenten delegierbar sein.** Speziell sollen Anzahl und Komplexität der nötigen Schritte von der ersten Anmeldung bis zu den Ergebnissen einer Anfrage durch Delegation minimiert werden.

6.4 Beschreibung der notwendigen Entwicklungen

Um die im vorausgehenden Abschnitt aufgestellten Forderungen umzusetzen, werden eine Reihe von Ergänzungen zu bizzyB vorgeschlagen. Diese sind in Abbildung 31 schematisch dargestellt und werden in den folgenden Abschnitten. leitfadenorientiert entwickelt.

[57] Durch die Mitarbeit seit Beginn des Projektes sind bereits viele meiner Vorstellungen, darunter auch im Leitfaden erwähnte Methoden, in das System eingeflossen. Trotzdem fanden sich genügend Ansatzpunkte für Verbesserungen am aktuellen System.

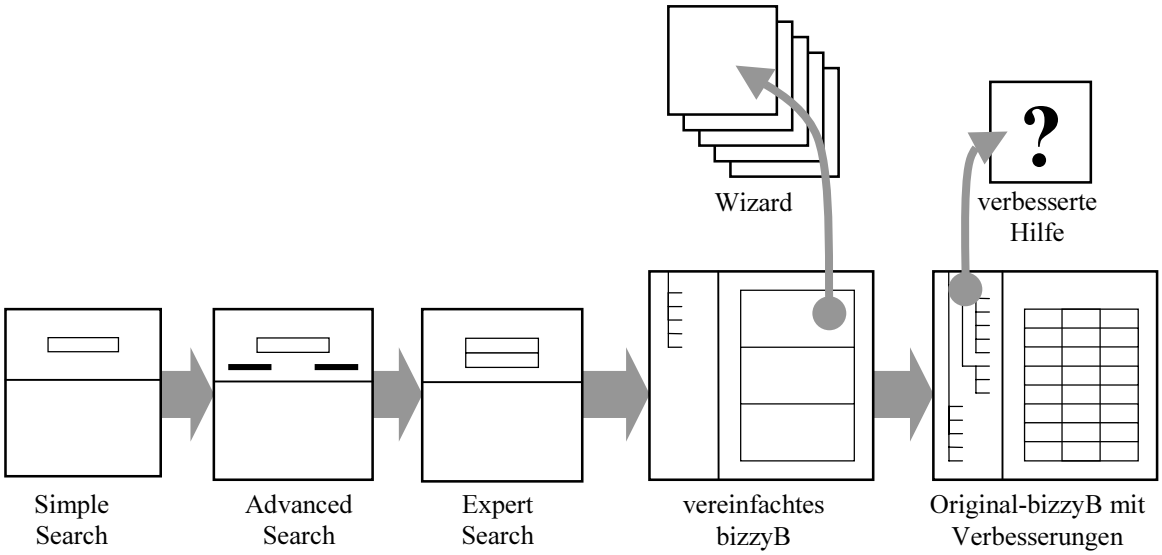


Abbildung 31: Übersicht und Zusammenhang der Verbesserungen bizzyBs

Die vorgeschlagenen Verbesserungen gruppieren sich rund um das bisherige bizzyB-System (rechts im Bild). Um die wesentlichen Funktionen von bizzyB zu erlernen, wird ihm zunächst ein in Abschnitt 6.4.1 beschriebenes System vorgeschaltet, welches das Brokeringsystem über das vereinfachende Conceptual Model einer Suchmaschine präsentiert. Indem der Benutzer auf mehreren Stufen (Simple Search, Advanced Search, Expert Search) durch zunehmend komplexere Anfragen das System exploriert, soll er allmählich ein Verständnis von der Basisfunktionalität bizzyBs gewinnen. Ist dieser mentale Schritt vollzogen, präsentiert sich das „eigentliche" bizzyB-System dem Benutzer und bietet mehrere „Wizards" (beschrieben in Abschnitt 6.4.2) an, die jeweils eine typische Brokeringaufgabe schrittweise und exemplarisch dem Benutzer nahebringen. Danach erfolgt ein fließender Übergang zu der üblichen Darstellung bizzyBs, das jedoch um zahlreiche Hilfefunktionen und delegierbare Aspekte ergänzt wurde (siehe Abschnitt 6.4.3). Beispielsweise werden die zunächst ausgeblendeten Elemente des Menübaums allmählich wieder eingeblendet und somit der Zugang zu den dahinterliegenden Funktionen ermöglicht.

Während der Schwerpunkt des neu entwickelten Conceptual Models auf der ersten Forderung nach **Vereinfachung** liegt, ist die zweite Forderung nach **Unterstützung** das Hauptmotiv für die übrigen Ergänzungen und Modifikationen. *Allen Verbesserungen gemeinsam* ist die **Delegation** von Aufgabenbereichen, wie sie in der dritten Forderung formuliert wurde.

6.4.1 Beschreibung eines vereinfachten Conceptual Models

Um vor allem die Forderung nach einem intuitiven Zugang zu bizzyB zu verwirklichen, ändere ich das Erscheinungsbild (Leitfaden, Abschnitt 5.2.2.1) bizzyBs dergestalt, dass das Conceptual Model der Suchmaschine kommuniziert wird. Dem Benutzer präsentiert sich bizzyB nicht als Brokeringsystem, sondern als *eine auf Firmeninformationen spezialisierte, besonders mächtige Suchmaschine*. Eine Vertrautheit mit diesem „einfachen" Konzept kann bei der Zielgruppe vorausgesetzt werden. Sämtlicher „Ballast" wie Anmelden und Profilerstellung entfallen, stattdessen kann, wie in Abbildung 32 dargestellt, durch direkte Eingabe des Such-

begriffes, beispielsweise der gesuchten Kategorie, unmittelbar auf eine Datenquelle zugegriffen werden. Somit ist es dem Benutzer möglich, sich durch wiederholtes Ausprobieren rasch ein Bild über die Qualität der vom System gelieferten Daten zu machen. Umgekehrt wird vermieden, den Benutzer zur Eingabe zahlreicher Profilinformationen zu zwingen, obwohl dieser keine Möglichkeit hat, die Relevanz dieser Informationen im einzelnen nachzuvollziehen.

Um die Lernförderlichkeit zu unterstützen, offenbart die Suchmaschine ihre gesamte Funktionsvielfalt nur schrittweise. Tabelle 18 illustriert, wie dies durch fortgesetztes Progressive Disclosure in mehreren Schritten erreicht wird.

Tabelle 18: Das Conceptual Model Suchmaschine in drei Komplexitätsstufen

Für den Benutzer nicht erkennbar werden Anfragen an das bisherige bizzyB-System weitergeleitet. Dazu werden alle im Unterschied zum Originalsystem „weggelassenen“ Schritte automatisch an „Simple Search“ delegiert: Anmeldung beim System mit einer individuellen Kennung, Erstellung der für eine Suchanfrage nötigen Objekte, insbesondere eines spezialisierten Profils. Noch nicht integriert ist eine Hilfsfunktion, die zweifellos auch für diese wenig komplexe Software nötig ist. Durch Progressive Disclosure (Leitfaden, Abschnitt 5.2.4.1), erkennbar an den Schaltflächen „Advanced Search“ bzw. „Expert Search“ in der rechten oberen Ecke des Bildschirms, soll der Benutzer dazu animiert werden, eigenständig explorierend weitere Systemfunktionalität zu erlernen. Dem Benutzer bleibt die Möglichkeit, jederzeit wieder zu einer einfacheren Stufe der Suchmaschine zurückzukehren.	 **Abbildung 32: Komplexitätsstufe 1: Suchmaschinenmetapher „Simple Search“**
Die in Abbildung 33 dargestellte erweiterte Suchmaschine „Advanced Search“ erlaubt neben der bekannten Suche nach Stichwörtern die Angabe einer speziellen Datenquelle und die Spezifizierung einer Menge von Suchkategorien. Damit soll dem Benutzer die Lernphase erleichtert und die Möglichkeit geboten werden, ohne Bevormundung herauszufinden, inwieweit die Verfeinerung einer Suchanfrage tatsächlich zu verbesserten Suchergebnissen führt. Die grafische Gestaltung dieser Stufe ist - bis auf die zusätzliche Funktionalität - konsistent mit der ersten (und auch der folgenden) Stufe.	 **Abbildung 33: Komplexitätsstufe 2: Erweiterte Optionen „Advanced Search“**

In der als „Expert Search“ bezeichneten dritten Stufe der Suchmaschine (dargestellt in Abbildung 34) wird erstmals die Eingabe der in bizzyB obligatorischen Benutzerkennung verlangt. Um dem Benutzer diesen möglicherweise als „lästig“ empfundenen Schritt zu erleichtern, wird auf bisher nicht erreichbare Suchfunktionen in Expert Search hingewiesen: Das parallele Suchen in mehreren Datenquellen, die Suche in kommerziellen Datenbanken sowie die Möglichkeit, Profile anzulegen und automatisiert auszuführen.

Nach dem Anmeldeschritt erwartet den Benutzer keine weitere, noch mächtigere Suchmaschine, sondern die volle Funktionalität des bizzyB-Systems, in dem zunächst die im folgenden Abschnitt beschriebenen Wizards angezeigt werden.

Abbildung 34: Komplexitätsstufe 3: Anmeldung für „Expert Search“

6.4.2 Beschreibung der Wizards

Durch die neu entwickelte Komponente „Wizards“ wird *durch geführte Benutzerdialoge die Systemfunktionalität anhand typischer Beispiele vermittelt*. Die Wizards stellen bereits ein Element der verbesserten Hilfefunktion dar. Wie bereits in Abschnitt 2.3.3.5 angesprochen, haben Wizards Agentenfunktionalität und sind deshalb für den Einsatz in Agentensystemen besonders geeignet (siehe Leitfaden, Abschnitt 5.2.1.1). Sie haben folgende Aufgaben:

- Sie decken typische Anfragen ab, mit denen Kunden an die Benutzer von bizzyB herantreten. Somit gewinnt der lernende Benutzer einen ersten Überblick über die Fähigkeiten und Anwendungsbereiche von bizzyB.
- Sie ermöglichen durch beispielhaftes Lernen die Entwicklung eines Verständnisses für Information Brokering-Systeme.
- Sie leiten den Benutzer schrittweise durch seine Aufgabe, bewahren ihn vor falschen oder irrelevanten Aktionen. Teile seiner Aufgabe werden dabei an den Wizard delegiert.

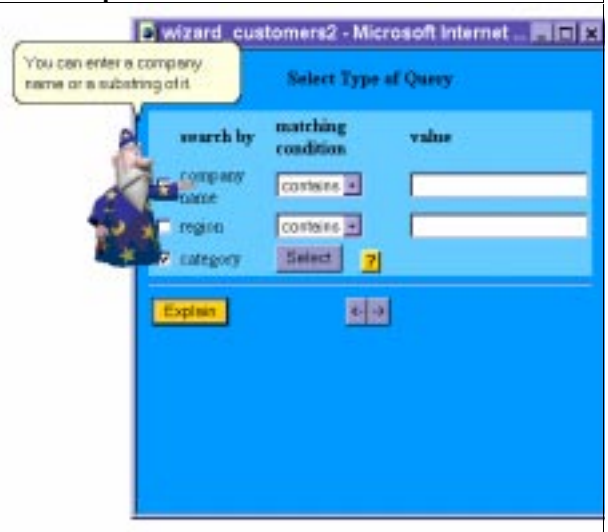
Abbildung 35: Startbildschirm mit Auswahl und Erläuterung der Wizards

Folgende drei Standardszenarien haben sich als für Mailänder Information Broker typische Aufgaben herausgestellt:

1. Ein Anbieter will den Absatz seiner Produkte erhöhen und benötigt zu Marketingzwecken möglichst viele, relevante und aktuelle Adressen potentieller Kunden.
2. Ein Unternehmen möchte eine Kooperation mit einem anderen Unternehmen eingehen. In diesem Fall sind präzise, detaillierte Informationen über einem kleinen Kreis möglicher Partner wichtig.
3. Möglichst umfangreiche Informationen über ein bereits bekanntes Unternehmen sollen eingeholt werden.

Jede dieser Standardsituationen wird, wie Abbildung 35 zeigt, durch einen Wizard exemplarisch vorgeführt werden. Von diesen Wizards möchte ich nun den Ablauf des ersten Wizards in der folgenden Tabelle genauer erläutern.

Tabelle 19: Programmdurchlauf des Wizard Customer

Jeder Wizard öffnet ein separates kleines Fenster. Unterstützend wird der Microsoft Agent, ein frei erhältlicher, flexibel programmierbarer und komfortabel (Leitfaden, Abschnitt 5.2.2.2) zu bedienender Präsentationsagent, eingesetzt. Er erklärt die Bedeutung der aktuellen Bildschirmseite durch Wahl der gelben „Explain"-Schaltfläche, auf einigen Seiten auch spontan bei Seitenaufruf. Weiterhin gibt er Zusatzinformationen zu einzelnen Seitenelementen (Kontexthilfe) und arbeitet an gewissen Stellen mit dem Benutzer zusammen. Der Microsoft Agent verfügt über Sprachausgabe, die gleichzeitig in Sprechblasen angezeigt wird. Auf diesem Bildschirm ist die **Auswahl an Datenquellen** vorzunehmen bzw. die bereits voreingestellte Auswahl zu bestätigen.	 **Abbildung 36: Eingabe der Datenquellen**
Ein Beispiel für die Zusammenarbeit zwischen Präsentationsagenten und Benutzer: Der Benutzer wählt eine beliebige Datenquelle aus, worauf der Agent Zusatzinformationen zu dieser Datenquelle liefert. Alle Bildschirme des Wizards sind konsistent gehalten bezüglich Aufbau, Dimension, Farbgebung sowie Plazierung von Schaltflächen und Agent. Für ein Lernsystem ist der durch die Animationen des Agenten eingeführte Spaßfaktor vertretbar (Leitfaden, Abschnitt 5.2.3).	 **Abbildung 37: Wizard gibt Auskunft über bestimmte Datenquellen**
Dieser Bildschirm enthält die **Auswahl des Anfragetyps**: Nach Firmenname, Region oder Kategorien. Die Erscheinungsform des Agenten ist nicht optimal, weil der Benutzer von Zauberern gemäß dem Leitfaden „für unmöglich gehaltene, überraschende Fähigkeiten" erwartet (Abschnitt 5.2.2.1). Dies ist aber hier nicht der Sinn des Agenten, obwohl Testbenutzer durchweg über die Qualität der Sprachausgabe und der Animationen überrascht sind. Eine sich anbietende Erscheinungsform für ein bizzyB (busy bee = fleißige Biene) genanntes System wäre hier eine Biene. Der Entwurf einer Biene als Agentencharakter ist mit Microsoft Agent zwar möglich, aber sehr aufwendig. Da ein Zauberer aber geeignet ist, Problemlösungswissen zu vermitteln (Leitfaden, Abschnitt 5.2.2.1), was hier tatsächlich eine seiner Aufgaben ist, habe ich mich unter den auswählbaren Charakteren für ihn entschieden.	 **Abbildung 38: Eingabe des Anfragetyps**

Nur im Falle des Anfragetyps „Kategoriensuche“ müssen nun eine **Auswahl passender Kategorien** gewählt bzw. die voreingestellten Kategorien betätigt werden.

Der Vorteil eines Präsentationsagenten ist die Möglichkeit, sich frei auf dem Bildschirm bewegen zu können, während seiner Erklärungen auf gemeinte Bildschirmelemente zu zeigen und diese bei Bedarf zu aktivieren.

Choreographie und Interaktionsverhalten des Agenten wurde gemäß den Richtlinien zum Interaktionsentwurf des Microsoft Agents (Microsoft 1998, 9-26) entwikkelt, verwendet also „natürliche“ anstelle formeller Sprache, spricht in kurzen Sätzen und unterstützt seine Mitteilungen durch eine ganze Reihe von Gesten. In nebenstehender Abbildung benutzt der Wizard beispielsweise die „erklären“-Geste.

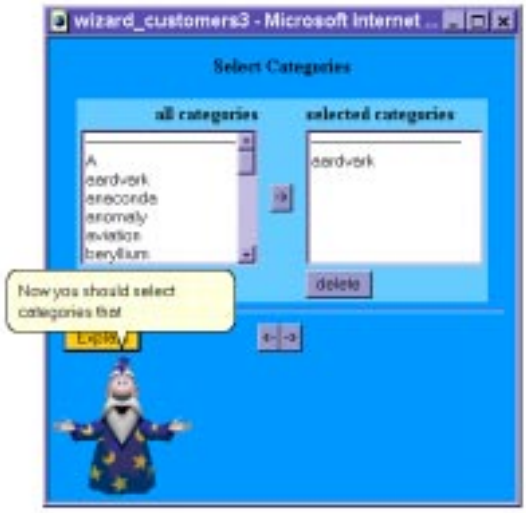


Abbildung 39: Auswahl geeigneter Kategorien

Der letzte Schritt der Profilerstellung beinhaltet die Angabe bzw. Bestätigung von **Zusatzinformationen** wie Zeitraum der Suchanfrage und Periodizität ihrer Ausführung.

Bis zu diesem Schritt kann zwischen den Bildschirmseiten beliebig vor- und zurückgeschaltet werden. Erst durch die Wahl der „Finish“-Schaltfläche wird die Anlage, Speicherung und Ausführung des Profils an den Agenten delegiert.

Abbildung 40: Eingabe zusätzlicher Profilinformationen

Die Ausgabe der Suchergebnisse zeigt Abbildung 41. Bewusste Constraints beschränken im Beispiel durch ein Popup-Menü die möglichen Antworten auf „ja“ oder „nein“.

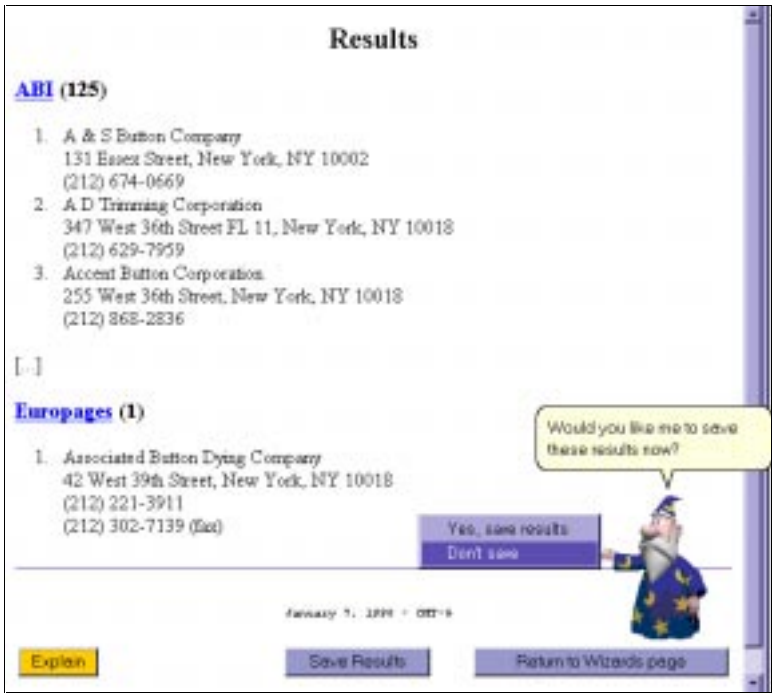


Abbildung 41: Ausgabe der Suchergebnisse

Die Qualität der Umsetzung des soeben beschriebenen ersten Wizards ist deutlich höher als die der beiden anderen Wizards. Diese reichen bezüglich Choreographie und Interaktionsreichtum nicht an den ersten Wizard heran.

6.4.3 Beschreibung weiterer Verbesserungsvorschläge zu bizzyB

Auch die nach dem Ausblenden der Wizards dargestellte „eigentliche“ Oberfläche bietet Verbesserungsansätze. Durch Progressive Disclosure und Constraints kann die Anzahl der Systemelemente im Vergleich zu der ursprünglichen Oberfläche (siehe Abbildung 30) stark reduziert werden:

1. Für **unerfahrene Benutzer** wird der im linken Fenster (Frame) des Originalsystems sichtbare Baum soweit beschnitten, dass lediglich die Profiles angezeigt werden, aber keine Hierarchie von Cases und Clients. Sinnvoll wäre die Einblendung aller momentan mit dem System verbundenen Broker (nicht aller *registrierten* Broker wie im Originalsystem), damit sie der Benutzer bei Problemen um Rat fragen kann. Erfolgreiche Tests vorausgesetzt kann auch das Blackboard, auf dem verschiedene Broker gleichzeitig schreiben und zeichnen können, dargestellt werden.
2. Die zweiten Stufe hat ein **kompetenter Benutzer** erreicht, sobald er eine größere Anzahl an Profilen erstellt hat und der Useragent ihm den Vorschlag macht, diese Profile zu Cases zu gruppieren.
3. Für **erfahrene Benutzer** wird auf der dritten Stufe der gesamte Menübaum eingeblendet, mit Ausnahme der Client-Objekte. Diese benötigen gelegentlicher Benutzer nicht vorrangig, sondern regelmäßig für Kunden arbeitender professioneller Broker.

Eine Umsetzung der Suche nach dem Namen nach bekannten Firmen durch das *in*FOCUS-Applet (siehe Abbildung 29) würde die Transparenz der Daten wesentlich erhöhen und die Effizienz beim Suchen steigern. Weiterhin sollten folgende Aktionen bei Bedarf an den Useragenten delegierbar sein:

- *Automatische Generierung neuer Cases und Profiles.* Dies würde den aufgabenfremden Aufwand zur Profilerstellung reduzieren.
- (Teilweises) *Vorausfüllen von Suchformularen*, analog zu den Formularen der Wizards, reduziert Fehlerquellen und beschleunigt die Generierung von Profiles.
- Umfangreicher Formulare können prinzipiell durch Wizards ausgefüllt werden, wodurch Falscheingaben reduziert werden.

Tabelle 20: Möglichkeiten zur Verbesserung der Hilfefunktion

Funktion	**Beschreibung**	**Agentenrelevanz**
Index	*Alphabetischer Hilfeindex*	gering
Topics	*Thematische Hilfe*. Traditionelles Handbuch	gering
Context**:**	*Hilfe im Kontext*. Dient der Orientierung. Beantwortet die Fragen „*Wo bin ich?“* und „*Was kann ich jetzt tun?“*	Agent kann über die bloße Beschreibung hinaus im Kontext sinnvolle Aktionen *demonstrieren*.
FAQ	Frequently Asked Questions. *Beantwortung typischer Fragen*	gering
Learn	bizzyB von Grund auf lernen	Agent kann den Unterricht „interaktiv“ gestalten. Kann auf Elemente deuten, die erklärt werden sollen.
Wizards	Beschreibung siehe vorhergehender Abschnitt	hoch
User to User help	Im System anwesende *Benutzer helfen sich gegenseitig*	gering
Proaktive Hilfe	Ohne explizites Aufrufen des Benutzers werden Tips und Hilfe gegeben.	Agent beobachtet den Benutzer und findet selbständig heraus - dass der Benutzer ein Problem hat - dass der Benutzer umständlich arbeitet
Just Ask	Benutzer gibt per Spracheingabe oder Tastatur *frei formulierte Fragen* ein.	Agent beobachtet Benutzer und kann daher den Fragekontext berücksichtigen. Agent versteht Spracheingabe.
Tip of the Day	Tagestip. Er vermittelt Problemlösungswissen der Form: Um A zu erreichen, tue B. Dadurch erhöht er auf beiläufige Weise die Lernförderlichkeit des Systems**.** Diese Methode wird in vielen professionellen Softwaresystemen verwendet (z.B. ICQ, Microsoft Office) Es ist bekannt, dass manche Benutzer diese Form der Hilfestellung als störend empfinden. Aus diesem Grund muss sie abschaltbar sein**.** Weiterhin muss die Möglichkeit vorhanden sein, weitere bzw. frühere Tips aufzurufen.	Vorteile der Agenteneinsatzes: Aktionen können nicht nur beschrieben, sondern auch demonstriert werden. Agent kann Menüs aufklappen oder auf Schaltflächen deuten.
Help on Help	Hilfe zur Bedienung der Hilfefunktion selbst.	Agent erläutert und aktiviert die entsprechenden Schaltflächen.

Eine ganze Reihe von Vorschlägen geht direkt auf die zweite Forderung nach verbesserten Hilfe- und Orientierungsmöglichkeiten ein. Viele der in Tabelle 20 zusammenfassend dargestellt Methoden sind Abschnitt 2.3.3.5, der Agenten mit Hilfefunktionen behandelte, entnommen. Beispielhaft umgesetzt wurden von diesen Vorschlägen - jeweils mit Hilfe des Mi-

crosoft Agents - die Punkte *Kontexthilfe, Wizards (siehe Abschnitt 6.4.2), Proaktive Hilfe und Tagestip.* Als ein Beispiel demonstriert Abbildung 42 - ähnlich wie Abbildung 37 - Kontexthilfe durch den Präsentationsagenten.

Abbildung 42: Mit Hilfe des Agenten realisierte durchgängige Hilfefunktion

6.5 Implementation

Sämtliche Verbesserungsvorschläge wurden über HTML-Seiten realisiert, die auch in verschiedene Skriptsprachen geschriebene Programme enthalten. JavaScripte ermöglichen Datenflüsse und die Navigation zwischen unterschiedlichen Seiten. Zur Steuerung des Microsoft Agent wurde zusätzlich eine das Sprachausgabesystem Lernout & Hauspie TruVoice eingesetzt. Eine spezielle, auf Visual Basic Script aufgesetzte Agenten-Scriptsprache steuert den Microsoft Agent selbst. Die Wahl eines Präsentationsagenten fiel auf den Microsoft Agent, weil dieser im Gegensatz zu vergleichbaren Systemen wie Persona und togglethis interactive characters unmittelbar verfügbar und flexibel einsetzbar ist.

bizzyB arbeitete ursprünglich nur mit dem Netscape Navigator zusammen. Da der Microsoft Agent jedoch nach dem Internet Explorer verlangt, war zunächst eine Anpassung des bestehenden Systems an den Internet Explorer nötig.

Ich habe den Quellcode nicht abgedruckt, weil er nur geringen wissenschaftlichen Informationswert besitzt, also nicht etwa neue Algorithmen enthält, und weil er sehr umfangreich ist. Mein Schwerpunkt liegt indessen auf der Optimierung der Benutzerschnittstelle, so dass ich dazu besonders viele Beispielbildschirme aufgeführt habe. Zahlreiche weitere Abbildungen sowie ein Video des kompletten Wizard Customer finden sich auf der diese Arbeit ergänzenden Homepage.

6.6 Diskussion der entwickelten Software

Nun werden die in Abschnitt 6.3 aufgestellten Forderungen an die Systemerweiterungen evaluiert, wobei zusätzlich zu theoretischen Überlegungen Fragen an Testanwender berücksichtigt werden. Diese Fragen können gemäß dem Leitfaden (siehe Abschnitt 5.3) in einer subjektiven Evaluation erhoben werden. Weniger geeignet ist hier die gebräuchliche Methode des „Thinking Aloud“ (siehe Reiterer u.a. 1993, 138f), wobei echte Anwender Aufgaben durchführen und dabei ihre Gedanken, Probleme und Meinungen verbalisieren. Denn noch existieren echte Anwender nicht, weil die Einführung von bizzyB an der Mailänder Handelskammer erst bevorsteht. Eine umfangreiche vergleichende Untersuchung von Original- und modifiziertem System als Alternative ist dagegen im Rahmen dieser Arbeit nicht vorgesehen, würde bei dem derzeitigen Integrationsgrad der verschiedenen Erweiterungen die Testpersonen auch zweifellos überfordern. Aus diesen Gründen wurde vorgeschlagen (Oppermann 1998), die Erweiterungen durch mit bizzyB vertrauten Personen nach dem Vorbild einer Expertenbefragung (Reiterer u.a. 1993, 138) durchzuführen.

Zu diesem Zweck testeten mit bizzyB vertraute Personen die einzelnen Erweiterungen und beantworteten verschiedene Fragen bezüglich der einzelnen Modifikationen sowie ihres Gesamteindrucks. Zusammenfassend lässt sich sagen, dass neben zahlreichen Verbesserungsvorschlägen an der Oberfläche eine bessere Erläuterung der für das Makeln wichtiger Begriffe angeregt wurde. Die Suchmaschinenmetapher wurde leicht positiv aufgenommen, aber als vermutlich sehr hilfreich für unerfahrene Benutzer erachtet. Durchweg positiv wurden die Wizards bewertet, wobei hier erwartungsgemäß der animierte Zauberer den größten Eindruck hinterließ. Das Konzept, Profilbildungen in mehrere, halbautomatisierte Schritte zu zerlegen, wurde - von der konkreten Reihenfolge einmal abgesehen - begrüßt. Weiterhin wurde angeregt, den Zauberer auch in anderen Systemkomponenten einzusetzen (so wie es in verschiedenen Hilfefunktionen bereits umgesetzt wurde), vor allem als Unterstützung in der Suchmaschine.

6.6.1 Überprüfung der aufgestellten Forderungen auf Umsetzung

Die zu untersuchenden, in Abschnitt 6.3 aufgestellten Forderungen waren die folgenden:

1. Benutzer müssen einen intuitiven, möglichst einfachen Zugang zum System erhalten.
2. Hilfs- und Orientierungsmöglichkeiten müssen verbessert werden. Denn es ist anzunehmen, dass die neuen Nutzer das System niemals vollständig erlernen und deshalb immer wieder Funktionen nachschauen müssen
3. Arbeitsanteile sollen an einen Agenten delegierbar sein. Speziell sollen Anzahl und Komplexität der nötigen Schritte von der ersten Anmeldung bis zu den Ergebnissen einer Anfrage durch Delegation minimiert werden.

Punkt 1 erwies sich, wie Abschnitt 6.4.1 zeigt, als mit dem vorgeschlagenen Conceptual Model einer Suchmaschine gut umsetzbar. Ein „möglichst einfacher Zugang“ wurde durch die Unterteilung in verschiedene komplexe Stufen und den Einstieg über die besonders stark vereinfachte Suchmaschine Simple Search erreicht.

Die zweite Forderung nach verbesserten Hilfs- und Orientierungsmöglichkeiten wurde durch eine Reihe von Hilfesystemen (Abschnitt 6.4.3), nicht zuletzt aber durch den Microsoft Agent (Abschnitt 6.4.2) umgesetzt.

Die letzte Forderung nach Delegation wurde in jeder einzelnen der in Abschnitt 6.4 beschriebenen Maßnahmen berücksichtigt. Die folgende Analyse bestätigt und präzisiert die Erfüllung der Forderung nach mehr Delegation und weniger auszuführenden Einzelschritten nach.

Tabelle 21: Vergleich der Anzahlen nötiger Einzelschritte, um ein Profil zu generieren

bizzyB	
1.	Benutzer anmelden
2.	„Brokerage Session“ starten
3.	Neuen Client erzeugen
4.	Neuen Case erzeugen
5.	Neues Profile erzeugen
6.	Quellenverwalter (Sources Manager) aufrufen
7.	Datenquelle auswählen
8.	Auswahl an Datenquellen hinzufügen.
9.	Anfragetyp „Kategoriensuche“ auswählen
10.	Kategorienmanager (WebCatNet) aufrufen
11.	Kategorie auswählen
12.	Auswahl an Kategorien hinzufügen
13.	Profildaten speichern
14.	Profil ausführen
15.	Dossier öffnen: enthält Suchergebnisse

Wizard Customer (Expert Search)	
1.	Benutzer anmelden
2.	„Wizard Customer“ starten: Vorausgefülltes Datenquellen-Menü erscheint
3.	Feld „weiter“ wählen: Vorausgefülltes Anfragetyp-Menü erscheint
4.	Aufruf der (vorausgefüllten) Kategorienauswahl
5.	Feld „weiter“ wählen: Voreingestelltes Menü mit Profil-Zusatzinformationen erscheint
6.	Profil ausführen: Profil wird gespeichert, eintreffende Suchergebnisse automatisch angezeigt.

Erweiterte Suchmaschine (Advanced Search)	
1.	Suchmodus „Advanced Search) aufrufen
2.	Suche mit voreingestellter Datenquelle, Kategoriesystem und Kategorienauswahl starten. Eintreffende Ergebnisse werden angezeigt.

Tabelle 21 listet sämtliche Einzelschritte auf, die ab Systemstart über das Formulieren einer Suchanfrage bis zum Erhalt der Ergebnisse mindestens notwendig sind. Jeder Schritt kann durch einen einzige Aktion (Mausklick) ausgeführt werden. Während im herkömmlichen bizzyB-System nicht weniger als 15 Schritte auszuführen sind, benötigt der Wizard Customer lediglich sechs, die Suchmaschine gar nur zwei Schritte.[58] Gegenüber dem Originalsystem spart der Wizard also 60 Prozent der Schritte, die Suchmaschine sogar 87 Prozent. Da in den Wizards sämtliche Schritte linear aufgereiht sind und Sprünge zwischen verschiedenen Systemkomponenten vollständig entfallen, reduziert sich darüberhinaus die Wahrscheinlichkeit von Falscheingaben. Diese Resultate sind starke Hinweise auf die Umsetzung der Forderungen, die Eignung von bizzyB für neue Benutzerkreise zu erhöhen und insbesondere die Anzahl nötiger Schritte zu minimieren.

[58]Simple Search wurde in diesem Vergleich nicht berücksichtigt, weil es keine Suche nach mehreren Kategorien zulässt.

Tabelle 22: Typische Arbeitschritte mit bizzyB und ihre Entsprechungen im Wizard Customer

	bizzyB	**Wizard Customer**
1.	Benutzer anmelden	Beibehalten
2.	„Brokerage Session“ starten	„Wizard Customer“ starten
3.	Neuen Client erzeugen	Entfällt
4.	Neuen Case erzeugen	Entfällt
5.	Neues Profile erzeugen	**Delegiert** an Wizard
6.	Quellenverwalter (Sources Manager) aufrufen	**Delegiert** an Wizard
7.	Datenquelle auswählen	**Delegiert** an Wizard
8.	Auswahl an Datenquellen hinzufügen.	Auswahl an Datenquellen bestätigen
9.	Anfragetyp „Kategoriensuche“ auswählen	**Delegiert** an Wizard
10.	Kategorienmanager (WebCatNet) aufrufen	Kategorienauswahl aufrufen
11.	Kategorie auswählen	**Delegiert** an Wizard
12.	Auswahl an Kategorien hinzufügen	Auswahl an Kategorien bestätigen
13.	Profildaten speichern	**Delegiert** an Wizard
14.	Profil ausführen	Beibehalten
15.	Dossier öffnen: enthält Suchergebnisse	**Delegiert** an Wizard

Doch wodurch wurden die im Vergleich zum Originalsystem eingesparten Schritte ersetzt? - Auf diese Frage gibt Tabelle 22 Auskunft: Von den verbleibenden sechs Schritten des Wizards wurde die Mehrzahl (4) modifiziert, nur zwei blieben unverändert. Weitere zwei Schritte - für das Anlegen neuer Clients und Cases - entfielen restlos. Client- und Caseobjekte sind vor allem für die Organisation der Profiles hilfreich, sobald diese eine gewisse Anzahl überschreiten. Doch in der Lernphase sind diese Objekte irrelevant, und für die Belange gelegentlicher Broker bleiben sie das auch mit hoher Wahrscheinlichkeit.

Die wichtigste Erkenntnis aus der Tabelle ist, dass alle übrigen sieben Schritte, also *annähernd die Hälfte aller Schritte an den Wizard delegiert* wurden. Zu etwa gleichen Teilen wurden anspruchsvolle (Vorauswahl von Daten) und einfache Aufgaben (Aufruf von Seiten, Speichern von Daten) an den Wizard delegiert. Dennoch wurde der in Abschnitt 4.3.3 aufgestellten Forderung, nur Teilaufgaben zu delegieren, entsprochen, und der Benutzer bleibt die oberste Kontrollinstanz**.**

Wir halten fest, dass der Leitfaden auf das Agentensystem bizzyB anwendbar war: Vereinfachungen, Verbesserungen der Hilfsmöglichkeiten, vor allem aber die Umsetzung von *Delegation war vielfach und erfolgreich möglich.*

6.6.2 Weitere Möglichkeiten der Evaluation

Die beste Evaluation ist der Bewährungsprobe im täglichen Einsatz, eingesetzt von echten Benutzern. Dies kann geschehen, sobald das meinen Entwicklungen zugrundeliegende bizzyB-System an der Mailänder Handelskammer zum Einsatz kommt. Bis dahin ist die vollständige Integration der entwickelten Komponenten in bizzyB sowie die Übersetzung des Gesamtsystems in die italienische Sprache geplant.

Dann aber kann eine wirkliche, vergleichende Evaluation durchgeführt werden, Stärken und Schwächen ermittelt und durch die Rückmeldungen der Anwender der Zyklus der Softwareverbesserungen beginnen.

6.6.3 Verallgemeinerung der Ergebnisse

Für das Agentensystem bizzyB haben sich die in dem Leitfaden gebündelten Erkenntnisse als anwendbar und hilfreich erwiesen. Es bleibt die Frage, ob der Leitfaden, entsprechend seinem eigenen Anspruch, *für Agentensysteme allgemein* eine Relevanz hat.

Zugegebenermaßen wurde der Leitfaden mit der konkreten Anwendung bizzyB vor Augen entwickelt, so dass eine schwächere Berücksichtigung von im konkreten Fall nicht benötigten Aspekten möglich ist. Deshalb wird ein Einsatz des Leitfadens in anderen Anwendungen von Zeit zu Zeit Konkretisierungen und Ergänzungen nötig machen. Doch sollte dies als Chance betrachtet werden, den Leitfaden auf einem aktuellen Stand zu halten und durch zusätzliche Beispiele und Methoden sogar zu verbessern.

Dennoch ist anzunehmen, dass der Leitfaden auch bei einem Einsatz in anderen Domänen erfolgreich sein kann: Da der Leitfaden gerade im Hinblick auf eine *möglichst breite Verwendung konzipiert* wurde, ist er allgemein und **frei** *von Implementierungsdetails* gehalten (vgl. Abschnitt 5.1.3). Keineswegs ist er auf die Einsatz in Brokeringsystemen beschränkt oder auf Hilfefunktionen spezialisiert. Er basiert vielmehr auf *allgemein gehaltenen Ausarbeitungen* im Rahmen dieser Arbeit auf den Gebieten Agenten, HCI und Delegation. Daher ist der Leitfaden in der vorliegenden Form bereits so allgemein wie möglich und für ein größtmögliches Anwendungsspektrum vorbereitet.

6.7 Zusammenfassung

Der in Kapitel 5 entwickelte Leitfaden wurde in diesem Kapitel erstmals zur agentenbasierten Softwareentwicklung eingesetzt. Dabei zeigten sich für das Brokeringsystem bizzyB zahlreiche Ansatzpunkte und Verbesserungsmöglichkeiten, die Forderungen nach Vereinfachung und Unterstützung entspringen. Die umgesetzten Verbesserungen haben die Benutzbarkeit des Systems für eine neue Gruppe von Benutzern erhöht und die allgemeine Anwendbarkeit des Leitfadens demonstriert.

7 Zusammenfassung und Ausblick

In diesem abschließenden Kapitel werden die wichtigsten Ergebnisse der Arbeit zusammengefasst. In einem Ausblick werden Verbesserungsmöglichkeiten des Leitfadens für agentenbasierte Softwareentwicklung genannt. Weiterhin wird der Versuch unternommen, den Leitfaden in Beziehung zu sich abzeichnenden Trends und Entwicklungen von Softwareagenten zu setzen.

7.1 Zusammenfassung

Im Rahmen dieser Arbeit sollte untersucht werden, wie sich mit softwareergonomischen Mitteln Softwareagenten die Mängel heutiger Agentensysteme vermeiden lassen. Dazu wurden in mehreren Anläufen jeweils stärkere Werkzeuge entwickelt, die sich, zu einem Leitfaden gebündelt, zur agentenbasierten Softwareentwicklung eignen und angewendet werden. Auf dem Weg dorthin wurde unter anderem eine Taxonomie von Agentensystemen entwickelt, agentenrelevante HCI-Gestaltungsrichtlinien identifiziert und transformiert, ein Modell der Delegation bei Agenten sowie der Leitfaden selbst erarbeitet. Diese (notwendige) mehrstufige Vorgehensweise spiegelt sich in der Struktur der Arbeit wieder, in der sich theoretische und praktischen Abschnitte abwechseln.

Zunächst wurden in Kapitel 2 Softwareagenten im Rahmen dieser Arbeit definiert, eine Taxonomie von Softwareagenten erstellt, die in drei Stufen verschiedene Blickwinkel auf Agenten erlaubt. Eine aufgabenorientierte Untersuchungsmethode von Agenten im praktischen Teil des Kapitels ließ zahlreiche (Funktions-) Mängel heutiger Softwareagenten erkennen und führte zur Formulierung dreier wesentlicher Hauptforderungen für Agenten.

Das dritte Kapitel untersuchte Probleme von Agentensystemen aus einem ergonomischen Blickwinkel. Nach Vorstellung und Vergleich der aussichtsreichsten Kandidaten zur Verbesserung von Agentensystemen - HCI und Normen - wurden aus diesen agentenrelevanten Gestaltungsrichtlinien extrahiert, zu Hypothesen transformiert und im Lichte der Hauptforderungen systematisiert. Der praktischen Teil des Kapitels zeigte die prinzipielle Anwendbarkeit, aber noch mangelnde Schärfe und Rechtfertigung der Kriterien und Hypothesen.

In Kapitel 4 wurde Delegation als das wichtigste, Agenten zugrundeliegende Konzept identifiziert, vertieft, und ihre Aspekte ausgearbeitet. Ein tieferes Verständnis für den *Sinn* der Verbesserung von Agentensystemen erwies sich als Schlüssel zur tatsächlichen Verbesserung. Im Anwendungsteil wurden ein aktuelles Agentensystem aufgabenorientiert, ergonomisch und konzeptionell untersucht.

Das Wissen über Aufgaben, HCI und Delegation bei Agenten wurden schließlich in Kapitel 5 zu einem Leitfaden gebündelt, der die Barriere überwand, die der Entwicklung agentenbasierter Software entgegensteht. Kapitel 6 demonstrierte die Einsatzmöglichkeiten des Leitfadens anhand des Brokeringsystems bizzyB. Sowohl bei der Systemanalyse als auch bei ihrer Verbesserung erwies sich der Leitfaden als hilfreich, und verschiedene Erweiterungen von bizzyB wurden leitfadenorientiert durchgeführt.

Der lange Weg war nötig, da jeder neue Schritt auf der Gesamtheit der vorausgehenden Schritte aufgebaut, und die Erkenntnisse aller Stufen wichtige Beiträge für den Leitfaden ge-

liefert haben. Rückblickend lässt sich sagen, dass Methoden zur agentenbasierten Softwareentwicklung nur dann erfolgreich sein können, wenn sie das Konzept Delegation berücksichtigen. Deshalb hätte sich die in dieser Arbeit verfolgte Methodensuche entscheidend verkürzen lassen, wenn gleich zu Anfang Delegation untersucht worden wäre. Doch diese Erkenntnis fällt nicht vom Himmel, was den langen Weg zum Ziel rechtfertigt. Denn erst die tiefere Beschäftigung mit Agentensystemen führte zur Erkenntnis, wo die Probleme heutiger Agentensysteme liegen und was verbessert werden muss.

Die Wahl von HCI war dennoch erfolgreich, lieferte es doch ein im Agentenkontext anwendbares Methodenarsenal (auch wenn ich vom heutigen Standpunkt aus teilweise andere, weniger redundante Gestaltungsrichtlinien gewählt hätte). Doch eine wirksame Kombination dieser Erkenntnisse wurde erst durch die Berücksichtigung der Delegation möglich, die sich somit als Schlüssel zur agentenbasierten Softwareentwicklung erwies. Der Leitfaden stellt die Quintessenz dieses Prozesses dar, mit spezieller Berücksichtigung seiner Verständlichkeit und Anwendbarkeit auf unterschiedliche Agentensysteme. Er mag kein Universalwerkzeug sein, und seine Aktualität ist angesichts der sich rapide entwickelnden Agententechnologie begrenzt, doch stellt er ein durchaus praktikables Werkzeug dar.

7.2 Ausblick

Außer geplanten Weiterentwicklungen der bizzyB-Erweiterungen und konkreten Verbesserungsvorschlägen für den Leitfaden sind mehrere Trends für zukünftige Softwaresysteme erkennbar, die für den Leitfaden relevant sind:

7.2.1 Leitfaden wird erweitert und verbessert

Die Mängel des bestehenden Leitfadens sowie die im folgenden angesprochenen Entwicklungen unterstreichen die Notwendigkeit von Modifikationen und Erweiterungen des Leitfadens.

- Wünschenswert ist eine *ausführlichere Ausarbeitung* des Leitfadens, mit mehr Beispielen, Methoden und Werkzeugen. Eine Ausweitung des Leitfadens auf die Evaluation bestehender Systeme, sogar deren halbautomatische Prüfung nach dem Vorbild Oppermanns u.a. (1992) ISO 9241-Evaluators ist denkbar!
- Entwickler würden von einer *Kurzform, etwa in Form einer Checkliste*, profitieren. Dadurch würde sich der Einarbeitungsaufwand minimieren, und dennoch würden eine Reihe möglicher Fehler bei der Agentenentwicklung vermieden. Diese Liste könnte mit wenig Aufwand und auf Basis der tabellarischen Zusammenfassungen der Leitfadenkomplexe erstellt werden.
- Eine *WWW-Umsetzung* des Leitfadens würde eine höhere Verbreitung sowie - durch Hyperlinks - eine bessere Strukturierung ermöglichen. Sie hätte weiterhin den Vorteil der größeren Aktualität, da der Leitfaden in der heutigen Form durch die schnelle technische Entwicklung zweifellos schnell veralten wird.

7.2.2 Werkzeuge zur Agentenentwicklung verbessern sich

Heutige Entwicklungsumgebungen zur Erstellung von Agentensystemen stecken noch in den Kinderschuhen. Diesem in Abschnitt 5.2.1.2 angesprochene Problem lässt sich beispielsweise durch komfortablere Entwicklungsumgebungen begegnen.

Auf einer niedrigeren Ebene könnten auf Agentensysteme angepasste Design Patterns (siehe Abschnitt 5.2.1.2) viel zur effektiveren Erstellung zuverlässigerer Softwareagenten beitragen. Design Patterns greifen - ebenso wie HCI - agentenspezifische Konzepte wie zum Beispiel Kapselung auf. Eine weitere Parallele zur HCI ist die starke Einbeziehung des Anwenders in den Designprozess. Die Entwicklung agentenspezifischer Patterns könnte hier lohnenswert sein.

Auf Implementationsebene schließlich könnten Bibliotheken generischer Agenten und Agentenbausteine erstellt werden, die, programmiert beispielsweise wie JATlite in Java, die Entwicklung von Agentensystemen automatisieren. Auch in COBRA wurde die Entwicklung einer solchen Bibliothek in Betracht gezogen.

7.2.3 Lohnende Anwendungsgebiete für Agenten wachsen

Agenten werden in immer weiteren Gebieten angewendet, teils weil sie die Gebrauchstauglichkeit von Software wirksam erhöhen, teil weil die Domäne selbst agentengeeigneter wird.

- Umfangreiche Softwaresysteme kranken an einem Übermaß von Befehlen, die sich kaum ein Anwender merken kann und die den gerade gesuchten Befehl verdecken. Schon die Fülle von Icons sprengt die Aufnahmefähigkeit des Monitors sowie des Benutzers. Hier bietet sich eine Agentenschnittstelle an, um die Benutzbarkeit dieser Software zu erhöhen bzw. wieder sinnvoll zu ermöglichen.
- Die Sprache XML (eXtensible Markup Language), entwickelt vom W3C, hat gute Aussichten, HTML als Standardsprache im WWW abzulösen. Ein großer Vorteil gegenüber HTML ist die Erweiterbarkeit von Dokumenten um beliebige Sprachelemente, was selbstbeschreibungsfähige Dokumente ermöglicht. Dies begünstigt all diejenigen Agenten, die den Inhalt von WWW-Seiten zu verstehen versuchen, also beispielsweise Recommender- und Information-Retrieval-Agenten.
- Die gerade entstehende Forschungsrichtung Wearable Computing ist in hohem Maße von verlässlicher Agententechnologie abhängig. Sie entwickelt am Körper dauerhaft zu tragende, permanent laufende Computersysteme, die dem Benutzer im Kontext helfen. Der Remembrance Agent von Rhodes (1997) ist solch ein prototypisches System, das beispielsweise situationsbezogene Textdaten auf einem vor dem Auge zu tragenden kleinen Bildschirm (Heads-Up Display) anzeigt und den Benutzer an wichtige Termine erinnert.

7.2.4 Agentenbasierte Softwareentwicklung setzt sich durch

Die folgende Entwicklung und zukünftige Bedeutung der Agententechnologie ist denkbar:

1. Die Auffassung von „Agenten als Designmodell“ setzt sich durch (vgl. Ovum '94, 27) und die agentenbasierte Softwareentwicklung nimmt stark zu. In ähnlicher Weise haben sich Softwaresysteme mehrfach in der Vergangenheit gewandelt, als sich Innovationen wie grafische Benutzeroberflächen, Hilfefunktionen oder Internetfähigkeiten zu Software-Standardelementen entwickelten. *In Folge besitzt die Mehrzahl an Softwaresystemen Agentenfunktionalität.* Allerdings heißen diese Systeme nicht mehr Agentensysteme - weil dieses Attribut nicht mehr zur Abgrenzung von Konkurrenzprodukten taugt.
2. Im zweiten Schritt geben die (ehemals Agentensysteme genannten) Systeme Teile ihrer Funktionalität an einen *persönlichen Useragenten des Benutzers*, der Bestandteil des Be-

triebssystems ist, ab. Das Betriebssystem ist in dieser Phase bereits selbst agentenbasiert.[59] Der Zugriff von Softwaresystemen auf standardisierte Betriebssystem-Agentenkomponenten ist vergleichbar mit heutigen Richtlinien zur Softwareerstellung, zum Beispiel grafische Elemente mit Rückgriff auf die Standardroutinen des Betriebssystems zu implementieren. Welche Teile des Systems an den Useragenten abgetreten werden, ist von System zu System unterschiedlich und weitgehend durch den Benutzer steuerbar. Ein Vorgeschmack dieser Entwicklung ist die Integration des (auf Betriebssystemebene verankerbaren) Microsoft Agent in bizzyB. Vermutlich wird der Microsoft Agent fester Bestandteil zukünftiger Versionen von Windows. Das würde einen starken zusätzlichen Anreiz zur Nutzung von Agenten darstellen. Denn es ist weitaus bequemer für Entwickler, bereits vorhandene Systemkomponenten mitzuverwenden als Agenten eigens entwickeln und implementieren zu müssen.

3. Im dritten und letzten Schritt *geben Softwaresysteme ihre gesamte Benutzerschnittstelle an den persönlichen Useragenten ab.* Sie besitzen keine direkte Schnittstelle zum Benutzer mehr, also auch kein „Erscheinungsbild“, sondern definieren sich ausschließlich über ihre Funktionalität. Der persönliche Useragent lernt vom Benutzer, ist der Mittler zwischen ihm und jeglicher Software im System und entscheidet über deren Erscheinungsbild, Mittel zur Zielerreichung und Umsetzung der Ein- und Ausgabe. Nur so ist die umfassende Steuerung aller Systemkomponenten durch eine einzige, intelligente Schnittstelle möglich, welche auch die in Abschnitt 2.3.3.2 angesprochenen zukünftigen Stellvertretereigenschaften integriert. Damit diese weitreichende Übertragung möglich ist, kommt es zu einer Normierung von Kommunikationsprotokollen und Schnittstellen zum persönlichen Agenten.

Der Leitfaden richtet sich nun in der aktuellen Version an Agenten, die auf oder noch vor der ersten Stufe stehen. Sollte die Entwicklung von Agenten so ablaufen wie hier angedeutet, wäre zu gegebener Zeit eine Modifikation des Leitfadens für Agentensysteme der zweiten Stufe nötig. In der dritten Stufe allerdings ist der Leitfaden nicht mehr notwendig. Da die gesamte Schnittstelle zum Benutzer nun über dessen persönlichen Agenten erfolgt, ist der schnittstellenorientierte Leitfaden in seiner Anwendbarkeit auf diesen einen Agenten reduziert.

[59] Es ist gut vorstellbar, dass ein solches Betriebssystem an der Benutzeroberfläche durch die in Abschnitt 3.5.7 beschriebene Bühnenmetapher Negropontes, welche die Desktopmetapher ablösen könnte, ergänzt wird. In dieser Metapher präsentieren sich die einzelnen Agenten als Akteure auf einer Art Theaterbühne.

8 Literaturverzeichnis

Die Veröffentlichungen sind denjenigen Aspekten zugeordnet, die sie überwiegend behandeln oder derentwegen ich sie zitiere.

8.1 Allgemeine Literatur

Burger u.a. 1996

Burger, Christoph; Entenmann, Mathias; Neputé, Stefano: „Konsumgüterindustrie und Handel im Jahr 2005: Handel endgültig entmachtet". In: A. Little (Hrsg.): *Management im vernetzten Unternehmen*, Wiesbaden, 1996, 127-157.

Kniesel 1996

Kniesel, Günter: *Encapsulation: Visibility and Access Rights*. Technical Report IAI-TR-96-12, Bonn, November 1996.

Kniesel 1997

Kniesel, Günter: What is (not) delegation. In: *The Darwin Project*, 1.12.1997. URL: http://javalab.cs.uni-bonn.de/research/darwin/darwin_eng.html (20.12.1997).

Lieberman 1986

Lieberman, Henry: Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems. *Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, 1986, 214-223.

Mertens u.a. 1996

Mertens, Peter und Schumann, Petra: Electronic Shopping. *Wirtschaftsinformatik*, Vol. 38, Nr. 5, 1996, 515-530.

8.2 HCI, Softwareergonomie, Normen

ACM SIGCHI 1992

ACM SIGCHI: Curriculum for Human-Computer Interaction. *ACM Special Interest Group on Computer-Human Interaction*. Curriculum Development Group, New York, 1992.

Apple 1993

Apple Computer: *Macintosh Human Interface Guidelines*, Addison-Wesley, Reading, 1993.

Apple 1997

Apple Computer: *Human Interface Principles*, 20.2.1997. URL: http://applenet.apple.com/hi/resources/principles/intro.html (25.7.1997).

Collins 1995

Collins, Dave: *Designing object-oriented user interfaces*, Benjamin/Cummings Publishing, Redwood City, 1995.

Cypher 1993

Cypher, Allan u.a.(Hrsg.): *Watch what I do: programming by demonstration*. MIT Press, London, 1993.

Dzida 1993

Dzida, Wolfgang: Anwendung software-ergonomischer Normen. In: E. Eberleh u.a. (Hrsg.): *Einführung in die Software-Ergonomie*. de Gruyter, Berlin, 1993.

Dzida 1994

Dzida, Wolfgang: Bestimmung und Anwendung ergonomischer Gestaltungskriterien im Prozeß der Software-Entwicklung. In*: Berichte des German Chapter of the ACM, Band 42*, Teubner, Stuttgart, 1994.

Dzida 1997

Dzida, Wolfgang: Persönliche Kommunikation vom 25.11.1997.

EG 1990

Europäische Gemeinschaft: *Richtlinie über die Mindestvorschriften bezüglich der Sicherheit und des Gesundheitsschutzes bei der Arbeit an Bildschirmgeräten*, 90/270/EWG, L156 14-18, 29. Mai 1990.

Erickson 1990

Erickson, Thomas: Creativity and Design. In: Laurel, Brenda (Hrsg.): *The Art of human-computer interface design*, Addison-Wesley, Reading, 1990. 1-4.

Hellbardt 1997

Hellbardt, Günter: *Die Entwicklung der Software-Ergonomie*. Universität Jena. URL: http://www1.informatik.uni-jena.de/Lehre/SoftErg/vor_hist.htm (26.6.1997.)

Informatik-Duden 1993

Duden „Informatik“: Ein Sachlexikon für Studium und Praxis - 2. vollst. überarb. und erw. Aufl., Lektorat des BI-Wiss.-Verl (Hrsg.). Dudenverlag, Mannheim, 1993.

ISO 9241

International Standards Organisation: ISO 9241. *Ergonomic requirements for office work with visual display terminals*. International Standard, 1996.

Negroponte 1990

Negroponte, Nicholas: Hospital Corners. In: Laurel, Brenda (Hrsg.): *The Art of human-computer interface design*, Addison-Wesley, Reading, 1990, 347-353.

Nelson 1990

Nelson, Theodore Holm: The Right Way to Think About Software Design. In: Laurel, Brenda (Hrsg.): *The Art of human-computer interface design*, Addison-Wesley, Reading, 1990, 235-243.

Norman 1986

Norman, Donald A.: Cognitive Engineering. In: D. Norman und S. Draper (Hrsg.): *User-Centered System Design.* Lawrence Erlbaum Associates, Hillsdale, 1986.

Norman 1988

Norman, Donald A.: *The Psychology of Everyday Things*. Basic Books, New York, 1988.

Norman 1992

Norman, Donald A.: *Turn Signals are the Facial Expressions of Automobiles*. Addison-Wesley, Reading, 1992.

Oppermann 1988

Oppermann, Reinhard: Software-ergonomische Evaluationsverfahren. In: H. Balzert u.a. (Hrsg.): *Einführung in die Software-Ergonomie* de Gruyter, Berlin, 1988.

Oppermann u.a. 1988

Oppermann, Reinhard u.a: Evaluation von Dialogsystemen. *Der software-ergonomische Leitfaden EVADIS.* de Gruyter, Berlin, 1988.

Oppermann u.a. 1992

Oppermann, Reinhard u.a: Software-ergonomische Evaluation. *Der Leitfaden EVADIS II. 2., neu bearb. u. erw. Aufl*. de Gruyter, Berlin, 1992.

Oppermann u.a. 1996

Oppermann, Reinhard u.a: Der ISO 9241-Evaluator. In: *Ergonomie&Informatik* 27, März 1996, 13-16.

Oppermann 1998

Oppermann, Reinhard: Persönliche Kommunikation vom 26.2.1998.

Preece u.a. 1994

Preece, Jenny u.a. (Hrsg.): *Human-Computer Interaction*. Addison-Wesley, Reading MA, 1994.

Reiterer u.a. 1993

Reiterer, Harald; Oppermann, Reinhard: Evaluation of user interfaces: EVADIS II - a comprehensive evaluation approach. In: *Behaviour & Information Technology*, Vol. 12, No. 3, 1993, 137-148.

Rödiger 1991

Rödiger, Karl-Heinz: Software-Ergonomie. *Gestaltungsgrundsätze der DIN-Norm 66 234 Teil 8 und ihre Umsetzung*, Technologieberatungsstelle beim DGB Landesbezirk NRW e.V. (Lothringer Str. 62, 4200 Oberhausen 1), 1991.

Shneiderman 1992

Shneiderman, Ben: *Designing the User Interface: strategies for effective human-computer interaction. 2nd Edition*. Addison-Wesley, Reading, 1992.

Shackel 1997

Shackel, Brian: Human-Computer Interaction — Whence and Whither? In: *Journal of the American Society for Information Science*, Vol. 48, Nr. 11, November 1997, 970-986.

Ulich 1992

Ulich, Eberhard: *Arbeitspsychologie*. 2., verb. Auflage. Verlag der Fachvereine, Zürich, 1992.

8.3 Agenten

Aridor u.a. 1998

Aridor, Yariv; Lange, Danny: *Agent Design Patterns: Elements of Agent Application Design.* Entwurf, akzeptiert zur Präsentation auf der Second Internation Conference on Autonomous Agents (Agents '98), 1998.

Bradshaw 1997

Bradshaw, Jeffrey M.: An Introduction to Software Agents. In: J. Bradshaw (Hrsg.): *Software Agents*, American Association for Artificial Intelligence, Menlo Park, 1997.

Doorenbos 1997

Doorenbos, Robert: *Re: Agents in Jango*. Persönliche E-Mail vom 18.12.1997.

Doorenbos u.a. 1997

Doorenbos, Robert; Etzioni, Oren; Weld, Daniel: A Scalable Comparison-Shopping Agent for the World-Wide Web. *Agents '97 Conference Proceedings*, 1997.

Franklin u.a. 1996

Franklin, Stan; Graesser, Art: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer, Berlin, 1996.

Gates 1996

Gates, Bill: *The Road Ahead. Completely revised and up-to-date*. Penguin Books, New York, 1996.

Kendall u.a. 1997

Kendall, Elizabeth; Malkoun, Margaret: Design Patterns for the Development of Multiagent Systems. In: C. Zhang, D. Lukose (Hrsg.): *Multi-Agent Systems: Methodologies and Applications; selected papers.* Springer, Berlin, 1997.

Krulwich 1998a

Krulwich, Bruce: *Re: Agent Application*. E-Mail in Agenten-Mailingliste agents@cs.umbc.edu vom 25.1.1998.

Krulwich 1998b

Krulwich, Bruce: *Re: Shopping Agents causing lawsuits.* E-Mail in Agenten-Mailingliste agents@cs.umbc.edu vom 9.3.1998.

Lange 1998

Lange, Danny: *Re: Agent Application*. E-Mail in Agenten-Mailingliste agents@cs.umbc.edu vom 25.1.1998.

Lanier 1995

Lanier, Jaron: Agents of Alienation, *Interactions*, Juli 1995, 66-72.

Laurel 1990

Laurel, Brenda: Interface Agents: Metaphors with Character. In: Laurel, Brenda (Hrsg.): *The Art of human-computer interface design*, Addison-Wesley, Reading, 1990, 347-353.

Lieberman 1997

Lieberman, Henry: *Autonomous Interface Agents*. MIT. URL: http://lcs.www.media.mit.edu/people/lieber/Lieberary/Letizia/AIA/AIA.html (24.6.1997).

Maes 1994

Maes, Pattie.: Agents that Reduce Work and Information Overload. *Communications of the ACM*, Vol. 37, Nr. 7, Juli 1994, 31- 40.

Microsoft 1998

Microsoft: *Developing for Microsoft Agent*. Microsoft Press, Redmond, 1998.

Minsky 1988

Minsky, Marvin: *The Society of Mind*. Touchstone Books, New York, 1988.

Müller 1996

Müller, Jörg P.: *The Design of Intelligent Agents: A layered Approach*. Springer, Berlin, 1996.

Netbot 1997

Netbot: *Excite, Inc. announces acquisition of Netbot, Inc. and break-through shopping service,* 16.10.1997. URL: http://www.jango.com/pressbox/releases/1016.html (2.11.1997).

Newell 1982

Newell, Allen: The Knowledge Level. In: *Artificial Intelligence*, 18, 1982, 87-127.

Norman 1997

Norman, Donald A.: How Might People Interact with Agents. In: *Communications of the ACM*, Vol. 37, Nr. 7, Juli 1994, 68-71.

Nwana 1996

Nwana, Hyacinth S.: Software Agents: An Overview. In: *Knowledge Engineering Review*, Vol. 11, Nr. 3, 1996.

Ovum 1994

Guilfoyle, Christine; Warner, Ellie*: Intelligent Agents: the New Revolution in Software.* Ovum Limited, London, 1994.

Ovum 1997

Guilfoyle, Christine; Jeffcoate, Judith, Stark, Heather: *Agents on the Web: Catalyst for E-commerce.* Ovum Limited, London, 1997.

Rhodes 1997

Rhodes, Bradley A.: The wearable remembrance agent. In: *Proceedings of The First International Symposium on Wearable Computers (ISWC '97)*, Oktober 1997, 123-128.

Shneiderman u.a. 1997

Shneiderman, Ben; Maes, Pattie: Direct Manipulation vs Interface Agents. In: *Interactions*, November + December 1997, 42-61.

Shoham 1993

Shoham, Yoav: Agent-oriented programming. In: *Artificial Intelligence 60*, 1993, 51-92.

SodaBot 1997

SodaBot: *The SodaBot Home Page*. URL: http://www.ai.mit.edu/people/sodabot/. MIT (22.6.1997).

Steels 1995

Steels, Luc: When are robots intelligent autonomous agents? In: *Journal of Robotics and Autonomous Systems* 15, 1995, 3-9.

Stoll 1995

Stoll, Clifford: *Silicon Snake Oil: Second Thoughts on the Information Highway*. Doubleday, New York, 1995.

Tegenbos u.a. 1997

Tegenbos, Jef; Nieuwenhuysen, Paul: My Kingdom for an Agent? Evaluation of Autonomy, an intelligent search agent for the internet. In: *Online & CDROM Review*, Vol. 21, No. 3, 1997, 139-148.

Thomas 1996

Thomas, Christoph: To Assist the User: On the Embeding of Adaptive and Agent-Based Mechanisms, *GMD-Bericht Nr. 258*, Oldenbourg, München, 1996.

Thomas u.a. 1997

Thomas, Christoph; Fischer, Gerhard: Using Agents to Personalize the Web, *Proceedings IUI97*, Januar 1997, 53-60.

Vereerstraeten 1998

Vereerstraeten, Eric: *RE: Intelligence vs. Mobility*. E-Mail in Agenten-Mailingliste agents@cs.umbc.edu vom 9.2.1998.

Wooldridge u.a. 1995

Wooldridge, Michael; Jennings, Nicholas S.: Intelligent agents: theory and practice. In: *The Knowledge Engineering Review*, Vol. 10, No. 2, 1995, 115-152.

Anhang: Liste erwähnter Agentenanwendungen

Agentensystem	Hersteller	Kategorie	Weitere Informationen
AgentBuilder	Reticular Systems	Entwicklungsumgebung	http://www.agentbuilder.com/
Agentware	Autonomy	Suchmaschine	http://www.autonomy.com
Aglets Workbench	IBM	Entwicklungsumgebung	http://www.trl.ibm.co.jp/aglets
Alexa	Alexa Internet	Recommender	http://www.alexa.com/
amazon.com BookMatcher	amazon.com	Recommender	http://www.amazon.com/exec/obidos/bookmatcher
BargainFinder	Anderson Consulting	Shoppingagent	http://bf.cstar.ac.com/bf/
blaxxun	blaxxun interactive	Stellvertreter	http://www.blacksun.com/
bizzyB	GMD	Information Broker	http://fit.gmd.de/projects/cobra
CDnow Album Advisor	CDnow	Recommender	http://www.cdnow.com/cgi-bin/mserver/SID=674930511/pagename=/RP/AI/album_advisor.html
D'Agents	Dartmouth College	Programmiersprache	http://www.cs.dartmouth.edu/~agent/
DataRover	DataRover Mobile Systems	Netzwerk mobiler Kleincomputer	http://www.datarover.com/
Firefly	Firefly Network	Recommender, Collaborative Filtering	http://www.firefly.net/
Jango	Netbot	Shoppingagent	http://www.jango.com/
Kasbah	MIT	Shoppingagenten auf elektronischem Marktplatz	http://kasbah.media.mit.edu/cgi-bin/KasbahLogin
KIF	UMBC Computer Science and Electrical Engineering	Programmiersprache	http://www.cs.umbc.edu/kse/kif/
KQML	UMBC Computer Science and Electrical Engineering	Protokoll, Agentensprache	http://www.cs.umbc.edu/kqml/
Letizia	MIT	Recommender	http://lcs.www.media.mit.edu/people/lieber/Lieberary/Letizia/AIA/AIA.html
LifestyleFinder	Anderson Consulting	Recommender	http://bf.cstar.ac.com
LiveAgent	AgentSoft	Parallele Suchmaschine	http://www.agentsoft.com/liveagen.htm
MetaCrawler	go2net	Parallele Suchmaschine	http://www.metacrawler.com/
Microsoft Agent	Microsoft	Präsentations-/Konversationsagent	http://www.microsoft.com/workshop/imedia/agent/default.asp
Microsoft Office Assistant	Microsoft	Hilfefunktion	http://microsoft.com/office/
Netscape Navigator Mailfilter	Netscape	Mailfilter	http://www1.netscape.com/browsers/using/messenger/organizing.html
Persona	DFKI	Präsentationsagent	http://www.dfki.de/imedia/ppp/
Remembrance Agent	MIT	Tragbare Orientierungs- und Erinnerungshilfe	http://rhodes.www.media.mit.edu/people/rhodes/Papers/wear-ra.html
ShopBot	University of Washington	Shoppingagent	ftp://ftp.cs.washington.edu/pub/etzioni/softbots/agents97.ps
SodaBot	MIT	Entwicklungsumgebung	http://www.ai.mit.edu/people/sodabot/
TeleScript	General Magic	Programmiersprache	http://www.genmagic.com/
togglethis interactive characters	togglethis	Präsentationsagent	http://www.togglethis.com/
Voyager	ObjectSpace	Entwicklungsumbegung	http://www.objectspace.com/voyager/index.html
Webdoggie	MIT	Recommender, Collaborative Filtering	weitere Informationen nicht mehr online abrufbar